ΠΑΝΕΠΙΣΤΗΜΙΟ ΣΤΕΡΕΑΣ ΕΛΛΑΔΟΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΪΑΤΡΙΚΗ

Ανάπτυξη οδηγού χρήσης του biojava για την ανάπτυξη εφαρμογών Βιοπληροφορικής

Παπαθεοδοσίου Μερσίνη

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υπεύθυνος
Κακαρούντας Αθανάσιος

Λαμία, 2010

ΠΑΝΕΠΙΣΤΗΜΙΟ ΣΤΕΡΕΑΣ ΕΛΛΑΔΟΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΪΑΤΡΙΚΗ

Ανάπτυξη οδηγού χρήσης του biojava για την ανάπτυξη εφαρμογών Βιοπληροφορικής

Παπαθεοδοσίου Μερσίνη

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υπεύθυνος
Κακαρούντας Αθανάσιος

Λαμία, 2010

Ευχαριστίες

Καθώς η εργασία αυτή ρίχνει σιγά σιγά την αυλαία της προπτυχιακής μου φοίτησης, θα ήταν μεγάλη παράλειψη από μέρους μου να μην εκφράσω την ευγνωμοσύνη μου προς όλους εκείνους που με βοήθησαν και μου συμπαραστάθηκαν, ο καθένας με το δικό του τρόπο, καθ' όλη τη διάρκειά της.

Κατ' αρχάς θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κο Κακαρούντα Αθανάσιο, τόσο για την εμπιστοσύνη που μου έδειξε με την ανάθεση της παρούσας πτυχιακής εργασίας όσο και για την υποστήριξη και την καθοδήγησή του σε κάθε φάση της εκπόνησής της.

Θα ήθελα επίσης να απευθύνω τις ευχαριστίες μου στον κο Τσαμαρδίνο Ιωάννη (Επίκουρο Καθηγητή του Τμήματος Επιστήμης Υπολογιστών) και τους καθηγητές μου κο Παυλόπουλο Γεώργιο και κο Μπάγκο Παντελεήμων για τις βασικές γνώσεις που μου προσέφεραν πάνω στη Βιοπληροφορική, για τη βοήθειά τους και ιδιαίτερα για το χρόνο που μου αφιέρωσαν.

Ένα μεγάλο ευχαριστώ οφείλω επίσης στο φίλο κο Γεωργίου Αθανάσιο για τις πολύτιμες συμβουλές του σχετικά με τη δημιουργία του ιστοχώρου.

Θα ήθελα να ευχαριστήσω τους φίλους μου για την υποστήριξη που μου προσέφεραν, ιδιαίτερα δε το Βαγγέλη, το Στέλιο, το Δημήτρη, την Ειρήνη και την Εμμανουέλα, χωρίς τη βοήθεια της οποίας, όλα αυτά τα χρόνια, δε θα βρισκόμουν σήμερα στην ευχάριστη θέση να συγγράφω την παρούσα εργασία.

Τέλος, πρέπει να εκφράσω την ευγνωμοσύνη μου στην οικογένειά μου για την αμέριστη συμπαράστασή τους, ηθική και υλική, όλα αυτά τα χρόνια από τότε που ήμουν μαθήτρια μέχρι και το πέρας της φοιτητικής μου ζωής.

Πρόλογος – Περίληψη

Με το πέρασμα των χρόνων παρατηρείται συνεχόμενη πρόοδος των επιστημών συνεπώς και του τομέα της Βιοπληροφορικής. Εμφανίζονται ολοένα και περισσότερα προγράμματα και γλώσσες προγραμματισμού με σκοπό την παροχή διευκολύνσεων σχετικά με βασικές βιολογικές διεργασίες.

Η παρούσα πτυχιακή εργασία αφορά ένα από αυτά τα εργαλεία, το project Biojava. Σκοπός είναι η ανάπτυξη ενός πλήρους, εύχρηστου οδηγού χρήσης του project, που θα βοηθά τον προγραμματιστή στα πρώτα του βήματα κατά την ενασχόλησή του με αυτό. Θα παρουσιαστούν βασικά προγράμματα – παραδείγματα και προτεινόμενες ασκήσεις για την καλύτερη κατανόησή του και θα πραγματοποιηθεί σύγκριση με αντίστοιχα προγράμματα της γλώσσας προγραμματισμού Java.

Πιο συγκεκριμένα, το 1ο κεφάλαιο αποτελεί μια εισαγωγή στην επιστήμη της Βιοπληροφορικής. Ξεκινά με τον ορισμό της και την εξέλιξή της με το πέρασμα των χρόνων και συνεχίζει με την παρουσίαση χαρακτηριστικών της. Στο τέλος του κεφαλαίου γίνεται αναφορά σε πεδία εφαρμογής της.

Συνεχίζοντας, στο 2ο κεφάλαιο, γίνονται περιγραφές κάποιων από τα σημαντικότερα εργαλεία που χρησιμοποιούνται τόσο από την πλευρά του λογισμικού (γλώσσες και ειδικά ολοκληρωμένα πακέτα) όσο και από την πλευρά του υλικού. Γίνεται επίσης σύντομη αναφορά στους σύγχρονους τομείς έρευνας – προκλήσεις που καλούνται να δώσουν λύση τα εργαλεία αυτά. Κατανοείται με αυτό τον τρόπο η ανάγκη για τη δημιουργία καινούριων, με καλύτερες επιδόσεις, εργαλείων. Τέλος γίνεται μια σύντομη περιγραφή του Biojava και καταγράφεται και ο στόχος της παρούσας πτυχιακής.

Αφού έχει δημιουργηθεί ένα «βιοπληροφορικό υπόβαθρο» προχωράμε πλέον στην παρουσίαση του project. Στο 3ο κεφάλαιο γίνεται η εισαγωγή στο Biojava σε μια προσπάθεια να αναδειχτούν τα βασικά χαρακτηριστικά – δυνατότητές του. αναφέρονται βασικά βιολογικά πακέτα που περιέχει η τελευταία έκδοση αλλά και η επέκταση BiojavaX και παρουσιάζεται σύγκριση και κάποια πλεονεκτήματα του Biojava έναντι της γλώσσας προγραμματισμού Java αλλά και των άλλων γλωσσών. Επιπλέον αναφέρεται η εξέλιξη που αναμένεται να έχει στο κοντινό μέλλον.

Κατόπιν, στο κεφάλαιο 4, αναπτύσσεται ο οδηγός χρήσης του project με παραδείγματα βασικών βιολογικών διεργασιών και παραπομπές στον αντίστοιχο, κάθε φορά, κώδικα και κάποια ολοκληρωμένα παραδείγματα με τον κώδικα και τα αποτελέσματά τους. Επιπρόσθετα, γίνεται παρουσίαση του εκπαιδευτικού υλικού το οποίο έχει τη μορφή ενοτήτων – ασκήσεων, καθώς επίσης παρατίθενται και αναλυτικές οδηγίες εγκατάστασης, με συχνή παράθεση εικόνων για καλύτερη κατανόηση.

Στο 5ο κεφάλαιο, λαμβάνει χώρα η παρουσίαση του ιστοχώρου, που δημιουργήθηκε με σκοπό να φιλοξενήσει τον οδηγό χρήσης και το εκπαιδευτικό υλικό. Αφού τελειώσει η αναφορά στο web design, στο πλάνο που ακολουθήθηκε, στις βασικές αρχές που πρέπει να ακολουθούνται και στις προϋποθέσεις που πρέπει να πληρούνται, παρουσιάζονται αναλυτικά ο τρόπος δημιουργίας του οδηγού χρήσης,

τα προγράμματα που χρησιμοποιήθηκαν, καθώς και εικόνες από το τελικό αποτέλεσμα.

Έπειτα, στο 6ο κεφάλαιο παρουσιάζονται τα αποτελέσματα της σύγκρισης του Biojava με τη Java. Διαγράμματα, ξεχωριστά για κάθε σύγκριση, όπου γίνεται αντιπαράθεση των χρόνων εκτέλεσης των προγραμμάτων στις δύο γλώσσες, οδηγούν σε μια πιο τεκμηριωμένη αξιολόγηση του project.

Η παραπάνω αξιολόγηση μαζί με τα συμπεράσματα που διεξήχθησαν με το πέρας της παρούσας πτυχιακής εργασίας συγκεντρώνονται στο τελευταίο κεφάλαιο.

Εν κατακλείδι παρουσιάζονται οι πηγές – αναφορές και τα παράρτημα με κώδικα για τις βασικές βιολογικές διεργασίες σε Biojava (παράρτημα 1) αλλά και σε Java (παράρτημα 2) και ολοκληρωμένα παραδείγματα (παράρτημα 3).

Λέξεις – κλειδιά

Βιοπληροφορική, Biojava, project, οδηγός χρήσης, βιολογικές διεργασίες εκπαιδευτικό υλικό, σύγκριση

Abstract

Over the years, science has been constantly progressing and, consequently, so has Bioinformatics. There is a constantly increasing number of programmes and programming languages, which aim to make basic biological procedures easier to be carried out.

This thesis concerns one of these tools: the Biojava Project. The aim is to develop a comprehensive and easy-to-use guide, which will help the programmers take their first steps in using the Project. Basic programmes– examples and suggested exercises will be presented that will help to understand the Project better – and there will also be a comparison to equivalent Java programmes.

More specifically, *Chapter 1* is an introduction to Bioinformatics. It starts by offering a definition of the term, as well as an account of how this science has been developed through the years, and continues to the presentation of its characteristics. At the end of the chapter, the application fields of Bioinformatics are mentioned.

Chapter 2 describes some of the most important tools that are currently used, both software (languages and completed packages) and hardware ones. There is also a brief mention of the contemporary research sectors, which, themselves, pose the challenges these tools must face. In this way, the need to create new, more efficient tools becomes clearer. Lastly, there is a short description of Biojava and a presentation of the aim of this thesis.

Having set the basis of Bioinformatics, we proceed to the presentation of the Project. In *Chapter 3*, there is an introduction to Biojava in an attempt to show its basic characteristics and potential. There is mention of basic biological packages the latest edition includes, as well as the BiojavaX extension, and there is a comparison of Biojava against the Java programming language, as well as other languages. Moreover, there is mention of Biojava's expected development in the near future.

In *Chapter 4*, the project's user's guide is presented, along with the installation instructions. This guide is frequently accompanied by images, in order to achieve better understanding, as well as examples of biological procedures with references to the equivalent codes. Furthermore, we present the educational material that is in the form of units – exercises and some complete examples with their codes and results.

In *Chapter 5*, there is a presentation of the website that has been created in order to host the user's guide and the educational material. After discussing the web design, the plan that was followed, the basic principles that have to be followed and the requirements that have to be met *(5.2)*, there is an analytical presentation of the way the user's guide was created, the programmes used, as well as images of the end result.

In *Chapter 6*, we present the results of the comparison between Biojava and Java. Charts, one for each comparison, present the execution times of programs in both languages and result in a more informed assessment of the project.

This assessment together with the findings conducted at the end of this thesis are gathered in the last chapter.

Finally, there is a presentation of the sources and references, as well as the appendixes with codes regarding the basic biological procedures both in Biojava *(appendix 1)* and Java *(appendix 2)* and some complete examples *(appendix 3)*.

Key Words
Bioinformatics, Biojava, project, user's guide, biological procedures, educational material, comparison

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΕΧΟΜΕΝΑ ΣΧΗΜΑΤΩΝ

ΠΕΡΙΕΧΟΜΕΝΑ ΕΙΚΟΝΩΝ

1ο ΚΕΦΑΛΑΙΟ
ΕΙΣΑΓΩΓΗ

Περίληψη

Στις μέρες μας παρατηρείται τόσο μεγάλη ανάπτυξη της επιστήμης των υπολογιστών όσο και σημαντική πρόοδος των επιτευγμάτων στον τομέα της Βιολογίας οι οποίες είναι σε πολύ μεγάλο βαθμό αλληλένδετες, καθώς η πρώτη συντελεί στην εξέλιξη της δεύτερης. Έτσι, λοιπόν, από τον συνδυασμό κυρίως αυτών των δύο γεννιέται μια νέα επιστήμη – η Βιοπληροφορική. Στόχος αυτού του κεφαλαίου είναι να περιγράψει εκτενώς τον όρο Βιοπληροφορική, το πώς αναπτύχθηκε με το πέρασμα των χρόνων, ποιες είναι οι επιστήμες τις οποίες συνδυάζει και ποια η χρησιμότητά της.

Βασικές έννοιες

- Βιοπληροφορική
- Πολυεπίπεδη επιστήμη
- Βάσεις δεδομένων
- Εργαλεία ανάλυσης βιολογικών δεδομένων
- Συστήματα ανάκτησης πληροφοριών
- Χρησιμότητα Βιοπληροφορικής
- Πεδία εφαρμογής

1.1 Εισαγωγή στη Βιοπληροφορική

1.1.1 Ορισμός

Σύμφωνα με το Εθνικό Ινστιτούτο Υγείας (National Institutes of Health- NIH) ο ορισμός της επιστήμης της Βιοπληροφορικής δίδεται ως εξής:

«Βιοπληροφορική είναι η έρευνα, η ανάπτυξη, ή η εφαρμογή υπολογιστικών εργαλείων και προσεγγίσεων για διεύρυνση της χρήσης βιολογικών, ιατρικών, συμπεριφοριστικών δεδομένων ή δεδομένων υγείας, συμπεριλαμβανομένων εκείνων για την απόκτηση, αποθήκευση, οργάνωση, ανάλυση, ή οπτικοποίηση τέτοιων στοιχείων». [9]

Αποτελεί, δηλαδή, έναν νέο επιστημονικό χώρο, όπου γίνεται εκμετάλλευση τόσο της υπολογιστικής ισχύος όσο και των γνώσεων που διατίθενται από διάφορες επιστήμες όπως η Βιολογία, η Βιοχημεία, η Πληροφορική, η Στατιστική, τα Εφαρμοσμένα Μαθηματικά (παραδείγματος χάριν διάφορα εργαλεία και μέθοδοι που προσφέρουν) προκειμένου να επιτευχθεί η επίλυση ποικίλων βιολογικών προβλημάτων, σε μοριακό κυρίως επίπεδο και η καλύτερη δυνατή εκμετάλλευση των βιολογικών δεδομένων για την εξαγωγή επιθυμητής γνώσης από αυτά.

1.1.2 Ετυμολογία

Ο όρος Βιοπληροφορική προέρχεται από τις λέξεις «βίος» (=ζωή) και «πληροφορία» ή ακόμα καλύτερα από τα συνθετικά «Βιο» και «Πληροφορική», όπου το πρώτο σχετίζεται με τη Μοριακή Βιολογία και το δεύτερο με την επιστήμη των υπολογιστών.

1.1.3 Ιστορική αναδρομή

Η Βιοπληροφορική αποτελεί έναν αναπτυσσόμενο κλάδο που γνωρίζει μεγάλη άνθιση και χαίρει ιδιαίτερης αναγνώρισης τα τελευταία χρόνια. Ωστόσο, αν ερευνήσουμε καλύτερα, διαπιστώνουμε ότι τα πρώτα βήματα στο χώρο αυτό βρίσκονται αρκετά πίσω στο χρόνο.

Από τη *δεκαετία του '50*, κιόλας ,έχουμε σημαντικά επιτεύγματα. Ήδη από το 1951 γίνεται πρόταση της δομής για την α-έλικα και τη β-πτυχωτή επιφάνεια (Pauling και Corey) και δυο χρόνια αργότερα λαμβάνει χώρα η πρόταση του μοντέλου της διπλής έλικας για το DNA από τους Watson και Crick. Η πρώτη πρωτεϊνική ακολουθία, η βοοειδής ινσουλίνη, ανακαλύπτεται το 1956 και αποτελείται από 51 κατάλοιπα. Επίσης, κατά την περίοδο αυτή, κατασκευάζεται το πρώτο ολοκληρωμένο κύκλωμα από τον Jack Kilby στο Texas Instruments και ιδρύεται η Advanced Research Projects Agency (ARPA) στις ΗΠΑ.

Μέσα στην *επόμενη δεκαετία* ο Pauling δημοσιεύει τη θεωρία του για την εξέλιξη των μορίων, γίνεται γνωστή η πρώτη ακολουθία νουκλεϊνικού οξέος (t-RNA της αλανίνης της μαγιάς με 77 βάσεις) και ένα χρόνο μετά, η Dayhoff χρησιμοποιεί όλα τα στοιχεία για τις διαθέσιμες ακολουθίες για να δημιουργήσει την πρώτη ολοκληρωμένη βάση δεδομένων (Άτλας πρωτεϊνικών ακολουθιών). Πρωτόκολλα δικτύου Packet-switching παρουσιάζονται στο ARPA και δημιουργείται το ARPANET με τη σύνδεση υπολογιστών στο Στάνφορντ, στο UCSB, στο πανεπιστήμιο της Γιούτα και στο UCLA.

Στη *δεκαετία του '70* ανακοινώνεται ο Αλγόριθμος Needleman-Wunsch και ο Ray Tomlinson (BBN) εφευρίσκει το πρόγραμμα ηλεκτρονικού ταχυδρομείου (email). Το 1972 δημιουργείται το πρώτο ανασυνδυασμένο μόριο DNA από τον Paul Berg και την ομάδα του και την ίδια χρονιά, ανακοινώνεται η Protein Databank, συμπληρώνοντας το έργο της Dayhoff, με μια συλλογή από 10 πρωτεϊνικές δομές προερχόμενες από κρυσταλλογραφία ακτινών χ. Μια ακόμη σημαντική ιδέα έρχεται από τους Vint Cerf και Robert Kahn οι οποίοι σκέφτονται τη σύνδεση των δικτύων ηλεκτρονικών υπολογιστών σε ένα «διαδίκτυο» και το Πρωτόκολλο Ελέγχου Μετάδοσης (TCP-Transmition Control Protocol). Ανακοινώνεται από τον P. H. O'Farrell η δισδιάστατη ηλεκτροφόρηση, όπου ο διαχωρισμός των πρωτεϊνών σε πήκτωμα πολυακρυλαμιδίου SDS συνδυάζεται με το διαχωρισμό σύμφωνα με ισοηλεκτρικά σημεία. Την ίδια χρονιά (1975) ιδρύεται η Microsoft Corporation από τους Bill Gates και Paul Allen και αργότερα αναπτύσσεται το Unix-To-Unix Copy Protocol (UUCP) από τα εργαστήρια Bell.

Στη *δεκαετία του '80*, όπως ήταν αναμενόμενο, οι εξελίξεις συνεχίζονται. Από τις πρώτες που έλαβαν χώρα ήταν η παρουσίαση των πολυδιάστατων NMR για τον προσδιορισμό της δομής των πρωτεϊνών *(Kumar, A.; Ernst, R.R.; Wüthrich, K.; Biochem. Biophys. Res. Comm., 1980, 95:, 1)*, καθώς επίσης και η ίδρυση της IntelliGenetics, Inc. στην Καλιφόρνια με πρώτο προϊόν της το πακέτο προγραμμάτων IntelliGenetics για την ανάλυση των ακολουθιών DNA και των πρωτεϊνών. Το 1981 δημοσιεύεται ο αλγόριθμος Smith-Waterman για τη στοίχιση ακολουθιών και η IMB εισάγει στην αγορά το δικό της προσωπικό ηλεκτρονικό υπολογιστή. Δημιουργείται το Genetics Computer Group (GCG) ως τμήμα του Πανεπιστημίου του Wisconsin του Wisconsin Biotechnology Center με πρώτο προϊόν της εταιρείας το πακέτο εργαλείων Μοριακής Βιολογίας Wisconsin. Ολοένα και περισσότεροι αλγόριθμοι έρχονται στην επιφάνεια όπως αυτός για την αναζήτηση βάσεων δεδομένων ακολουθιών *(Wilbur-Lipman)* και ο FASTP για γρήγορη αναζήτηση ομοιότητας ακολουθίας. Το 1986 δημιουργείται η βάση δεδομένων SWISS-PROT και ένα χρόνο αργότερα εμφανίζεται μια εκ των πιο διαδεδομένων γλωσσών στο χώρο της Βιοπληροφορικής, η Perl (Practical Extraction Report Language), από τον Larry Wall. Η δεκαετία αυτή κλείνει με την ίδρυση του NCBI (National Center for Biotechnology Information) το 1988.

Από το *1990* και μέσα στα επόμενα δέκα χρόνια έχουμε εφαρμόζεται ευρέως το πρόγραμμα BLAST(Altschul, κ. συν.), ιδρύεται το Molecular Applications Group στην Καλιφόρνια από τους Michael Levitt και Chris Lee με πρώτα προϊόντα που χρησιμοποιούνται για τη μοντελοποίηση μορίων και τη σχεδίαση πρωτεϊνών και η InforMax με προϊόντα που επιχειρούν ανάλυση ακολουθιών, διαχείριση δεδομένων και βάσεων δεδομένων, αναζήτηση, χαρτογράφηση κ.ά.. Περιγράφεται η δημιουργία και η χρήση των επεκφρασμένων ετικετών ακολουθίας (*ESTs – Expressed Sequence Tags*) *(J. Craig Venter, κ. συν., Science, 252: 1651-1656)* και παράλληλα ιδρύεται το ινστιτούτο για την έρευνα του γονιδιώματος από τον Craig Venter. Η βάση δεδομένων πρωτεϊνικών μοτίβων PRINTS δημοσιεύεται από τους Attwood και Beck και ιδρύεται το Ευρωπαϊκό Ινστιτούτο Βιοπληροφορικής (European Bioinformatics Institute-EBI) της EMBL. Η Microsoft παρουσιάζει την πρώτη έκδοση του Internet Explorer, η Sun την πρώτη έκδοση της Java και σε συνεργασία με τη Netscape παρουσιάζουν την πρώτη έκδοση της JavaScript. Το 1996 ανακοινώνεται η βάση δεδομένων Prosite από τον Bairoch και τους συνεργάτες του και ένα χρόνο μετά δημοσιεύεται το γονιδίωμα της E. coli (4.7 Mbp). Επιπροσθέτως έχουμε ίδρυση αρκετών εταιρειών όπως της LION BioScience AG (μια ολοκληρωμένη εταιρεία γονιδιωματικής με μεγάλη έμφαση στη Βιοπληροφορική), του ελβετικού Ινστιτούτου Βιοπληροφορικής σαν ένα μη κερδοσκοπικό ίδρυμα, της Inpharmatica (μιας νέας εταιρείας Γονιδιωματικής και Βιοπληροφορικής) και της GeneFormatics (μιας εταιρείας επικεντρωμένης στην ανάλυση και την πρόβλεψη της δομής και της λειτουργίας των πρωτεϊνών).

Από το *2000 και μετά* οι προσπάθειες συνεχίζονται. Έχουμε εύρεση της ακολουθίας των γονιδίων A. thaliana (100 Mb) και D. melanogaster (180Mb) με αποκορύφωμα την αποκωδικοποίηση του ανθρώπινου γονιδιώματος που γεννά

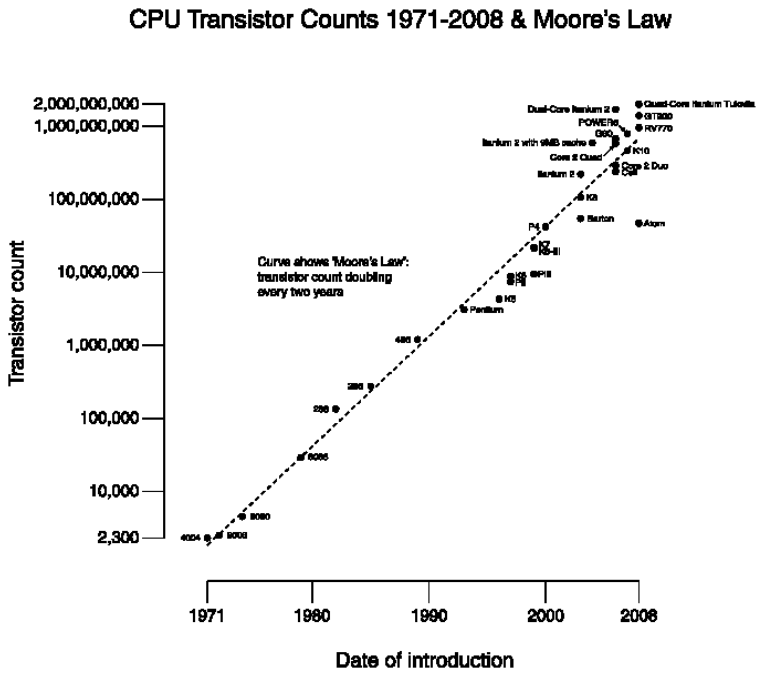
ελπίδες για το μέλλον της ανθρωπότητας και χαρακτηρίζεται ισοδύναμη με την ανακάλυψη του τροχού.

Με το πέρασμα των χρόνων συνεχίζεται η ανάπτυξη του τομέα της Βιοπληροφορικής, ως ενιαίου πλέον κλάδου. Βλέπουμε προσανατολισμό διαφόρων επιστημόνων στον τομέα αυτό και παρουσίαση ολοένα και περισσότερων καινοτομιών που κάνουν την έρευνα ευκολότερη. [10]

1.2 Χαρακτηριστικά Βιοπληροφορικής

1.2.1 Ανάπτυξη

Τα τελευταία χρόνια παρατηρείται τεράστια ανάπτυξη της τεχνολογίας και της επιστήμης της Πληροφορικής και ενσωμάτωσή αυτής σε όλους τους τομείς της καθημερινής μας ζωής, αλλά και σε άλλες επιστήμες με σκοπό την ενίσχυση και την περεταίρω ανάπτυξή τους. Σύμφωνα μάλιστα και με το νόμο του Moore (*«Η χωρητικότητα των επεξεργαστών σε transistors (συνεπώς και η ταχύτητά τους) διπλασιάζεται κάθε 2 χρόνια»*) [3] υπάρχει ταχεία ανάπτυξη της υπολογιστικής ισχύος πράγμα το οποίο αποδείχτηκε και όντως συμβαίνει κάθε 18 μήνες.



Εικόνα 1: Mooore's law (Πηγή:[6])

Παράλληλα βλέπουμε ότι τις τελευταίες δεκαετίες η πρόοδος στον τομέα της Μοριακής Βιολογίας σε συνδυασμό με την πρόοδο της Γονιδιωματικής Τεχνολογίας έχουν οδηγήσει σε εκρηκτική ανάπτυξη της βιολογικής πληροφορίας. Από το 1953, με την ανακάλυψη του μοντέλου διπλής έλικας από τους Watson και Crick, ανοίχτηκαν νέοι ορίζοντες όσον αφορά την έρευνα στους τομείς αυτούς. Με το πέρασμα των χρόνων παρατηρούνται ολοένα και περισσότερες ανακαλύψεις, οι

οποίες, βέβαια, φτάνουν στο αποκορύφωμα τους τον Απρίλιο του 2003 με την αποκωδικοποίηση του ανθρώπινου γονιδιώματος.

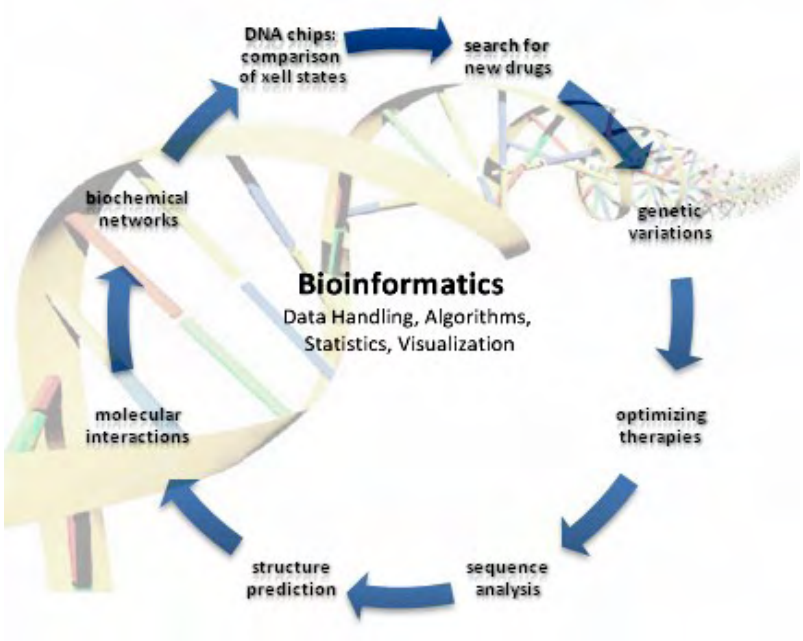
Κάνουμε λόγο πλέον για τεράστιο όγκο πληροφοριών (πλήρη γονιδιώματα βιολογικών ειδών συμπεριλαμβανομένου και του ανθρώπινου, πρωτεϊνικές ακολουθίες, τρισδιάστατες δομές πρωτεϊνών, πληροφορίες για βιοποικιλότητα, κ.λπ.) που είναι αδύνατο πια να γίνει η διαχείρισή τους, ο έλεγχος και η ανάλυσή τους από τον άνθρωπο, εφόσον κάτι τέτοιο θα ήταν εξαιρετικά δύσκολο (με μεγάλη πιθανότητα λάθους) και ιδιαίτερα χρονοβόρο. Οπότε αναλαμβάνουν τις εργασίες αυτές οι μηχανές/υπολογιστές και παρουσιάζεται επιτακτική η ανάγκη συνεργασίας της Πληροφορικής με τη Βιολογία, δύο τομέων, που εκ πρώτης όψεως φαίνονταν να είναι ασυμβίβαστοι μεταξύ τους.

Είναι προφανές ότι έπρεπε οπωσδήποτε να αναπτυχθούν κάποια εργαλεία λογισμικού ή και υλικού ή να προσαρμοστούν, σε πολλές περιπτώσεις, τα ήδη υπάρχοντα ώστε να γίνει η διαχείριση όλης αυτής της πληροφορίας.

1.2.2 Βιοπληροφορική: πολυεπίπεδη επιστήμη

Πρόκειται για ένα διεπιστημονικό τομέα που συνδυάζει τις παρακάτω επιστήμες:

- Βιολογία (κυρίως Μοριακή)
- Πληροφορική
- Βιοχημεία
- Εφαρμοσμένα Μαθηματικά και Στατιστική
- Ιατρική
- Φυσική



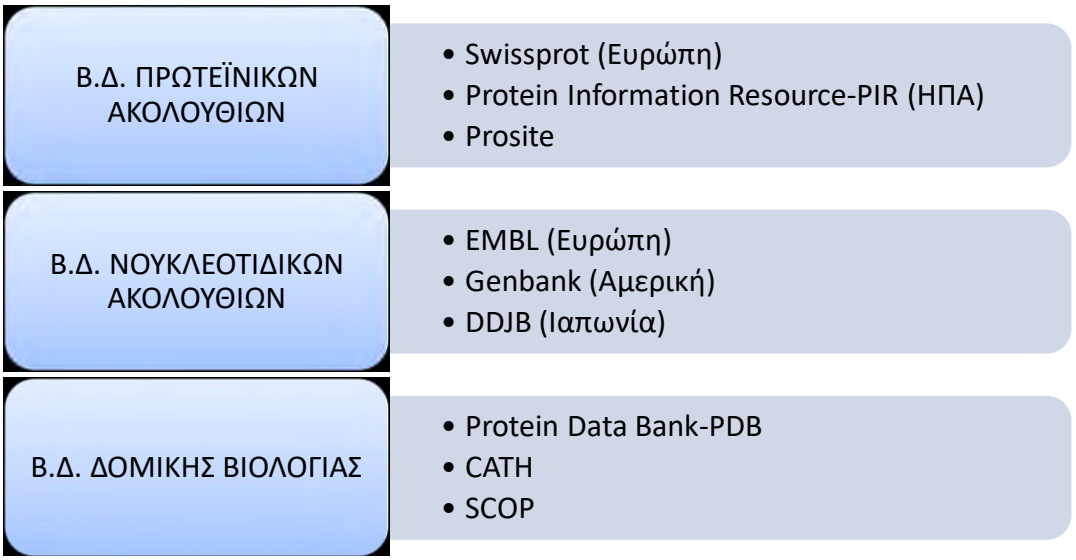
Εικόνα 2: Τομείς έρευνας στη Βιοπληροφορική

1.2.3 Τομείς έρευνας-χρησιμότητα

– *Αποδοτική οργάνωση και αποθήκευση των βιολογικών δεδομένων*

Δημιουργία τεράστιων βάσεων δεδομένων που καθιστούν εύκολη την αναζήτηση και την ανάκτηση δεδομένων από τους ερευνητές καθώς και την ενημέρωσή τους, όταν προκύπτουν νέες πληροφορίες.

Υπάρχουν διάφοροι τύποι βάσεων δεδομένων [11]. Στο παρακάτω διάγραμμα παρατίθενται οι τύποι αυτοί και αναφέρονται ονομαστικά οι κυριότεροι εκπρόσωποί τους.



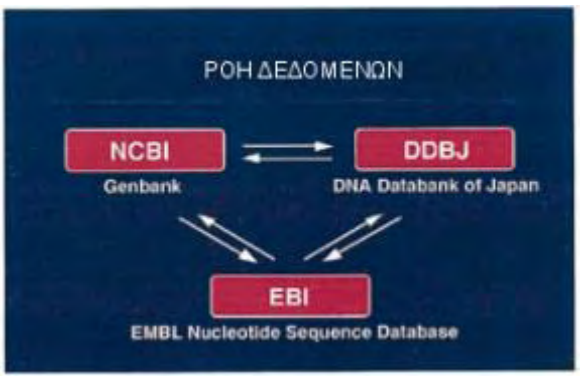
Σχήμα 1: Βιολογικές Βάσεις Δεδομένων

Η **SWISS-PROT** [12] είναι μια βάση δεδομένων πρωτεϊνικών ακολουθιών που ιδρύθηκε το 1986 και στις μέρες μας συντηρείται από το Ελβετικό ινστιτούτο Βιοπληροφορικής (Swiss Institute of Bioinformatics) σε συνεργασία με το Ευρωπαϊκό Ινστιτούτο Βιοπληροφορικής (European Bioinformatics Institute). Η παρούσα έκδοση της SWISS-PROT περιέχει 104948 καταχωρήσεις (Ιανουάριος 2002) στις οποίες εκτός από την ακολουθία υπάρχουν και συμπληρωματικά σχόλια όπως, βιβλιογραφικές αναφορές, γενικά στοιχεία δευτεροταγούς δομής, σύνδεσμοι σε άλλες βάσεις δεδομένων σχετικές με κάθε εγγραφή καθώς και σημειώσεις για τη βιολογική λειτουργία (αν είναι γνωστή) και άλλες χρήσιμες πληροφορίες.

Η **Protein Information Resource (PIR)** [13] εδράζεται στο Πανεπιστήμιο του Georgetown και αποτελεί τμήμα του Εθνικού Ιδρύματος Βιοϊατρικής Έρευνας (NBRF) των Η.Π.Α. Η PIR περιλαμβάνει μια σειρά από βάσεις δεδομένων που σχετίζονται με τη μελέτη των πρωτεϊνών με κυριότερη από αυτές την PIR-International Protein Sequence Database (PSD). Η PSD αποτελεί όπως και η SWISS-PROT μια βάση δεδομένων πρωτεϊνικών ακολουθιών συνοδευόμενη από συμπληρωματικά σχόλια. Τα δεδομένα της PSD προκύπτουν από την συνεργασία της PIR με το Munich Information Center for Protein Sequences (MIPS) και την Japanese International Protein Information Database (JIPID). Η τελευταία έκδοση της PSD

(Rel. 71.03, Φεβρουάριος 2002) περιλαμβάνει 283138 εγγραφές. Πρέπει να σημειωθεί ότι η PIR-PSD χωρίζεται σε 4 υποενότητες τις PIR1, PIR2, PIR3 και PIR4. Μεταξύ των PIR1 και PIR2 δεν υπάρχει ουσιαστική διαφορά. Ο διαχωρισμός διατηρείται κυρίως για ιστορικούς λόγους. Οι ενότητες αυτές περιέχουν το 99% των εγγραφών της βάσης και τα κριτήρια ταξινόμησης και σχολιασμού είναι ακριβώς τα ίδια. Αντίθετα οι εγγραφές της PIR3 δεν έχουν ακόμη υποστεί έλεγχο και σχολιασμό. Τα περιεχόμενα της PIR4 είναι ακολουθίες οι οποίες είτε δεν συναντώνται στη φύση είτε δεν εκφράζονται υπό φυσιολογικές συνθήκες. Επίσης μπορεί να περιέχει ακολουθίες που έχουν συντεθεί de novo σε εργαστήριο. Σε κάθε περίπτωση πάντως έχουν υποστεί έλεγχο και σχολιασμό από τους υπεύθυνους της βάσης.

Η **PROSITE** [14] είναι μια βάση, ταξινόμησης σε οικογένειες, πρωτεϊνικών ακολουθιών και αυτοτελών περιοχών ακολουθιών (sequence domains). Βασίζεται στη γενικότερη παρατήρηση ότι ενώ υπάρχει ένας τεράστιος αριθμός διαφορετικών πρωτεϊνών στη φύση, αυτές μπορούν να ομαδοποιηθούν, με βάση την ομοιότητα στην ακολουθία τους, σε ένα μικρό αριθμό οικογενειών. Οι πρωτεΐνες ή οι αυτοτελείς δομικές περιοχές που ανήκουν στην ίδια οικογένεια έχουν την ίδια λειτουργία και προέρχονται από κοινό πρόγονο. Είναι φανερό ότι πρωτεΐνες που ανήκουν στην ίδια οικογένεια, έχουν τμήματα της ακολουθίας τους που είναι περισσότερο συντηρημένα στην πορεία της εξέλιξης τους. Αυτές οι περιοχές σχετίζονται άμεσα με τη λειτουργία τους και με τη δομή των πρωτεϊνών στο χώρο. Αναλύοντας τις ακολουθίες πρωτεϊνών που ανήκουν στην ίδια οικογένεια είναι δυνατό να προκύψει ένα «αποτύπωμα» χαρακτηριστικό για κάθε ομάδα, ικανό ώστε να τη διαχωρίζει από τις άλλες πρωτεϊνικές αλληλουχίες που δεν ανήκουν στην οικογένεια αυτή. Μια ανάλογη περίπτωση αποτελεί η λήψη αποτυπωμάτων από την αστυνομία. Ένα «αποτύπωμα» είναι ικανό για να ταυτοποιήσει ένα άτομο. Παρόμοια και στις πρωτεΐνες η χρήση ενός τέτοιου «αποτυπώματος» μπορεί να χρησιμεύσει για να ταξινομηθεί μια άγνωστη πρωτεϊνική αλληλουχία σε μια γνωστή οικογένεια πρωτεϊνών δίνοντάς μας ενδείξεις για την πιθανή λειτουργία της. Αυτή τη στιγμή η PROSITE περιέχει «αποτυπώματα» για 1000 περίπου οικογένειες. Για κάθε οικογένεια υπάρχει λεπτομερής ανάλυση για τη δομή και τη λειτουργία των πρωτεϊνών που ανήκουν σ' αυτή.



Εικόνα 3: Ροή της πληροφορίας των βάσεων – συνεχής επικοινωνία (Πηγή: [3])

Η **EMBL** Nucleotide Sequence Database [15] αποτελεί τη μεγαλύτερη βάση νουκλεοτιδικών αλληλουχιών στην Ευρώπη και βρίσκεται υπό την αιγίδα του Ευρωπαϊκού Εργαστηρίου Μοριακής Βιολογίας (EMBL). Εδράζεται και συντηρείται στο Ευρωπαϊκό Ινστιτούτο Βιοπληροφορικής (EBI) στο Cambridge, UK. Τα δεδομένα προέρχονται από ανεξάρτητα ερευνητικά εργαστήρια καθώς και από ομάδες που ασχολούνται με τον προσδιορισμό των γονιδιωμάτων διαφόρων οργανισμών. Η κατάθεση ακολουθιών στην EMBL-Bank είναι μια διαδικασία απλή και πραγματοποιείται μέσω του Διαδικτύου κατ' αντίστοιχο τρόπο με αυτό της GENBANK. Στη συνέχεια οι νεοεισερχόμενες ακολουθίες υφίστανται επεξεργασία και σχολιασμό από τους υπευθύνους της βάσης προτού γίνουν διαθέσιμες στην επιστημονική κοινότητα. Επιπλέον μέσω του Διαδικτύου παρέχονται μια σειρά από εργαλεία ανάλυσης ακολουθιών (π.χ. Fasta, BLAST). Η τελευταία έκδοση της EMBL-Bank (Rel. 69 - Μάρτιος 2002) περιέχει 15960527 εγγραφές, ενώ ο συνολικός αριθμός των νουκλεοτιδίων φτάνει τα 17868806247.

Η **GENBANK** [16] αποτελεί μια βάση δεδομένων νουκλεοτιδικών αλληλουχιών που είναι ελεύθερα διαθέσιμη στην επιστημονική κοινότητα. Βρίσκεται υπό την αιγίδα του Εθνικού Ινστιτούτου Υγείας των Η.Π.Α. Η κύρια πηγή της πληροφορίας που περιέχεται στην GENBANK προέρχεται από απευθείας υποβολές δεδομένων, όπως προκύπτουν από πειραματικές διεργασίες διαφόρων ερευνητικών ομάδων. Τα νεοεισερχόμενα δεδομένα υφίστανται επεξεργασία και προστίθενται σχόλια (annotation) για την διευκόλυνση των ερευνητών. Ανά τακτά χρονικά διαστήματα τα ήδη κατατεθειμένα δεδομένα επανεξετάζονται και γίνονται διορθώσεις αν προκύπτουν νέα δεδομένα σχετικά με τις εγγραφές. Η διαδικασία κατάθεσης των δεδομένων μπορεί να πραγματοποιηθεί πολύ γρήγορα μέσω του Διαδικτύου με την συμπλήρωση κατάλληλης φόρμας και στη συνέχεια οι υπεύθυνοι της βάσης αναλαμβάνουν το σχολιασμό της εγγραφής και τη δημοσιοποίηση της στην βάση. Έχει διαπιστωθεί ότι κάθε 14 μήνες ο αριθμός των νουκλεοτιδικών βάσεων που περιέχονται στην GENBANK διπλασιάζεται με αποτέλεσμα η παρούσα έκδοση (Rel. 128, Φεβρουάριος 2002) να περιέχει 15465325 ακολουθίες με τον συνολικό αριθμό βάσεων να φτάνει τις 17089143893.

Η **DNA Databank of Japan (DDJB)** [17] ιδρύθηκε το 1986 στο Εθνικό Ινστιτούτο Γενετικής (NIG) το οποίο βρίσκεται υπό την αιγίδα του Υπουργείου Παιδείας, Επιστημών και Αθλητισμού της Ιαπωνίας. Αποτελεί τη μοναδική διεθνώς αναγνωρισμένη βάση νουκλεοτιδικών αλληλουχιών στην Ιαπωνία, ενώ η κύρια πηγή δεδομένων της είναι οι εργασίες Ιαπώνων ερευνητών. Επιπλέον παρέχονται μια σειρά από εργαλεία για την ανάλυση των νουκλεοτιδικών αλληλουχιών. Η παρούσα έκδοση της DDJB (Rel. 48, Ιανουάριος 2002) περιέχει 15016100 εγγραφές, ενώ ο συνολικός αριθμός των νουκλεοτιδικών βάσεων που περιέχονται στις ακολουθίες είναι 16197713855.

Η **Protein Data Bank (PDB)** [18] αποτελεί τη μοναδική βάση παγκοσμίως όπου είναι κατατεθειμένες οι τρισδιάστατες δομές βιολογικών μακρομορίων. Ιδρύθηκε το 1971 στα Brookhaven National Laboratories (BNL) των ΗΠΑ και περιελάμβανε 7

δομές μακρομορίων, όπως αυτές προέκυψαν από κρυσταλλογραφικές μελέτες. Ο ρυθμός αύξησης των εγγραφών στη δεκαετία του '70 ήταν πολύ μικρός. Από το 1980 και μετά, λόγω της τεχνολογικής εξέλιξης σε κάθε στάδιο του προσδιορισμού των δομών, ο ρυθμός προσθήκης δεδομένων στην PDB αυξήθηκε δραματικά. Πλέον στην βάση περιλαμβάνονται και δομές όπως προκύπτουν με φασματοσκοπία μάζας (NMR). Στην παρούσα φάση (Μάρτιος 2002) η PDB περιλαμβάνει 17493 δομές βιομορίων. Οι εγγραφές στην PDB εκτός από τις συντεταγμένες των ατόμων που απαρτίζουν τη δομή περιλαμβάνουν και επιπρόσθετα βοηθητικά στοιχεία, όπως βιβλιογραφικές αναφορές, λεπτομέρειες για τον προσδιορισμό της δομής, καθώς και άλλα στοιχεία που προκύπτουν από τη συγκεκριμένη δομή. Κάθε δομή προτού διατεθεί στο κοινό υφίσταται έλεγχο για την ορθότητα της με τη χρήση ειδικού λογισμικού. Στη συνέχεια εφόσον περάσει τις δοκιμές με επιτυχία αποκτά ένα χαρακτηριστικό κωδικό και προστίθεται στη βάση.

Η **CATH** [19] είναι μια βάση ιεραρχικής ταξινόμησης των πρωτεϊνικών δομών που είναι κατατεθειμένες στην PDB με βάση τις αυτοτελείς δομικές περιοχές (domains) που τις απαρτίζουν. Για τον καταρτισμό της CATH δεν λαμβάνονται υπόψη μη πρωτεϊνικές δομές, ενώ οι πρωτεϊνικές δομές που περιέχονται πρέπει να είναι προσδιορισμένες σε διακριτικότητα υψηλότερη των 3 Angstroms. Η CATH χρησιμοποιεί κυρίως αυτοματοποιημένες μεθόδους για την ταξινόμηση. Σε ειδικές περιπτώσεις, όμως, τα ανθρώπινα κριτήρια είναι δυνατόν να δώσουν καλύτερα αποτελέσματα από τις αυτοματοποιημένες μεθόδους, όποτε και προτιμούνται.

Τα 4 κύρια επίπεδα της ιεραρχίας είναι η Τάξη (Class), η Αρχιτεκτονική (Architecture), η Τοπολογία (Οικογένεια διπλώματος) (Topology (fold family)) και η Ομόλογη Οικογένεια (Homologous superfamily).

Οι πρωτεΐνες που αποτελούνται από περισσότερα του ενός domains αναλύονται στα επιμέρους στοιχεία αυτόματα με βάση ειδικούς αλγόριθμους αναγνώρισης domains. Με την αυτόματη διαδικασία κατατάσσονται το 53% των δομών. Οι υπόλοιπες διαχωρίζονται στα επιμέρους domains με παρατηρήσεις που προκύπτουν είτε από τους αλγόριθμους αυτόματου διαχωρισμού είτε από τη βιβλιογραφία. Η ταξινόμηση πραγματοποιείται μόνο στις αυτοτελείς δομικές περιοχές.

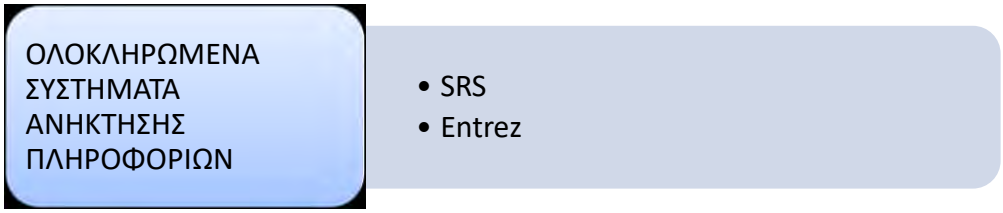
Η βάση **SCOP** [20] έχει σαν βασικό στόχο την ανάλυση των δομικών και εξελικτικών σχέσεων μεταξύ όλων των πρωτεϊνών γνωστής δομής κατατεθειμένων στην Protein Data Bank (PDB). Για την αναγνώριση των παραπάνω σχέσεων και την ταξινόμηση, με βάση τις σχέσεις αυτές, των πρωτεϊνών η διαδικασία που διεξάγεται δεν είναι αυτοματοποιημένη, αλλά πραγματοποιείται αποκλειστικά με βάση τον ανθρώπινο παράγοντα μετά από λεπτομερή μελέτη και σύγκριση των πρωτεϊνικών δομών. Αυτοματοποιημένες μέθοδοι χρησιμοποιούνται μόνο για την ομοιογένεια των δεδομένων που περιέχονται στη βάση.

Η ταξινόμηση των πρωτεϊνών, όπως αναφέρθηκε παραπάνω γίνεται με βάση δομικές και εξελικτικές σχέσεις. Τα βασικά επίπεδα ταξινόμησης είναι η οικογένεια (Family), υπέρ-οικογένεια (Superfamily), το δίπλωμα (Fold) και η τάξη (Class).

- Οικογένεια (Family): Ξεκάθαρη εξελικτική σχέση μεταξύ των μελών.
- Υπέρ-οικογένεια (Superfamily): Τα μέλη της έχουν πιθανά προέλθει από κοινό πρόγονο.

- Δίπλωμα (Fold): Εμφάνιση ομοιότητας σε επίπεδο δομής.
- Τάξη (Class): Τέσσερις κύριες δομικές κατηγορίες πρωτεϊνών έχουν ταυτοποιηθεί με βάση το δίπλωμα των στοιχείων δευτεροταγούς δομής τους, τις all-α (η δομή σχηματίζεται από α-έλικες), all-β (η δομή αποτελείται από β-πτυχωτές επιφάνειες), α/β (α-έλικες και β-πτυχωτές επιφάνειες εναλλάσσονται στην δομή της πρωτεΐνης) και α+β (α-έλικες και β-πτυχωτές επιφάνειες βρίσκονται σε διακριτές περιοχές της δομής).

Επίσης, υπάρχουν και ολοκληρωμένα συστήματα τα οποία χρησιμοποιούνται από τους χρήστες προκειμένου να ανακτήσουν τις πληροφορίες που προσφέρουν οι βάσεις δεδομένων. Τέτοια είναι τα παρακάτω:



Σχήμα 2: Ολοκληρωμένα συστήματα ανάκτησης πληροφοριών από τις Βάσεις Δεδομένων

Το **SRS** [11] είναι ένα ισχυρό, εύχρηστο σύστημα διαχείρισης δεδομένων το οποίο διατίθεται από την εταιρία LION Bioscience. Το SRS, μέσω ενός φιλικού προς το χρήστη, γραφικού περιβάλλοντος δίνει την δυνατότητα αναζήτησης και ανάκτησης δεδομένων από περισσότερες από 400 βάσεις δεδομένων, οι οποίες μπορεί να είναι αποθηκευμένες στον ίδιο κεντρικό υπολογιστή. Το μεγάλο του πλεονέκτημα είναι ότι μπορεί κάποιος να πραγματοποήσει ταυτόχρονη αναζήτηση για ένα ζήτημα άμεσου ενδιαφέροντος σε παραπάνω από μια βάσεις δεδομένων, που δεν περιέχουν ανάλογου είδους πληροφορία, και η μορφοποίηση των δεδομένων σε καθεμιά να είναι διαφορετική. Ένα άλλο μεγάλης σημασίας πλεονέκτημα είναι η ταχύτητα με την οποία εκτελούνται οι αναζητήσεις, παρά το γεγονός ότι διαχειρίζεται πραγματικά τεράστιο όγκο πληροφορίας λόγω του μεγάλου αριθμού βάσεων που μπορεί να διαχειρίζεται ταυτόχρονα. Τέλος δίνεται η δυνατότητα στον κάτοχο του συστήματος να ενσωματώνει σε αυτό βάσεις που δημιούργησε ο ίδιος ή προγράμματα για κάθε είδους υπολογιστική ανάλυση χωρίς να επηρεάζεται η απόδοση του συστήματος.

Το **Entrez** [11] αποτελεί ένα σύστημα διαχείρισης, ανάλογο του SRS, για την αναζήτηση και ανάκτηση πληροφοριών όλων των βάσεων δεδομένων που περιέχονται στο NCBI (National Center for Biotechnology Information) των ΗΠΑ. Το Entrez δίνει τη δυνατότητα αναζήτησης σε βάσεις δεδομένων νουκλεοτιδικών και πρωτεϊνικών ακολουθιών, δομές βιομορίων, γονιδιωμάτων και στη βάση για την αναζήτηση βιβλιογραφίας MEDLINE μέσω του ίδιου γραφικού περιβάλλοντος επιτρέποντας και πιο πολύπλοκες αναζητήσεις ανάμεσα στα στοιχεία τους. Βέβαια το γεγονός ότι περιορίζεται μόνο στις βάσεις δεδομένων του NCBI και ότι δεν επιτρέπει πολύπλοκες αναζητήσεις το καθιστούν υποδεέστερο έναντι του SRS.

– *<u>Ανάπτυξη εργαλείων που επιτρέπουν την αποδοτική ανάλυση των βιολογικών δεδομένων</u>*

Ενώ η συλλογή και η οργάνωση των δεδομένων είναι μια εργασία ουσιαστικής σημασίας, όλες αυτές οι καταχωρημένες πληροφορίες θα ήταν άχρηστες αν δε βρισκόταν κάποιος τρόπος να αναλυθούν. Επίσης είναι γνωστό ότι η ακολουθία καθορίζει την τρισδιάστατη δομή και αυτή με τη σειρά της τη λειτουργία (λειτουργικά χαρακτηριστικά) μιας πρωτεΐνης. Επομένως, είναι φανερό γιατί η Βιοπληροφορική επικεντρώθηκε στην ανάλυση τόσο των ακολουθιών όσο και των δομών. [21] [22] [8]

~ *Ανάλυση ακολουθιών*

Περιλαμβάνει διάφορες ενέργειες με πρωταρχική αυτή της ανάπτυξης αλγορίθμων για σύγκριση (πρωτεϊνικών και νουκλεοτιδικών ακολουθιών). Μέσω αυτών γίνεται σύγκριση και εύρεση, εάν βέβαια υπάρχει, ομοιότητας μεταξύ των μελών μεγάλων συνόλων δεδομένων. Υπάρχουν δύο τρόποι στοίχισης ακολουθιών, κατά ζεύγη και πολλαπλή. Αλγόριθμοι που χρησιμοποιούνται για κατά ζεύγη στοίχιση είναι οι Smith-Waterman (τοπική στοίχιση) και Needleman-Wunsch (ολική στοίχιση), καθώς και ευριστικοί αλγόριθμοι όπως ο Blast, ο Fasta και ο αλγόριθμος κοπής γωνιών, οι οποίοι είναι ευρέως χρησιμοποιούμενοι λόγω της μείωσης του υπολογιστικού χρόνου που προσφέρουν. Όσον αφορά στην πολλαπλή στοίχιση, δηλαδή στην ταυτόχρονη στοίχιση περισσότερων από δύο ακολουθιών, έχουμε τους αλγορίθμους Feng&Doolitle, ClustalW, Dialign, T-Coffee κ.ά. Μέσω αυτών μπορούμε να πάρουμε πολλές σημαντικές πληροφορίες για την προέλευσή τους, αφού είναι γνωστό πως η ομοιότητα δύο ακολουθιών αντανακλά κατά βάση κοινή εξελικτική προέλευση ή να καθορίσουμε οικογένειες πρωτεϊνών ή ακόμη και να αποδώσουμε λειτουργικό και δομικό ρόλο σε μια άγνωστη πρωτεΐνη. Επιπλέον, πέραν της σύγκρισης, δίνεται η δυνατότητα πρόβλεψης των λειτουργικών και δομικών χαρακτηριστικών (για παράδειγμα πρόβλεψη δευτεροταγούς ή τριτοταγούς δομής), εύρεσης προτύπων (με αυτόν τρόπο μπορούμε, όσον αφορά τις ακολουθίες DNA να προσδιορίσουμε τις περιοχές όπου βρίσκονται τα γονίδια, σημεία από όπου ξεκινά ή σταματά η αντιγραφή του DNA κ.ά.), ανάλυσης γονιδιωμάτων καθώς και πολλές άλλες.

~ *Ανάλυση δομών*

Περιλαμβάνει αναπαράσταση/οπτικοποίηση των δομών (Jmol, Pymol, Rasmol, Grasp, Cn3D κ.ά.), υπέρθεση δομών- στοίχιση (SSAP, MASS, MAMOOTH,VAST κ.ά.), αναγνώριση πρωτεϊνικού διπλώματος (UCLA-DOE, 3D-PSSM, 123D, FFAS), συγκριτική προτυποποίηση με βάση την ομολογία κ.ά..

– *<u>Ανάπτυξη εργαλείων για την ερμηνεία αποτελεσμάτων</u>*

Ένα χαρακτηριστικό παράδειγμα τέτοιων εργαλείων είναι τα φυλογενετικά δένδρα, τα οποία παρουσιάζουν με συστηματικό τρόπο τα αποτελέσματα των συγκρίσεων. Συγκεκριμένα οπτικοποιούν την εξέλιξη από ένα κοινό πρόγονο και μας δίνουν τη δυνατότητα να ανιχνεύουμε μακρινές ομοιότητες ανάμεσα σε οργανισμούς.

Επίσης, με τη χρήση κάποιων από αυτά γίνεται καθορισμός των τρισδιάστατων δομών πρωτεϊνών από τις βιολογικές ακολουθίες.

1.3 Πεδία εφαρμογής

Ο απώτερος σκοπός όλων αυτών των ερευνών και γενικότερα της ανάπτυξης της Βιοπληροφορικής είναι να εξαχθούν συμπεράσματα χρήσιμα για την κατανόηση της δομής και της λειτουργίας του έμβιου κόσμου. Με αυτόν τον τρόπο καθίσταται δυνατή η αντιμετώπιση επιδημιών ή κληρονομικών ασθενειών, η γονιδιακή θεραπεία, η ανάπτυξη ιστοκαλλιεργειών καθώς και η ανακάλυψη νέων φαρμάκων και εμβολίων. Εκτός από τη, σύμφωνα με τα παραπάνω, εμφανή συμβολή της στην Ιατρική υπάρχουν κι άλλες περιοχές για τις οποίες αποτελεί αναπόσπαστο κομμάτι όπως για παράδειγμα η Λειτουργική Γονιδιωματική και η Πρωτεομική. Συνιστά γεγονός πως υπάρχουν και πιο σύγχρονοι ακόμα τομείς έρευνας, όπως η γενετική τροποποίηση φυτών και ζώων (κλωνοποίηση υψηλότερων ζώων, κλωνοποίηση φυτών- για παράδειγμα δημιουργία ποικιλιών ανθεκτικών στην ξηρασία). Ακόμη χρησιμοποιείται σε ζητήματα σχετικά με τη βιοποικιλότητα και το περιβάλλον, σε μελέτες για την κλιματική αλλαγή και για εναλλακτικές πηγές ενέργειας καθώς και στη δημιουργία βιολογικών όπλων. Όλα αυτά λοιπόν έχουν αναδείξει τη Βιοπληροφορική, όχι μόνο ένα κλάδο υποστήριξης των επιστημών αλλά, σαν έναν τομέα που κατευθύνει τη μελλοντική πορεία της έρευνας στις βιολογικές επιστήμες και στη Βιοτεχνολογία.

2ο ΚΕΦΑΛΑΙΟ
ΕΡΓΑΛΕΙΑ ΒΙΟΠΛΗΡΟΦΟΡΙΚΗΣ

Περίληψη

Όπως ήδη επισημάνθηκε στο προηγούμενο κεφάλαιο, ένας από τους κύριους τομείς έρευνας στη Βιοπληροφορική είναι η ανάλυση των βιολογικών δεδομένων και η ερμηνεία των αποτελεσμάτων. Εκτός από τα εργαλεία που, συνοπτικά, αναφέρθηκαν παραπάνω, υπάρχει επιπροσθέτως ένα μεγάλο πλήθος εργαλείων από τα οποία κάποια εκτελούν συγκεκριμένες εργασίες - έχουν αναπτυχθεί αποκλειστικά για βιολογικές εφαρμογές - και κάποια που αν και έχουν ευρεία χρήση, χρησιμοποιούνται και για τέτοιου είδους εφαρμογές. Στο κεφάλαιο αυτό θα γίνει προσπάθεια για μια λίγο εκτενέστερη αναφορά τέτοιων εργαλείων –των πιο ευρέως διαδεδομένων - τόσο από την πλευρά του λογισμικού (software) όσο και από την πλευρά του υλικού (hardware). Επίσης, θα καταγραφούν κάποιοι σύγχρονοι τομείς έρευνας που ενδεχομένως αποτελούν προκλήσεις για όσους ασχολούνται με τη Βιοπληροφορική, οι ανάγκες που προκύπτουν και θα παρουσιαστεί το εργαλείο Biojava καθώς και ο στόχος της παρούσας πτυχιακής εργασίας.

Βασικές έννοιες

- Λογισμικό (Software)
- Γλώσσες προγραμματισμού
- Ολοκληρωμένα πακέτα
- Εργαλεία Βιοπληροφορικής
- Προσομοίωση
- Ανάλυση δεδομένων
- Πρωτεομική
- Δομική Βιολογία
- Βιολογικά δέντρα
- Φυλογενετική ανάλυση
- Υλικό (Hardware)
- Σύγχρονοι τομείς έρευνας
- Biojava
- Στόχος πτυχιακής

2.1 Λογισμικό - Software

2.1.1 Γλώσσες προγραμματισμού

Οι γλώσσες προγραμματισμού που χρησιμοποιούνται κατά κόρον σε εφαρμογές Βιοπληροφορικής διακρίνονται σε τρεις επιμέρους κατηγορίες:

– *Γλώσσες προγραμματισμού γενικού σκοπού*

Αυτού του είδους οι γλώσσες προγραμματισμού βρίσκουν εφαρμογές και στη Βιοπληροφορική επειδή παρέχουν, εκτός των άλλων, δυνατότητες για γρήγορη αναζήτηση (search) σε διάφορες βάσεις δεδομένων. Τέτοιες είναι οι παρακάτω:

~ *C [23] [24]*

Αναπτύχτηκε από τον Dennis Ritchie, στα εργαστήρια Bell Labs με σκοπό να χρησιμοποιηθεί για την ανάπτυξη του λειτουργικού συστήματος UNIX και αρχικά υλοποιήθηκε σε έναν υπολογιστή DEC PDP-11 το 1972. Έκτοτε χρησιμοποιείται ευρύτατα τόσο για ανάπτυξη λογισμικού συστήματος (system software) όσο και για απλές εφαρμογές. Χαρακτηρίζεται για την ταχύτητά της και για το γεγονός ότι είναι διαθέσιμη στα περισσότερα σημερινά λειτουργικά συστήματα, λόγοι που εξηγούν τη ραγδαία ανάπτυξή της.

~ *C++ [23] [24]*

Αναπτύχθηκε από τον Bjarne Stroustrup, στα εργαστήρια Bell, το 1979. Πρόκειται για μια βελτίωση της C καθώς περιλαμβάνει, μεταξύ άλλων, κλάσεις, εικονικές συναρτήσεις, υπερφόρτωση τελεστών, πολλαπλή κληρονομικότητα κ.ά.. Η γλώσσα C++ ακολουθεί τις έννοιες του αντικειμενοστραφούς προγραμματισμού κάτι που δεν συμβαίνει στην C. Είναι εξίσου γρήγορη όσο η C και χρησιμοποιείται κυρίως για υλοποίηση εφαρμογών που απαιτούν γρήγορους υπολογισμούς. Οι γλώσσες C και C++ δεν συνιστώνται για διαδικτυακές εφαρμογές, για εφαρμογές που υποστηρίζουν υψηλής ποιότητος γραφικά καθώς και δεν είναι διαδεδομένες στο χώρο της Βιοπληροφορικής σε περιπτώσεις όπου ο χρήστης προσπαθεί να αντλήσει πληροφορίες από πολλές πηγές δεδομένων κάτι που απαιτεί την διαχείριση αρχείων μεγάλου όγκου δεδομένων.

~ *Java [25] [26]*

Σχεδιάστηκε από την εταιρεία πληροφορικής Sun Microsystems και "πατέρας" της θεωρείται ο James Gosling. Από την ημέρα της δημοσιοποίησής της, τον Μάιο του 1995, η Java εξαπλώθηκε ιδιαίτερα στην επιχειρηματική κοινότητα λόγω του μεγάλου ενδιαφέροντος για το World Wide Web. Έχει κιόλας λύσει τα περισσότερα προβλήματα στο μοντέλο client/server και έχει προάγει την χρήση του World Wide Web. Πρόκειται για μια αντικειμενοστρεφή γλώσσα, που είναι απλή, συμβατή με δικτυακά πρωτόκολλα, ουδέτερη της υποκείμενης αρχιτεκτονικής, φορητή, ασφαλής, υψηλής απόδοσης, δυναμική, σταθερή, interpreted, multithreaded και ανεξάρτητη από λειτουργικό σύστημα και πλατφόρμα (πράγμα το οποίο είναι κι ένα από τα βασικότερα πλεονεκτήματά της έναντι των άλλων γλωσσών προγραμματισμού).

– *Scripting Languages (γλώσσες συγγραφής σεναρίων)*

Είναι οι πιο ευρέως χρησιμοποιούμενες γλώσσες στις βιολογικές επιστήμες. Ένας από τους λόγους που τις κάνουν τόσο χρήσιμες είναι η δυνατότητα που έχουν να

δουλεύουν με κανονικές εκφράσεις (regular expressions) - μια πολύ συνηθισμένη εργασία, για παράδειγμα, για αυτούς που ασχολούνται με τη Βιοπληροφορική είναι η εύρεση προτύπων (patterns) για τα οποία χρησιμοποιούνται κανονικές εκφράσεις. Τόσο οι κανονικές εκφράσεις, το εύκολο συντακτικό τους όσο και οι μέθοδοι που παρέχουν διευκολύνουν πολύ την ανάλυση, την επεξεργασία, την εύρεση προτύπων καθώς και την τροποποίηση και τον μετασχηματισμό αρχείων με μεγάλο όγκο πληροφορίας. Ειδικότερα σε περιπτώσεις όπου συχνά οι χρήστες επιθυμούν να διαχειριστούν μεγάλες βάσεις δεδομένων που βρίσκονται ελεύθερα στο διαδίκτυο σε μορφή κειμένου οι γλώσσες συγγραφής σεναρίων αποτελούν ένα πολύτιμο εργαλείο για αυτούς..

~ *Perl [27] [28]*

Το όνομα Perl προέρχεται από τα αρχικά “Practical Extraction and Report Language” (Γλώσσα πρακτικής εξαγωγής και αναφοράς). Πρόκειται για μια γλώσσα προγραμματισμού η οποία δεν είναι κατεξοχήν αντικειμενοστρεφής αλλά παραταύτα τα τελευταία χρόνια γίνεται μία σημαντική προσπάθεια να υποστηρίξει αντικειμενοστρεφείς μεθόδους. Είναι εξαιρετικά εύκολη στο συντακτικό και είναι η πλέον διαδεδομένη στο χώρο της Βιοπληροφορικής εφόσον υπάρχουν πολλά κομμάτια ανοιχτού κώδικα ελεύθερα στο διαδίκτυο για αναλύσεις που αφορούν πολλούς τομείς της Βιοπληροφορικής. Σχεδιάστηκε από τον Larry Wall, πρωτοεμφανίστηκε το 1987 και η πιο πρόσφατη έκδοσή της είναι η 5.10.1.

~ *Python [29]*

Το όνομα Python προέρχεται από μία ομάδα Άγγλων κωμικών με το όνομα Monty Python. Διακρίνεται για την αναγνωσιμότητα του κώδικά της, την ευκολία χρήσης της, τις πολλές βιβλιοθήκες που διαθέτει και την ταχύτητα εκμάθησης της. Η γλώσσα Python γίνεται ολοένα και πιο διαδομένη με την πάροδο του χρόνου εφόσον είναι επίσης και αρκετά γρήγορη. Δημιουργήθηκε από τον Ολλανδό Guido van Rossum το 1990 και η τρέχουσα έκδοσή της είναι η 3.1.

~ *Ruby [30]*

Είναι μια δυναμική γλώσσα προγραμματισμού που συνδυάζει σύνταξη εμπνευσμένη από τις Perl και Smalltalk. Επινοήθηκε το 1993 από τον Yukihiro Matsumoto, ο οποίος, σύμφωνα με δήλωση του, δημιούργησε τη Ruby επειδή επιθυμούσε μια γλώσσα πιο ισχυρή από την Perl και πιο αντικειμενοστρεφή από την Python. Επιτρέπει πλέον την ανάπτυξη διαδικτυακών εφαρμογών και οδεύει στο να γίνει μία αυτόνομη λύση τόσο για την ανάπτυξη κώδικα που αφορά επεξεργασία πληροφορίας όσο και για τη διαδικτυακή παρουσίαση αυτής.

– *Γλώσσες προγραμματισμού ειδικού σκοπού*

Πρόκειται ουσιαστικά για κάποια projects, κάποιες επεκτάσεις ευρέως διαδεδομένων γλωσσών προγραμματισμού, που έχουν εστιάσει στη Βιοπληροφορική

και έχουν αναπτύξει συγκεκριμένα εργαλεία για τη διευκόλυνση των εργασιών στον τομέα αυτόν.

~ *Bioperl [31]*

Πρόκειται αναμφισβήτητα για το πιο δημοφιλές, μέχρι στιγμής, έργο (project). Ο κώδικας του Bioperl είναι μια εκτεταμένη βιβλιοθήκη των βασικών ενοτήτων γραμμένες σε Perl για την υποστήριξη της διαχείρισης και επεξεργασίας των βιολογικών πληροφοριών - μεταξύ άλλων, υπάρχουν ενότητες που σχετίζονται με φυλογενετική ανάλυση, πολλαπλές στοιχίσεις ακολουθιών και την ανάλυση μικροσυστοιχιών (microarrays). Είναι κυρίως κατάλληλο για την επεξεργασία δεδομένων ακολουθιών και διασυνδέεται σε βάσεις δεδομένων, υποστηρίζοντας, επίσης, οπτικοποίηση αλληλουχιών και ερωτήματα για εξωτερικούς σχολιασμούς. Είναι εξαιρετικά φορητό, ανοικτό και ελεύθερο με καθιερωμένες διεπαφές με άλλες γλώσσες προγραμματισμού. Προσπαθεί να μιμηθεί το αντικειμενοστρεφές πρότυπο προγραμματισμού με τη χρήση των ενοτήτων Perl και με τήρηση τριών αρχών σχεδιασμού. Η πρώτη αρχή είναι να διαχωρίσει το περιβάλλον από την εφαρμογή. Η δεύτερη αρχή είναι να παράσχει ένα πλαίσιο αναφοράς (base framework) για τη σχετική λειτουργία γενικεύοντας κοινές ρουτίνες σε μια ενιαία ενότητα. Η τρίτη και τελευταία αρχή είναι να χρησιμοποιήσει τα Factory και Strategy πρότυπα (patterns) , όπως ορίζεται από τον Erich Gamma.

~ *Biopython [32]*

Το Biopython ιδρύθηκε το 1999 και διαμορφώθηκε μετά το επιτυχές Bioperl έργο. Είναι μια διεθνής ένωση για την ανάπτυξη ελεύθερα διαθέσιμων εργαλείων Python και βιβλιοθηκών για την υπολογιστική μοριακή βιολογία. Η λειτουργικότητα του Biopython περιλαμβάνει διεπαφή με το πρόγραμμα/ αλγόριθμο στοίχισης ClustalW και κώδικα για την ταξινόμηση των δεδομένων. Το γραφικό περιβάλλον του χρήστη (GUI - Graphical User Interface) περιορίζεται σε βασικούς χειρισμούς των ακολουθιών, μεταφράσεις, εκτελέσεις του αλγορίθμου BLAST κ.τ.λ.

~ *Bioruby [33]*

Το Bioruby παρέχει ένα ολοκληρωμένο περιβάλλον για τη Βιοπληροφορική βασισμένο στη γλώσσα Ruby (π.χ. ανάλυση ακολουθιών, BLAST, HMMER), η οποία έχει εγγενή υποστήριξη για αντικειμενοστρεφή προγραμματισμό με απλή αλλά ισχυρή σύνταξη. Ολόκληρο το πακέτο Bioruby είναι γραμμένο με τη χρήση απλά και μόνο της γλώσσας Ruby, οπότε δεν υπάρχει θέμα εξάρτησης από το λειτουργικό σύστημα. Υποστηρίζεται από το Πανεπιστήμιο του Τόκιο (Human Genome Center), το Πανεπιστήμιο του Κιότο (Βιοπληροφορική Center) και το Open Bioinformatics Foundation (OBF).

~ *Biobike - Biolisp [34]*

Πρόκειται, ουσιαστικά, για μια ακόμη γλώσσα προγραμματισμού ειδικά για τη Βιολογία. Το BioBike έχει ενσωματωμένα όλα τα ευρέως χρησιμοποιούμενα εργαλεία στον τομέα της Βιοπληροφορικής (Blast, Clustal, κ.λπ.). Η κύρια γλώσσα του BioBike ονομάζεται BioLisp - μια διάλεκτος της Lisp, στην οποία έχουν προστεθεί οι κοινές βιολογικές λειτουργίες. Είναι το περιβάλλον στο οποίο μπορεί κανείς να γράψει και να εκτελέσει κώδικα γραμμένο σε BioLisp. Με το Biolisp, μεταξύ άλλων, μπορεί κανείς να εκτελέσει απλή βιολογική επεξεργασία φυσικής γλώσσας για το PubMed, να εργαστεί πάνω στις ακολουθίες, να αναπαραστήσει και να εξερευνήσει γραφήματα, να παράγει δέντρα και να εκτελέσει ταξινόμηση δεδομένων μικροσυστοιχιών. Ωστόσο, δεν παρέχεται οπτικοποίηση.

~ *The R project for Statistical Computing [35]*

Είναι ένα περιβάλλον λογισμικού ανοιχτού κώδικα για Υπολογιστική Στατιστική και γραφικά, το οποίο «τρέχει» σε μια μεγάλη ποικιλία από πλατφόρμες UNIX, Windows και MacOS. Υπάρχουν διεπαφές R για τις σημαντικότερες γλώσσες προγραμματισμού, όπως MATLAB, Perl, Python, Java, C, C + + και Fortran. Το πακέτο R εξασφαλίζει υλοποιήσεις για ένα ευρύ φάσμα στατιστικών και γραφικών τεχνικών, συμπεριλαμβανομένης της μοντελοποίησης και της ανάλυσης διασποράς. Είναι δημοφιλές στον κόσμο της Βιοπληροφορικής λόγω του ότι διατίθενται, σε αντίθεση με άλλα (π.χ. Matlab), δωρεάν. Συνιστάται ιδιαίτερα για τη βιολογική ανάλυση, δεδομένου ότι πολλά σχετικά έγγραφα είναι διαθέσιμα στο διαδίκτυο (online).

~ *Bioconductor [36]*

Βασίζεται στο R project. Το BioConductor είναι ένα, ανοιχτού κώδικα και ελεύθερης ανάπτυξης, λογισμικό που στοχεύει στην παροχή πρόσβασης σε μια σειρά στατιστικών και γραφικών μεθόδων και εργαλείων για την ανάλυση και την κατανόηση δεδομένων της Γονιδιωματικής. Για παράδειγμα, υπάρχουν διαθέσιμα πακέτα για ανάλυση και στατιστικές ή γραφικές μέθοδοι για: προεπεξεργασία δεδομένων από cDNA πίνακες, εντοπισμό γονιδίων που εκφράζονται διαφορετικά, θεωρητικές αναλύσεις γραφημάτων, απεικόνιση στοιχείων γονιδιωματικής. Είναι κατάλληλο για την στατιστική ανάλυση των πειραμάτων σε μικροσυστοιχίες, την προεπεξεργασία συστοιχιών και τον έλεγχο ποιότητας, την κανονικοποίηση μιας συστοιχίας και μεταξύ δύο συστοιχιών, τη σύνδεση της βοηθητικής μεταβλητής και των δεδομένων σχεδιασμού με τα στοιχεία έκφρασης, καθώς και για εξαγωγή συμπερασμάτων πάνω σε βιολογικά και κλινικά ζητήματα.

2.1.2 Ολοκληρωμένα πακέτα ειδικά για Βιοπληροφορική

– *Εργαλεία προσομοίωσης*

~ *Neuron [37]*: Μοντελοποιεί ανεξάρτητους νευρώνες και νευρωνικά δίκτυα.

- *Ανάλυση ακολουθιών- αλληλεπιδράσεις πρωτεϊνών*
  - *ClustalW [38]*: Είναι ένα γενικού σκοπού πρόγραμμα πολλαπλών στοιχίσεων για ακολουθίες DNA ή πρωτεϊνών. Υπολογίζει το καλύτερο match (ταίριασμα) για τις επιλεγμένες ακολουθίες και τις κατατάσσει/στοιχίζει με τέτοιο τρόπο έτσι ώστε να μπορούν να διακριθούν εύκολα η ταυτότητα, οι ομοιότητες και οι διαφορές τους.
  - *ClustalX [38]*: Ανανεωμένη έκδοση του ClustalW που προσφέρει στο πρόγραμμα γραφικό περιβάλλον
  - *T-Coffee [39]*: Είναι ένα πακέτο για πολλαπλή στοίχιση ακολουθιών. Μπορεί είτε να χρησιμοποιηθεί απλά για να γίνει η στοίχιση των ακολουθιών είτε για να συνδυάσουμε τις μεθόδους στοίχισης που προτιμούμε (Clustal, Mafft, Probcons, Muscle κ.ά.) σε μια μόνο στοίχιση (M-Coffee).
  - *MUSCLE [40]*: Προέρχεται από τα αρχικά Multiple Sequence Comparison by Log-Expectation και είναι εργαλείο για πολλαπλή στοίχιση ακολουθιών.

- *Ανάλυση δεδομένων μικροσυστοιχιών*
  - *TM4 [41]*: Σουίτα λογισμικού που επιτρέπει στους χρήστες να συλλαμβάνουν, να διαχειρίζονται και να αναλύουν τα στοιχεία από πειράματα μικροσυστοιχιών DNA.

- *Ανάλυση δεδομένων*
  - *GEMS [42]*: Εργαλείο για την αυτοματοποιημένη ανακάλυψη μοριακής υπογραφής και την κατασκευή μοντέλων πρόβλεψης.
  - *Causal Explorer [43]*: Βιβλιοθήκη αλγορίθμων για εκμάθηση Bayesian δικτύων και causal μοντέλα.

- *Ανάλυση εικόνων και βίντεο*
  - *AxioVision and ZEN 2009*: Λογισμικό για ψηφιακή επεξεργασία εικόνας στα μικροσκόπια.
  - *Volocity [44]*: Λογισμικό υψηλών προδιαγραφών για οπτικοποίηση σε 3 και 4 διαστάσεις.
  - *Huygens Deconvolution Software [45]:* Προγράμματα υψηλής ποιότητας για αποκατάσταση και βελτίωση ανάλυσης εικόνων από μικροσκόπια.

- *Δομική Βιολογία*
  - *AutoDock [46]*: Σχεδιασμένο για να προβλέψουμε πώς μικρά μόρια, όπως πιθανών των φαρμάκων, δεσμεύονται σε έναν υποδοχέα με γνωστή τρισδιάστατη δομή.
  - *Insight [47]*: Μοντελοποίηση μακρομορίων.
  - *NAMD2 [48]*: Λογισμικό για υψηλών προδιαγραφών προσομοίωση μεγάλων βιομορίων.
  - *XPLOR [49] [50]*: Λογισμικό για δυναμική μοντελοποίηση μορίων.

~ *GROMACS [51]*: Λογισμικό για δυναμική μοντελοποίηση μορίων.

– *Πρωτεομική*
  ~ *ProteoIQ [52]*: Λογισμικό για συγκριτική και ποσοτική Πρωτεομική.
  ~ *ProSightPC 2.0 [53]*: Λογισμικό για την από πάνω προς τα κάτω ανάλυση δεδομένων Πρωτεομικής.
  ~ *Mascot Integra [54]*: Επιτρέπει διαχείριση δεδομένων για την Πρωτεομική.
  ~ *MS-LIMS [55]*: Ελαφριά, φορητή, λύση για διαχείριση δεδομένων Πρωτεομικής που βασίζονται σε φασματομετρία μάζας
  ~ *Trans proteomics Pipeline [56]*: Συλλογή εργαλείων για την Πρωτεομική
  ~ *OpenMS [57]*: Πρόκειται για open-source λογισμικό – βιβλιοθήκη της C++ για την διαχείριση και ανάλυση δεδομένων από LC/MS.

– *Ανάλυση βιολογικών δικτύων*
  ~ *Medusa [58]*: Είναι μια εφαρμογή Java και είναι διαθέσιμη σαν applet. Βασίζεται στον αλγόριθμο Fruchterman-Reingold και παρέχει δισδιάστατες απεικονίσεις δικτύων μέτριου μεγέθους μέχρι μερικές χιλιάδες κόμβους και ακμές.
  ~ *Cytoscape [59]*: Μοντελοποιεί τη μοριακή αλληλεπίδραση μεταξύ των δικτύων παρέχοντας δισδιάστατες απεικονίσεις και είναι κατάλληλη για υψηλής κλίμακας ανάλυση δικτύων με εκατοντάδες χιλιάδες κόμβους και ακμές. Ολοκληρώνει τις αλληλεπιδράσεις αυτές με τα gene expression profiles και με άλλα στατικά δεδομένα.
  ~ *BioLayout Express [60]*: Είναι ένα εργαλείο για την οπτικοποίηση και ομαδοποίηση δικτύων μεγάλης κλίμακας και υποστηρίζει τόσο σταθμισμένα όσο και μη σταθμισμένα γραφήματα.
  ~ *Osprey [61]*: Εφαρμογή που επιτρέπει δισδιάστατες αναπαραστάσεις κατευθυνόμενων, μη κατευθυνόμενων και σταθμισμένων δικτύων.
  ~ *ProViz [62]*: Εφαρμογή για δισδιάστατες και ψευδο-τρισδιάστατες απεικονίσεις. Είναι δυνατό στον τομέα των πρωτεϊνών – στις αλληλεπιδράσεις πρωτεϊνικών δικτύων.
  ~ *Ondex [63]*: Επιτρέπει δισδιάστατες αναπαραστάσεις κατευθυνόμενων, μη κατευθυνόμενων και σταθμισμένων δικτύων. Μπορεί να χειριστεί μεγάλης κλίμακας δίκτυα με εκατοντάδες χιλιάδες κόμβους. Το δυνατό του σημείο είναι η ικανότητά του να συνδυάζει ετερογενείς τύπους δεδομένων σε ένα δίκτυο.
  ~ *PATIKA [64]*: Προέρχεται από τα αρχικά **P**athway **A**nalysis **T**ools for **I**ntegration and **K**nowledge **A**cquisition. Είναι web-based εφαρμογή και παρέχει δισδιάστατες απεικονίσεις μονών ή κατευθυνόμενων γραφημάτων, χωρίς να υπάρχουν περιορισμοί στο μέγεθός τους. Είναι ένα ολοκληρωμένο περιβάλλον λογισμικού με στόχο να προσφέρει στους ερευνητές μια ολοκληρωμένη λύση για την μοντελοποίηση και ανάλυση των κυτταρικών

διαδικασιών. Είναι ένα από τα λίγα εργαλεία που δίνει τη δυνατότητα αποτελεσματικής απεικόνισης των μεταβάσεων.

~ *PIVOT [65]*: Σχεδιάζει τα πάντα σε δύο διαστάσεις και χρησιμοποιεί απλές μη κατευθυνόμενες γραμμές για να δείξει τις σχέσεις μεταξύ βιοοντοτήτων. Δεν έχει περιορισμό στο μέγεθος των δεδομένων που μπορεί να παρουσιάσει. Είναι ιδιαίτερα κατάλληλο για την οπτικοποίηση των αλληλεπιδράσεων μεταξύ πρωτεϊνών και τον προσδιορισμό των σχέσεων μεταξύ τους, για παράδειγμα η περίπτωση των ομολόγων.

~ *Pajek [66]*: Προσφέρει δισδιάστατες και ψευδο-τρισδιάστατες αναπαραστάσεις και υποστηρίζει μονά, κατευθυνόμενα και σταθμισμένα γραφήματα. Είναι κατάλληλο για μεγάλης κλίμακας δίκτυα με χιλιάδες, ακόμη και εκατομμύρια κόμβους και ακμές.

– <u>*Οπτικοποίηση βιολογικών δέντρων και φυλογενετική ανάλυση*</u>

~ *Dendroscope [67] [68]*: Είναι μια πλατφόρμα java η οποία είναι σε θέση να απεικονίσει φυλογενετικά δέντρα σε δύο διαστάσεις πολύ αποτελεσματικά. Συνιστάται για πολύ μεγάλα σύνολα δεδομένων με εκατοντάδες χιλιάδες ταξινομικές κατηγορίες.

~ *iTOL [69] [68]*: Είναι μια web εφαρμογή για την απεικόνιση και το χειρισμό φυλογενετικών δέντρων σε δύο διαστάσεις. Χρησιμοποιεί Shockwave Flash και Javascript για να εμφανίσει δέντρα και διαβάζει μορφές αρχείων (file formats) Newick και Nexus.

~ *HyperTree [70] [68]*: Είναι μια ελεύθερα διαθέσιμη java εφαρμογή που αναπτύχθηκε κυρίως για την προβολή, την επεξεργασία και το χειρισμό μεγάλων συνόλων δεδομένων σε δύο διαστάσεις. Διαβάζει και γράφει τα δέντρα σε μορφή αρχείου PHYLIP.

~ *NJPlot [71] [68]*: Είναι μια αυτόνομη εφαρμογή ανοιχτού κώδικα που αναπτύχθηκε κυρίως για την οπτικοποίηση δέντρων σε δύο διαστάσεις. Διαβάζει τη μορφή αρχείων Newick ως είσοδο (input) και επιτρέπει μεγέθυνση, αντιμετάθεση κλαδιών (branch swapping), εμφάνιση αποτελεσμάτων bootstrap και υποστηρίζει τη σχεδίαση πολλαπλών διακλαδισμένων δέντρων. Εμφανίζει περιορισμένη λειτουργικότητα σε σύγκριση με άλλα εργαλεία και δεν συνιστάται για μεγάλα σύνολα δεδομένων.

~ *HyperGeny [72] [68]*: Προέρχεται από τις λέξεις HYPERbolic phyloGENY. Είνια μια java εφαρμογή ανεξάρτητη πλατφόρμας που αναπτύχθηκε κυρίως για την υπερβολική (hyperbolic) οπτικοποίηση μεγάλων δενδρικών δομών. Λαμβάνει ως είσοδο και παρέχει ως έξοδο δέντρα και υπόδεντρα μόνο σε μορφή Newick.

~ *CTree [73] [68]*: Είναι μια java εφαρμογή που αναπτύχθηκε κυρίως για την προβολή, την ανάλυση και την επεξεργασία φυλογενετικών δέντρων σε δύο διαστάσεις. Μπορεί να διαβάσει και να γράψει δέντρα σε μορφές Newick και

.pdf. Δεν είναι πολύ διαδραστικό, αλλά είναι σε θέση να χειριστεί μεγάλα σύνολα δεδομένων.

~ *Άλλα* παρόμοια εργαλεία που χρησιμοποιούνται γενικά στον τομέα της φυλογενετικής ανάλυσης είναι τα: Phylowidget [74], BAOBAB [75], MEGA (Molecular Evolutionary Genetics Analysis) [76], PHYLIP [77], HCE (Hierarchical Clustering Explorer) [78] κ.ά..

2.2 Υλικό- Hardware

Πέρα από τα διάφορα εργαλεία λογισμικού (εφαρμογές, γλώσσες προγραμματισμού κ.λπ.) που αναφέρθηκαν παραπάνω υπάρχουν και κάποιες προσπάθειες ανάπτυξης υλικού προσαρμοσμένο στις ανάγκες της Βιοπληροφορικής. Οι προσπάθειες αυτές, οι περισσότερες στο παρελθόν (παράλληλος προγραμματισμός, χρήση FPGAs), τείνουν σήμερα να εκλείψουν. Η ανάπτυξη της υπολογιστικής ισχύος είναι ολοένα και μεγαλύτερη ώστε οι ανάγκες που προκύπτουν καλύπτονται από τους διαθέσιμους υπολογιστές, χωρίς να είναι αναγκαία η δημιουργία άλλων «δυνατότερων». Ωστόσο, ένα παράδειγμα hardware που είναι σχεδιασμένο για να καλύπτει τις ανάγκες τις Βιοπληροφορικής και αξίζει να σημειωθεί είναι το παρακάτω:

CLC Bioinformatics Cell *[2]*

Με το CLC bioinformatics Cell, η βιοπληροφορική εργασία μπορεί να επιταχυνθεί έως και 100 φορές, δίνοντας μεγαλύτερη βιολογική διορατικότητα και υψηλότερης ποιότητας αποτελέσματα σε λιγότερο χρόνο.

Με το CLC bioinformatics Cell, αντιλαμβανόμαστε αμέσως σημαντικές επιταχύνσεις των τριών σημαντικών και ευρέως χρησιμοποιούμενων, στο χώρο της Βιοπληροφορικής, αλγορίθμων. Έχει τη δυνατότητα να δώσει μια γρήγορη εκτέλεση του ClustalW και τις ταχύτερες υλοποιήσεις που έχουν σημειωθεί ποτέ σε πρότυπο υλικό (standard hardware) για τα Smith-Waterman (130x) και HMMER (37x).

Εικόνα 4: CLC Bioinformatics Cell
(Πηγή: [2])

Ολοκληρωμένο

Το Cell είναι πλήρως συμβατό και εξοπλισμένο με τις πιο περιεκτικές/ολοκληρωμένες εφαρμογές λογισμικού που κυκλοφορούν στην αγορά: το CLC DNA Workbench, CLC RNA Workbench, το CLC Protein Workbench και το CLC Main Workbench. Επομένως, το Cell επωφελείται από τα χαρακτηριστικά όλων αυτών των βιολογικών Workbenches όπως είναι η γραφική διεπαφή χρήστη, η επιλογή πολλαπλών αναλύσεων σε δέσμες εκτελέσεων (batch run), οι δυνατότητες αποθήκευσης δεδομένων, η χρήση γραφημάτων και πινάκων επισκόπησης/σύνοψης των αποτελεσμάτων κ.λπ..

Πέραν της φιλικής γραφικής διεπαφής χρήστη υπάρχει και η δυνατότητα εκτέλεσης των εργασιών μέσω της γραμμής εντολών. Αυτό σημαίνει ότι δίνεται η επιλογή της ενσωμάτωσης υπολογισμών υψηλής απόδοσης σε υφιστάμενα σενάρια ροών εργασίας Βιοπληροφορικής και σε πρότυπα προγράμματα που έχουν αναπτυχθεί ενδο-εταιρικά.

Επίσης δίνεται και μια τρίτη επιλογή – η χρησιμοποίηση του Cell μέσω του ολοκληρωμένου web portal tool, μέσω του οποίου παρέχεται πρόσβαση στη χρήση του Cell από όλο τον κόσμο.

Ευέλικτο

Αυτή η καινοτόμος προσέγγιση για την επιτάχυνση του υλικού στο χώρο της Βιοπληροφορικής προσφέρει ευελιξία σε διάφορους τομείς:

- *Φυσική τοποθεσία*: Δεν υπάρχει ανάγκη για ένα «κρύο» δωμάτιο στο οποίο θα είναι τοποθετημένος ο κεντρικός υπολογιστής (server) ή για νέο ακριβό υλικό (hardware) για την εκτέλεση υψηλών επιδόσεων υπολογισμών. Το μόνο που χρειάζεται είναι ένας προσωπικός υπολογιστής.
- *Λειτουργικό σύστημα*: Εφ 'όσον ο υπολογιστής διαθέτει επεξεργαστή Intel ή AMD, το Cell λειτουργεί στα λειτουργικά συστήματα Windows, Linux και Mac OS X.
- *Χρησιμοποίηση λογισμικού*: Το Cell μπορεί να χρησιμοποιηθεί ως αναπόσπαστο μέρος όλων των βιολογικών workbenches, είτε μέσω μιας διεπαφής γραμμής εντολών.
- *Προσαρμογή*: Με το CLC Developer Kit, οι χρήστες μπορούν να δημιουργήσουν τις δικές τους ροές εργασίας συμπεριλαμβανομένων των λειτουργικών δυνατοτήτων του Cell.

Πλήρης αξιοποίηση της ισχύς του υπολογιστή του χρήστη

Εάν ο χρήστης έχει πολλούς υπολογιστές που δουλεύουν παράλληλα, ή αν έχει έναν υπολογιστή με πολλούς επεξεργαστές ή πυρήνες, το Cell παρέχει μια λύση που επιταχύνει τους υπολογισμούς, ανάλογα με τον αριθμό των Cells στα οποία αντιστοιχεί.

Αναζητήσεις HMMER

Το HMMER είναι μια υλοποίηση του προφίλ των HMM μοντέλων για την ανάλυση ακολουθιών πρωτεϊνών.

Πολλοί ερευνητές χρησιμοποιούν το HMMER για εντατικές και χρονοβόρες εργασίες και έχουν αναπτύξει ως εκ τούτου μια έκδοση του HMMER βασισμένη στο SIMD που επιταχύνει τα Hmmpfam και Hmmsearch μέχρι και 30 φορές σε ένα σύγχρονο υπολογιστή, με αποτέλεσμα σημαντικές μειώσεις των χρόνων αναζήτησης.

Αναζητήσεις Smith Waterman

Η εύρεση ομόλογων DNA, RNA ή πρωτεϊνικών αλληλουχιών σε μια βάση δεδομένων μπορεί να γίνει με πολλούς τρόπους. Η αναζήτηση που είναι βασισμένη στον αλγόριθμο Smith-Waterman είναι η μόνη μέθοδος που προσδιορίζει όλα τις πραγματικές επιτυχίες (hits), αλλά ο αλγόριθμος είναι πολύ αργός όταν εργαζόμαστε σε μεγάλα σύνολα δεδομένων και οι περισσότεροι επιστήμονες επομένως χρησιμοποιούν τον, πολύ πιο γρήγορο, αλγόριθμο αναζήτησης BLAST.

Ωστόσο, η ταχύτητα του BLAST αποβαίνει εις βάρος της ποιότητας. Η ευαισθησία του BLAST είναι στην πραγματικότητα τόσο χαμηλή, ώστε υπάρχει σημαντικός κίνδυνος εξαφάνισης σημαντικών ενδιαφερουσών ακολουθιών. Σε σύγκριση με τις αναζητήσεις που βασίζονται στον αλγόριθμο Smith Waterman, έως και 50% των search hits δε γίνονται με τη χρήση του BLAST.

Με το Cell, είναι δυνατή η επιτάχυνση της αναζήτησης Smith-Waterman που προηγουμένως χρειαζόταν μισή μέρα, σε περίπου 6 λεπτά και αποκλείει την περίπτωση της χρήσης BLAST - τουλάχιστον στις περιπτώσεις που στην αναζήτηση δεδομένων δεν χρειάζονται μόνο μερικές από τις απαντήσεις αλλά όλες οι απαντήσεις.

Στοιχίσεις ClustalW

Ο ClustalW είναι μια από τις πιο συχνά χρησιμοποιούμενες μεθόδους για την εκτέλεση πολλαπλών στοιχίσεων ακολουθιών. Ωστόσο, κατά τη στοίχιση πολλών ακολουθιών και κατά τη στοίχιση μεγάλων ακολουθιών η ταχύτητα του αλγορίθμου δεν είναι και τόσο εντυπωσιακή.

Αφού όμως ο ClustalW είναι τόσο δημοφιλής, στο Cell έχει εφαρμοστεί μια SIMD εκδοχή του, με αποτέλεσμα την επιτάχυνση του έως και 10 φορές στους περισσότερους υπολογιστές.

2.3 Σύγχρονοι τομείς έρευνας

Πέραν από τους συγκεκριμένους τομείς έρευνας που υπάρχουν για όσους ασχολούνται με τη Βιοπληροφορική (κεφάλαιο 1.2.3), συνεχώς, καθώς η τεχνολογία εξελίσσεται και η επιστήμη κάνει μεγάλα άλματα, προκύπτουν ολοένα και περισσότερες ανάγκες. Έτσι, ενώ στα πρώτα βήματα της Βιοπληροφορικής οι ανάγκες επικεντρώνονταν στη δημιουργία μεγάλων βάσεων δεδομένων ή

ολοκληρωμένων εργαλείων που θα ανέλυαν σε μικρό χρόνο τα βιολογικά δεδομένα, στις μέρες οι ανάγκες στον τομέα αυτό είναι πιο συγκεκριμένες απ' ότι παλαιότερα και μάλιστα έχουν επιτακτικό χαρακτήρα. Κάποιες από αυτές, ενδεικτικά, παρουσιάζονται παρακάτω:

- Υλοποίηση και σχεδιασμός υπολογιστικών εργαλείων για αυτόματη ανάκτηση γνώσης από Βάσεις Βιολογικών Δεδομένων
- Ανάλυση Ακολουθιών Βιολογικών Δεδομένων (ακριβής επανάληψη μοτίβων, προσεγγιστική επανάληψη μοτίβων)
- Κατηγοριοποίηση Βιολογικών Δεδομένων (βάσει ακολουθίας, βάσει συμπεριφοράς, βάσει στερεοδιαμόρφωσης)
- Μοριακή Μοντελοποίηση (στερεοδιαμόρφωση σε κατάλληλη ενεργειακή κατάσταση)
- Ανάλυση Πρωτεϊνών (εντοπισμός ενεργού κέντρου προσάραξης, συμπληρωματικότητα ως προς μορφή και ενεργειακή κατάσταση)
- Σχεδιασμός Φαρμάκων με χρήση Η/Υ

2.4 Ένα νέο εργαλείο - Biojava

Το BioJava είναι ένα project ανοικτού κώδικα το οποίο καθιστά ικανή τη γρήγορη ανάπτυξη εφαρμογών Βιοπληροφορικής στη γλώσσα προγραμματισμού Java. Παρέχει, δηλαδή, στον προγραμματιστή, μια πλατφόρμα Java εμπλουτισμένη με δυνατές ρουτίνες ανάλυσης και στατιστικές, εργαλεία για τη συντακτική ανάλυση κοινών μορφών αρχείων και πακέτα για το χειρισμό (δεδομένων) ακολουθιών και τρισδιάστατων δομών.

Περιλαμβάνει έναν αριθμό από ώριμα APIs, από τα οποία τα 10 πιο γνωστά είναι τα εξής:

1) Αλφάβητα αμινοξέων και νουκλεοτιδίων
2) Συντακτικός αναλυτής BLAST
3) Είσοδοι- έξοδοι ακολουθιών
4) Δυναμικός προγραμματισμός
5) Είσοδοι- έξοδοι και χειρισμός δεδομένων δομών
6) Χειρισμός δεδομένων ακολουθίας
7) Γενετικοί αλγόριθμοι
8) Στατιστικές κατανομές
9) Γραφικές διεπαφές χρήστη
10) Σειριοποίηση σε βάσεις δεδομένων

Στόχος

Το Biojava, εκμεταλλευόμενο τα πλεονεκτήματα της Java, έχει σκοπό να απλοποιήσει την επαναχρησιμοποίηση κώδικα και να παρέχει υλοποιήσεις που είναι εύκολο να συνδεθούν σε εξωτερικά σενάρια και εφαρμογές. Παρέχεται, λοιπόν, ένα ευέλικτο εργαλείο που θα επιτρέψει την ταχεία ανάπτυξη εφαρμογών Βιοπληροφορικής.

2.5 Στόχος πτυχιακής εργασίας

Η παρούσα πτυχιακή εργασία αφορά στην ανάπτυξη ενός πλήρους οδηγού προγραμματισμού σε BioJava, το οποίο, όπως αναφέραμε, δημιουργήθηκε για την αποτελεσματική επεξεργασία βιολογικών δεδομένων.

Θα μελετηθεί το υπερσύνολο εντολών και χειριστηρίων που προσφέρει το BioJava, και θα πραγματοποιηθεί σύγκριση με τις τυπικές εντολές και τα διαθέσιμα χειριστήρια της Java. Θα γίνει προσπάθεια να αναδειχτούν τα δυνατά σημεία του BioJava και να αναπτυχθούν προγράμματα - παραδείγματα. Ο οδηγός (ηλεκτρονικός και έντυπος) που θα αναπτυχθεί, θα υποδεικνύει τον σωστό τρόπο με τον οποίο πρέπει να υλοποιούνται συχνά χρησιμοποιούμενες διεργασίες σε αλγορίθμους Βιοϊατρικής.

Συνοψίζοντας, στόχος της παρούσας πτυχιακής είναι η εξοικείωση με ένα εξειδικευμένο περιβάλλον ανάπτυξης εφαρμογών και η ανάπτυξη υποδειγμάτων κώδικα που απαιτούνται για εξειδικευμένες εφαρμογές που άπτονται του αντικειμένου του Τμήματος Πληροφορικής με Εφαρμογές στη Βιοϊατρική. Επιπρόσθετος σκοπός είναι η δημιουργία εκπαιδευτικού υλικού που θα παρέχει βασικές γνώσεις και προτεινόμενες ασκήσεις στους αναγνώστες του οδηγού. Η ολοκλήρωσή της θα αποφέρει τον πρώτο οδηγό αυτού του είδους και θα αποτελέσει βάση για τη χρήση του συγκεκριμένου project για την ανάπτυξη εφαρμογών Βιοπληροφορικής.

3ο ΚΕΦΑΛΑΙΟ
ΕΙΣΑΓΩΓΗ ΣΤΟ BIOJAVA

Περίληψη

Μετά από την περιγραφή όλων αυτών των γλωσσών και των εργαλείων ένα εύλογο ερώτημα που μας δημιουργείται αμέσως είναι το εάν και σε ποιο βαθμό ικανοποιούν τις ανάγκες που προκύπτουν. Εν μέρει ικανοποιούνται αλλά καθώς οι ανάγκες γίνονται ολοένα και πιο μεγάλες και πιο συγκεκριμένες, είναι λογικό να χρειαζόμαστε και εργαλεία που θα επιδεικνύουν καλύτερες επιδόσεις σε αξιοπιστία και ιδιαίτερα σε ταχύτητα, η οποία παίζει σημαντικό ρόλο σε όλες τις επιστήμες, άρα και στις βιολογικές. Στόχος λοιπόν είναι να δημιουργούμε προγράμματα που θα είναι γρηγορότερα, πιο ευέλικτα και πιο φιλικά προς το χρήστη, με έτοιμα εργαλεία ώστε να μπορούμε να κάνουμε την επεξεργασία πιο εύκολα γλιτώνοντας κόπο και χρόνο. Ένα τέτοιο εργαλείο θα παρουσιάσουμε παρακάτω – το Biojava. Όπως ήδη αναφέραμε πρόκειται για ένα έργο (project) βασισμένο στη γλώσσα προγραμματισμού Java με στόχο να διευκολύνει την εκτέλεση βιολογικών διεργασιών. Στο παρόν κεφάλαιο θα γίνει εκτενής παρουσίασή των χαρακτηριστικών του και των πλεονεκτημάτων του σε σύγκριση με τον απλό προγραμματισμό σε γλώσσα Java και άλλες γλώσσες προγραμματισμού.

Βασικές έννοιες

- Biojava
- Open source project
- APIs
- Δυνατότητες
- Χαρακτηριστικά
- Ακολουθίες
- Σύμβολα
- Συγκριτικά πλεονεκτήματα

3.1 Τι είναι το BioJava

Το Biojava [79] είναι ένα ώριμο έργο ανοιχτού κώδικα (open source project) το οποίο παρέχει ένα ολοκληρωμένο πακέτο λογισμικού για την επεξεργασία βιολογικών δεδομένων.

Δημιουργήθηκε από εθελοντές και είναι ενταγμένο στον OBF (Open Bioinformatics Foundation).

Επινοήθηκε το 1999 από τους Thomas Down και Matthew Pocock. Ξεκίνησε σαν μια διασύνδεση προγραμματισμού εφαρμογών (API-Application Programming

Interface) και εξελίχτηκε σε ένα πλήρες πακέτο λογισμικού που περιέχει λειτουργικές υπομονάδες (modules) για την εκτέλεση ποικίλων συνηθισμένων εργασιών της Βιοπληροφορικής.

Έχει ήδη χρησιμοποιηθεί σε κάποιες εφαρμογές πραγματικού κόσμου και έχει γίνει λόγος γι' αυτό σε περισσότερες από 50 δημοσιεύσεις.

3.2 Δυνατότητες και χαρακτηριστικά

Οι βασικές δυνατότητες και τα ιδιαίτερα χαρακτηριστικά που επιδεικνύει παρουσιάζονται παρακάτω:

❖ *Τα σύμβολα και τα αλφάβητα*

Όταν έγινε για πρώτη φορά διαθέσιμη η βιολογική πληροφορία ήταν απαραίτητο να βρεθεί ένας κατάλληλος τρόπος για να τη διαχειριστούμε. Μια λογική προσέγγιση ήταν η αναπαράσταση κάθε μονομερούς σε ένα μακρομόριο με ένα μόνο γράμμα - συνήθως το αρχικό γράμμα της χημικής οντότητας που περιγράφεται, για παράδειγμα στο DNA χρησιμοποιούμε το γράμμα "T" για τη Θυμίνη. Έτσι όταν έγινε η εισαγωγή των δεδομένων στους υπολογιστές ήταν λογικό να χρησιμοποιηθεί η ίδια ιδέα. Στην πορεία όμως προέκυψαν κάποια προβλήματα όπως αυτό της επικύρωσης (αν δε γίνεται επικύρωση είναι δυνατόν κάθε συμβολοσειρά που εισάγεται σε μια ρουτίνα να θεωρείται βιολογική ακολουθία) ή της σημασίας κάθε συμβόλου (πολλές φορές μπορεί να μην είναι ξεκάθαρη και ως αποτέλεσμα να έχουμε διφορούμενα σύμβολα).

Για να αποφευχθούν τα παραπάνω προβλήματα το BioJava υιοθετεί μια διαφορετική προσέγγιση για τα δεδομένα βιολογικών ακολουθιών. Το κάθε σύμβολο δεν είναι απλά ένας ASCII χαρακτήρας, αλλά ένα αντικείμενο που «κουβαλάει» μαζί του αρκετές πληροφορίες.

```
public interface Symbol {
    public String getName();           //όνομα
    public Annotation getAnnotation(); //πληροφορίες (π.χ.
                                         στοιχεία για τη χημική
                                         σύσταση)
    public Alphabet getMatches();      //μόνο για ασαφή σύμβολα
   }

//Η παραπάνω κλάση είναι μέρος του πακέτου org.biojava.bio.symbol
```

Όλα τα σύμβολα, λοιπόν, έχουν κάποιο συγκεκριμένο όνομα (name property - παραδείγματος χάριν, Θυμίνη). Μπορούν, προαιρετικά, να διαθέτουν κάποιες πρόσθετες πληροφορίες, που σχετίζονται με αυτά (παραδείγματος χάριν, πληροφορίες για τις χημικές ιδιότητες μιας βάσης DNA), οι οποίες καταχωρούνται σε μια πρότυπη δομή δεδομένων BioJava που ονομάζεται Annotation (σχολιασμός). Τα Annotations είναι απλά σύνολα δεδομένων με αξία κλειδιού (key-value data). Η τελευταία μέθοδος, η getMatches, είναι σημαντική μόνο για τα διφορούμενα σύμβολα.

Αν και η δημιουργία συμβόλων είναι ιδιαίτερης σημασίας εξίσου απαραίτητη και σημαντική είναι η δημιουργία αλφαβήτων από αυτά. Γνωρίζουμε πως τα αλφάβητα, τα οποία είναι διαφορετικά σε κάθε περίπτωση (π.χ. για το DNA το (A,G,T,C), για το RNA το (A,G,U,C)), μας παρουσιάζουν ποια σύμβολα έχουμε τη δυνατότητα να χρησιμοποιούμε κάθε φορά, δηλαδή τους δομικούς λίθους των ακολουθιών.

Με αυτόν τον τρόπο, όλα τα σύμβολα (symbol objects) που μπορούμε να συναντήσουμε σε ένα συγκεκριμένο τύπο ακολουθίας καθορίζονται σε ένα αλφάβητο (Alphabet). Αν και δίνεται η δυνατότητα να καθοριστούν τα σύμβολα και τα αλφάβητα, το BioJava παρέχει κάποια προκαθορισμένα αλφάβητα για την αναπαράσταση βιολογικών μορίων.

```
FiniteAlphabet dna = DNATools.getDNA();
Iterator dnaSymbols = dna.iterator();
while (dnaSymbols.hasNext()) {
    Symbol s = (Symbol) dnaSymbols.next();
    System.out.println(s.getName());
}
```

❖ *SymbolList: η απλή ακολουθία*

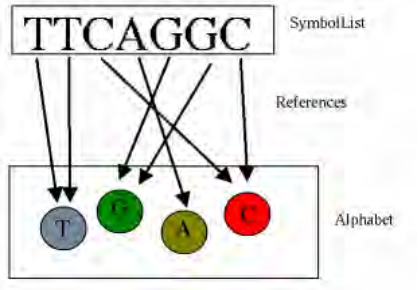
Η βασική διεπαφή για τα δεδομένα ακολουθίας στο BioJava είναι η SymbolList. Κάθε SymbolList έχει ένα σχετικό αλφάβητο και μπορεί να περιέχει σύμβολα μόνο από αυτό. Οι SymbolLists πρέπει να εξετάζονται σαν συμβολοσειρές οι οποίες αποτελούνται όχι από χαρακτήρες, όπως συνηθίζαμε μέχρι τώρα, αλλά από αντικείμενα Symbol. Η διεπαφή προσδιορίζει μεθόδους για να βρίσκουμε το αλφάβητο που χρησιμοποιείται σε μια ακολουθία και το μήκος της και να έχουμε πρόσβαση στα σύμβολα, όπως βλέπουμε στο παρακάτω παράδειγμα:

```
SymbolList seq = getSomeSequence();
System.out.println("Alphabet = " + seq.getAlphabet().getName());
System.out.println("Length = " + seq.length());
System.out.println("First symbol = " +
seq.symbolAt(1).getName());
```

Εκτός από τις παραπάνω, υπάρχουν διάφορες άλλες πρότυπες μέθοδοι στη διεπαφή SymbolList. Η μέθοδος *subList* επιστρέφει μια λίστα συμβόλων που αντιπροσωπεύει μέρος της ακολουθίας, ακριβώς όπως η μέθοδος substring της κλάσης String. Η μέθοδος *seqString* επιστρέφει μια αναπαράσταση της ακολουθίας με απλή συμβολοσειρά.

Σε αυτό το σημείο κρίνεται απαραίτητο να απαντήσουμε σε ένα εύλογο ερώτημα που δημιουργείται στους περισσότερους σχετικά με το Symbol/SymbolList μοντέλο του BioJava που παρουσιάσαμε παραπάνω. Αν δηλαδή χρειάζεται πολύ περισσότερη μνήμη από μια απλή προσέγγιση βασισμένη σε συμβολοσειρές. Πρέπει να τονιστεί πως κάτι τέτοιο δεν ισχύει κι αυτό γιατί δεν χρησιμοποιείται ένα ξεχωριστό αντικείμενο για την αναπαράσταση κάθε νουκλεοτιδίου σε μια μεγάλη ακολουθία DNA. Στην πραγματικότητα, υπάρχουν ακριβώς τέσσερα singleton αντικείμενα Symbol που αναπαριστούν τα σύμβολα που βρίσκονται στο αλφάβητο του DNA.

Όποτε, λοιπόν για παράδειγμα, ένα κατάλοιπο Θυμίνης καταχωρείται σε μια ακολουθία, στην πραγματικότητα καταχωρείται μια αναφορά στο singleton αντικείμενο της Θυμίνης. Επίσης, ένα ακόμη παράδειγμα εξοικονόμησης μνήμης είναι στην περίπτωση που θέλουμε να βρούμε τη συμπληρωματική μιας αλυσίδας. Κάτι τέτοιο είναι δυνατό με τη χρήση ενός απλού προγραμματιστικού μετασχηματισμού, χρησιμοποιώντας απλά μερικά bytes μνήμης ανεξάρτητα από το μήκος της, σε αντίθεση με τα megabytes που είναι απαραίτητα για την αναπαράσταση μιας ακολουθίας σα συμβολοσειρά.



Σχήμα 3: Τρόπος ανάγνωσης μιας DNA ακολουθίας με τη γλώσσα Biojava (Πηγή:[7])

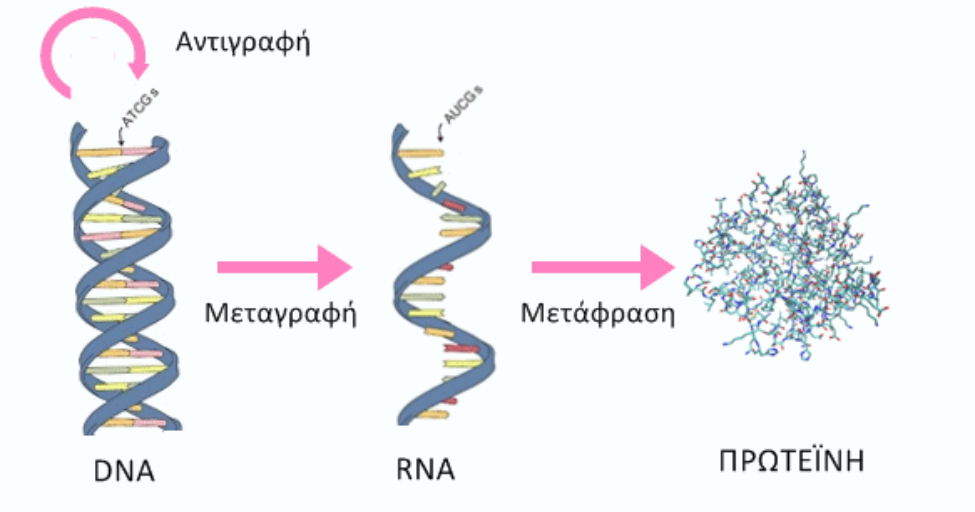
❖ *Ακολουθίες*

Δεδομένου ότι στις βιολογικές επιστήμες και κατ' επέκταση στη Βιοπληροφορική οι ακολουθίες (πρωτεϊνικές και νουκλεοτιδικές) και ο χειρισμός τους καταλαμβάνουν πρωταρχική θέση, δεν θα μπορούσε να λείπει από το BioJava κάποια διευκόλυνση σχετικά με αυτές.

Μέχρι αυτό το σημείο, επικεντρωθήκαμε στη διεπαφή SymbolList που, όπως καταλαβαίνουμε από το όνομά της, είναι μια ακατέργαστη λίστα Symbol αναφορών. Ωστόσο, οι πραγματικές καταχωρήσεις στις βιολογικές βάσεις δεδομένων είναι πιο περίπλοκες: οι ακολουθίες έχουν σχεδόν πάντα κάποιο είδος ID κώδικα ή κάποια σχετική περιγραφή και πολλές συνοδεύονται ακόμη και από πίνακες με σχόλια. Στο BioJava, το Sequence (δηλαδή η ακολουθία) είναι μια υποδιεπαφή της SymbolList και κατά συνέπεια όλες οι μέθοδοι που χρησιμοποιούνται για την πρόσβαση στα δεδομένα ακολουθίας σε μια SymbolList μπορούν εξίσου να εφαρμοστούν σε μια ακολουθία (Sequence) και οι ακολουθίες μπορούν να περάσουν ως είσοδοι σε οποιεσδήποτε μεθόδους ανάλυσης που κανονικά αναμένουν μια SymbolList. Η διεπαφή ακολουθίας προσθέτει δύο τύπους στοιχείων σε μια λίστα συμβόλων: τους global σχολιασμούς, όπως τα ονόματα, τα προσδιοριστικά βάσεων δεδομένων και βιβλιογραφικές αναφορές και τους σχολιασμούς συγκεκριμένου τόπου (Location-specific) που αποκαλούνται και features.

Εκτός από τους δύο παραπάνω τύπους σχολιασμού σε μια ακολουθία μπορεί να συνδέονται κι άλλα σχόλια κι αυτό γιατί το BioJava παρέχει μια ειδική διεπαφή για την εισαγωγή σχολίων με το όνομα Annotation – η διεπαφή αυτή χρησιμοποιείται και σε άλλα αντικείμενα και όχι μόνο στη Sequence.

Κατά γενικό κανόνα η διεπαφή Sequence χρησιμοποιείται συνήθως για τις ακολουθίες που έχουν φορτωθεί σε ένα πρόγραμμα από αρχεία ή βάσεις δεδομένων. Η διεπαφή SymbolList είναι, ίσως, πιο κατάλληλη για τις ακολουθίες που παράγονται εσωτερικά από ένα πρόγραμμα ανάλυσης.



Σχήμα 4: Κεντρικό δόγμα της Μοριακής Βιολογίας

❖ *Χειρισμός ακολουθιών*

Μετά την αναπαράσταση και την απόκτηση των ακολουθιών πολύ σημαντική είναι η επεξεργασία τους. Το BioJava κάνει ευκολότερο το χειρισμό τους απλοποιώντας λειτουργίες ζωτικής σημασίας.

Παραδείγματος χάριν διευκολύνει την αντιγραφή DNA και RNA ακολουθιών. Όλοι γνωρίζουμε τη σημασία της αντιγραφής του DNA κατά την οποία αυτό αυτοδιπλασιάζεται προκειμένου να διατηρήσει και να μεταβιβάσει τη γενετική πληροφορία από κύτταρο σε κύτταρο. Για να πραγματοποιηθεί όμως η αντιγραφή θα πρέπει να βρεθεί πρώτα η συμπληρωματική του αλυσίδα. Κάτι τέτοιο καθίσταται εύκολα δυνατό με τη χρήση της μεθόδου `DNATools.reverseComplement(SymbolList sl)`.

Ακόμη ένα παράδειγμα είναι η εύκολη μεταγραφή του DNA σε RNA, δηλαδή το πρώτο στάδιο της γονιδιακής έκφρασης κατά το οποίο δημιουργείται ένα μόριο RNA, με χρήση μιας αλυσίδας DNA ως προτύπου. Η διαδικασία αυτή γίνεται εύκολα με τη χρήση της μεθόδου `DNATools.toRNA()`.

Μια επίσης πολύ σημαντική διεργασία είναι η μετάφραση, δηλαδή η έκφραση της πληροφορίας στη γλώσσα των αμινοξέων. Μετά τη μεταγραφή του DNA σε RNA, εκείνο με τη σειρά του (συγκεκριμένα το mRNA), βάση του γενετικού κώδικα, μεταφράζεται στην αλληλουχία αμινοξέων μιας πρωτεΐνης. Στο BioJava η διαδικασία αυτή λαμβάνει χώρα με τη χρήση της `RNATools.translate()`.

Άλλες δυνατότητες είναι η δημιουργία της κανονικής έκφρασης μιας ακολουθίας, αλλαγή των ονομάτων των ακολουθιών και πολλές άλλες.

❖ *Ακολουθίες εισόδου εξόδου*

Υπάρχουν αρκετοί τρόποι με τους οποίους μπορεί να αναπαρασταθεί μια βιολογική αλληλουχία (π.χ. με μια απλή ακολουθία χαρακτήρων ή με ένα sequence αντικείμενο όσον αφορά το BioJava ή με ένα fasta αρχείο όσον αφορά τις ακολουθίες που ανακτώνται κατευθείαν από βάσεις δεδομένων) οπότε θα πρέπει το πρόγραμμά μας να είναι κάθε φορά σε θέση να δέχεται οποιονδήποτε τύπο εισόδου.

Ο κώδικας ακολουθιών εισόδου-εξόδου στο BioJava έχει σχεδιαστεί με σκοπό να είναι εύκαμπτος και να προσαρμόζεται εύκολα σε μια ευρεία ποικιλία εισόδων. Ωστόσο, μπορούν να χρησιμοποιηθούν και απλές μέθοδοι που εγκαθιστούν σαρωτές για την ανάγνωση διαφόρων ειδών αρχείων. Κατά γενικό κανόνα, αυτές οι μέθοδοι παίρνουν ένα αντικείμενο BufferedReader της Java και επιστρέφουν έναν επαναλήπτη που επιτρέπει τη σάρωση των ακολουθιών ενός αρχείου. Παραδείγματος χάριν:

```
BufferedReader br = new BufferedReader(new
FileReader(fileName));
SequenceIterator stream = SeqIOTools.readFastaDNA(br);
while (stream.hasNext()) {
    Sequence seq = stream.nextSequence();
    // do something with the sequence.
}
```

Όπως αναφέραμε και παραπάνω υπάρχουν αρκετοί τρόποι γραφής μιας ακολουθίας οι οποίοι διαφέρουν μεταξύ τους περισσότερο ως προς το μέγεθος σχολιασμού που διαθέτουν και τον τρόπο με τον οποίο αυτός είναι γραμμένος. Για παράδειγμα ένα χαρακτηριστικό αρχείο βιολογικής ακολουθίας περιέχει global πληροφορίες (π.χ. ταυτότητα), σχολιασμούς σε συγκεκριμένες περιοχές της ακολουθίας και πραγματικά δεδομένα ακολουθίας. Τα είδη αρχείων, βέβαια, που χρησιμοποιούνται στην πραγματικότητα (π.χ. fasta) περιέχουν τα καθαρά στοιχεία της ακολουθίας με μια μόνο γραμμή περιγραφής, ενώ από την άλλη πλευρά έχουμε και τα GFF αρχεία που είναι μια απλή λίστα χαρακτηριστικών γνωρισμάτων χωρίς καθόλου δεδομένα ακολουθίας.

Ένα sequence αντικείμενο της BioJava περιέχει τα ίδια είδη πληροφοριών με ένα αρχείο ακολουθίας. Ο αρχικός στόχος της αρχιτεκτονικής εισόδου είναι προφανώς να ληφθεί ένα stream που περιέχει τα δεδομένα αρχείων ακολουθίας, και να επιστραφούν ένα ή περισσότερα sequence αντικείμενα.

Το πλαίσιο εισόδου ακολουθίας είναι βασισμένο στη διεπαφή SequenceBuilder (αυτή είναι στην πραγματικότητα μια υποεπαφή του SeqIOListener, αλλά γι αυτούς τους σκοπούς χρησιμοποιείται συνήθως η κλάση SequenceBuilder). Ο ρόλος ενός "οικοδόμου ακολουθίας"(sequence builder) είναι να συσσωρεύσει τις πληροφορίες που ανακαλύπτονται αναλύοντας ένα αρχείο ακολουθίας και, τελικά, να κατασκευάζει ένα αντικείμενο ακολουθίας. Υπάρχουν δύο είδη εφαρμογών SequenceBuilder: οι οικοδόμοι (Builders) και τα φίλτρα (Filters).

❖ *Features*

Ένα feature (χαρακτηριστικό γνώρισμα) αντιπροσωπεύει μια περιοχή μιας ακολουθίας που έχει μερικές καθορισμένες ιδιότητες, δηλαδή μπορούν να αναπαριστούν δομές όπως για παράδειγμα γονίδια και επαναλαμβανόμενα στοιχεία σε χρωμοσώματα ή α-έλικες σε πρωτεΐνες. Σαν διεπαφή της Java, το feature έχει τις ακόλουθες βασικές ιδιότητες:

– Μια θέση(location) μέσα στην ακολουθία με καθορισμένη έναρξη και μια λήξη (ίσες στην περίπτωση των σημειακών θέσεων (point locations)).
– Ένα τύπο (type) (παραδείγματος χάριν, "γονίδιο" ή "έλικα").
– Μια πηγή (source) (συχνά το όνομα του προγράμματος που ανακάλυψε το feature).
– Ένα αντικείμενο σχολιασμών (Annotation object), το οποίο μπορεί να περιέχει οποιαδήποτε άλλα στοιχεία.

Τα σύνολα των features καταχωρούνται σε αντικείμενα εφαρμόζοντας τη διεπαφή FeatureHolder. Το feature το ίδιο επίσης επεκτείνει το FeatureHolder, δίνοντας τη δυνατότητα αναπαράστασης "εμφωλευμένων" χαρακτηριστικών γνωρισμάτων. Παραδείγματος χάριν, ένα feature που αντιπροσωπεύει μια μεγάλη γενετική ρυθμιστική περιοχή μπορεί να περιέχει sub-features σχολιάζοντας επιμέρους παράγοντες μεταγραφής θέσεων δέσμευσης.

❖ *Μετρήσεις και κατανομές (counts and distributions)*

Το BioJava περιλαμβάνει εκτός των άλλων, και έτοιμες μεθόδους μέτρησης κάποιου συγκεκριμένου καταλοίπου σε μια ακολουθία, υπολογισμού της συχνότητας εμφάνισης ενός symbol σε μια ακολουθία, μετατροπής της μέτρησης σε κατανομή και άλλες που μας βοηθούν να αποκτούμε μια γενική άποψη για τις ακολουθίες και σε περίπτωση που λαμβάνει χώρα κάποια αλλαγή να εντοπίζεται γρήγορα.

Γνωρίζουμε πως αρκετές ασθένειες, οι λεγόμενες γενετικές, προκαλούνται είτε από γονιδιακές (με αντικατάσταση, προσθήκη ή αφαίρεση βάσης – δρεπανοκυτταρική αναιμία, φαινυλκετονουρία κλπ) είτε από χρωμοσωμικές (δομικές και αριθμητικές – σύνδρομο Down, σύνδρομο Klinefelter, σύνδρομο Turner, πολυπλοειδισμός κλπ) μεταλλάξεις. Αυτές οι μεταλλάξεις έχουν ως συνέπεια την αύξηση ή μείωση χρωμοσωμάτων, αλλαγές σε βάσεις ή σε ολόκληρα γονίδια, με αποτέλεσμα να αλλάζει ο αριθμός τους και να παρατηρούνται αλλαγές στη συχνότητα και στα ποσοστά εμφάνισής τους. Κάτι τέτοιο, λοιπόν, θα ήταν ιδιαίτερα χρήσιμο στις περιπτώσεις των μεταλλάξεων.

Μια άλλη εφαρμογή τους θα μπορούσε να είναι στη δημιουργία profiles, όπου επίσης χρειάζονται μετρήσεις για το ποια κατάλοιπα συναντώνται σε κάθε θέση στοίχισης και πόσο συχνά εμφανίζεται ένα κατάλοιπο σε κάθε θέση.

❖ *Πίνακες βαρών και δυναμικός προγραμματισμός*

Ένα απλό μοντέλο που περιγράφει ένα μοτίβο ή ένα μέρος πολλαπλής τοπικής στοίχισης είναι αυτό των πινάκων βαρών, στο οποίο ένα μοτίβο χαρακτηρίζεται με πιθανότητες (εμφάνισης κάποιας βάσης) σε κάθε θέση. Ένας τέτοιος πίνακας για ένα πρότυπο μήκους n καθορίζεται σαν ένας πίνακας αποτελούμενος από αριθμούς $W_{i,x}$ όπου το i κυμαίνεται από 1 μέχρι n και το x είναι το αλφάβητο (για παράδειγμα το {A,T,G,C} για το DNA).

Όσον αφορά το δυναμικό προγραμματισμό στον τομέα της Βιοπληροφορικής έχουμε αρκετά παραδείγματα όπως η χρήση των πινάκων που περιγράφηκαν παραπάνω, οι κατά ζεύγη στοιχίσεις με τη βοήθεια των HMM, καθώς και οι αλγόριθμοι για ολική και τοπική στοίχιση.

3.3 Biojava 1.7.1 - API

Παρακάτω παρουσιάζονται τα πακέτα (packages) που περιλαμβάνει η τελευταία έκδοση του Biojava – αναφέρονται ενδεικτικά τα πιο σημαντικά από κάθε κατηγορία (αναλυτικότερα μπορεί κάποιος να τα βρει στον ιστοχώρο του BioJava [9]).

- Βασικά βιολογικά πακέτα (packages)
  - `org.biojava.bio.alignment`
    Κλάσεις για τη δημιουργία και την περιγραφή στοιχίσεων ακολουθιών.
  - `org.biojava.bio.seq.db`
    Συλλογή από δεδομένα βιολογικών ακολουθιών.
- Συνιστώσες της διεπαφής χρήστη
  - `org.biojava.bio.gui`
    Γραφικές διεπαφές για τα αντικείμενα του Biojava.
  - `org.biojava.bio.gui.sequence`
    Γραφική παρουσίαση των ακολουθιών και των σχετικών με αυτές σχολιασμών.
- Βάσεις δεδομένων ακολουθιών και μορφές αρχείων
  - `org.biojava.bio.program.homologene`
    Κλάσεις υποστήριξης για δεδομένα από το Homologene
  - `org.biojava.bio.seq.db.biosql`
    Γενικού σκοπού αποθήκευση ακολουθιών σε μια σχετική βάση δεδομένων.
- Χειρισμός δεδομένων εξόδου (outputs) από άλλα εξωτερικά εργαλεία
  - `org.biojava.bio.program.hmmer`

Εργαλεία για την εργασία με Hidden Markov μοντέλα από το πακέτο `HMMer`.

  - `org.biojava.bio.program.sax`
    Σαρωτές που προσφέρουν αναπαραστάσεις των εξόδων διαφόρων βιολογικών εργαλείων σε XML.

- Πακέτα Δυναμικού Προγραμματισμού
  - `org.biojava.bio.dp`
    Hidden Markov Models και αλγόριθμοι δυναμικού προγραμματισμού.

- Χρωματογραφήματα
  - `org.biojava.bio.chromatogram`
    Κλάσεις και διεπαφές για δεδομένα από χρωματογραφήματα, όπως αυτά προκύπτουν από τον προσδιορισμό της ακολουθίας DNA.
  - `org.biojava.bio.chromatogram.graphic`
    Εργαλεία για την αναπαράσταση των χρωμογραφημάτων.

- Μακρομοριακή δομή
  - `org.biojava.bio.structure`
    Κλάσεις και διεπαφές για την πρωτεϊνική δομή.
  - `org.biojava.bio.structure.io`
    Είσοδοι και έξοδοι δομών.

- Χρήσιμα προγράμματα και πακέτα ανάπτυξης
  - `org.biojava.ontology.io`
    Εργαλεία για το «κατέβασμα» και την αποθήκευση ακολουθιών.
  - `org.biojava.utils.math`
    Χρήσιμες κλάσεις σχετικές με τα Μαθηματικά.

- Πακέτα Μοριακής Βιολογίας
  - `org.biojava.bio.molbio`
    Οι κλάσεις και οι διεπαφές σε αυτό το πακέτο καλύπτουν συνηθισμένες τεχνικές της Μοριακής Βιολογίας, όπως είναι η PCR (Protein Chain Reaction).
  - `org.biojava.bio.proteomics`
    Διευκολύνσεις που βοηθούν στην εκτέλεση διαφόρων φυσικών αναλύσεων των πρωτεϊνών.

- Πειραματικά πακέτα
  - `org.biojava.bio.seq.distributed`
    Ακολουθίες και βάσεις δεδομένων ακολουθιών, οι οποίες αποτελούνται από δεδομένα προερχόμενα από διαφορετικές πηγές.
  - `org.biojava.bio.seq.homol`
    Κλάσεις και διεπαφές για τον καθορισμό της ομοιότητας μεταξύ ακολουθιών και της ομολογίας.

- Πλαίσιο (framework) για γενετικούς αλγόριθμους

  - `org.biojavax.ga`
    Κλάσεις για την παροχή ενός πλαισίου για γενετικούς αλγόριθμους.
  - `org.biojavax.ga.util`
    Χρήσιμες λειτουργίες και βοηθητικές κλάσεις.

- Πακέτα για πειράματα Φυλογενετικής
  - `org.biojavax.bio.phylo`
    Κλάσεις για την υποστήριξη των αντικειμένων Φυλογενετικής (phylogeny objects).
  - `org.biojavax.bio.phylo.io.phylip`
    Κλάσεις για την υποστήριξης γραφής και ανάγνωσης αρχείων μορφής PHYLIP.

- Άλλα πακέτα
  - `org.biojava.bio.structure.gui`
    Περιλαμβάνει κλάσεις που βοηθούν το χρήστη να δει πρωτεϊνικές δομές με το Jmol (εάν βρίσκεται στο μονοπάτι), να υπολογίσει τη στοίχιση πρωτεϊνικών δομών και να ανακαλύψει λεπτομέρειες για τον αλγόριθμο στοίχισης πρωτεϊνών.
  - `org.biojava.bio.structure.io.mmcif`
    Είσοδοι και έξοδοι mmcif αρχείων.

3.4 BiojavaX

Το BiojavaX είναι μια επέκταση για το υπάρχον project Biojava. Δεν πρόκειται για κάποια καινούρια έκδοση όπως είναι, για παράδειγμα, το Biojava 3. Σε αντίθεση λοιπόν με τις καινούριες εκδόσεις στις οποίες χρειάζεται να γραφτεί ο κώδικας από την αρχή, το BiojavaX είναι απόλυτα συμβατό με το υπάρχον project. Οτιδήποτε είναι γραμμένο σε Biojava μπορεί να λειτουργήσει στο BiojavaX και το αντίστροφα.

3.4.1 Τι δεν άλλαξε;

Υφιστάμενες διεπαφές

Η συμβατότητα με τις ήδη υπάρχουσες εκδόσεις είναι πάντα ένα σημαντικό ζήτημα όταν πρόκειται να αναπτυχθεί ένα νέο τμήμα λογισμικού είτε πρόκειται για μια απλή επέκταση είτε για μια καινούρια έκδοση. Το BiojavaX αντιμετωπίζει αυτό το πρόβλημα κρατώντας όλες τις καινούριες κλάσεις και διεπαφές μέσα σε ένα πακέτο (package), το org.biojavax. Ωστόσο καμία από τις παλιές διεπαφές δεν τροποποιήθηκαν, οπότε οποιοδήποτε κομμάτι κώδικα εξαρτάται από αυτές δε θα παρουσιάσει καμία διαφορά. Εκτός από τα συνεχώς εξελισσόμενα bugfixes (διορθωτές σφαλμάτων) ο τρόπος με τον οποίο λειτουργούσαν οι παλιές κλάσεις επίσης δεν άλλαξε. Οι καινούριες διεπαφές που παρουσιάζονται στο BiojavaX επεκτείνουν αυτές που παρουσιάζονται στα ήδη υπάρχοντα πακέτα του project.

Ελεγκτές αλλαγών

Το μοντέλο ελέγχου αλλαγών του Biojava παρέμεινε αμετάβλητο. Οι καινούριες κλάσεις ορίζουν μία σειρά νέων τύπων αλλαγών που προστίθενται σ' αυτούς που ήδη βρίσκονται στις κλάσεις. Κάτι τέτοιο σημαίνει πως οι υπάρχοντες ελεγκτές αλλαγών έχουν τη δυνατότητα να προσκολληθούν στα αντικείμενα του BiojavaX και να λαμβάνουν πληροφορίες, όπως ακριβώς συνέβαινε και πριν.

Σάρωση αρχείων βάσει γεγονότων

Το BiojavaX χρησιμοποιεί ακόμη σάρωση αρχείων βασισμένη σε γεγονότα για την ανάγνωση και τη γραφή αρχείων, ακριβώς με τον ίδιο τρόπο που γινόταν στις παλαιότερες κλάσεις.

3.4.2 Τι άλλαξε;

Απαιτήσεις συστήματος

Η Java 1.4 είναι απαραίτητη για όλα τα πακέτα BioJavaX.

Πλούσιες διεπαφές

Το BioJavaX ορίζει μια νέα σειρά διεπαφών για την ενασχόληση με αντικείμενα ακολουθίας. Οι νέες διεπαφές επεκτείνουν κατά το δυνατό τις παλιές με στόχο να διατηρηθεί συμβατότητα. Οι διεπαφές, αυτές γνωστές ως πλούσιες (rich) διεπαφές, θα μπορούσαν να θεωρηθούν ότι είναι «εμπλουτισμένες» εκδόσεις των διεπαφών που επεκτείνουν. Οι νέες αυτές διεπαφές παρουσιάζονται παρακάτω:

- `ComparableOntology`
- `ComparableTerm`
- `ComparableTriple`
- `RichSequenceIterator`
- `RichSequence`
- `RichLocation`
- `RichFeature`
- `RichFeatureHolder`
- `RichAnnotatable`
- `RichAnnotation`
- `BioSQLFeatureFilter`
- `RichSequenceDB`

BioSQL persistence

Το BioJavaX παρουσιάζει ένα νέο τρόπο συνεργασίας με τις βάσεις δεδομένων BioSQL. Αντί να γίνεται προσπάθεια να ξαναανακαλυφθεί ο τροχός με ένα ακόμη νέο αντικείμενο-σχεσιακό σύστημα χαρτογράφησης, το BioJavaX χρησιμοποιεί τις υπηρεσίες του Hibernate για να κάνει «όλη τη δουλειά γι 'αυτό». Στην πραγματικότητα, δεν υπάρχει ούτε μία δήλωση SQL σε οποιοδήποτε σημείο του κώδικα του BioJavaX.

Καλύτερη σάρωση αρχείων

Οι παλιοί σαρωτές αρχείων του Biojava φόρτωναν όλες τις πληροφορίες στη μνήμη, αλλά δεν ασχολούνταν ιδιαίτερα με την κατανόηση του περιεχομένου των αρχείων. Συνεπώς, συχνά αποτύγχαναν κατά τη μετατροπή των αρχείων από μια

μορφή σε κάποια άλλη. Οι καινούριοι σαρωτές καταβάλλουν μεγάλη προσπάθεια να χωρέσουν τα δεδομένα από μυριάδες μορφές αρχείων σε μια μορφή που θα μπορεί να αναπαρασταθεί με το BioSQL και ως εκ τούτου με το νέο μοντέλο αντικειμένων του BiojavaX. Φυσικά κάτι τέτοιο δεν είναι πάντα εφικτό, αλλά δουλεύει πολύ καλύτερα απ' ότι παλαιότερα. Σαρώνοντας δεδομένα σε ένα σταθερό μοντέλο αντικειμένων αντί να αποθηκεύονται τα πάντα ως σχολιασμοί (όπως συνέβαινε, για παράδειγμα, με τους παλιούς σαρωτές SwissProt), η μετατροπή από μια μορφή αρχείου σε μια άλλη γίνεται πολύ ευκολότερη. Οι νέοι σαρωτές επιτρέπουν, επίσης, στο χρήστη να παρακάμψει εντελώς μη ενδιαφέροντα τμήματα, κάτι που μπορεί να επιταχύνει σε μεγάλο βαθμό απλές διεργασίες, όπως η καταμέτρηση των ακολουθιών σε ένα αρχείο.

NCBI ταξινόμηση

Ένας σαρωτής παρέχεται για τη φόρτωση της NCBI Ταξινομικής βάσης δεδομένων σε ένα σύνολο αντικειμένων BioJavaX NCBITaxon. Ο σαρωτής, αυτός, διαβάζει τα αρχεία node.dmp και names.dmp που παρέχονται από το NCBI και κατασκευάζει την κατάλληλη ιεραρχία αντικειμένων.

Πεδίο ονομάτων

Όλες οι ακολουθίες στο BioJavaX πρέπει να ανήκουν σε κάποιο πεδίο ονομάτων (namespace).

Singletons

Το BioJavaX προσπαθεί να χρησιμοποιήσει τα Singletons όσο το δυνατόν περισσότερο. Αυτό γίνεται για τους εξής λόγους:

*Μείωση της χρήσης μνήμης.

* Αποτροπή προβλημάτων με διπλά κλειδιά στη BioSQL.

Γενετικοί αλγόριθμοι

Το BioJavaX εισάγει ένα νέο πακέτο για την εργασία με γενετικούς αλγορίθμους.

3.4.3 Μελλοντικά σχέδια

Μερικά από τα μελλοντικά σχέδια είναι η συμβατότητα με το BioPerl και τη βάση δεδομένων του, η πιο αποτελεσματική σάρωση και η υποστήριξη περισσότερων μορφών αρχείων.

3.4.4 Πακέτα (packages) που περιλαμβάνει

- `org.biojavax`
  Κλάσεις και διεπαφές που επεκτείνουν τις βασικές λειτουργίες του Biojava.
- `org.biojavax.bio`
  Κλάσεις για την αναπαράσταση βιολογικών οντοτήτων και των μεταξύ τους σχέσεων.
- `org.biojavax.bio.db`
  Αλληλεπιδράσεις μεταξύ των αντικειμένων του BiojavaX και μιας βάσης δεδομένων.
- `org.biojavax.bio.db.biosql`

Διεπαφή μεταξύ του Biojava και βάσεων δεδομένων BioSQL.

- `org.biojavax.bio.db.ncbi`
  Διεπαφές για NCBI δεδομένα.
- `org.biojavax.bio.seq`
  Πλούσιες υλοποιήσεις ακολουθιών, τοποθεσιών (locations) και χαρακτηριστικών γνωρισμάτων (features).
- `org.biojavax.bio.seq.io`
  Κλάσεις για την υποστήριξη της εισόδου και εξόδου των RichSequences και των Bioentry αντικειμένων.
- `org.biojavax.bio.taxa`
  Κλάσεις που μοντελοποιούν το NCBI ταξινομικό σχήμα όπως περιγράφεται στο BioSQL σχήμα.
- `org.biojavax.bio.taxa.io`
  Κλάση για «φόρτωμα» ταξινομικών στοιχείων NCBI.
- `org.biojavax.ontology`
  Επεκτάσεις στο μοντέλο οντολογιών του Biojava που αναπαριστά την οντολογία BioSQL.
- `org.biojavax.utils`
  Χρήσιμες κλάσεις που χρησιμοποιούνται από τα αντικείμενα του BiojavaX, αλλά μπορούν να έχουν ευρεία χρησιμότητα και σε άλλες εφαρμογές.

3.5 Πλεονεκτήματα

3.5.1 Πλεονεκτήματα έναντι της Java

- Αναδεικνύει τα πλεονεκτήματα της Java και παρέχει μια βιβλιοθήκη διεπαφών που ενσωματώνουν έννοιες με πυρήνα τη Βιοπληροφορική. Ενώ, δηλαδή, η Java είναι μια γλώσσα προγραμματισμού που χρησιμοποιείται σε ποικίλες εφαρμογές και απλά εξυπηρετεί και εργασίες Βιοπληροφορικής, το Biojava επικεντρώνεται σε αυτές.
- Προσφέρει έτοιμα πακέτα. Ενώ χρησιμοποιώντας τη Java ο χρήστης θα πρέπει να γράψει κώδικα, να δημιουργήσει δηλαδή μόνος του το πρόγραμμα, στο Biojava κάτι τέτοιο δεν είναι απαραίτητο.
  Π.χ. για τον εντοπισμό μιας συγκεκριμένης ακολουθίας ανάμεσα σε κάποιους χαρακτήρες στη Java θα χρειαζόταν να γραφτεί πρόγραμμα, ενώ με το Biojava απλά χρησιμοποιούμε την κλάση Locate-Sequence.
- Υπάρχουν λοιπόν αντικείμενα για να αναπαριστούν:
  - ακολουθίες
  - στοιχίσεις
  - σχολιασμούς γονιδιωμάτων
  - βάσεις δεδομένων
  - αλγορίθμους σύγκρισης ακολουθιών
  - γραφικές αναπαραστάσεις ακολουθιών

3.5.2 Πλεονεκτήματα έναντι άλλων γλωσσών

- Αρχικά, το Biojava, όσον αφορά τη γλώσσα προγραμματισμού δε χρησιμοποιεί τίποτα παραπάνω από την ευρέως διαδεδομένη και γνωστή στους περισσότερους Java. Είναι εύκολο λοιπόν να προσαρμοστούν και να το χρησιμοποιήσουν αυτοί που ξέρουν Java - κάτι που βοηθά ιδιαίτερα τους προγραμματιστές που ίσως να μην έχουν ιδιαίτερη σχέση με τη Βιολογία - χωρίς να χρειάζεται να μάθουν από την αρχή κάποια άλλη γλώσσα, όπως π.χ. συμβαίνει στην περίπτωση της Perl.
- Προσφέρει υψηλή αντικειμενοστρέφεια επιτρέποντας εκτεταμένη παραμετροποίηση χωρίς να απαιτείται μεγάλος κώδικας.
- Εκμεταλλεύεται τα πλεονεκτήματα της Java έναντι άλλων γλωσσών προγραμματισμού, τα οποία είναι γενικά τα εξής:
  - Απλή και εύκολη στη χρήση.
  - Ανεξάρτητη από το λειτουργικό σύστημα - τα προγράμματα που είναι γραμμένα σε Java τρέχουνε ακριβώς το ίδιο σε Windows, Linux, Unix και Macintosh χωρίς να χρειαστεί να ξαναγίνει μεταγλώττιση (compiling) ή να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα.
  - Ασφαλής. Σε περίπτωση κάποιας δικτυακής εφαρμογής, ελαχιστοποιεί την πιθανότητα προσβολής του συστήματος του χρήστη από κάποιο κακόβουλο πρόγραμμα.
  - Πολυνηματική, δηλαδή ένα απλό πρόγραμμα σε Java μπορεί να κάνει πολλά, διαφορετικά προγράμματα ανεξάρτητα και αλληλεπιδρώντα.
  - Περιλαμβάνει πρότυπες βιβλιοθήκες οι οποίες παρέχουν, μεταξύ άλλων, υποστήριξη για συλλογές, δικτύωση, SQL, γραφικές διεπαφές χρήστη (GUIs), εκτύπωση, νήματα και CORBA.
  - Μόλις εμφανίζεται κάποια καινούρια τεχνολογία, η Java είναι συχνά ανάμεσα στις πρώτες γλώσσες που την αποδέχεται και την ενσωματώνει (π.χ. XML).
  - Η συλλογή «σκουπιδιών κώδικα» (garbage collection), η έλλειψη δεικτών (pointers), ο αυτόματος έλεγχος των ορίων του πίνακα και ο αυστηρός έλεγχος τύπων κάνουν τη Java δυνατή, ενώ, παράλληλα, το μοντέλο αντικειμένων και το toolkit με τα πρότυπα APIs που περιέχει βοηθούν στη γρήγορη ανάπτυξη εφαρμογών.
  - Υποστηρίζει εφαρμογές πολυμέσων.

3.6 Μελλοντική εξέλιξη

Στόχος του Biojava είναι να παρέχει ένα API το οποίο θα μπορεί να χρησιμοποιηθεί από οποιονδήποτε χρησιμοποιεί τη Java για να αναπτύξει λογισμικό σχετικά με τη Βιοπληροφορική ανεξάρτητα από την ειδικότητά του. Τα

χαρακτηριστικά του γονιδιώματος θα πρέπει προς το παρόν να τα χειριζόμαστε αναφορικά με την υποκείμενη γονιδιωματική ακολουθία, το οποίο μπορεί να κάνει την εργασία με μετα-γονιδιωματικά σύνολα δεδομένων, όπως τα αποτελέσματα μικροσυστοιχιών, ιδιαίτερα πολύπλοκη.

Παρόλο που το Blast API είναι ευρέως χρησιμοποιούμενο δεν υποστηρίζει όλες τις μορφές εξόδων της «οικογένειας» Blast. Συνεχίζεται λοιπόν η προσπάθεια να προστεθούν σαρωτές (parsers) για το PSI-Blast και άλλες μορφές που δεν υποστηρίζονται σήμερα.

Με το πέρασμα των χρόνων και με τις συνεχιζόμενες προσπάθειες το Biojava προβλέπεται να εξελιχθεί σε ένα πολύ δυνατό εργαλείο παρέχοντας ολοένα και περισσότερες κλάσεις ή ολόκληρα πακέτα με σκοπό να διευκολύνει τους χρήστες του. Ήδη βρίσκεται υπό κατασκευή μια νεότερη έκδοση του Biojava το Biojava 3.

3.6.1 Biojava3 (BJ3)

3.6.1.1 Βασικές αρχές

- Ενσωμάτωση χαρακτηριστικών γνωρισμάτων (features) του Java 6.
- Πρακτικός σχεδιασμός χωρίς κυκλικές εξαρτήσεις, με χωριστά JAR αρχεία για τις βασικές συνιστώσες (είσοδοι/έξοδοι, βάσεις δεδομένων, γενετικοί αλγόριθμοι, χειρισμός ακολουθιών, κ.λπ.).
- Διαχωρισμός των APIs από τον κώδικα της εφαρμογής με τη βοήθεια των πακέτων.
- Το όνομα του βασικού πακέτου είναι org.biojava3 (για την αποφυγή συγκρούσεων με το org.biojava και το org.biojavax, για τα οποία θα υπάρχουν επεκτάσεις συμβατές με το BJ3, ώστε να επιτρέπεται η χρησιμοποίηση και του παλιού κώδικα).
- Πλήρης σχολιασμός του κώδικα.
- Ανάπτυξη και εφαρμογή ενός σταθερού στυλ κωδικοποίησης.
- Δεν θα περιλαμβάνεται χρήση κώδικα Swing διότι είναι αδύνατον να γραφτεί με τέτοιο τρόπο ώστε να ενσωματώνει πλήρως τις απαιτήσεις των προγραμμάτων του κάθε χρήστη. Η χρήση, ωστόσο, κώδικα για τα γραφικά θα είναι επιτρεπτή, αφού είναι προφανές πως χρησιμοποιείται για χρηστικούς λόγους όπως για τις πρωτεϊνικές δομές.
- Προσπάθεια να κρατηθεί όσο πιο απλό γίνεται. Δε θα γίνεται χρήση των αντικειμένων παρά μόνο σε περίπτωση που κάτι τέτοιο κρίνεται απολύτως απαραίτητο. Για παράδειγμα, οι ακολουθίες θα μπορούν να αντιμετωπίζονται σαν απλές συμβολοσειρές και θα χρειάζεται να μετατραπούν σε πιο πολύπλοκα μοντέλα αντικειμένων μόνο σε περίπτωση που ο χρήστης θέλει πιο πολύπλοκους χειρισμούς, όπως η αποθήκευση βασικών πληροφοριών.
- Θα παρέχεται διαχωρισμός των λειτουργιών. Παραδείγματος χάριν δε θα υπάρχει προεπιλογή για να «φορτώνονται» μαζί με τις ακολουθίες τα features τους και αντίστοιχα με τα features οι ακολουθία τους. Με αυτό τον τρόπο εξοικονομείται μνήμη και επιτρέπεται στις εργασίες να γίνονται ξεχωριστά.

- Σίγουρα σωστή εκτέλεση των εντολών equals, compareTo και hashCode.
- Οι εξαιρέσεις (exceptions) προβλέπονται να είναι, σε γενικές γραμμές, RuntimeExceptions και unchecked (όπου δεν έχει γίνει έλεγχος) , οι οποίες μάλιστα θα εμφανίζουν χρήσιμα μηνύματα. Στην παρούσα έκδοση υπάρχουν ήδη εξαιρέσεις οι οποίες θα συμπληρωθούν με τις καινούριες.

3.6.1.2 Δομή βασισμένη σε ενότητες (module structure)

Επειδή η ενασχόληση με μεγάλα projects είναι μια δύσκολη διαδικασία, δόθηκε έμφαση στο χωρισμό του κώδικα σε μικρότερες ενότητες (modules) με μικρή εξάρτηση μεταξύ τους. Πυρήνα (δηλαδή το Biojava3 - core) αποτελούν μια σειρά από κλάσεις που θα είναι κοινές σε διάφορες ενότητες (modules). Τα κοινά στοιχεία για όλες τις ενότητες είναι η ανάγνωση, η γραφή και η αναπαράσταση δεδομένων ακολουθίας. Επίσης, μια σημαντική πρόταση είναι η χρήση της Java για τη μοντελοποίηση των βιολογικών σχέσεων μεταξύ των ακολουθιών με όση μεγαλύτερη ακρίβεια αυτό είναι δυνατό. Με αυτό τον τρόπο το biojava3 api θα καθορίσει συγκεκριμένες σχέσεις που θα βοηθούν τον προγραμματιστή να αντιλαμβάνεται τη Βιολογία μέσω του κώδικα και το βιολόγο να αισθάνεται οικειότητα κατά τη συγγραφή κάποιου κώδικα.

Κάποια από τα βασικά modules είναι τα εξής:

- Ενότητα Biojava3 (Biojava3 module)
  Η ενότητα api περιέχει δημιουργό αντικειμένων (object builder) ο οποίος δημιουργεί τα αντικείμενα από γεγονότα, όπως περίπου γίνεται και στην περίπτωση του αναλυτή SAX (Simple API for XML). Οι χρήστες, επίσης, μπορούν να επιλέξουν να αποθηκεύσουν δεδομένα στη μνήμη, στο δίσκο, να δημιουργήσουν ένα δείκτη προς την πηγή και να τα ανακτήσουν και να τα διαβάσουν αργότερα ή να τα χρησιμοποιήσουν με όποιο άλλο τρόπο επιθυμούν. Η βελτιστοποίηση με αυτό τον τρόπο γίνεται ευκολότερη, αφού οι χρήστες είναι σε θέση να αποφασίσουν τι ακριβώς θα κρατήσουν στη μνήμη και τι όχι.
- Ενότητα ακολουθίας (Sequence module)
  Αυτή η ενότητα api καθορίζει ολόκληρο το μοντέλο αντικειμένων ακολουθίας του Biojava (παρόμοιο με αυτό που υπάρχει σήμερα αλλά επιτρέπει ακολουθίες που δεν είναι βασισμένες σε σύμβολα και διαχωρίζει τις ακολουθίες από τα features). Το api έχει υποκλάσεις με δημιουργούς αντικειμένων για τις ακολουθίες, παρέχει μετατροπή των συμβολοσειρών σε ακολουθίες βασισμένες στα σύμβολα και το αντίθετο καθώς και απλοποίηση της έννοιας του αλφαβήτου, κ.ά..
- Ενότητα χαρακτηριστικών (Feature module)
  Η ενότητα api καθορίζει ολόκληρο το μοντέλο αντικειμένων feature του Biojava (παρόμοιο με αυτό που υπάρχει σήμερα αλλά διαχωρίζει τις ακολουθίες από τα features). Έχει υποκλάσεις με δημιουργούς

αντικειμένων για features και επιτρέπεται η ονοματοδοσία των features με βάση πρότυπες οντολογίες.

- Ενότητα εισόδου/εξόδου (IO module)

  Το api παρέχει βασικές λειτουργίες ανάγνωσης (read()) και γραφής (write()). Επίσης περιλαμβάνει την έννοια της RecordSource η οποία μπορεί να είναι είτε ένα αρχείο είτε μια ομάδα αρχείων είτε μια βάση δεδομένων ή μια υπηρεσία web κ.λπ., τα οποία (όλα) εφαρμόζουν ένα είδος διεπαφής RecordProvider για την επανάληψη πάνω σε αντικείμενα. Αυτά τα αντικείμενα μπορεί να είναι ακολουθίες, χαρακτηριστικά, κ.λπ.. Ακόμη υπάρχει η ενότητα με το όνομα εφαρμογή (implementation), συγκεκριμένα μία για κάθε είδος αρχείου (fasta, Genbank, κ.λπ.), καθεμιά από τις οποίες έχει προεπιλεγμένα το μοντέλο αντικειμένων και το δημιουργό που ταιριάζουν ακριβώς σε αυτή τη μορφή αρχείου.

  Οι ενότητες εισόδου/εξόδου είναι ανεξάρτητες μεταξύ τους και τελείως διαφορετικές ως προς τον τρόπο που χρησιμοποιούνται.

- Άλλες ενότητες
  - Χειρισμός οντολογιών (Ontology handling)
  - Πρωτεϊνική δομή (Protein structure)
  - Ανάλυση μικροσυστοιχιών (Microarray analysis)
  - Φυλογενετική (Phylogenetics)
  - κλπ

4ο ΚΕΦΑΛΑΙΟ
ΟΔΗΓΟΣ ΧΡΗΣΗΣ ΚΑΙ ΕΚΠΑΙΔΕΥΤΙΚΟ ΥΛΙΚΟ

Περίληψη

Στο παρόν κεφάλαιο θα υλοποιηθεί το βασικό κομμάτι της παρούσας πτυχιακής εργασίας που αφορά στον οδηγό χρήσης του project Biojava. Θα αναφερθούν παραδείγματα (με παραπομπές στους κώδικες του παραρτήματος) στα οποία χρησιμοποιείται το Biojava για την εκτέλεση βασικών βιολογικών διεργασιών, καθώς και υλοποιήσεις ολοκληρωμένων προβλημάτων με τον πηγαίο κώδικα και τα αποτελέσματά τους. Θα παρουσιαστεί το εκπαιδευτικό υλικό που θα περιέχεται σε μορφή ενοτήτων, για τη σταδιακή εκμάθηση του project. Τέλος, θα δοθούν αναλυτικές οδηγίες σχετικά με την εγκατάστασή του και κάποιες επισημάνσεις για τυχόν λάθη και παραλείψεις που μπορούν να δημιουργήσουν προβλήματα στα πρώτα βήματα.

Βασικές έννοιες

- Οδηγίες εγκατάστασης
- Αλφάβητα
- Χειρισμός ακολουθιών
- Μετάφραση
- Στοίχιση
- Δυναμικός προγραμματισμός
- Εκπαιδευτικό υλικό
- Ασκήσεις

4.1 Παραδείγματα προγραμμάτων γραμμένα σε Biojava

Στο προηγούμενο κεφάλαιο κάναμε λόγο για τις ιδιαίτερες δυνατότητες και τα χαρακτηριστικά του Biojava. Σε αυτό το κεφάλαιο θα υλοποιήσουμε συγκεκριμένα παραδείγματα επισημαίνοντας και τη βιολογική τους σημασία.

4.1.1 Τα σύμβολα και τα αλφάβητα

Κατά την ενασχόλησή μας με βιολογικά δεδομένα και ιδιαίτερα με ακολουθίες (DNA, πρωτεϊνικές κλπ) είναι αναγκαίος ο καθορισμός, των απαραίτητων κάθε φορά, αλφαβήτων. Με αυτό τον τρόπο είμαστε σε θέση να ελέγξουμε τις ακολουθίες και να εξασφαλίσουμε ότι δεν θα χρησιμοποιήσουμε κάποια που περιέχει λάθος σύμβολα.

Όπως προαναφέραμε τα αλφάβητα είναι συλλογές συμβόλων, τα οποία είτε υπάρχουν ήδη και απλά μπορούμε να τα καλέσουμε είτε πρέπει να δημιουργηθούν από την αρχή.

- *Κλήση ήδη υπαρχόντων αλφάβητων*

  Όσον αφορά στην πρώτη αυτή κατηγορία, εδώ ανήκουν τα ήδη γνωστά και πιο συχνά χρησιμοποιούμενα αλφάβητα όπως αυτά του DNA (αποτελούμενο από τέσσερα σύμβολα – A, G, C ,T), του RNA (αποτελούμενο από τέσσερα σύμβολα – A, G, C, U) και το πρωτεϊνικό (αποτελούμενο από 26 διαφορετικά σύμβολα). Σε αυτά μπορούμε να έχουμε πρόσβαση με δύο διαφορετικούς τρόπους:

  1) Αφού έχουν καταχωρηθεί μέσω του AlphabetManager κατά την εκκίνηση μπορούμε να έχουμε πρόσβαση απλά καλώντας τα με το όνομά τους.
     Π.χ. dna = AlphabetManager.alphabetForName("DNA");
  2) Ένας εξίσου απλός τρόπος είναι να τα πάρουμε από τις Tools classes του Biojava.
     Π.χ. dna = DNATools.getDNA();

  (βλέπε παράρτημα 1 - κώδικας 1.1)

DNA:	A--Αδενίνη	RNA:	A--Αδενίνη
	C--Κυτοσίνη		C--Κυτοσίνη
	G--Γουανίνη		G--Γουανίνη
	T--Θυμίνη		U--Ουρακίλη

Πρωτεΐνες:	A--Αλανίνη	I--Ισολευκίνη	R--Αργινίνη
	C--Κυστεΐνη	K--Λυσίνη	S--Σερίνη
	D--Ασπαραγανικό οξύ	L--Λευκίνη	T--Θρεονίνη
	E-Γλουταμινικό οξύ	M--Μεθειονίνη	V--Βαλίνη
	F--Φαινυλαλανινη	N--Ασπαραγίνη	W--Τ(Θ)ρυπτοφάνη
	G--Γλυκίνη	P--Προλίνη	Y--Τυροσίνη
	H--Ιστιδίνη	Q--Γλουταμίνη	

Εικόνα 5: Αλφάβητα (DNA, RNA, πρωτεϊνικό)

- *Δημιουργία αλφαβήτων και κλήση αυτών*

  Όσον αφορά στη δεύτερη αυτή κατηγορία, εδώ απαιτείται μια μικρή διαδικασία. Θα πρέπει πρώτα να καθοριστούν τα σύμβολα του αλφάβητου, να δημιουργήσουμε το αλφάβητο, να το καταχωρήσουμε και τέλος εάν θέλουμε να το καλέσουμε με τον πρώτο τρόπο που αναφέραμε παραπάνω.
  (βλέπε παράρτημα 1 - κώδικας 1.2)

- *CrossProductAlphabets*

  Εκτός από αυτά τα απλά αλφάβητα έχουμε τη δυνατότητα να δημιουργήσουμε και CrossProductAlphabets (π.χ. ένα αλφάβητο με τα κωδικόνια), που προκύπτουν από τον πολλαπλασιασμό των ήδη υπαρχόντων αλφαβήτων. Μπορούμε ακόμη να κάνουμε και την αντίθετη διαδικασία, δηλαδή αν έχουμε ένα CrossProductAlphabet να δούμε από ποια σύμβολα αποτελείται.
  (βλέπε παράρτημα 1 - κώδικας 1.3)

4.1.2 Χειρισμός ακολουθιών

Όσον αφορά τις ακολουθίες υπάρχουν αρκετές διευκολύνσεις σχετικά με τους βασικούς χειρισμούς τους. Παρακάτω θα αναφέρουμε μερικές από αυτές:

- *Μετατροπή συμβολοσειράς σε ακολουθία (sequence) και ακολουθίας σε συμβολοσειρά*

  Οι συμβολοσειρές είναι ο πιο βολικός και συχνά χρησιμοποιούμενος τρόπος αναπαράστασης των ακολουθιών αλλά και των μεγαλύτερων βιολογικών πολυμερών. Το Biojava κάνει χρήση των SymbolLists και των Sequences για την αναπαράσταση των πολυμερών. Με αυτόν τον τρόπο δεν αποθηκεύονται σαν απλές συμβολοσειρές αλλά, κάνοντας χρήση των αντικειμένων από διαφορετικά αλφάβητα, είμαστε σε θέση κάθε φορά να αναγνωρίζουμε αν πρόκειται για ακολουθία DNA, RNA ή πρωτεϊνική. Το βασικό, λοιπόν, σημείο είναι να δίνεται η δυνατότητα στον προγραμματιστή να μετατρέπει το απλό κείμενο (συμβολοσειρά) σε SymbolLists ή Sequence και το αντίθετο. Αυτό γίνεται με την κλήση απλών μεθόδων.

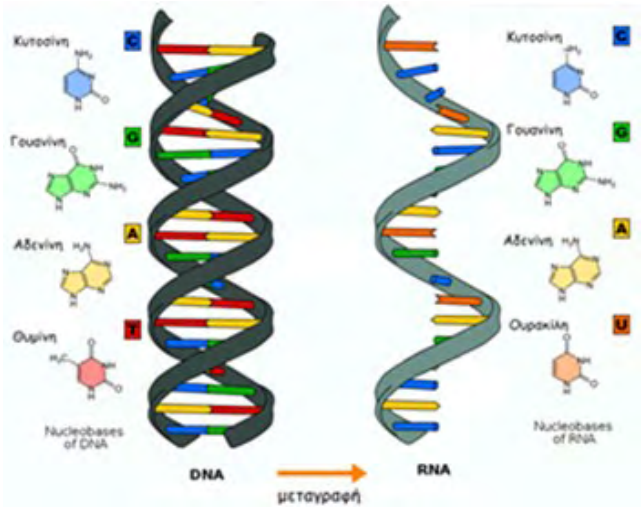
  (βλέπε παράρτημα 1 - κώδικας 2.1)
- *Απόκτηση ενός τμήματος μιας ακολουθίας*

  Τις περισσότερες φορές δεν επιθυμούμε να επεξεργαστούμε μια ολόκληρη ακολουθία. Για παράδειγμα θέλουμε μόνο τις 10 πρώτες βάσεις της ή κάποιες βάσεις ανάμεσα σε δύο σημεία που θα καθορίσουμε. Αυτά τα τμήματα τα παίρνουμε με τις κατάλληλες μεθόδους. Προσοχή η αρίθμηση των βάσεων στο Biojava ξεκινά από το 1 και όχι από το 0 όπως στις απλές συμβολοσειρές. Αν ζητήσουμε κάποιο τμήμα έξω από τα όρια, το αίτημά μας δεν θα ικανοποιηθεί.

  (βλέπε παράρτημα 1 - κώδικας 2.2)
- *Μεταγραφή DNA σε RNA*

  Η διαδικασία της μεταγραφής γίνεται εύκολα μέσω της μεθόδου `DNATools.toRNA()`. Παλαιότερα χρησιμοποιούνταν η μέθοδος `transcribe()`, αλλά μετά το Biojava 1.4 σταμάτησε να χρησιμοποιείται.

  (βλέπε παράρτημα 1 - κώδικας 2.3)



Εικόνα 6: Μεταγραφή DNA σε RNA

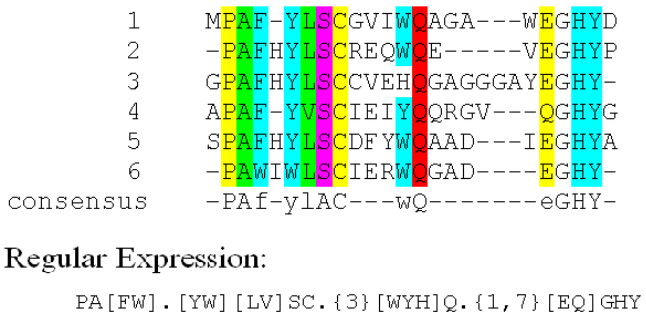
– *Εύρεση της συμπληρωματικής αλυσίδας*

Για να βρούμε το αντίστροφο συμπλήρωμα μιας DNA SymbolList ή Sequence θα πρέπει να χρησιμοποιήσουμε τη μέθοδο `DNATools.reverseComplement()`.
Αντίστοιχα αν θέλουμε να βρούμε το αντίστροφο συμπλήρωμα μιας RNA SymbolList ή Sequence καλούμε τη μέθοδο `RNATools.reverseComplement()`.
(βλέπε παράρτημα 1 - κώδικας 2.4)

Εικόνα 7: Εύρεση συμπληρωματικών αλυσίδων

– *Δημιουργία κανονικών εκφράσεων*

Μια από τις πιο ενδιαφέρουσες εργασίες που μπορεί κανείς να πραγματοποιήσει με τη βοήθεια του MotifTools είναι δημιουργία προτύπων (patterns) κανονικών εκφράσεων.
(Ένα απλό πρόγραμμα – παράδειγμα παρουσιάζεται στο παράρτημα 1 - κώδικας 2.5)



Εικόνα 8: Εξαγωγή κανονικής έκφρασης από πολλαπλή στοίχιση (Πηγή:[8])

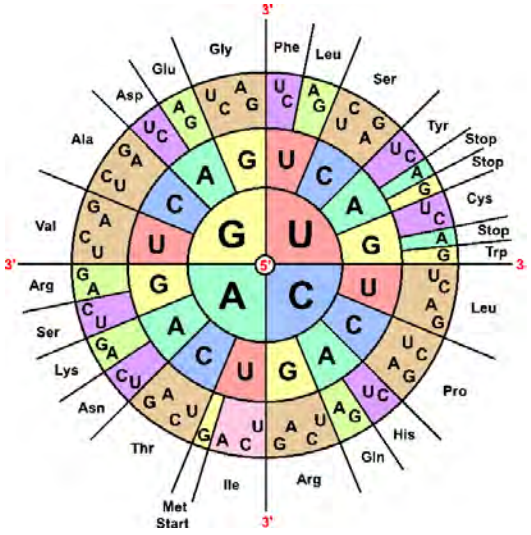
4.1.3 Μετάφραση

– *Μετάφραση DNA/ RNA SymbolList ή Sequence σε πρωτεΐνη*

Τα βήματα που πρέπει να ακολουθήσει κάποιος προκειμένου να κάνει μετάφραση μιας ακολουθίας DNA σε πρωτεΐνη είναι τα εξής:

1) Μεταγραφή σε RNA (προφανώς το βήμα αυτό παραλείπεται σε περίπτωση που θέλουμε να μεταφράσουμε ακολουθία RNA).
2) Χωρισμός της ακολουθίας σε τριπλέτες (κωδικόνια)
3) Μετάφραση σε πρωτεΐνη

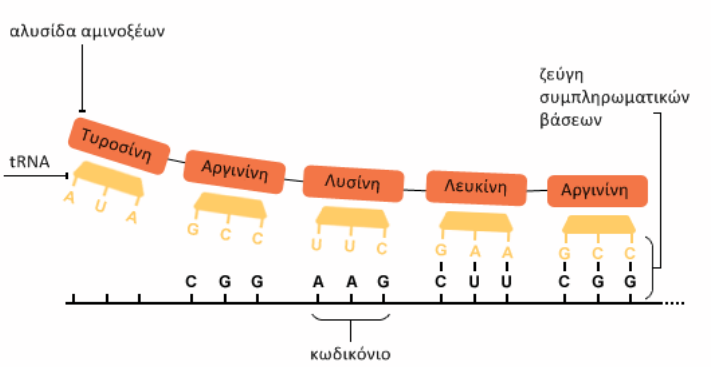
Η διαδικασία αυτή στο Biojava λαμβάνει χώρα με την απλή χρήση δύο βασικών μεθόδων – της `DNATools.toRNA()`για μεταγραφή και της `RNATools.translate()` για μετάφραση.
(βλέπε παράρτημα 1 - κώδικας 3.1)



Σχήμα 5: Πίνακας μετάφρασης κωδικονίων σε αμινοξέα (Πηγή:[5])

– *Μετάφραση ενός κωδικονίου σε ένα αμινοξύ*

Στην περίπτωση που κάποιος θέλει να μεταφράσει ένα μόνο κωδικόνιο σε αμινοξύ χρειάζεται ένα διαφορετικό σύνολο εντολών, το οποίο αν και εκ πρώτης όψεως δείχνει να είναι πιο πολύπλοκο, δίνει την ευκαιρία στο χρήστη να καταλάβει πως γίνεται η μετάφραση και να δει λεπτομέρειες που κρύβονται πίσω από την εύχρηστη μέθοδο RNATools.translate().(βλέπε παράρτημα 1 - κώδικας 3.2)



Εικόνα 9: Μετάφραση (φάση μετατροπής κωδικονίου σε αμινοξύ)

4.1.4 Είσοδοι / έξοδοι ακολουθιών

– *Γραφή – εκτύπωση ακολουθιών σε μορφή FASTA*
Η μορφή FASTA είναι μια πρότυπη, ευρέως χρησιμοποιούμενη, μορφή εξόδου (output) βιολογικών διεργασιών, που είναι αρκετά εύκολο να διαβαστεί. Εδώ, λοιπόν, παρουσιάζουμε παραδείγματα σχετικά με την εκτύπωση ακολουθιών ή ολόκληρων βάσεων δεδομένων ακολουθιών σε μορφή FASTA με ένα OutputStream όπως το απλό System.out.
(βλέπε παράρτημα 1 - κώδικας 4.1)

```
gi|50302071|gb|AY675564.1| Mus musculus molossinus mitochondrion,
omplete genome
ITAATGTAGCTTAATAACAAAGCAAAGCACTGAAAATGCTTAGATGGATAATTTTATCCCATAAACACA
AGGTTTGGTCCTGGCCTTATAATTAATTAGAGGTAAAGTTACACATGCAAACCTCCATAGACCGGTGTA
AATCCCTTAAACATTTGCTTAAACTTTAAGGAGAGGGTATCAAGCACATTAAAATAGCTTAAGACACCT
GCCTAGCCACACCCCCACGGGACTCAGCAGTGATAAATATTAAGCAATAAACGAAAGTTTGACTAAGTT
IACCTCTTAGGGTTGGTAAATTTCGTGCCAGCCACCGCGGTCATACGATTAACCCAAACTAATTATTTT
GGCGTAAAACGTGTCAACTATAAATAAATAAATAGAATTAAAATCCAACTTATATGTGAAAATTCATTG
IAGGACCTAAACTCAATAACGAAAGTAATTCTAATCATTTATAATACACGACAGCTAAGACCCAAACTG
GATTAGATACCCCACTATGCTTAGCCATAAACCTAAATAATTAAATTTAACAAAACTATTTGCCAGAGA
CTACTAGCCATAGCTTAAAACTCAAAGGACTTGGCGGTACTTTATATCCATCTAGAGGAGCCTGTTCTA
AATCGATAAACCCCGCTCTACCTCACCATCTCTTGCTAATTCAGCCTATATACCGCCATCTTCAGCAAA
CCTAAAAAGGTATTTAAGTAAGCAAAAGAATCAAACATAAAAACGTTAGGTCAAGGTGTAGCCAATGAA
IGGGAAGAAATGGGCTACATTTTCTTATAAAAGAACATTACTATATCCTTTATGAAACTAAAGGACTAA
GAGGATTTAGTAGTAAATTAAGAATAGAGAGCTTAATTGAATTGAGCAATGAAGTACGCACACACCGCC
```

Εικόνα 10: Αρχείο σε μορφή fasta

– *Ανάγνωση ακολουθιών από ένα αρχείο FASTA*
Μια από τις πιο συνηθισμένες διεργασίες εισόδου/εξόδου είναι η ανάγνωση ενός αρχείου που περιέχει κάποια ακολουθία και βρίσκεται στη μνήμη. Το SeqIOTools παρέχει κάποιες βασικές στατικές μεθόδους για την ανάγνωση αρχείων στο Biojava. Υπάρχουν περισσότεροι από ένας τρόποι.
(βλέπε παράρτημα 1 - κώδικας 4.2)
– *Εξαγωγή ακολουθιών από τις GenBank/ EMBL/ SwissProt και επανεγγραφή σε μορφή FASTA*
Όπως αναφέραμε και παραπάνω η πιο γνωστή, ίσως, μορφή βιολογικών δεδομένων είναι η μορφή fasta. Είναι, επομένως, χρήσιμο να υπάρχει ένα

εργαλείο που θα διαβάζει αρχεία από διάφορες βάσεις δεδομένων και θα εξάγει τις ακολουθίες που βρίσκονται σε αυτά σε μορφή fasta.
(βλέπε παράρτημα 1 - κώδικας 4.3)

4.1.5 Σχολιασμοί

- *Καταγραφή σχολίων σε μια ακολουθία*
  Όταν διαβάζουμε ένα αρχείο από την GenBank ή από την EMBL, πάντα, περιέχονται, εκτός από την ακολουθία, και διάφορες άλλες πληροφορίες. Οι πληροφορίες αυτές καταλήγουν είτε σαν απλό Feature είτε, αν είναι πιο γενικές, σαν Annotations.
  (βλέπε παράρτημα 1 - κώδικας 5.1)

4.1.6 Τοποθεσίες και features

- *Δημιουργία ενός feature*
  Στο Biojava ένα feature μοιάζει με σχολιασμό σε κάποιο συγκεκριμένο σημείο. Υπάρχουν διάφοροι τύποι features, οι οποίοι, όλοι μαζί, αποτελούν τη διεπαφή Feature. Όλες οι εφαρμογές feature περιέχουν μια εσωτερική κλάση που ονομάζεται "Template". Η κλάση αυτή καθορίζει την ελάχιστη ποσότητα πληροφορίας που χρειάζεται για τη δημιουργία ενός feature. Ένα feature υλοποιείται όταν ένα feature template δίνεται σαν παράμετρος στη μέθοδο createFeature της διεπαφής FeatureHolder.
  Η κλάση Sequence έχει σαν υποδιεπαφή την κλάση FeatureHolder οπότε μπορεί να κρατά features, ενώ, αντίθετα, μια SymbolList δε μπορεί. Επίσης και η κλάση Feature έχει σαν υποδιεπαφή τη FeatureHolder πράγμα το οποίο σημαίνει ότι υπάρχει η δυνατότητα να δημιουργηθούν εμφωλευμένα features.
  (βλέπε παράρτημα 1 - κώδικας 6.1)
- *Αφαίρεση ενός feature*
  Όταν επεξεργαζόμαστε ένα αντικείμενο Sequence μπορεί να θέλουμε να διαγράψουμε κάποια features. Στο πρόγραμμα – παράδειγμα παρουσιάζεται πώς διαγράφουμε όλα τα features που πληρούν ορισμένες προϋποθέσεις.
  (βλέπε παράρτημα 1 - κώδικας 6.2)

4.1.7 Μετρήσεις και κατανομές (counts and distributions)

- *Καταμέτρηση των βάσεων σε μια ακολουθία*
  Η καταμέτρηση των βάσεων σε μια ακολουθία είναι μια αρκετά συνηθισμένη εργασία. Γενικά χρειάζεται να δημιουργηθεί ένας πίνακας και να χρησιμοποιηθεί ένα οποιοδήποτε σύστημα που να κάνει καταλογογράφηση. Ακόμα καλύτερα μπορεί να χρησιμοποιηθεί ένα AlphabetIndex το οποίο θα δημιουργήσει ένα τυποποιημένο πίνακα/ευρετήριο. Το πρόγραμμα που παρουσιάζεται διαβάζει μια ακολουθία από κάποιο τύπο αρχείου, μετράει τις βάσεις και τυπώνει το αποτέλεσμα. Προσοχή: το παράδειγμα που παρουσιάζεται δεν χειρίζεται διφορούμενα σύμβολα – σε περίπτωση που

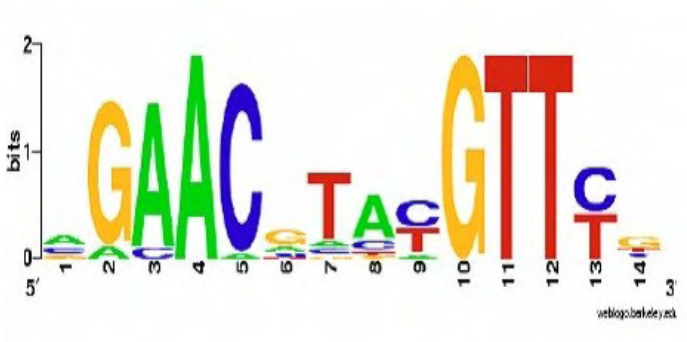
θέλουμε κάποια τέτοια εργασία θα πρέπει να προσθέσουμε ξεχωριστό μετρητή.
(βλέπε παράρτημα 1 - κώδικας 7.1)

- *Υπολογισμός της συχνότητας εμφάνισης ενός συμβόλου σε μια ακολουθία*
  Μία από τις πιο χρήσιμες κλάσεις στο Biojava είναι η Distribution (κατανομή). Μια κατανομή είναι μια αναλογία συμβόλων – συχνοτήτων. Οι κατανομές δημιουργούνται από τα Symbols που έχουν παρατηρηθεί με τη χρήση ενός DistributionTrainerContext, το οποίο μπορεί να διευθύνει διάφορες καταγεγραμμένες κατανομές και να χειριστεί οποιοδήποτε σύμβολο (Symbol) από οποιοδήποτε αλφάβητο (Alphabet).
  (βλέπε παράρτημα 1 - κώδικας 7.2)
- *Μετατροπή μιας μέτρησης σε κατανομή*
  Μια μέτρηση μπορεί απλά να μετατραπεί σε κατανομή με τη χρήση της μεθόδου `countToDistribution()` της κλάσης DistributionTools.
  (βλέπε παράρτημα 1 - κώδικας 7.3)
- *Δημιουργία τυχαίας ακολουθίας από μια κατανομή*
  Τα Distribution αντικείμενα του Biojava έχουν μια μέθοδο για να κάνουν δειγματοληψία συμβόλων. Με διαδοχική δειγματοληψία αρκετών συμβόλων μπορεί να δημιουργηθεί μια τυχαία ακολουθία. Επειδή αυτή είναι μια κοινή εργασία, προβλέπεται στη DistributionTools η στατική μέθοδος `generateSequence()`. Το πρόγραμμα που παρουσιάζεται παράγει τυχαία ακολουθία με ομοιόμορφη κατανομή των συμβόλων του DNA αλφαβήτου. Η παραγόμενη, κάθε φορά, ακολουθία θα διαφέρει από τις προηγούμενες αν και το ποσοστό εμφάνισης της κάθε βάσης θα είναι πάντα περίπου 25%. Μη ομοιόμορφη κατανομή μπορεί να χρησιμοποιηθεί για τη δημιουργία κάποιας επιθυμητής προδιαγεγραμμένης ακολουθίας.
  (βλέπε παράρτημα 1 - κώδικας 7.4)
- *Εύρεση του ποσού πληροφορίας ή της εντροπίας σε μια κατανομή*
  Η πληροφορία κατά Shannon επιστρέφεται σαν μια τιμή double και αντανακλά τη συνολική πληροφορία. Η εντροπία, από την άλλη πλευρά, επιστρέφεται σαν HashMap του κάθε συμβόλου και της αντίστοιχης εντροπίας του. Το παράδειγμα υπολογίζει και τα δύο για μια προδιαγεγραμμένη κατανομή.
  (βλέπε παράρτημα 1 - κώδικας 7.5)

4.1.8 Πίνακες βαρών και δυναμικός προγραμματισμός

- *Χρήση πίνακα βαρών για την εύρεση μοτίβου*
  Ένας πίνακας βαρών είναι ένας χρήσιμος τρόπος αναπαράστασης μιας στοίχισης ή ενός μοτίβου. Μπορεί επίσης να χρησιμοποιηθεί σαν ένας πίνακας βαθμολόγησης (scoring) για να ανιχνεύσει/διακρίνει ένα παρόμοιο μοτίβο σε μια ακολουθία. Το Biojava περιλαμβάνει μια κλάση με το όνομα WeightMatrix στο πακέτο (package) org.biojava.bio.dp. Υπάρχει επίσης και

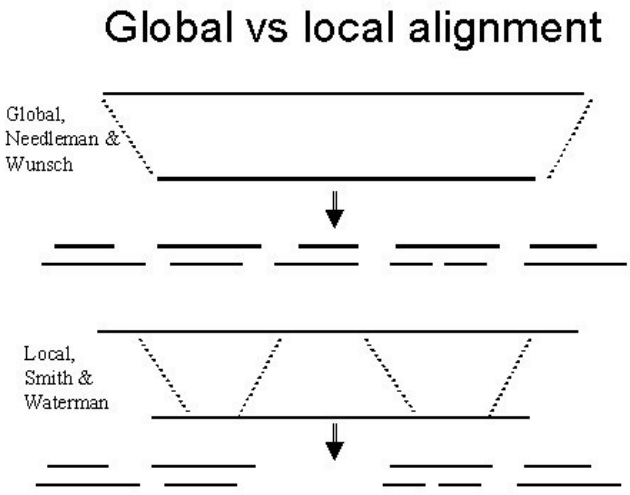
ένας WeightMatrixAnnotator που χρησιμοποιεί τον πίνακα βαρών για να προσθέσει Features σε οποιοδήποτε τμήμα της ακολουθίας που εξετάζεται το οποίο υπερβαίνει κάποιο συγκεκριμένο κατώφλι. Το πρόγραμμα – παράδειγμα παράγει ένα πίνακα βαρών από μια στοίχιση και τον χρησιμοποιεί για να προσθέσει σχόλια σε μια ακολουθία με κατώφλι 0,1.
(βλέπε παράρτημα 1 - κώδικας 8.1)



Εικόνα 11: Μοτίβο (Πηγή [1])

– *Ολική ή τοπική στοίχιση με βάση τον αλγόριθμο Needleman – Wunsch ή τον Smith – Waterman*

Μια συχνή προσέγγιση στην κατά ζεύγη στοίχιση ακολουθιών είναι η χρήση των αλγορίθμων Needleman – Wunsch και Smith – Waterman, για ολική και τοπική στοίχιση αντίστοιχα. Στο παράδειγμα παρουσιάζεται η υλοποίηση και των δύο αλγορίθμων.
(βλέπε παράρτημα 1 - κώδικας 8.2)



Σχήμα 6: Ολική και τοπική στοίχιση (Πηγή:[4])

4.2 Ολοκληρωμένα παραδείγματα - προβλήματα

4.2.1 Πρόβλημα 1

Ανάγνωση μιας ακολουθίας σε μορφή fasta και εύρεση του αντίστροφου συμπληρώματος.

Το πρόγραμμα παίρνει ως είσοδο ένα αρχείο σε μορφή fasta που περιέχει μία ή περισσότερες ακολουθίες, τις διαβάζει και επιστρέφει το αντίστροφο συμπλήρωμά τους. Συγκεκριμένα αντικαθιστά τα κατάλοιπα μιας ακολουθίας βάση της συμπληρωματικότητας των βάσεων (η Αδενίνη είναι συμπληρωματική της Θυμίνης (A-T) και η Γουανίνη συμπληρωματικής της Κυτοσίνης (G-C)).

Το παρόν πρόγραμμα μας δίνει, επίσης, στην αρχή, τη δυνατότητα παραμετροποίησης της μορφής fasta που θα έχουν η/οι ακολουθία/ες που θα παράγονται ως έξοδος. Μπορούμε, δηλαδή, με αυτό τον τρόπο, να επιλέξουμε εμείς ποιες ακριβώς θα είναι οι πληροφορίες που θα εμφανίζονται στην κεφαλίδα. Για παράδειγμα, μπορεί κάποιος να επιθυμεί την εμφάνιση μόνο του αριθμού πρόσβασης ή της περιγραφής ή οποιουδήποτε άλλου συνδυασμού στοιχείων μεταξύ αυτών που δίνονται σαν μέθοδοι στην κλάση FastaHeader. Στην παρούσα υλοποίηση έχουν απενεργοποιηθεί όλες οι επιλογές.
(βλέπε παράρτημα 3 – κώδικας 1)

4.2.2 Πρόβλημα 2

Παρουσίαση της δυνατότητας απευθείας πρόσβασης σε βάσεις δεδομένων.

Συγκεκριμένα πραγματοποιείται πρόσβαση στη βάση δεδομένων νουκλεοτιδικών αλληλουχιών Genbank και στη βάση δεδομένων πρωτεϊνικών αλληλουχιών Genpet και δίνονται παραδείγματα για αναζήτηση εμφάνισης μιας ακολουθίας τόσο με τον αριθμό πρόσβασης (accession number) αλλά και με τον αριθμό gi.
(βλέπε παράρτημα 3 – κώδικας 2)

4.2.3 Πρόβλημα 3

Ανάγνωση ενός αρχείου τύπου embl, εξαγωγή της ακολουθίας dna σε μορφή fasta, μεταγραφή σε rna και μετάφραση.

Το παρόν πρόγραμμα παίρνει ως είσοδο ένα αρχείο μορφής embl και εξάγει την ή τις ακολουθίες DNA, που εμπεριέχονται, σε μορφή fasta. Στη συνέχεια μετατρέπει την ακολουθία σε SymbolList η οποία και δίνεται είσοδος σε κατάλληλη μέθοδο του Biojava με την οποία λαμβάνει χώρα η μεταγραφή σε RNA. Τέλος, η ακολουθία που είναι αποτέλεσμα της μεταγραφής εξετάζεται για να βρεθούν τα κωδικόνια έναρξης (AUG) και λήξης (UAA ή UAG ή UGA), ώστε να πραγματοποιηθεί μετάφραση του κατάλληλου τμήματος σε πρωτεΐνες.

Δίνεται, επίσης, στην αρχή, τη δυνατότητα παραμετροποίη σης της μορφής fasta που θα έχουν η/οι ακολουθία/ες που θα παράγονται ως έξοδος.

(βλέπε παράρτημα 3 – κώδικας 3)

4.2.4 Πρόβλημα 4

Ανάγνωση ενός αρχείου τύπου embl, μέτρηση των βάσεων των ακολουθιών που περιέχει, εξαγωγή κατανομής/ών και δημιουργία τυχαίας ακολουθίας βάση της κατανομής αυτής.

Το πρόγραμμα που υλοποιήθηκε δέχεται τρεις παραμέτρους – η πρώτη με το όνομα του αρχείου που δίνεται ως είσοδος, η δεύτερη με τον τύπο του αρχείου και η τρίτη με το αλφάβητο που θα χρησιμοποιηθεί. Εντοπίζονται η/οι ακολουθία/ες που βρίσκονται στο αρχείο, και για καθεμιά από αυτές γίνεται καταμέτρηση των βάσεων, εξαγωγή της κατανομής και στη συνέχεια δημιουργία μιας τυχαίας συμβολοσειράς βάση αυτής.
(βλέπε παράρτημα 3 – κώδικας 4)

4.2.5 Πρόβλημα 5

Ολική στοίχιση δύο ακολουθιών.

Το παράδειγμα αυτό υλοποιεί ολική στοίχιση δύο ακολουθιών DNA σύμφωνα με τον αλγόριθμο Needleman – Wunsch. Το πρόγραμμα δέχεται τρεις εισόδους – η πρώτη με το όνομα του αρχείου που περιέχει την πρώτη, προς στοίχιση, ακολουθία, η δεύτερη με το όνομα του αρχείου που περιέχει τη δεύτερη, προς στοίχιση, ακολουθία και η τρίτη με τον πίνακα υποκαταστάσεων.
(βλέπε παράρτημα 3 – κώδικας 5)

4.3 Εκπαιδευτικό υλικό

Στα πλαίσια της ενότητας αυτής θα αναπτυχθεί το εκπαιδευτικό υλικό που θα ενσωματωθεί στον ηλεκτρονικό οδηγό χρήσης. Η ανάγκη να δοθεί εκπαιδευτικός χαρακτήρας στον οδηγό χρήσης οδήγησε στη δημιουργία ενοτήτων – ασκήσεων με αυξανόμενο δείκτη δυσκολίας, με σκοπό να βοηθήσουν όσους επιθυμούν να μάθουν να προγραμματίζουν με το Biojava.

4.3.1 Δομή – οργάνωση των ενοτήτων

Το εκπαιδευτικό υλικό αποτελείται από εννέα ενότητες οι οποίες διαθέτουν συγκεκριμένη δομή. Στην αρχή της καθεμιάς προσδιορίζεται ο στόχος, δηλαδή οι γνώσεις που θα πρέπει να αποκομίσει ο εκπαιδευόμενος μετά το τέλος της. Στη συνέχεια, παρουσιάζεται η βασική θεωρία που είναι απαραίτητη για την κατανόηση των ζητημάτων που πραγματεύονται κάθε φορά οι ασκήσεις και γίνεται επεξήγηση αλγορίθμων όπου αυτό κρίνεται απαραίτητο. Έπειτα παρέχεται βοήθεια, επισημαίνοντας ποια πακέτα πρέπει να χρησιμοποιηθούν για την επίλυση της εκάστοτε άσκησης. Τέλος, δίνεται η εκφώνηση της άσκησης ή του σετ ασκήσεων που περιέχονται στην ενότητα. Οι λύσεις των ασκήσεων δεν παρέχονται ξεχωριστά, ωστόσο, μπορούν όλες να εντοπιστούν σε κάποιο σημείο του οδηγού χρήσης, αφού

αποτελούν είτε υλοποιήσεις πρότυπων αλγορίθμων είτε ολοκληρωμένα παραδείγματα.

4.3.2 Παρουσίαση ενοτήτων

Παρακάτω παρουσιάζονται οι εννέα ενότητες, όπως ακριβώς απαντώνται και στον ηλεκτρονικό οδηγό χρήσης.

ΕΝΟΤΗΤΑ 1
Εισαγωγή στα αλφάβητα

Στόχος
Η εξοικείωση του εκπαιδευόμενου με τις έννοιες του συμβόλου και του αλφαβήτου που απαντώνται συχνά στους κώδικες του project. Θα πρέπει στο τέλος του εργαστηρίου να είναι σε θέση να δημιουργεί και να καλεί αλφάβητα.

Βασική θεωρία
Το DNA (δεσοξυ-ριβονουκλεϊκό οξύ) είναι το γενετικό υλικό του κυττάρου και περιλαμβάνει τα γονίδια που περιέχουν τις πληροφορίες για την κατασκευή όλων των απαραίτητων, για τον οργανισμό, πρωτεϊνών. Πρόκειται για ένα δίκλωνο μόριο του οποίου δομικά συστατικά είναι τα νουκλεοτίδια που αποτελούνται από μία αζωτούχο βάση Αδενίνη (A), Θυμίμη (T), Γουανίνη (G) και Κυτοσίνη (C), μια πεντόζη (δεσοξυριβόζη) και μια φωσφορική ομάδα. Κατά συνέπεια το αλφάβητο του DNA, δηλαδή τα γράμματα που αποτελούν την ακολουθία του είναι τα A, T, G, C.
Το RNA (ριβονουκλεϊκό οξύ) συνιστά ένα πολυμερές νουκλεϊκό οξύ, το οποίο είναι μονόκλωνο και δημιουργείται με εκμαγείο το μόριο DNA. Τα δομικά συστατικά του είναι τα νουκλεοτίδια που αποτελούνται από μία αζωτούχο βάση Αδενίνη (A), Ουρακίλη (U), Γουανίνη (G) και Κυτοσίνη (C), μια πεντόζη (ριβόζη) και μια φωσφορική ομάδα. Συνεπώς το αλφάβητο του RNA, δηλαδή τα γράμματα που αποτελούν την ακολουθία του είναι τα A, U, G, C.
Οι πρωτεΐνες είναι μεγάλα σύνθετα βιομόρια οι οποίες αποτελούν είτε δομικό συστατικό του κυττάρου είτε συνεργούν σε κάποια συγκεκριμένη λειτουργία. Αποτελούνται από αμινοξέα, τα οποία ενώνονται μεταξύ τους με πεπτιδικούς δεσμούς σχηματίζοντας μια γραμμική αλυσίδα. Οπότε, το πρωτεϊνικό αλφάβητο αποτελείται από 20 διαφορετικά γράμματα τα οποία αναπαριστούν αυτά τα αμινοξέα. Τα εν λόγω γράμματα είναι τα εξής: A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y.

Βοηθητικό υλικό για την επίλυση των ασκήσεων
Τα παραπάνω αλφάβητα δε χρειάζεται να δημιουργηθούν από το χρήστη – βρίσκονται ήδη στην κατάλληλη βιβλιοθήκη του Biojava.
Καλείστε να διερευνήσετε και να χρησιμοποιήσετε τα ακόλουθα πακέτα:

- `org.biojava.bio`
- `org.biojava.bio.seq`
- `org.biojava.bio.symbol`

Άσκηση 1
Δημιουργείστε ένα πρόγραμμα το οποίο θα καλεί ένα αλφάβητο (όποιο επιθυμείτε) με δύο διαφορετικούς τρόπους. Ο πρώτος τρόπος θα είναι με χρήση της κλάσης AlphabetManager και ο δεύτερος με χρήση της αντίστοιχης κλάσης Tools (ανάλογα με το αλφάβητο – DNATools, RNATools, ProteinTools).

Άσκηση 2
Δημιουργείστε το δικό σας αλφάβητο. Αρχικά θα πρέπει να δημιουργήσετε τα σύμβολα που θα περιέχει το αλφάβητό σας, να τα συγκεντρώσετε σε ένα HashSet και εν συνεχεία να δημιουργήσετε το αλφάβητο με το όνομα που επιθυμείτε.

Άσκηση 3
Με τη μέθοδο που ακολουθήσατε στη δεύτερη άσκηση δημιουργείστε ένα ακόμη αλφάβητο και συγκρίνετε τα προκειμένου να παρατηρήσετε αν είναι ίσα και τυπώστε το αποτέλεσμα στην οθόνη.

ΕΝΟΤΗΤΑ 2
Εισαγωγή στις ακολουθίες

Στόχος
Η εξοικείωση του εκπαιδευόμενου με την έννοια της ακολουθίας, που συνδέεται με όλα τα βασικά βιολογικά ζητήματα. Με το πέρας του εργαστηρίου θα πρέπει ο εκπαιδευόμενος να είναι σε θέση να δημιουργεί ακολουθίες από συμβολοσειρές και το αντίστροφο, καθώς επίσης και να εξάγει κάποιο τμήμα μιας ακολουθίας.

Βασική θεωρία
Το DNA, το RNA και οι πρωτεΐνες είναι όλα ακολουθίες. Μία ακολουθία θα μπορούσαμε να την περιγράψουμε σαν μια σειρά – εναλλαγή από γράμματα (του αντίστοιχου κάθε φορά αλφαβήτου – περισσότερες πληροφορίες για τα αλφάβητα βλέπε στην ενότητα 1).
Στις περισσότερες γλώσσες προγραμματισμού η αναπαράσταση βιολογικών πολυμερών γίνεται με συμβολοσειρές. Η έννοια της ακολουθίας ταυτίζεται με την έννοια της συμβολοσειράς, δηλαδή με την απλή είσοδο χαρακτήρων από το πληκτρολόγιο. Στο Biojava κάτι τέτοιο δεν ισχύει. Για την αναπαράσταση των πολυμερών χρησιμοποιεί τις έννοιες συμβολοσειρά (SymbolList) και ακολουθία (Sequence), οι οποίες διαφέρουν από τις απλές συμβολοσειρές, καθώς δημιουργούνται με τη βοήθεια αυτών.

Βοηθητικό υλικό για την επίλυση των ασκήσεων
Καλείστε να διερευνήσετε και να χρησιμοποιήσετε τα ακόλουθα πακέτα:

- `org.biojava.bio.seq`
- `org.biojava.bio.symbol`

Άσκηση 1
Δημιουργείστε τη συμβολοσειρά “ATGGGTCCATGA” και την ακολουθία “CGTAACGTTACTA”.

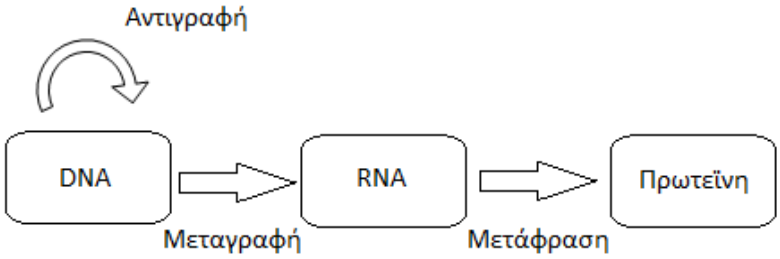
Άσκηση 2
Μετατρέψτε τη συμβολοσειρά της άσκησης 1 σε ακολουθία και σε συμβολοσειρά και εκτυπώστε την.

Άσκηση 3
Δημιουργείστε μια συμβολοσειρά RNA και πάρτε ένα τμήμα της. Για παράδειγμα πάρτε το πρώτο γράμμα της, μια τυχαία υπο-ακολουθία δίνοντας το πρώτο και τα δύο τελευταία γράμματα. Γενικά κάντε συνδυασμούς – δημιουργείστε κι άλλες συμβολοσειρές DNA ή πρωτεϊνικές και δοκιμάστε να ζητάτε και να εκτυπώνετε διάφορα τμήματά τους.

ΕΝΟΤΗΤΑ 3
Συμπληρωματικότητα και μεταγραφή

Στόχος
Κατανόηση των εννοιών της συμπληρωματικότητας των βάσεων και της μεταγραφής του DNA. Με την ολοκλήρωση της παρούσας άσκησης ο εκπαιδευόμενος θα πρέπει να μπορεί να εξάγει τη συμπληρωματική μιας αλυσίδας και να μεταγράφει μια ακολουθία DNA σε RNA.

Βασική θεωρία
Το DNA είναι δίκλωνο μόριο το οποίο αποτελείται από δύο πολυνουκλεοτιδικές αλυσίδες, οι οποίες ενώνονται μεταξύ τους με δεσμούς υδρογόνου σχηματίζοντας μια δεξιόστροφη έλικα. Ωστόσο ισχύει ο κανόνας της συμπληρωματικότητας των βάσεων, δηλαδή η κάθε βάση ενώνεται με την αντίστοιχη/συμπληρωματική της. Έτσι η Αδενίνη ενώνεται με τη Θυμίνη (με δύο δεσμούς υδρογόνου) και η Γουανίνη με την Κυτοσίνη (με τρεις δεσμούς υδρογόνου).
Στο κεντρικό δόγμα της βιολογίας το οποίο φαίνεται στο παρακάτω σχήμα, παρουσιάζεται η διαδικασία σύνθεσης πρωτεϊνών από το DNA με ενδιάμεσο σταθμό το RNA. Το πρώτο λοιπόν στάδιο είναι η μεταγραφή του DNA σε RNA. Κατά τη διαδικασία αυτή η διπλή αλυσίδα του DNA σπάει και με πρότυπο τη μία εξ αυτών δημιουργείται το RNA με βάση την συμπληρωματικότητα των βάσεων που αναφέρθηκε παραπάνω. Η μόνη διαφορά είναι πως η Αδενίνη δεν έχει ως συμπληρωματική τη Θυμίνη αλλά την Ουρακίλη.



Σχήμα 7: Κεντρικό δόγμα Μοριακής Βιολογίας

Βοηθητικό υλικό για την επίλυση των ασκήσεων
Οι κατάλληλες μέθοδοι που θα πρέπει να χρησιμοποιήσετε βρίσκονται στα παρακάτω πακέτα:

- `org.biojava.bio.seq`
- `org.biojava.bio.symbol`

Άσκηση 1
Δημιουργείστε μια SymbolList και βρείτε το συμπλήρωμα και το αντίστροφο συμπλήρωμά της.

Άσκηση 2
Μεταγράψτε την ακολουθία της πρώτης άσκησης σε RNA και εμφανίστε την στην οθόνη.

ΕΝΟΤΗΤΑ 4
Μετάφραση

Στόχος

Κατανόηση της διαδικασίας της μετάφρασης του RNA σε πρωτεϊνη. Μετά την ολοκλήρωση της άσκησης ο εκπαιδευόμενος θα πρέπει να είναι σε θέση να δημιουργεί ολοκληρωμένο πρόγραμμα που θα εκτελεί τις διαδικασίες που απαρτίζουν το κεντρικό δόγμα της Μοριακής Βιολογίας.

Βασική θεωρία

Μετάφραση είναι η διαδικασία σύνθεσης πρωτεϊνών βάσει του RNA, οι οποίες είναι υπεύθυνες για τη δομή και τη λειτουργία των κυττάρων και κατ' επέκταση των οργανισμών. Ο γενετικός κώδικας είναι το σύνολο των κανόνων σύμφωνα με τους οποίους η γενετική πληροφορία που περιέχεται στο γενετικό υλικό (DNA ή mRNA) μεταφράζεται σε πρωτεϊνες, δηλαδή σε αλληλουχία αμινοξέων από τα κύτταρα όλων των οργανισμών στη γη. Συγκεκριμένα ο γενετικός κώδικας αντιστοιχίζει ένα αμινοξύ σε μια τριπλέτα νουκλεοτιδίων του DNA με ενδιάμεσο μόριο μια συμπληρωματική αλληλουχία RNA.

Το τμήμα της αλληλουχίας βάσεων ενός γονιδίου και του mRNA που κωδικοποιεί ένα πεπτίδιο αρχίζει με το κωδικόνιο έναρξης (AUG) και τελειώνει με ένα κωδικόνιο λήξης (UGA, UAG, UAA). Ο όρος κωδικόνιο χρησιμοποιείται στο mRNA για τις τριάδες βάσεων των νουκλεοτιδίων του που μεταφράζονται.

Για την καλύτερη κατανόηση παρατίθεται πίνακας με τα κωδικόνια μεταφρασμένα σε αμινοξέα.

First base	Second base: U	Second base: C	Second base: A	Second base: G	Third base
U	UUU Phe	UCU Ser	UAU Tyr	UGU Cys	U
	UUC Phe	UCC Ser	UAC Tyr	UGC Cys	C
	UUA Leu	UCA Ser	UAA Stop	UGA Stop	A
	UUG Leu	UCG Ser	UAG Stop	UGG Trp	G
C	CUU Leu	CCU Pro	CAU His	CGU Arg	U
	CUC Leu	CCC Pro	CAC His	CGC Arg	C
	CUA Leu	CCA Pro	CAA Gln	CGA Arg	A
	CUG Leu	CCG Pro	CAG Gln	CGG Arg	G
A	AUU Ile	ACU Thr	AAU Asn	AGU Ser	U
	AUC Ile	ACC Thr	AAC Asn	AGC Ser	C
	AUA Ile	ACA Thr	AAA Lys	AGA Arg	A
	AUG Met	ACG Thr	AAG Lys	AGG Arg	G
G	GUU Val	GCU Ala	GAU Asp	GGU Gly	U
	GUC Val	GCC Ala	GAC Asp	GGC Gly	C
	GUA Val	GCA Ala	GAA Glu	GGA Gly	A
	GUG Val	GCG Ala	GAG Glu	GGG Gly	G

Σχήμα 8: Σχηματισμός αμινοξέων από κωδικόνια

Βοηθητικό υλικό για την επίλυση των ασκήσεων

Οι κατάλληλες μέθοδοι που θα πρέπει να χρησιμοποιήσετε βρίσκονται στα παρακάτω πακέτα:

- `org.biojava.bio.seq`
- `org.biojava.bio.symbol`
- `org.biojavax.bio.seq.io`

Άσκηση 1

Δημιουργείστε μια ακολουθία RNA, εκτελέστε τη διαδικασία της μετάφρασης και εμφανίστε το αποτέλεσμα στην οθόνη.

Άσκηση 2

Δημιουργείστε μια ακολουθία DNA, μεταγράψτε τη σε RNA και στη συνέχεια μεταφράστε τη. Προσοχή: θα πρέπει να πληρούνται οι προϋποθέσεις για τη μετάφραση – δηλαδή να ξεκινάει με κωδικόνιο έναρξης και να τελειώνει σε κωδικόνιο λήξης.

ΕΝΟΤΗΤΑ 5
Βάσεις δεδομένων

Στόχος
Να γνωρίσει ο εκπαιδευόμενος τις βάσεις δεδομένων, ώστε να είναι σε θέση να κάνει αναζητήσεις. Με το πέρας της άσκησης θα πρέπει να έχει αποκτήσει την ικανότητα να δημιουργεί πρόγραμμα που θα «κατεβάζει» την ακολουθία που επιθυμεί κατευθείαν από τη βάση δεδομένων.

Βασική θεωρία
Οι βάσεις δεδομένων που χρησιμοποιούνται στον τομέα της Βιοπληροφορικής μπορούν να χωριστούν στις εξής τρεις βασικές κατηγορίες:

- Βάσεις δεδομένων πρωτεϊνικών ακολουθιών
  - Swissprot (Ευρώπη) - www.expasy.ch/sprot
  - Protein Information Resource (PIR) - pir.georgetown.edu
  - Prosite - www.expasy.ch/prosite
- Βάσεις δεδομένων νουκλεοτιδικών ακολουθιών
  - EMBL (Ευρώπη) - www.ebi.ac.uk/embl
  - Genbank (Αμερική) - www.ncbi.nlm.nih.gov/Genbank/index.html
  - DDJB (Ιαπωνία) - www.ddbj.nig.ac.jp
- Βάσεις δεδομένων Δομικής Βιολογίας
  - Protein Data Bank (PDB) - www.rcsb.org/pdb/home/home.do
  - CATH - www.cathdb.info
  - SCOP - scop.mrc-lmb.cam.ac.uk/scop/index.html

Σημείωση: Θα μπορούσατε για καλύτερη κατανόηση να διερευνήσετε τις παραπάνω βάσεις και να δοκιμάσετε να «κατεβάσετε» αρχεία σε διάφορες μορφές.

Βοηθητικό υλικό για την επίλυση των ασκήσεων
Οι εντολές που απαιτούνται για το άνοιγμα των αρχείων δε διαφέρουν σε τίποτα από τις κλασικές εντολές της Java. Ωστόσο, για την απευθείας πρόσβαση και απόκτηση ακολουθιών από τις βάσεις δεδομένων, θα πρέπει να ενασχοληθείτε και να μάθετε να χρησιμοποιείτε και τα παρακάτω πακέτα:

- `org.biojava.bio.symbol.SymbolList`
- `org.biojavax.bio.db.ncbi.GenbankRichSequenceDB`
- `org.biojavax.bio.db.ncbi.GenpeptRichSequenceDB`
- `org.biojavax.bio.seq.RichSequence`

Άσκηση 1
Πραγματοποιείστε πρόσβαση στη βάση δεδομένων νουκλεοτιδικών αλληλουχιών Genbank και αναζητήστε μια ακολουθία τόσο με τον αριθμό πρόσβασης (accession number) αλλά και με τον αριθμό gi. Το αποτέλεσμα θα πρέπει να εμφανίζεται στη

οθόνη (μπορείτε να ψάξετε την ακολουθία με Accession number M98343 και gi 182086)

Άσκηση 2

Πραγματοποιείστε πρόσβαση στη βάση δεδομένων νουκλεοτιδικών αλληλουχιών Genpept και αναζητήστε μια ακολουθία τόσο με τον αριθμό πρόσβασης (accession number) αλλά και με τον αριθμό gi. Το αποτέλεσμα θα πρέπει να εμφανίζεται στη οθόνη (μπορείτε να ψάξετε την ακολουθία με Accession number AAD51968 και gi 5805369)

ΕΝΟΤΗΤΑ 6
Αρχεία

Στόχος
Μετά το τέλος της παρούσας άσκησης θα πρέπει ο εκπαιδευόμενος να έχει κατανοήσει τη χρήση αρχείων και συγκεκριμένα το άνοιγμα αρχείων για την ανάγνωση ακολουθιών και τη μετατροπή της μιας μορφής αρχείου σε άλλη.

Βασική θεωρία

Όλες οι διεργασίες που παρουσιάστηκαν παραπάνω μπορούν να επεκταθούν και να γενικευτούν χρησιμοποιώντας αρχεία από τα οποία εξάγονται οι ακολουθίες. Τα αρχεία αυτά συνήθως είναι εξαγωγές από βάσεις δεδομένων νουκλεοτιδικών ή πρωτεϊνικών αλληλουχιών και εκτός από τις ακολουθίες περιέχουν και άλλα χρήσιμα σχόλια.

Για να υποστεί όμως μια ακολουθία επεξεργασία θα πρέπει να μετατραπεί σε μια μορφή κατάλληλη προς ανάγνωση απαλλαγμένη από τις περιττές πληροφορίες. Η πιο συνηθισμένη μορφή που χρησιμοποιείται στη Βιοπληροφορική είναι η μορφή fasta. Παρακάτω παρουσιάζονται παραδείγματα αρχείων, ένα της μορφής embl (το ονομα προέρχεται από την ομώνυμη βάση) και ένα της μορφής fasta.

```
ID   AB000263 standard; RNA; PRI; 368 BP.
XX
AC   AB000263;
XX
DE   Homo sapiens mRNA for prepro cortistatin like peptide, complete cds.
XX
SQ   Sequence 368 BP;
     acaagatgcc attgtccccc ggcctcctgc tgctgctgct ctccggggcc acggccaccg       60
     ctgccctgcc cctggagggt ggccccaccg gccgagacag cgagcatatg caggaagcgg      120
     caggaataag gaaaagcagc ctcctgactt tcctcgcttg gtggtttgag tggacctccc      180
     aggccagtgc cgggcccctc ataggagagg aagctcggga ggtggccagg cggcaggaag      240
     gcgcacccca ccagcaatcc gcgcgccggg acagaatgcc ctgcaggaac ttcttctgga      300
     agacctttct ctcctgcaaa taaaacctca cccatgaatg ctcacgcaag tttaattaca      360
     gacctgaa                                                               368
//
```

Εικόνα 12: Αρχείο της μορφής embl

```
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA for prepro cortistatin like peptide, complete cds.|len=368
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCC
CCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGC
CTCCTGACTTTCCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCTCATAGGAGAGG
AAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCCAGCAATCCGCGCGCCGGGACAGAATGCC
CTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAG
TTTAATTACAGACCTGAA
```

Εικόνα 13:Αρχείο της μορφής fasta

Βοηθητικό υλικό για την επίλυση ασκήσεων

Το άνοιγμα των αρχείων πραγματοποιείται με τον ίδιο τρόπο που γίνεται και στη γλώσσα προγραμματισμού Java. Ωστόσο στο Biojava υπάρχουν ειδικές μέθοδοι για την ανάγνωση, γραφή και μετατροπή μιας μορφής αρχείου σε άλλη. Αναλυτικότερες πληροφορίες μπορείτε να βρείτε μελετώντας τα πακέτα που προτάθηκαν και στις προηγούμενες ενότητες. Εκτός από τα παραπάνω πακέτα θα χρειαστείτε και τα παρακάτω πακέτα εκ των οποίων το δεύτερο είναι από την επέκταση BiojavaX:

- `org.biojava.bio.seq.db`
- `org.biojavax.bio.seq`

Άσκηση 1
Δημιουργείστε δύο διαφορετικές ακολουθίες με τα ονόματα της αρεσκείας σας, τοποθετείστε τες σε μία βάση δεδομένων και στη συνέχεια εξάγετέ τες σε μορφή fasta.

Άσκηση 2
«Κατεβάστε» από τον ομώνυμο ιστοχώρο της βάσης δεδομένων ένα αρχείο της μορφής embl και αποθηκεύστε τον στον υπολογιστή σας. Στη συνέχεια δημιουργείστε πρόγραμμα Biojava, το οποίο θα ανοίγει το αρχείο (το όνομα του αρχείου θα δηλώνεται ως παράμετρος και όχι μέσα στον κώδικα) και στη συνέχεια μετατρέψτε και εμφανίστε το σε μορφή fasta.

ΕΝΟΤΗΤΑ 7
Μετρήσεις και κατανομές

Στόχος

Σκοπός της παρούσας άσκησης είναι η εξοικείωση με την έννοια της κατανομής. Ο εκπαιδευόμενος θα πρέπει να μάθει να χρησιμοποιεί τις κατάλληλες μεθόδους του Biojava για να μετρά τις βάσεις μιας ακολουθίας ανά κατηγορία, να βρίσκει τη συχνότητα εμφάνισής τους και να δημιουργεί την κατανομή τους.

Βασική θεωρία

Η κατανομή αποτελεί τον τρόπο περιγραφής των τυχαίων μεταβλητών (στην περίπτωση μας συμβόλων).

Η πληροφορία είναι ένα μέγεθος το οποίο μετρά την αβεβαιότητα να παρουσιαστεί κάποιο γεγονός όταν ένα πείραμα επαναλαμβάνεται. Όσο μεγαλύτερη είναι η αβεβαιότητα τόσο περισσότερη είναι η πληροφορία που λαμβάνεται, δηλαδή όταν η πιθανότητα εμφάνισης μιας τυχαίας μεταβλητής (στην περίπτωσή μας ενός συμβόλου) μεγιστοποιείται ή ελαχιστοποιείται η ποσότητα της πληροφορίας μικραίνει.

Η εντροπία συνιστά ένα μέγεθος το οποίο δηλώνει το μέτρο της τυχαιότητας σε ένα σύστημα. Ειδικότερα, με τον όρο εντροπία αναφερόμαστε στη μέση πληροφορία.

Βοηθητικό υλικό για την επίλυση ασκήσεων

Καλείστε να χρησιμοποιήσετε μεθόδους από τα παρακάτω πακέτα:

- `org.biojava.bio.dist`
- `org.biojava.bio.seq`
- `org.biojava.bio.seq.io`
- `org.biojava.bio.symbol`
- `org.biojava.utils`

Άσκηση 1

Δημιουργείστε μια ακολουθία της αρεσκείας σας τουλάχιστον δέκα βάσεων και μετρείστε πόσες συναντώνται από κάθε είδος. Στη συνέχεια υπολογίστε το ποσοστό εμφάνισής καθεμιάς και εμφανίστε το αποτέλεσμα στην οθόνη.

Άσκηση 2

Βρείτε και αποθηκεύστε στον υπολογιστή σας μια ακολουθία μορφής fasta. Πραγματοποιείστε καταμέτρηση των βάσεων, όπως και στην προηγούμενη άσκηση, και βάση αυτής δημιουργείστε την αντίστοιχη κατανομή. Τέλος, βάση της κατανομής δημιουργείστε μια τυχαία ακολουθία. Η μέτρηση, η κατανομή και η τυχαία ακολουθία θα εμφανίζονται στην οθόνη.

Άσκηση 3

Δημιουργείστε μια κατανομή της αρεσκείας σας και ορίστε το βάρος του κάθε συμβόλου. Υπολογίστε το ποσό πληροφορίας και την εντροπία για καθένα από τα σύμβολα και εμφανίστε τα αποτελέσματα στην οθόνη.

ΕΝΟΤΗΤΑ 8
Εισαγωγή στη στοίχιση ακολουθιών
Ολική στοίχιση – αλγόριθμος Needleman Wunsch

Στόχος
Ο εκπαιδευόμενος στα πλαίσια της παρούσας ενότητας θα πρέπει να κατανοήσει την έννοια της στοίχισης των ακολουθιών με τη χρήση του αλγορίθμου Needleman - Wunsch και να είναι σε θέση να εκτελεί τέτοιου είδους στοιχίσεις.

Βασική θεωρία

Η στοίχιση ακολουθιών (sequence alignment) είναι μια διαδικασία κατά την οποία, δύο ή περισσότερες ακολουθίες (DNA, RNA ή πρωτεϊνικές) τοποθετούνται η μία κάτω από την άλλη με τέτοιο τρόπο έτσι ώστε τα κοινά τους σύμβολα να είναι τοποθετημένα στην ίδια θέση. Σκοπός είναι η βέλτιστη στοίχιση. Είναι μια δύσκολη διαδικασία και έχουν αναπτυχθεί αρκετοί αλγόριθμοι τόσο για την κατά ζεύγη (ολική ή τοπική στοίχιση) όσο και για την πολλαπλή στοίχιση. Η χρησιμότητά της είναι ιδιαίτερη, καθώς ομοιότητες μεταξύ των ακολουθιών συνήθως υποδηλώνουν κοινή εξελικτική προέλευση ή κοινή λειτουργία, οπότε είμαστε σε θέση ακόμη και να ταυτοποιήσουμε μια άγνωστη πρωτεΐνη.

Η κατά ζεύγη στοίχιση μπορεί να πραγματοποιηθεί με δύο τρόπους, είτε να εφαρμοστεί κατά μήκος όλης της ακολουθίας είτε σε ένα τμήμα της.

Το πρώτο από αυτά τα είδη στοίχισης ονομάζεται ολική (global). Ο πιο διαδεδομένος αλγόριθμος που χρησιμοποιείται για τέτοιου είδους στοιχίσεις είναι ο Needleman – Wunsch. Η βασική ιδέα είναι η χρήση ενός δισδιάστατου πίνακα σκορ του οποίου η μία διάσταση αναφέρεται στα σύμβολα της μιας ακολουθίας και η άλλη διάσταση στα σύμβολα της δεύτερης. Αρχικά, λοιπόν, τοποθετούνται οι τιμές των σκορς στην πρώτη γραμμή και στην πρώτη στήλη και στη συνέχεια γεμίζουν και τα υπόλοιπα κελιά του πίνακα βάση των τιμών που έχουν οι γείτονες. Τέλος ανατρέχουμε τον πίνακα προς τα πίσω για να βρούμε την επιθυμητή στοίχιση.

Βοηθητικό υλικό για την επίλυση ασκήσεων

Όλα τα πακέτα που θα χρειαστείτε για την εκτέλεση της στοίχισης είναι τα παρακάτω:

- `org.biojava.bio.alignment.NeedlemanWunsch`
- `org.biojava.bio.alignment.SequenceAlignment`
- `org.biojava.bio.alignment.SubstitutionMatrix`
- `org.biojava.bio.seq.DNATools`
- `org.biojava.bio.seq.Sequence`
- `org.biojava.bio.symbol.AlphabetManager`
- `org.biojava.bio.symbol.finitealphabet`

Άσκηση 1

Δημιουργείστε δύο ακολουθίες DNA και πραγματοποιείστε ολική στοίχιση. Για καλύτερη εμπέδωση επαναλάβετε το ίδιο για δύο ακολουθίες RNA καθώς και για δύο πρωτεϊνικές. Σε κάθε περίπτωση τα αποτελέσματα θα πρέπει να εμφανίζονται στην οθόνη.

Άσκηση 2

Δημιουργείστε δύο αρχεία που περιέχουν ακολουθίες σε μορφή fasta. Πραγματοποιείστε ολική στοίχιση και εμφανίστε τα αποτελέσματα.

ΕΝΟΤΗΤΑ 9
Τοπική στοίχιση – αλγόριθμος Smith Waterman

Στόχος
Ο εκπαιδευόμενος στα πλαίσια της παρούσας ενότητας θα πρέπει να κατανοήσει την έννοια της στοίχισης των ακολουθιών με τη χρήση του αλγορίθμου Smith – Waterman και να είναι σε θέση να εκτελεί τέτοιου είδους στοιχίσεις.

Βασική θεωρία
Όσον αφορά την τοπική στοίχιση, σε αυτή την περίπτωση εφαρμόζουμε τον αλγόριθμο Smith – Waterman.Τα βήματα που ακολουθούνται είναι περίπου τα ίδια με αυτά που περιγράψαμε στον προηγούμενο αλγόριθμο, δηλαδή η αρχικοποίηση (σε αυτή την περίπτωση όμως η πρώτη γραμμή και η πρώτη στήλη αρχικοποιούνται με την τιμή μηδέν), στη συνέχεια γέμισμα του πίνακα με χρήση της κατάλληλης σχέσης και τέλος σάρωση του πίνακα προς τα πίσω ξεκινώντας από το σημείο που εμφανίζεται το μεγαλύτερο σκορ του πίνακα και τελειώνει στο κελί με το προκαθορισμένο ελάχιστο κατώφλι.

Βοηθητικό υλικό για την επίλυση ασκήσεων
Όλα τα πακέτα που θα χρειαστείτε για την εκτέλεση της στοίχισης είναι τα παρακάτω:

- `org.biojava.bio.alignment.SmithWaterman`
- `org.biojava.bio.alignment.SequenceAlignment`
- `org.biojava.bio.alignment.SubstitutionMatrix`
- `org.biojava.bio.seq.DNATools`
- `org.biojava.bio.seq.Sequence`
- `org.biojava.bio.symbol.AlphabetManager`
- `org.biojava.bio.symbol.finitealphabet`

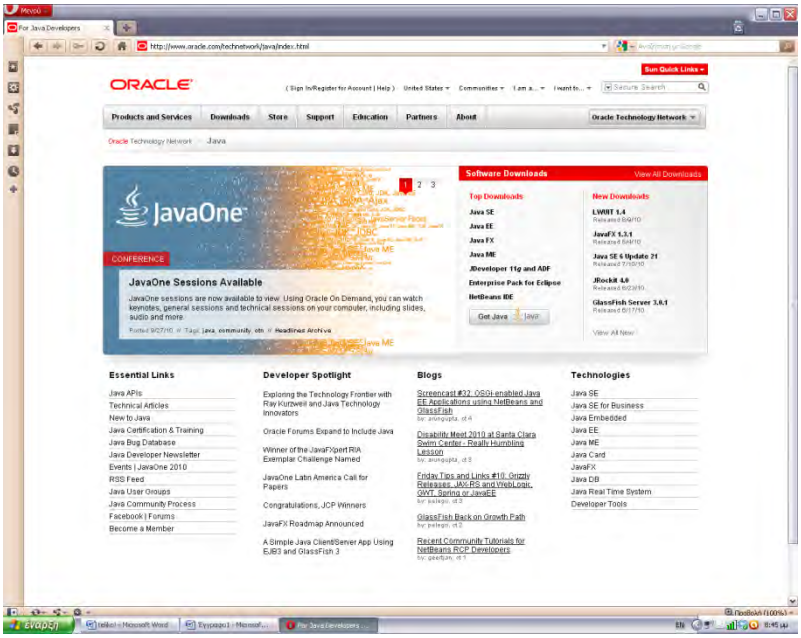
Άσκηση 1
Δημιουργείστε δύο ακολουθίες DNA και πραγματοποιείστε τοπική στοίχιση. Για καλύτερη εμπέδωση επαναλάβετε το ίδιο για δύο ακολουθίες RNA καθώς και για δύο πρωτεϊνικές. Σε κάθε περίπτωση τα αποτελέσματα θα πρέπει να εμφανίζονται στην οθόνη.

Άσκηση 2
Δημιουργείστε δύο αρχεία που περιέχουν ακολουθίες σε μορφή fasta. Πραγματοποιείστε τοπική στοίχιση και εμφανίστε τα αποτελέσματα.

4.4 Οδηγίες Εγκατάστασης

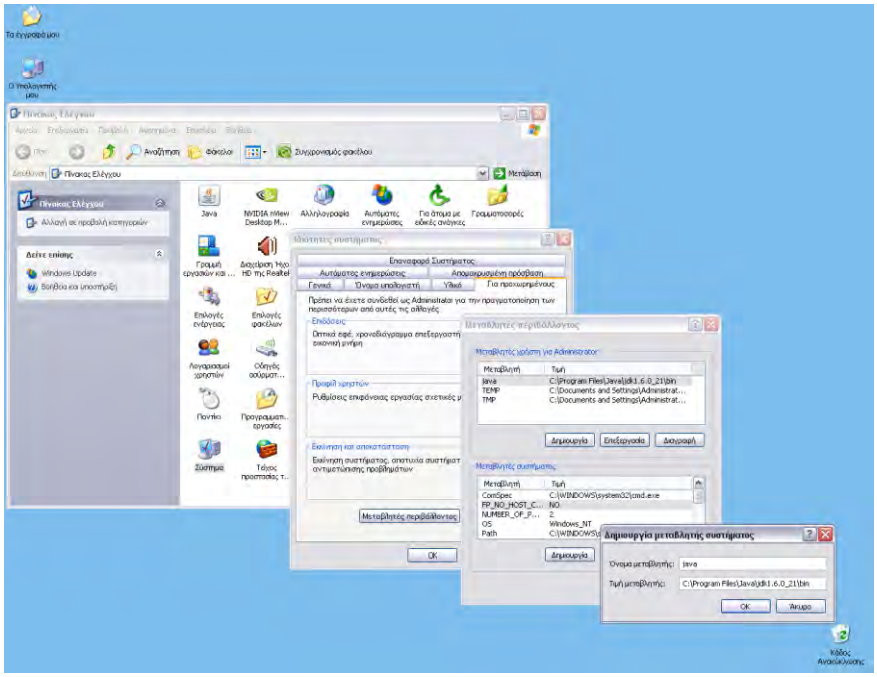
Παρακάτω παρουσιάζονται ένα ένα τα βήματα εγκατάστασης των Java, Netbeans και Biojava στα Windows. Ωστόσο, οδηγίες βρίσκονται και στον επίσημο ιστοχώρο του Biojava [79] στο "How to integrate BioJava in NetBeans IDE".

1) Μετάβαση στο επίσημο ιστοχώρο της Java [80] και απόκτηση του Java JDK (όχι του JRE).
   Προσοχή: το Biojava δεν θα τρέξει εάν δεν πληρούνται οι προδιαγραφές της Java 2 Standard Edition (J2SE) 1.4 έκδοσης ή κάποιας πιο πρόσφατης.



Εικόνα 14: Αρχική σελίδα του ιστοχώρου java.sun.com

2) Πριν το ξεκίνημα της εγκατάστασης θα ήταν χρήσιμο να γίνει έλεγχος αν το παραπάνω αρχείο έχει το προβλεπόμενο μέγεθος. Εάν είναι πλήρες γίνεται μετάβαση στο βήμα 3, ειδάλλως πρέπει να γίνει καινούρια προσπάθεια απόκτησης του αρχείου.
3) Η εγκατάσταση ξεκινά με «διπλό κλικ» στο αρχείο exe.
4) Αφού έχει τελειώσει η εγκατάσταση είναι χρήσιμο να οριστεί η μεταβλητή path. Η διαδικασία που πρέπει να ακολουθηθεί είναι η εξής:
   - Έναρξη → Πίνακας ελέγχου → Σύστημα → Για προχωρημένους → Μεταβλητές περιβάλλοντος (Start → Control Panel → System →Advanced → Environment Variables).
   - Προσθήκη του μονοπατιού που οδηγεί στο bin φάκελο του JDK και στις μεταβλητές συστήματος και στις μεταβλητές χρήστη. Συνήθως το μονοπάτι έχει τη μορφή: C:\Program Files\Java\jdk1.6.0_<version>\bin.

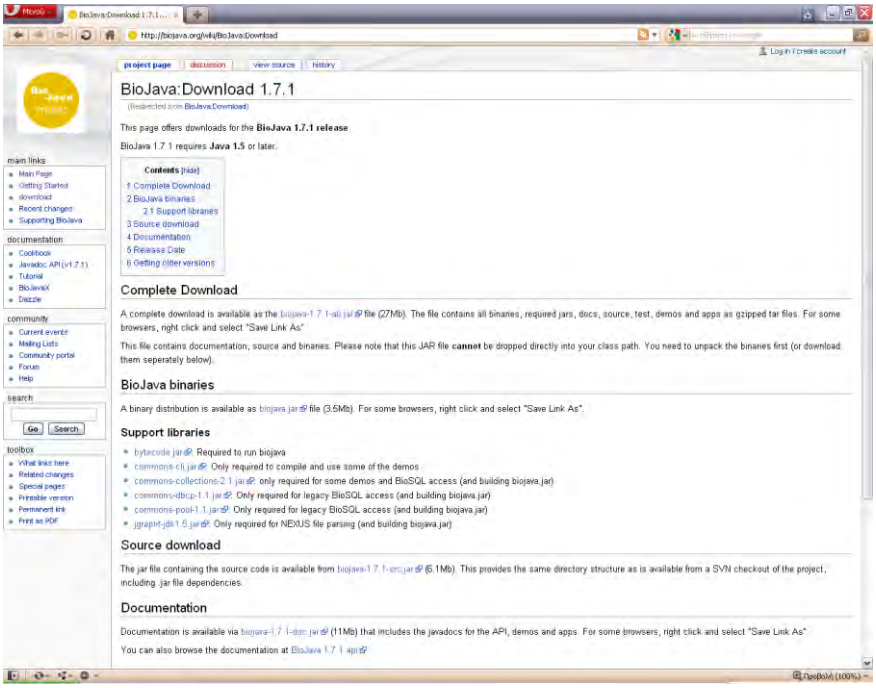


Εικόνα 15: Διαδικασία ορισμού μεταβλητής path

5) Στη συνέχεια προτείνεται μετάβαση στον επίσημο ιστοχώρο του Netbeans [81] και απόκτηση του Netbeans 6.9 (είναι η νεότερη έκδοση που είναι προς το παρόν διαθέσιμη). Επιλογή της γλώσσας και του λειτουργικού συστήματος (στο πάνω αριστερό μέρος της σελίδας) και κάποιου από τα πακέτα (bundles) που έχουν σχέση με τη Java (το Java είναι το πιο πλήρες).
Προσοχή: το Biojava δουλεύει καλύτερα εάν έχει γίνει εγκατάσταση κάποιας έκδοση μετά την 5.0.

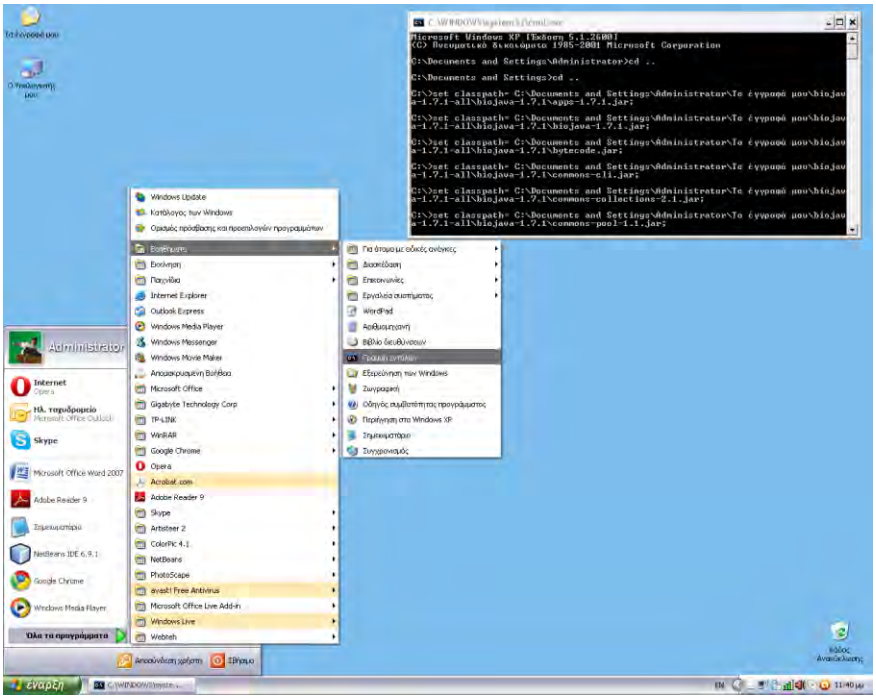
Εικόνα 16: Αρχική σελίδα του ιστοχώρου netbeans.org

6) Η εγκατάσταση ξεκινά με «διπλό κλικ» στο αρχείο.
7) Πλέον, το μόνο που έχει μείνει είναι η απόκτηση του Biojava από τον επίσημο ιστοχώρο του [79] στο Biojava: Download. Υπάρχει δυνατότητα επιλογής απόκτησης είτε των επιμέρους τμημάτων είτε του συνολικού αρχείου, για μεγαλύτερη ευκολία. Σε περίπτωση επιλογής του δεύτερου τρόπου είναι απαραίτητη η αποσυμπίεση.



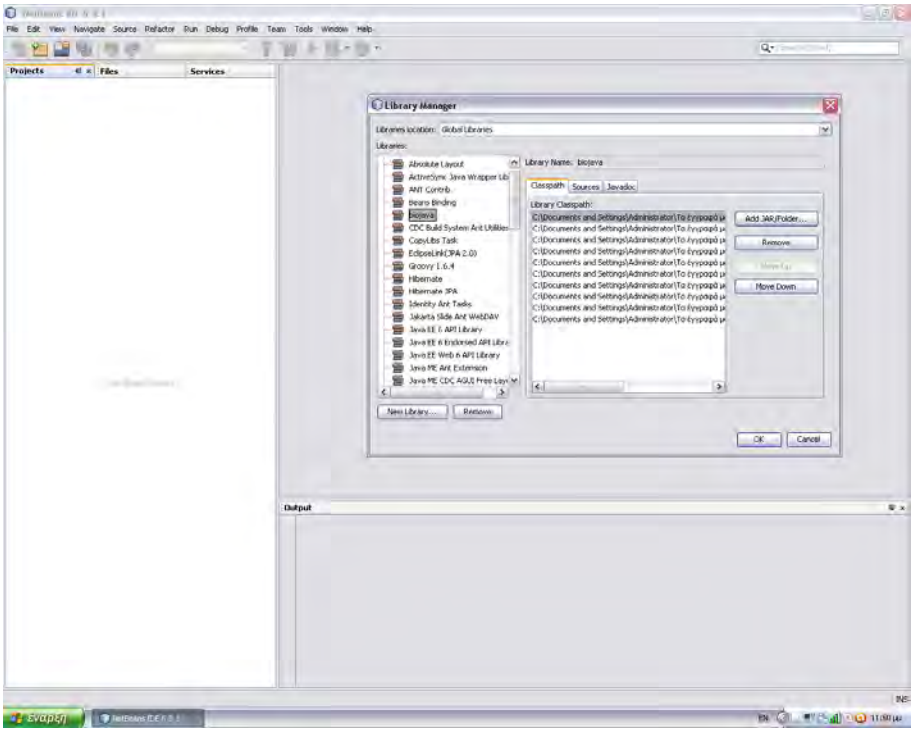
Εικόνα 17: Σελίδα από το biojava.org για την απόκτηση του biojava

8) Σε αυτό το σημείο θα πρέπει να γίνει δήλωση του μονοπατιού που οδηγεί στο φάκελο – αρχείο του βήματος 7. Συγκεκριμένα θα πρέπει να χρησιμοποιηθεί μια εντολή για κάθε jar αρχείο. Επόμενη κίνηση είναι η μετάβαση στη γραμμή εντολών και πληκτρολόγηση μιας εντολής set classpath για καθένα από αυτά τα αρχεία. Η εντολή είναι περίπου η εξής: set classpath= C:\<όνομα αρχείου>.jar;



Εικόνα 18: Δηλώσεις μονοπατιών για τα jar αρχεία

9) Ακολουθεί άνοιγμα του Netbeans 6.9 και δημιουργία μιας νέας βιβλιοθήκη (Tools → Libraries → New Library). Στη βιβλιοθήκη αυτή, η οποία μπορεί να έχει είτε το όνομα Biojava είτε οποιοδήποτε άλλο, θα πρέπει να προστεθεί ο φάκελος που αποκτήθηκε στο βήμα 7 με τη βοήθεια της επιλογής "Add JAR/Folder".



Εικόνα 19: Ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) NetBeans – δημιουργία βιβλιοθήκης

10) Κατά τη δημιουργία η καινούρια βιβλιοθήκη θα πρέπει να προστεθεί στις βιβλιοθήκες του project. Είναι επομένως απαραίτητη και η ακόλουθη διαδικασία: μετάβαση στο αριστερό παράθυρο με το όνομα "Projects" και «δεξί κλικ» στην επιλογή Libraries → Add Library → Biojava (ή όποιο άλλο όνομα έχει χρησιμοποιηθεί) → Add Library. Το Biojava είναι πλέον έτοιμο προς χρήση.
    Προσοχή: σε περίπτωση που εμφανιστεί σφάλμα κατά την εκτέλεση κάποιου παραδείγματος προτείνεται επιστροφή στο βήμα 9 και προσθήκη, με την επιλογή "Add JAR/Folder", των jar αρχείων ένα ένα.

5ο ΚΕΦΑΛΑΙΟ
ΗΛΕΚΤΡΟΝΙΚΟΣ ΟΔΗΓΟΣ ΧΡΗΣΗΣ

Περίληψη

Στο κεφάλαιο αυτό θα γίνει λόγος για τον ηλεκτρονικού οδηγό χρήσης που δημιουργήθηκε στα πλαίσια της παρούσας πτυχιακής εργασίας και αποτελεί συνέχεια του έντυπου που παρουσιάστηκε στο προηγούμενο κεφάλαιο. Αρχικά θα γίνει εκτενής αναφορά στο σχεδιασμό ιστοχώρων, στις βασικές αρχές που τον διέπουν και στα βήματα που ακολουθούνται. Επίσης, θα δοθούν σχηματικές αναπαραστάσεις και διαγράμματα για την καλύτερη κατανόηση, όπου κρίνεται απαραίτητο. Τέλος, θα γίνει αναλυτική παρουσίαση τόσο του ιστοχώρου όσο και του τρόπου δημιουργίας του και των εργαλείων που χρησιμοποιήθηκαν.

Βασικές έννοιες

- Σχεδιασμός διαδικτυακών εφαρμογών (web design)
- Βασικές αρχές σχεδιασμού
- Βήματα σχεδιασμού
- Χρήστες – απαιτήσεις χρηστών
- Χάρτης ιστοχώρου (sitemap)
- Σενάριο διάδρασης (storyboard)
- Ηλεκτρονικός οδηγός χρήσης

5.1. Ανάγκη δημιουργίας ηλεκτρονικού οδηγού

Ο οδηγός αυτός αποτελεί την ηλεκτρονική επέκταση του έντυπου με σκοπό την καλύτερη διάχυση πληροφοριών. Αποτελεί τον πρώτο οδηγό χρήσης του project Biojava στη ελληνική γλώσσα που θα είναι διαθέσιμος στο διαδίκτυο και ενδέχεται να αποτελέσει πολύτιμο σύμβουλο για όσους (ερευνητές, εκπαιδευτικούς ή φοιτητές κυρίως από τους τομείς της Βιοπληροφορικής, της Πληροφορικής και της Βιολογίας) ασχοληθούν με το εν λόγω project.

Οι λόγοι που οδήγησαν στην υλοποίησή του σε ηλεκτρονική μορφή είναι, θα λέγαμε, προφανείς. Ο πρώτος λόγος είναι η δημοτικότητα του διαδικτύου (internet). Ζούμε στην εποχή της τεχνολογικής έκρηξης και της πληροφορίας. Το διαδίκτυο αποτελεί την κύρια πηγή άντλησης πληροφοριών και το πιο ευρέως χρησιμοποιούμενο μέσο επικοινωνίας. Είναι αναμενόμενο, λοιπόν, κάποιος που θα θέλει να ενημερωθεί για το Biojava να κάνει αρχικά μια σχετική αναζήτηση στο διαδίκτυο.

Τη δημοτικότητα αυτή έρχεται να συμπληρώσει και η προσβασιμότητα στο διαδίκτυο, η οποία αυξάνεται με εκθετικούς ρυθμούς τα τελευταία χρόνια (σύμφωνα με τη Στατιστικά Υπηρεσία Ελλάδος η πρόσβαση των νοικοκυριών της χώρας μας

στο διαδίκτυο από το 2005 έως το 2009 έχει αυξηθεί κατά 95% [82]. Οι χρήστες είναι σε θέση να μελετούν τον οδηγό οποτεδήποτε, αφού είναι διαθέσιμος όλο το εικοσιτετράωρο, και οπουδήποτε στην Ελλάδα αλλά και σε ολόκληρο τον κόσμο.

Ένα ακόμη στοιχείο του ηλεκτρονικού οδηγού είναι η άμεση αλληλεπίδραση που προσφέρει, η οποία έρχεται σε αντίθεση με τη συντηρητική ανάγνωση του έντυπου οδηγού. Ο χρήστης έχει τη δυνατότητα διεξαγωγής αναζήτησης και χρήσης υπερσυνδέσμων για λήψη χρήσιμων πληροφοριών από άλλους διαδικτυακούς τόπους, καθώς και αποστολής ηλεκτρονικού μηνύματος σε περίπτωση που χρειάζεται βοήθεια ή επιπρόσθετη ενημέρωση.

Τέλος, σαν ένα ακόμη βασικό πλεονέκτημα θα πρέπει να αναγνωριστεί και η δυνατότητα ανανέωσης του υλικού. Όταν ένα project βρίσκεται στην αιχμή του είναι λογικό να προκύπτουν συνεχώς καινούρια δεδομένα, υλοποιήσεις ή ακόμη και καινούριες εκδόσεις του (π.χ. αναμένεται το Biojava 3), τα οποία θα πρέπει να παρουσιάζονται στους χρήστες. Κάτι τέτοιο, ωστόσο, θα ήταν αδύνατο αν είχαμε αρκεστεί στη δημιουργία του έντυπου οδηγού χρήσης.

Ο ηλεκτρονικός οδηγός χρήσης περιλαμβάνει οτιδήποτε αναφέρθηκε στις προηγούμενες ενότητες, από τον ορισμό και τις πληροφορίες σχετικά με τη Βιοπληροφορική, τις πληροφορίες σχετικά με το Biojava, μέχρι υλοποιήσεις πρότυπων αλγορίθμων, ολοκληρωμένα παραδείγματα και εκπαιδευτικό υλικό υπό την μορφή ασκήσεων.

5.2. Web design (σχεδιασμός ιστοχώρου)

5.2.1 Ορισμός του web design

Πρόκειται για τη *«διαδικασία χαρτογράφησης του τρόπο δράσης για την ανάπτυξη ενός έργου που προορίζεται για το διαδίκτυο (π.χ. κατασκευή portal) με την οργάνωση των απαιτήσεών του και τη δημιουργία προσχεδίων υλοποίησής του»*. [83]

Ενώ ο σχεδιασμός είναι η τέχνη και η διαδικασία του συνδυασμού επιμέρους στοιχείων του σχεδίου (γραμμές, σχήμα, υφή, χρώμα) σε μια επιθυμητή συμφωνία – ταξινόμηση, το web design είναι η τέχνη και η διαδικασία δημιουργίας μιας ιστοσελίδας ή ενός ολόκληρου ιστότοπου, που αφορά τόσο την αισθητική όσο και το μηχανισμό λειτουργίας του ιστότοπου.

5.2.2 Website planning (πλάνο – σχέδιο ιστοχώρου) -σχεδιαστική προσέγγιση

Ο ακριβής σχεδιασμός του ιστοχώρου πριν από τη δημιουργία του είναι μια σύνθετη, αλλά ουσιαστικής σημασίας, δραστηριότητα. Ιδιαίτερα αν αναλογιστούμε τις συνέπειες που μπορεί να έχουν τα λάθη κατά την υλοποίηση, κατανοούμε αμέσως τη σημασία αποφυγής τους. Θα πρέπει, λοιπόν, να ληφθούν υπόψη αρκετές παράμετροι για να θεωρηθεί ένας σχεδιασμός ολοκληρωμένος. Οι παράμετροι αυτές παρουσιάζονται παρακάτω. [84]

❖ ***Σκοπός***

Ένα από τα πρώτα βήματα της διαδικασίας σχεδιασμού είναι ο καθορισμός του σκοπού δημιουργίας του ιστοχώρου. Μια τέτοια δήλωση θα πρέπει να επικεντρώνεται στο ή στους στόχους που θα εκπληρώσει ο ιστοχώρος και ποια η χρησιμότητά του για τους υποψήφιους χρήστες. Ένας σαφώς καθορισμένος σκοπός θα βοηθήσει στην υπόλοιπη διαδικασία σχεδιασμού, αφού αμέσως προσδιορίζεται το κοινό στο οποίο απευθύνεται και το περιεχόμενο του ιστοχώρου. Επίσης ο καθορισμός όχι ενός και μοναδικού σκοπού, αλλά πολλών επιμέρους, βραχυπρόθεσμων και μακροπρόθεσμων στόχων, θα βοηθήσει σε πιθανή τροποποίηση στο μέλλον.

Ο γενικός και προφανής σκοπός του δικού μας δικτυακού τόπου με τον οδηγό χρήσης του Biojava είναι η παροχή πληροφοριών στους χρήστες σχετικά με το project.

Οι επιμέρους στόχοι που θα πρέπει να ικανοποιούνται είναι οι εξής:

- Διάθεση κώδικα σε Biojava.
- Παρουσίαση ολοκληρωμένων παραδειγμάτων.
- Ανάδειξη των πλεονεκτημάτων του Biojava έναντι της γλώσσας προγραμματισμού Java αλλά και άλλων γλωσσών.
- Παροχή γενικών πληροφοριών σχετικά με τη Βιοπληροφορική.
- Παροχή συνδέσμων για χρήσιμους ιστότοπους με σκοπό την άντληση περεταίρω πληροφοριών.
- Ευκαιρία να διαβάζουν οι χρήστες ενδιαφέροντα άρθρα.
- Εξασφάλιση της γρήγορης προσβασιμότητας και της διαθεσιμότητας.
- Παροχή δυνατότητας αναζήτησης τόσο μέσα στον ίδιο τον ιστοχώρο, όσο και σε άλλους.
- Μελλοντική προσθήκη νέου υλικού και ανανέωση των υπάρχοντος.

❖ ***Κοινό - Δυνητικοί χρήστες***

Επίσης ένα βήμα – κλειδί κατά το σχεδιασμό είναι ο προσδιορισμός του κοινού στο οποίο απευθύνεται ο ιστοχώρος, δηλαδή των υποψήφιων χρηστών. Πρόκειται, όπως είναι προφανές για διαφορετικές κατηγορίες χρηστών με διαφορετικά χαρακτηριστικά, γνώσεις, προτιμήσεις που θα πρέπει να ληφθούν υπόψη και απαιτήσεις που θα πρέπει να εκπληρωθούν.

Οι χρήστες του παρόντος δικτυακού τόπου μπορούν να χωριστούν σε δύο κατηγορίες: το χρήστη – διαχειριστή (administrator) και το χρήστη – επισκέπτη (visitor). Η πρώτη κατηγορία περιλαμβάνει το άτομο (σε άλλες περιπτώσεις μπορεί οι διαχειριστές να είναι περισσότεροι από ένας) που έχει αναλάβει εξ' ολοκλήρου, αρχικά, τη δημιουργία και, μετέπειτα, τη διαχείριση του ιστοχώρου. Η δεύτερη κατηγορία, σαφώς μεγαλύτερη από την πρώτη, περιλαμβάνει όλα εκείνα τα άτομα που θα επισκεφτούν τον ιστότοπο. Πρόκειται για ερευνητές, εκπαιδευτικούς, προγραμματιστές, βιολόγους, πανεπιστημιακούς ή μη φοιτητές και γενικά άτομα που ασχολούνται με τους τομείς της Πληροφορικής και της Βιολογίας, ιδιαίτερα δε της Βιοπληροφορικής. Στη συνέχεια παρουσιάζονται οι ιδιαίτερες απαιτήσεις της καθεμιάς από της παραπάνω ομάδες.

Απαιτήσεις διαχειριστή

- Πλήρη δικαιώματα σχετικά με τη συντήρηση και την αναβάθμιση του συστήματος.
- Δυνατότητα ενημέρωσης του υλικού που παρέχεται – προσθήκη καινούριου, διόρθωση ή διαγραφή υπάρχοντος.
- Διαχείριση εικόνων και άλλων μέσων (π.χ. modules).
- Αλλαγή/ενημέρωση των υπερσυνδέσμων.
- Δυνατότητα επέμβασης ή/και ολοκληρωτικής αλλαγής του τρόπου παρουσίασης των πληροφοριών και της μορφής του ιστοχώρου.
- Πρόσβαση στη σελίδα διαχείρισης από οποιοδήποτε σημείο με οποιονδήποτε φυλλομετρητή (browser).

Απαιτήσεις επισκέπτη

- Πρόσβαση στον ιστοχώρο από οποιοδήποτε σημείο παρέχεται πρόσβαση στο διαδίκτυο.
- Εξασφάλιση της ίδιας ποιότητας παρουσίασης του ιστοχώρου με τη χρήση οποιουδήποτε προγράμματος περιήγησης. Ο χρήστης δε θα πρέπει να είανι αναγκασμένος να εγκαταστήσει επιπλέον λογισμικό.
- Ευχρηστία και ευκολία στην πλοήγηση.
- Παροχή των πληροφοριών σε μορφή pdf με δυνατότητα αντιγραφής και αποθήκευσης.
- Δυνατότητα εκτέλεσης αναζήτησης.
- Δυνατότητα αποστολής ηλεκτρονικού μηνύματος στο διαχειριστή.
- Πρόσβαση σε άλλους διαδικτυακούς τόπους μέσω συνδέσμων για αναζήτηση περεταίρω πληροφοριών.

❖ ***Περιεχόμενο***

Η εκτίμηση και αξιολόγηση του περιεχομένου απαιτεί ο σκοπός της σελίδας να είναι καθορισμένος. Με αυτόν τον τρόπο είναι εύκολο να προσδιοριστεί, αρχικά, ποιο ακριβώς θα είναι το περιεχόμενο (αυτό δηλαδή που θα εκπληρώνει το σκοπό δημιουργίας του ιστοχώρου) και στη συνέχεια αφού συγκεντρωθεί το απαιτούμενο υλικό (κείμενα, εικόνες, κ.λπ.) να κατηγοριοποιηθεί και να οργανωθεί.

Ο ιστοχώρος επικεντρώνεται και οργανώνεται σε τρεις βασικές θεματικές ενότητες:

- Πληροφορίες σχετικά με τη Βιοπληροφορική – ορισμός της έννοιας, ιστορική εξέλιξη, χρησιμότητα και εργαλεία που χρησιμοποιούνται για την εκτέλεση εφαρμογών στον τομέα αυτό.
- Πληροφορίες σχετικά με το project Biojava – τι είναι, βασικά στοιχεία, πλεονεκτήματα, σύγκριση με τη γλώσσα προγραμματισμού Java και μελλοντική εξέλιξη.
- Ανάπτυξη του οδηγού χρήσης του project – οδηγίες εγκατάστασης, παρουσίαση κώδικα για συνηθισμένες βιολογικές διεργασίες και ολοκληρωμένων παραδειγμάτων.

❖ ***Συμβατότητα και περιορισμοί***

Λόγω της ολοένα και μεγαλύτερης ανάπτυξης του web και των τεχνολογιών γύρω από αυτό, είναι πιθανό να προκύψουν κάποια προβλήματα. Συγκεκριμένα, λόγω της ευρείας γκάμας επιλογών σε προγράμματα περιήγησης (browsers) αλλά και σε μορφές αρχείων εικόνων (image file formats) που αυτά υποστηρίζουν, η συμβατότητα της ιστοσελίδας με όλους τους χρήστες είναι περιορισμένη. Ο μόνος τρόπος να πούμε με σιγουριά ότι η ιστοσελίδα είναι συμβατή είναι η δοκιμή διαφόρων προγραμμάτων περιήγησης.

Ο συγκεκριμένος διαδικτυακός τόπος δοκιμάστηκε και εμφανίζεται με επιτυχία στα εξής προγράμματα περιήγησης: Opera, Mozilla Firefox και Google Chrome.

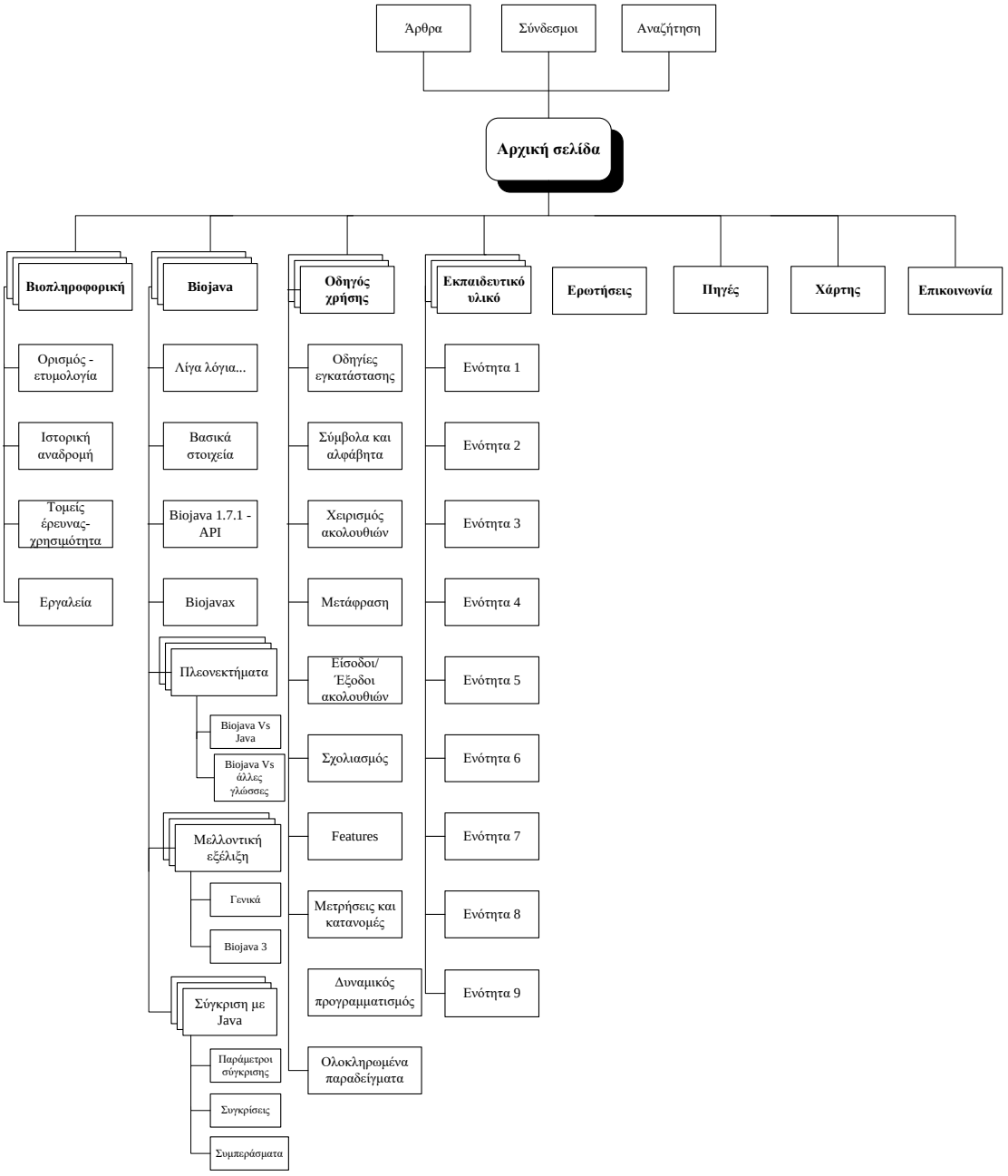
❖ ***Τεκμηρίωση του σχεδιασμού***

Η τεκμηρίωση χρησιμοποιείται για τον οπτικό σχεδιασμό του ιστοχώρου. Λαμβάνοντας υπόψη το σκοπό, τους χρήστες και το περιεχόμενο σχεδιάζονται η δομή, το περιεχόμενο και οι αλληλεπιδράσεις που κρίνονται καταλληλότερες κάθε φορά. Η τεκμηρίωση μπορεί να θεωρηθεί σαν ένα πρότυπο – ένα μοντέλο που θα επιτρέπει στη διάταξη του ιστοχώρου να αναθεωρηθεί (με σκοπό αλλαγές και βελτιώσεις).

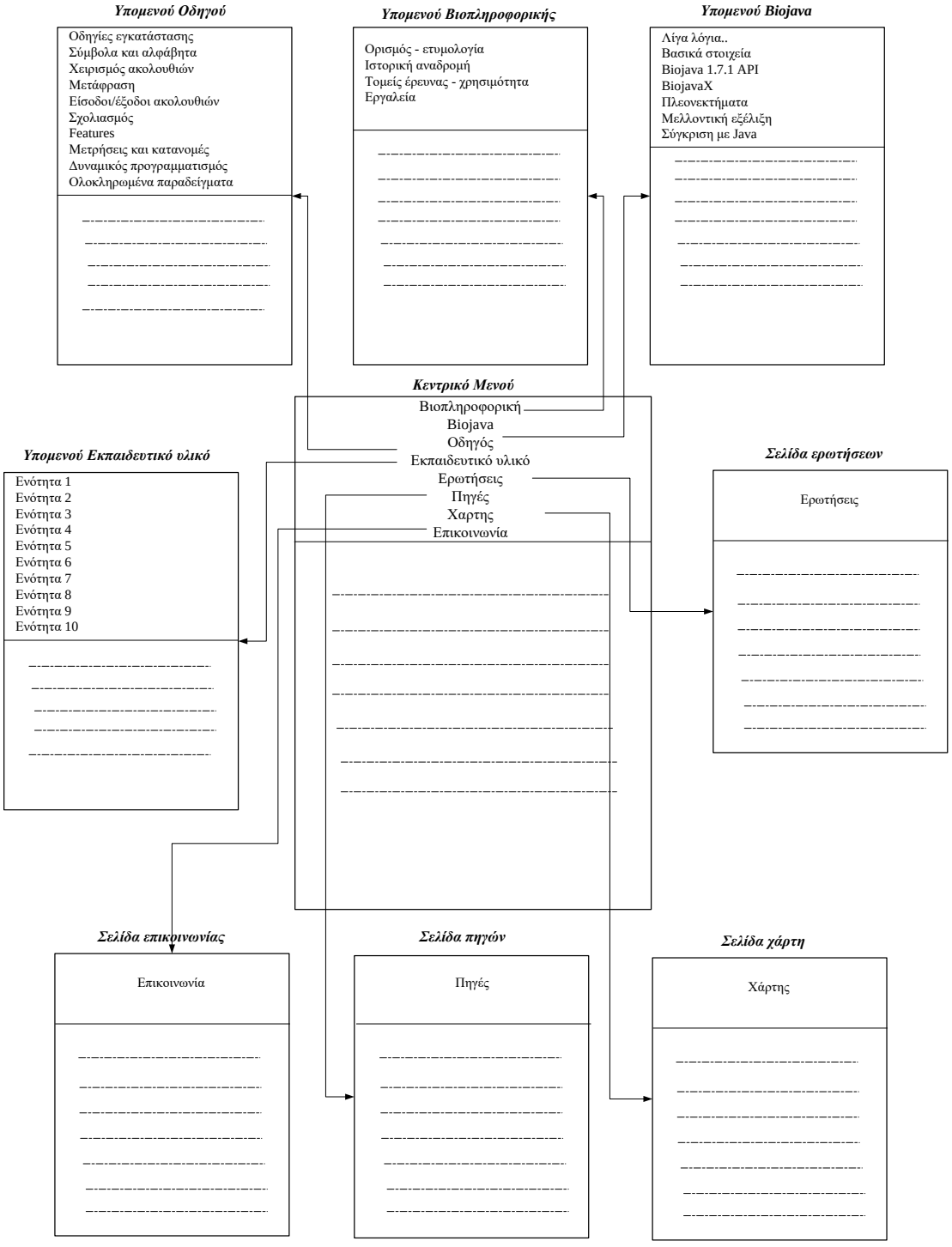
Το πρώτο βήμα μπορεί απλά να περιλαμβάνει πληροφορίες για την αρχιτεκτονική σύμφωνα με τις οποίες κατηγοριοποιείται το περιεχόμενο και διαμορφώνεται η δομή των πληροφοριών. Η δομή αυτή χρησιμοποιείται για τη δημιουργία ενός διαγράμματος που ονομάζεται χάρτης του ιστοχώρου (sitemap). Έτσι δημιουργείται μια εικόνα για το πώς οι ιστοσελίδες ή το περιεχόμενο θα συνδέονται μεταξύ τους, κάτι το οποίο μπορεί να συμβάλλει στην απόφαση του περιεχομένου που θα δοθεί σε κάθε σελίδα.

Εκτός από το σχεδιασμό της δομής, η διάταξη και η διασύνδεση των επιμέρους σελίδων μπορεί να σχεδιαστεί χρησιμοποιώντας ένα storyboard. Κατά τη διαδικασία του storyboarding δημιουργείται ένα έγγραφο με τον τίτλο, την περιγραφή και το σκοπό της κάθε σελίδας του ιστοχώρου, οι οποίες συνδέονται μεταξύ τους με τον πιο αποτελεσματικό τρόπο (υπό τη μορφή διαγράμματος).

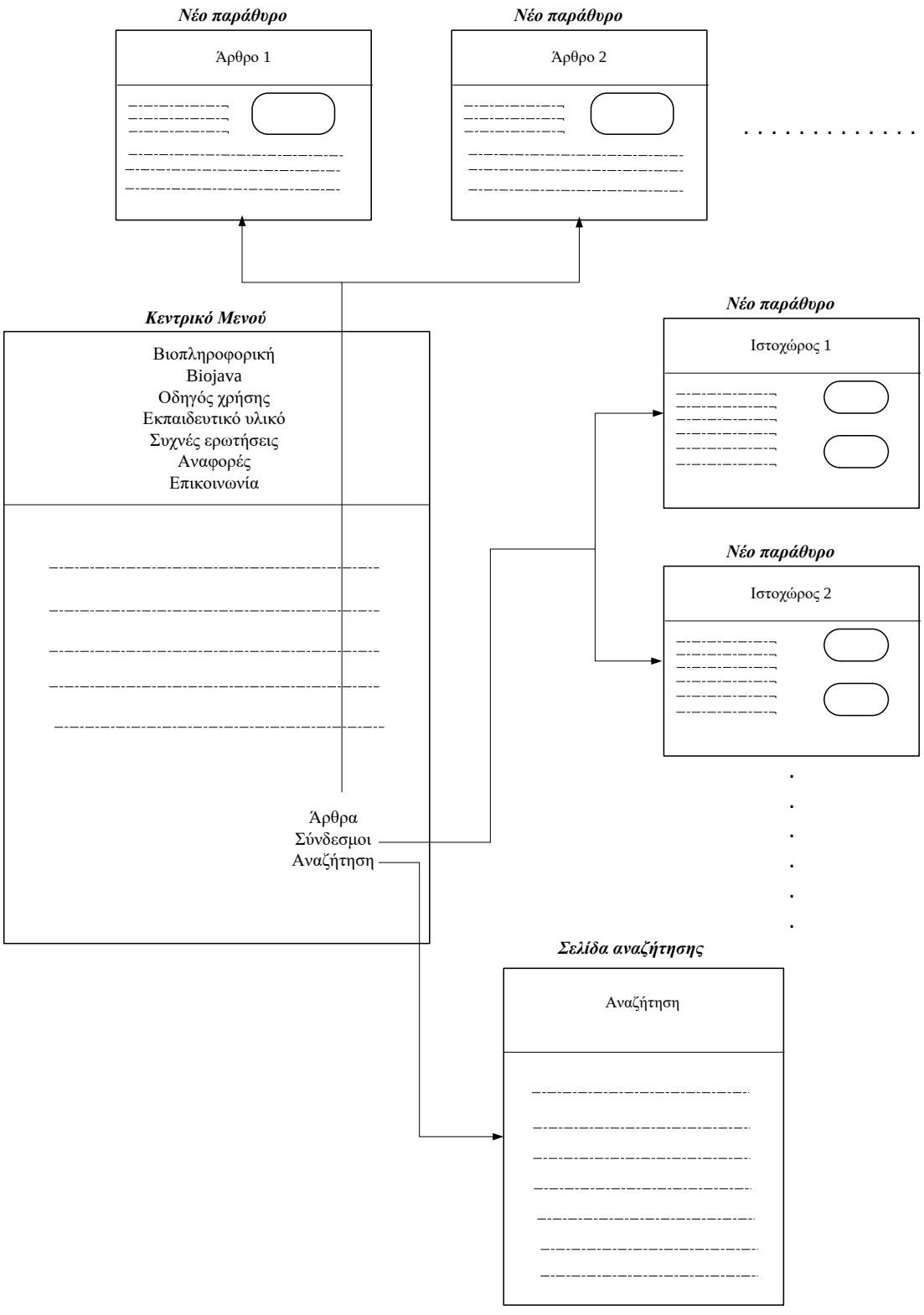
Παρακάτω παρουσιάζονται τόσο ο χάρτης πλοήγησης όσο και το σενάριο διάδρασης του διαδικτυακού χώρου. Όσον αφορά το δεύτερο, έχει γίνει χωρισμός του σε επίπεδα για την καλύτερη κατανόησή του και σε μέρη λόγω έλλειψης χώρου.



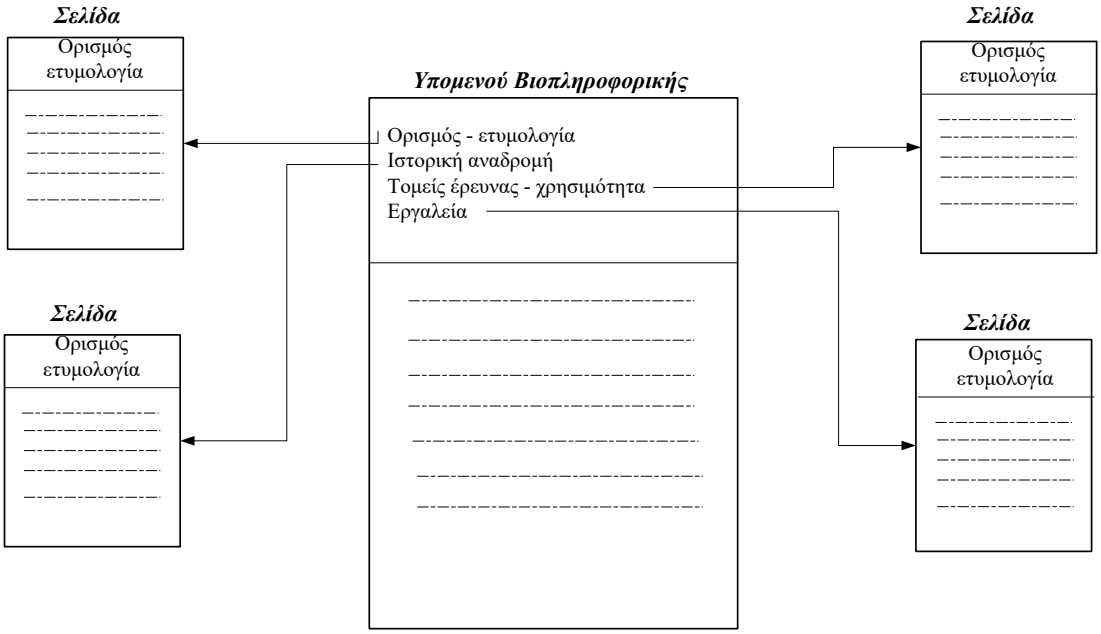
Σχήμα 9: Χάρτης (sitemap) του ιστοχώρου www.eviorama.com/biojava

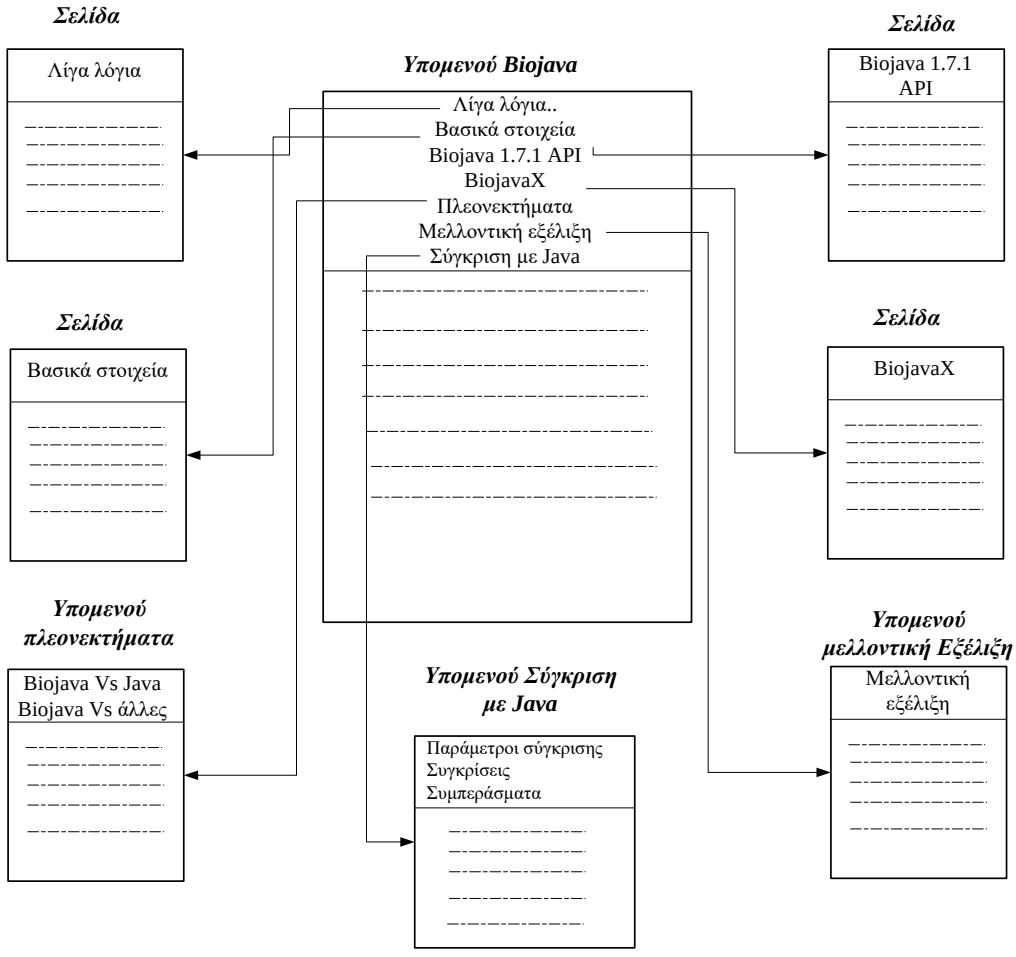


Σχήμα 10: Storyboard – επίπεδο 1 – μέρος 1 από 2

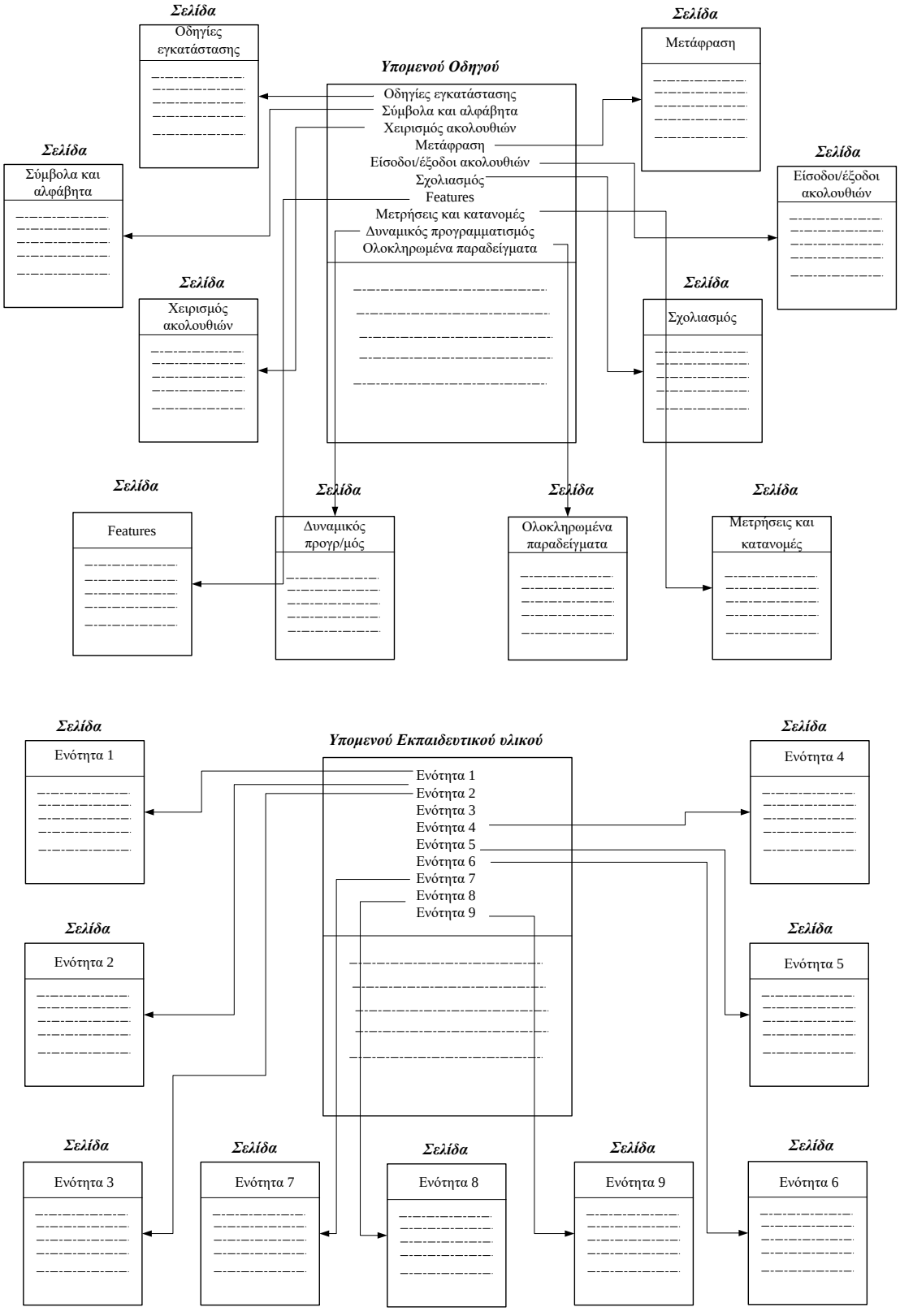


Σχήμα 11: Storyboard – επίπεδο 1 – μέρος 2 από 2

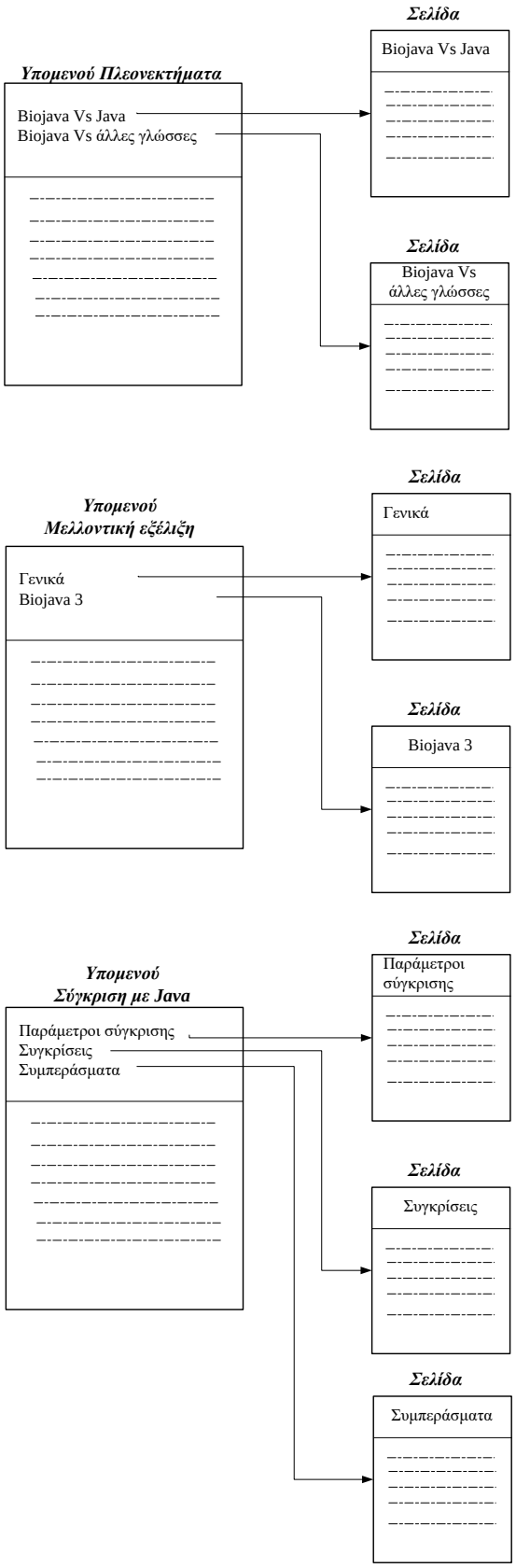




Σχήμα 12: Storyboard – επίπεδο 2 – μέρος 1 από 2



Σχήμα 13: Storyboard – επίπεδο 2 – μέρος 2 από 2



Σχήμα 14: Storyboard – επίπεδο 3

5.2.3 Website design (σχεδιασμός ιστοχώρου)

Αφού ολοκληρωθεί η δημιουργία του πλάνου – σχεδίου που θα καθορίζει τη δομή του ιστοχώρου, δηλαδή το περιεχόμενο των ιστοσελίδων και οι μεταξύ τους συνδέσεις είναι η ώρα για το σχεδιασμό. Ας αναλογιστούμε ότι η δημιουργία ενός ιστοχώρου μοιάζει με την παραδοσιακή συγγραφή βιβλίου, οπότε κάθε ιστοχώρος μπορεί να θεωρηθεί σαν ένα βιβλίο και κάθε ιστοσελίδα σαν μια απλή σελίδα. Βρισκόμαστε λοιπόν αναλογικά στο σημείο όπου έχουμε καθορίσει τη δομή των κεφαλαίων και το περιεχόμενό τους και ετοιμαζόμαστε να προσδιορίσουμε την ακριβή δομή της κάθε σελίδας – την εμφάνισή της.

Ωστόσο, στην περίπτωση του ιστοχώρου η διαδικασία αυτή είναι λίγο δυσκολότερη. Από τη φύση του, ο σχεδιασμός ιστοχώρων είναι μια αντιφατική εργασία, αφού απαιτείται να βρεθεί το σημείο τομής ανάμεσα στην άκαμπτη τεχνική συμμόρφωση και στην προσωπική δημιουργικότητα. Ένας σχεδιαστής μπορεί να δώσει περισσότερη προσοχή στην εμφάνιση της ιστοσελίδας, παραμελώντας άλλες λειτουργίες copywriting ή βελτιστοποίησης μηχανών αναζήτησης, όπως είναι η αναγνωσιμότητα του κειμένου, η ευκολία πλοήγησης ή η ευκολία εύρεσης της ιστοσελίδας. Σαν αποτέλεσμα θα είναι η δυσαρέσκεια των χρηστών και η αμφισβήτηση των μεθόδων και των πολλών διακοσμητικών γραφικών που χρησιμοποιεί ο σχεδιαστής. Θα πρέπει, συνεπώς να λαμβάνονται υπόψη τόσο η μορφή όσο και η λειτουργικότητα. Αξίζει, βέβαια, να σημειωθεί ότι, στις μέρες μας, σε ορισμένες περιπτώσεις η μορφή ακολουθεί τη λειτουργικότητα. Για παράδειγμα στις περισσότερες ιστοσελίδες υπάρχουν δυναμικά μενού ή slideshows για την παρουσίαση άρθρων, νέων ή ανακοινώσεων, τα οποία υπακούν τόσο σε κανόνες αισθητικής όσο και λειτουργικότητας.

❖ ***Ευχρηστία/ λειτουργικότητα***

Σύμφωνα με το πρότυπο (standard) ISO 9241 (Εργονομικές Απαιτήσεις για την Εργασία Γραφείου με Τερματικά με Οπτικές Οθόνες), ευχρηστία είναι: *«… ο βαθμός στον οποίο ένα προϊόν μπορεί να χρησιμοποιηθεί από συγκεκριμένους τελικούς χρήστες για να επιτύχουν συγκεκριμένους στόχους με αποτελεσματικότητα, αποδοτικότητα και ικανοποίηση μέσα σε ένα σε συγκεκριμένο πλαίσιο χρήσης»* [85]. Αν ο ιστοχώρος που θα δημιουργήσουμε δε διακρίνεται από ευχρηστία μπορούμε να μιλάμε με βεβαιότητα για αποτυχία, καθώς οι χρήστες θα αναγνωρίζουν τη μη λειτουργικότητά του και δε θα είναι σε θέση να το χρησιμοποιήσουν.

Οι βασικότερες από τις αρχές που θα πρέπει να εξασφαλιστούν στον εν λόγω διαδικτυακό τόπο είναι οι εξής [86]:

- *Ομοιομορφία και συνέπεια:* Τα γραφικά και το κείμενο επιβάλλεται να ακολουθούν ένα ενιαίο ύφος σε όλο το φάσμα του ιστοχώρου.
- *Οικειότητα:* Ο ιστοχώρος θα πρέπει να είναι φιλικός προς το χρήστη με τη δημιουργία εύκολης και αξιόπιστης διεπαφής και εύκολης πλοήγησης παρόμοιας με αυτή που χρησιμοποιείται σε άλλους ιστοχώρους.

- *Μινιμαλισμός:* Αποφυγή περιττών στοιχείων ώστε να μη δυσκολεύεται και να μην κουράζεται ο χρήστης κατά την ανάγνωση.
- *Χρήση κατανοητής προς τους χρήστες γλώσσας: Α*ποφυγή δύσκολων ορολογιών και εννοιών.
- *Επαρκής υποστήριξη:* Δυνατότητα αναζήτησης, χάρτης πλοήγησης, δυνατότητα αποστολής μηνύματος ηλεκτρονικού ταχυδρομείου για περαιτέρω βοήθεια.

❖ ***Τεχνικά θέματα***

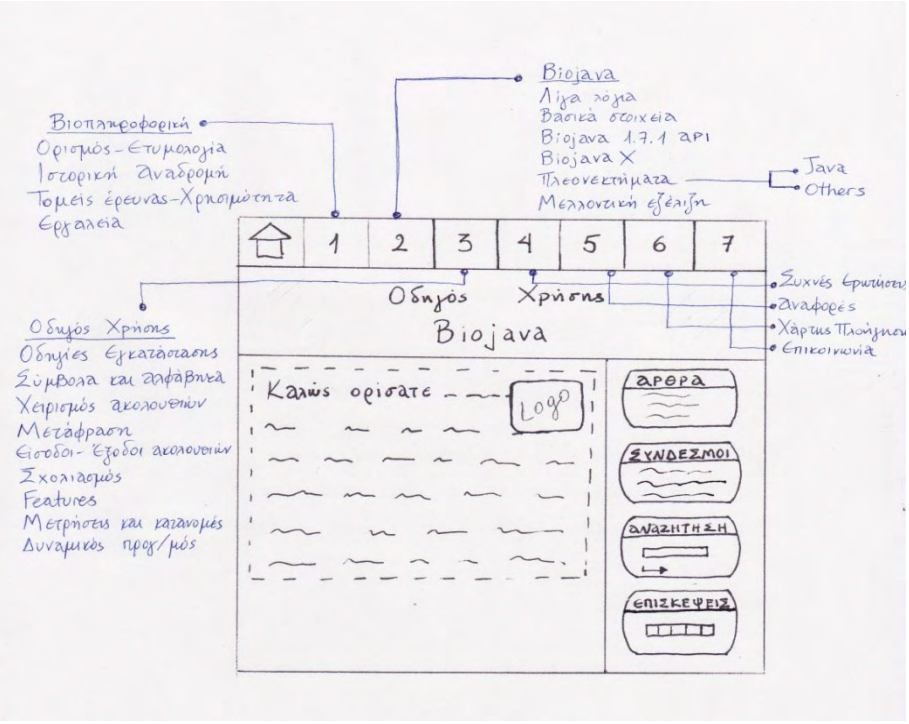
Όπως αναφέραμε και πρωτύτερα κατά το σχεδιασμό πρέπει να δίνεται βαρύτητα τόσο στη λειτουργικότητα όσο και στη μορφή που θα έχει ο ιστοχώρος. Όσον αφορά τη λειτουργικότητα θα πρέπει να ακολουθούνται οι παραπάνω απαιτήσεις. Όσον αφορά τη μορφή, πέρα από τις αρχές τις αισθητικής που θα πρέπει να τον διέπουν, είναι ανάγκη να ξεκαθαριστούν και κάποια τεχνικά θέματα. Κυριότερα είναι αυτά που ακολουθούν [84] [86]:

- *Ανάλυση οθόνης:* Επειδή ο ιστοχώρος αναμένεται να προβληθεί σε διάφορους χρήστες των οποίων οι οθόνες δεν έχουν απαραίτητα τα ίδια χαρακτηριστικά και την ίδια ανάλυση, θα πρέπει να βεβαιωθούμε ότι οι ιστοσελίδες φαίνονται καλά ακόμη και στη μικρότερη δυνατή ανάλυση (600x480).
- *Βάθος χρώματος:* Οι ιστοσελίδες θα πρέπει να σχεδιάζονται σε παλέτα 2^8=256 χρωμάτων, αφού όσο περισσότερα χρώματα χρησιμοποιούνται τόσο πιο πραγματικές φαίνονται οι εικόνες στο ανθρώπινο μάτι. Ωστόσο, τα γραφικά είναι ανάγκη να ελέγχονται ακόμη και σε 16 για να διασφαλίζεται ότι φαίνονται ομαλά. (τα λιγότερα υποστηριζόμενα χρώματα είναι τα 2^4=16 χρώματα)
- *Το μέγεθος των εγγράφων:* Είναι ένας από τους σημαντικότερους παράγοντες που θα πρέπει να λαμβάνεται υπόψη. Αν ληφθεί υπόψη ότι όσο μεγαλύτερο είναι το μέγεθος των αρχείων τόσο περισσότερος είναι ο χρόνος που απαιτείται για να «κατέβει» η σελίδα, ένας πολύ «φορτωμένος» ιστότοπος, λόγω αυτής της καθυστέρησης θα προκαλέσει δυσανασχέτηση στο χρήστη. Σημειώνουμε επίσης ότι το μέγεθος του ιστοχώρου υπολογίζεται από το άθροισμα των μεγεθών των εξής: κώδικας HTML + γραφικά + μέγεθος άλλων αρχείων (audio, video, java applets, κ.λπ.)
- *Διάταξη σελίδας:* Το περιεχόμενο θα πρέπει να κατανέμεται στη σελίδα λαμβάνοντας τον τρόπο ανάγνωσης του χρήστη. Έχει παρατηρηθεί όταν κάποιος επισκέπτεται για πρώτη φορά μια ιστοσελίδα, διαβάζει σε σχήμα *F*. Διενεργούν, δηλαδή, πρώτα μια ανάγνωση στο πάνω μέρος της σελίδας με οριζόντια φορά, έπειτα μια παρόμοια λίγο πιο κάτω και στη συνέχεια μια ανάγνωση στο αριστερό μέρος με κάθετη φορά. Είναι χρήσιμο επίσης να καθοριστεί το μέγεθος έτσι ώστε να μη χρειάζεται ολίσθηση και ο χρήστης να έχει την πλήρη εποπτεία στα πλαίσια μιας οθόνης.

❖ ***Σχεδιασμός αρχικής σελίδας***

Η πρώτη σελίδα ενός ιστότοπου είναι γνωστή σαν αρχική σελίδα (Home Page) ή σελίδα περιεχομένων (Index Page). Κάποιοι ιστοχώροι έχουν και μια σελίδα πριν από την αρχική, η οποία ονομάζεται Splash Page και ενδέχεται να περιλαμβάνει ένα μήνυμα καλωσορίσματος, επιλογή γλώσσας, κ.λπ.. Αφού, λοιπόν, η αρχική σελίδα είναι η πρώτη σελίδα με την οποία έρχεται σε επαφή ο χρήστης κατά την επίσκεψή του στον ιστότοπο είναι αναγκαίο να της δοθεί και η απαραίτητη προσοχή. Είναι, ίσως, η μοναδική σελίδα που μπορεί να διαφέρει στην πολυπλοκότητα της δομής της από τις υπόλοιπες σελίδες του ιστοτόπου. Σε καμιά περίπτωση όμως δεν πρέπει να διαφέρει στο ύφος ή τους χρωματισμούς της. Οι κανόνες ακολουθούνται κατά το σχεδιασμό της παρουσιάζονται παρακάτω:

- Η αρχική σελίδα θα πρέπει να λειτουργεί σαν πίνακας περιεχομένων και να περιλαμβάνει συνδέσμους (μέσω μενού πλοήγησης) προς όλες τις σελίδες του ιστοχώρου, αναδεικνύοντας στοιχεία εσωτερικού περιεχομένου.
- Επιβάλλεται να μην περιέχει μεγάλο κείμενο αλλά μια σύντομη αναφορά στο λόγο ύπαρξης και τη χρησιμότητα του ιστοχώρου.
- Το παράθυρο θα πρέπει να διαθέτει όνομα που θα την κατατάσσει εύκολα στις μηχανές αναζήτησης.
- Θα πρέπει να δίνεται έμφαση στα σημεία υψηλής προτεραιότητας.
- Δεν πρέπει να γίνεται υπερβολική χρήση γραφικών (ιδιαίτερα σε σημαντικό περιεχόμενο) και πολλών μορφών αρχείων (video, εικόνες, κ.λπ.) που θα καθιστούν το «φόρτωμα της σελίδας αργό.
- Είναι χρήσιμο να περιλαμβάνει «κουτάκι» αναζήτησης και διακριτούς συνδέσμους προς άλλους διαδικτυακούς τόπους με συναφές περιεχόμενο.



Εικόνα 20: Αρχικό σχέδιο για τη homepage

5.3. Υλοποίηση

5.3.1 Προγράμματα και συστήματα που χρησιμοποιήθηκαν

5.3.1.1 Artisteer 2

❖ ***Τι είναι;***

Είναι εμπορικό προϊόν που αυτοματοποιεί το σχεδιασμό ιστοσελίδων. Πρόκειται ουσιαστικά για ένα πρόγραμμα το οποίο δημιουργεί αυτόματα πρότυπα ιστοχώρων (website templates) και θέματα ιστολογίων. Δοκιμαστική έκδοσή του είναι διαθέσιμη στον επίσημο ιστότοπο του Artisteer [87].

❖ ***Τι προσφέρει;***

Προσφέρει μια εξαιρετικά εύχρηστη διεπαφή, ακόμα για άτομα χωρίς καθόλου εμπειρία στο σχεδιασμό ιστοσελίδων, που θυμίζει ιδιαίτερα το περιβάλλον εργασίας του Microsoft Office 2007. Με αυτό τον τρόπο καθίσταται εφικτός ο επαγγελματικός σχεδιασμός χωρίς απαίτηση για περεταίρω γνώσεις Photoshop, CSS, HTML, Dreamweaver ή άλλων τεχνολογιών. Τέλος, ξεχωρίζει για τη δυνατότητα πειραματισμού που δίνει στο χρήστη καθώς και την εις βάθος παραμετροποίηση. Είναι προφανείς, λοιπόν οι λόγοι που αποφασίστηκε να γίνει ο σχεδιασμός του προτύπου με το εν λόγω πρόγραμμα.

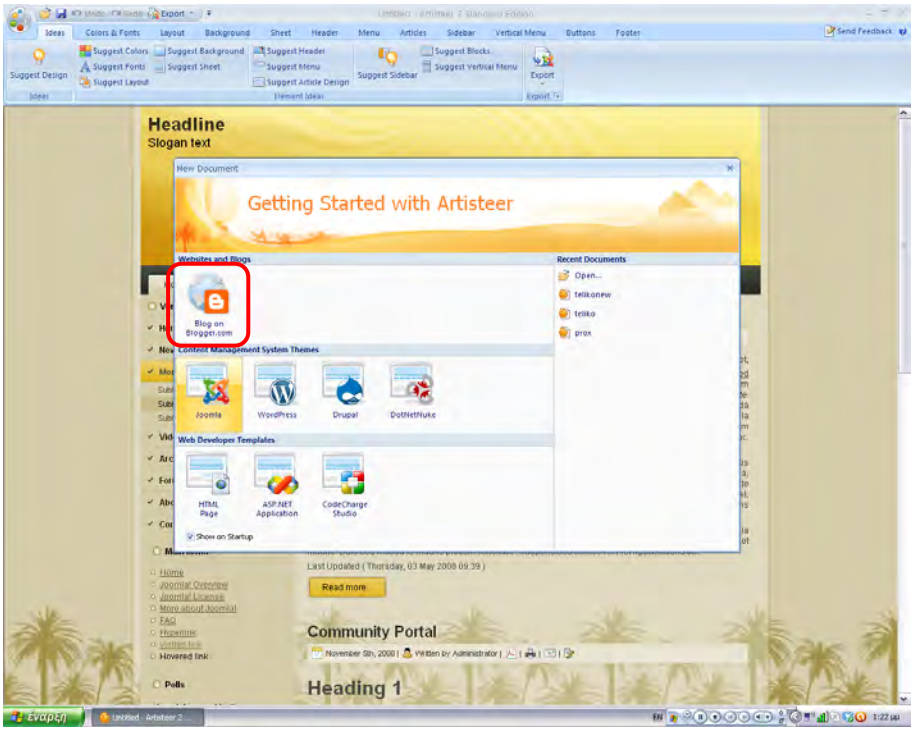
❖ ***Πώς λειτουργεί;***

Αφού επιλέξουμε με τι ακριβώς θέλουμε να ασχοληθούμε από το παράθυρο που εμφανίζεται κατά την είσοδό μας στο πρόγραμμα (στην προκειμένη περίπτωση με πρότυπο Joomla) (εικόνα 5.3) βρισκόμαστε στην αρχική σελίδα όπου προτείνεται και το πρώτο πρότυπο.

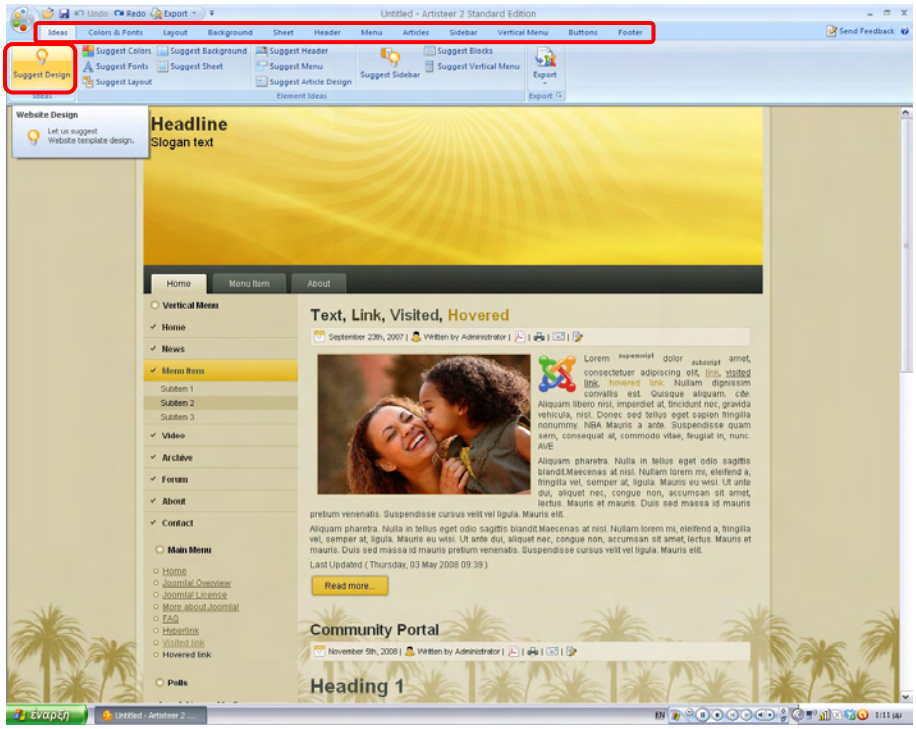
Το πρόγραμμα διαθέτει μια εργαλειοθήκη με 12 διαφορετικές επιλογές (εικόνα 5.4). Η βασική ιδέα είναι η επιλογή του προτύπου και στη συνέχεια η διαμόρφωσή του με τις δυνατότητες που δίνουν οι υπόλοιπες 11 επιλογές. Αναλυτικότερα οι επιλογές αυτές είναι:

1. *Ιδέες (Ideas):* Προτείνονται με ψευδο-τυχαία σειρά πρότυπα και για το καθένα χρώματα, διατάξεις, φόντο, γραμματοσειρές, κ.λπ. και παρέχεται η επιλογή εξαγωγή (export), την οποία χρειαζόμαστε μόλις ολοκληρώσουμε το σχεδιασμό του προτύπου για να αποκτήσουμε το Joomla template που δημιουργήσαμε σε μορφή zip αρχείου (εικόνα 5.5) (η διαδικασία αυτή είναι απαραίτητη για να είναι το πρότυπό μας διαχειρήσιμο από το σύστημα Joomla).
2. *Χρώματα και γραμματοσειρές (Colors & Fonts):* Δίνεται η δυνατότητα να βρεθεί ο επιθυμητός συνδυασμός χρωμάτων και η κατάλληλη γραμματοσειρά.
3. *Διάταξη (Layout):* Εύρεση της επιθυμητής διάταξης της σελίδας, δηλαδή ο συνδυασμός μενού – άρθρου ή ο αριθμός στηλών της σελίδας.
4. *Φόντο (Background):* Χειρισμός του φόντου (χρώμα, σχέδιο, εισαγωγή εικόνας κ.λπ.).
5. *Σελίδα (Sheet):* Καθορισμός του πλάτους της σελίδας, του χρώματός της, των περιθωρίων κ.λπ..

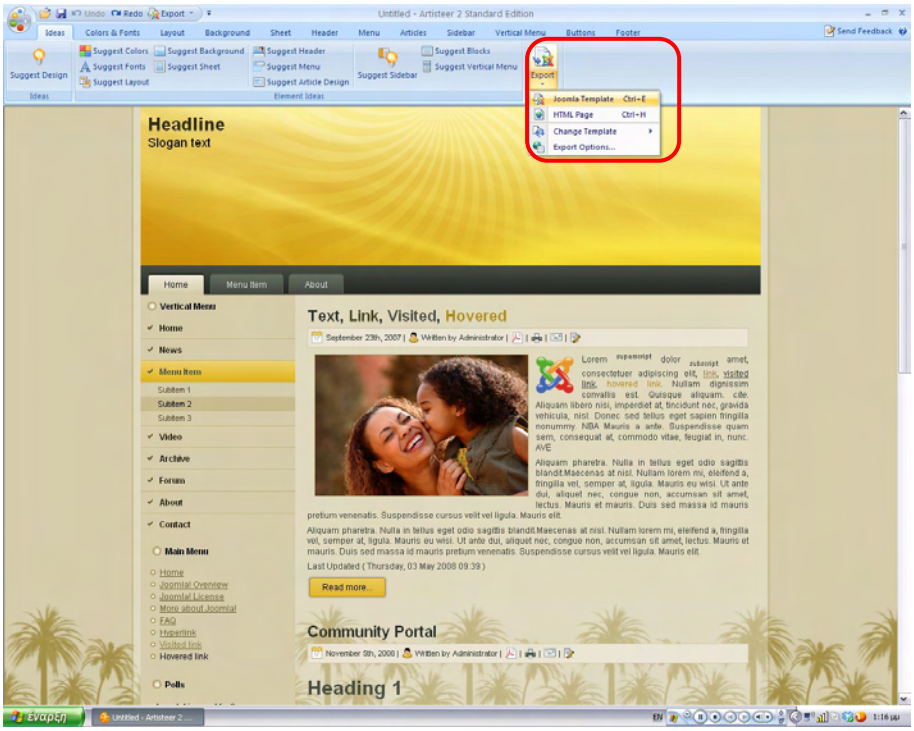
6. *Επικεφαλίδα (Header):* Διαμόρφωση ολόκληρου του πλαισίου που θα αποτελέσει επικεφαλίδα – εύρεση κατάλληλου ύψους, πλάτους, χρώματος, εικόνας, κτλ..
7. *Μενού (Menus):* Επιλογή του στυλ και του χρωματισμού που θα έχει το κεντρικό μενού – ξεχωριστή διαμόρφωση για τη μπάρα (menu bar) για τα κουμπιά/ αντικείμενα (item) και για τα κουμπιά/αντικείμενα σε δεύτερο ή μεγαλύτερο επίπεδο (subitem).
8. *Άρθρα (Articles):* Επιλογή του τρόπου εμφάνισης των άρθρων (μέγεθος και χρώμα άρθρου, μέγεθος, χρώμα και είδος γραμματοσειράς, διαμόρφωση επικεφαλίδας κ.ά.).
9. *Πλάγια μπάρα (Sidebar):* Καθορισμός του στυλ και του χρώματος της πλάγιας μπάρας – εάν υπάρχει.
10. *Κάθετο μενού (Vertical menu):* Επιλογή του στυλ και του χρωματισμού που θα έχει το κάθετο μενού – ξεχωριστή διαμόρφωση για τη μπάρα (menu bar) για τα κουμπιά/ αντικείμενα (item) και για τα κουμπιά/αντικείμενα σε δεύτερο ή μεγαλύτερο επίπεδο (subitem).
11. *Κουμπιά (Buttons):* Προσδιορισμός της εμφάνισης των κουμπιών (σχήμα, χρώμα κ.λπ.) τα οποία μπορούν να εμφανίζονται σε οποιοδήποτε σημείο του άρθρου ή της πλάγιας μπάρας.
12. *Footer:* Επιλογή του χρώματος και του στυλ που θέλουμε να διαθέτει καθώς και του τρόπου εμφάνισης των γραμμάτων μέσα σε αυτό.



Εικόνα 21: Επιλογή του Joomla



Εικόνα 22: Αρχική σελίδα του Artisteer όπου ξεχωρίζει το μενού και η επιλογή για τις προτάσεις προτύπου



Εικόνα 23: Επιλογή εξαγωγής του προτύπου σε αρχείο μορφής zip

5.3.1.2 Joomla

- ***Τι είναι;***

Πρόκειται για ένα βραβευμένο σύστημα διαχείρισης περιεχομένου (Content Management System - CMS) που καθιστά δυνατή τη δημιουργία ιστοσελίδων και ολοκληρωμένων online εφαρμογών. Το Joomla είναι γραμμένο σε PHP, αποθηκεύει τα δεδομένα σε μια βάση MySQL. Είναι μια εφαρμογή ανοιχτού κώδικα που διατίθεται δωρεάν σε όποιον θέλει να ασχοληθεί με το σχεδιασμό ιστού από τον επίσημο ιστοχώρο του [88] αλλά και από την ελληνική κοινότητα υποστήριξης [89].

- ***Τι προσφέρει;***

Είναι μια ευέλικτη εφαρμογή με εύχρηστο περιβάλλον φιλικό προς το χρήστη. Ιδιαίτερο χαρακτηριστικό του είναι αναρίθμητες δυνατότητες που προσφέρει. Για παράδειγμα μπορεί κάποιος να «ανεβάζει» περιεχόμενο από οποιονδήποτε υπολογιστή, ο οποίος διαθέτει σύνδεση στο διαδίκτυο. Η δημοσίευση σελίδων είναι απεριόριστη και υπάρχουν ακόμη εκατοντάδες πρόσθετες εφαρμογές, όπως προσθήκη forum, παρουσιάσεις φωτογραφιών, φόρμες επικοινωνίας και πολλά άλλα.

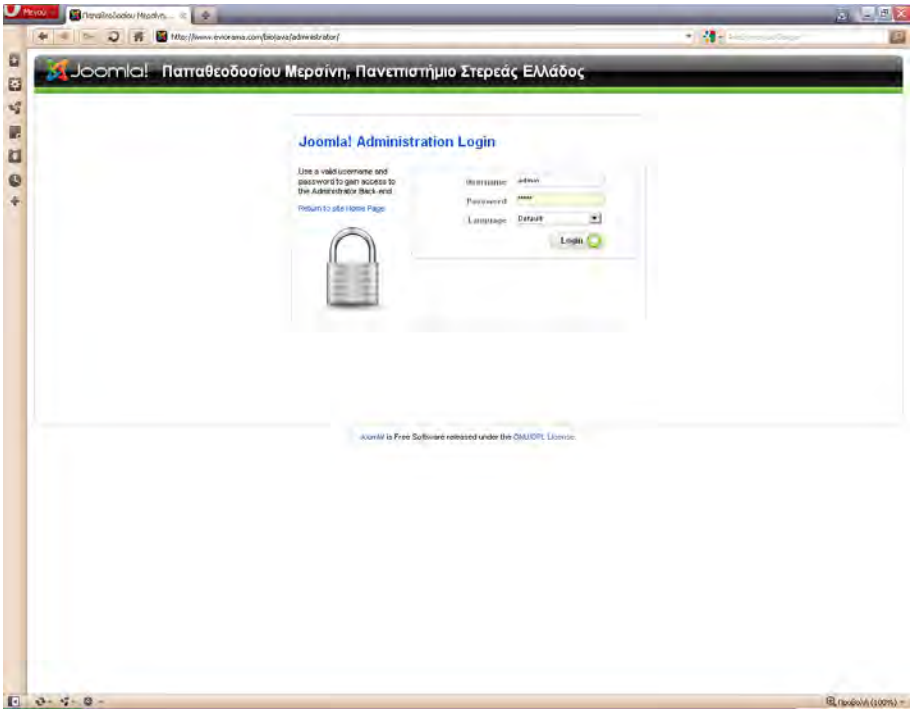
- ***Πώς λειτουργεί;***

Το Joomla εγκαθίσταται σε έναν κεντρικό υπολογιστή, τον web server. Ο διαχειριστής έχει πρόσβαση στο περιβάλλον διαχείρισης μέσω ενός προγράμματος περιήγησης (browser), όπως είναι, για παράδειγμα το Internet Explorer ή το Mozilla Firefox. Μπορεί με αυτό τον τρόπο να κάνει οποιαδήποτε προσθήκη ή τροποποίηση στο περιεχόμενο της ιστοσελίδας. Το περιεχόμενο οργανώνεται αποδοτικά σε αρκετές κατηγορίες και υποκατηγορίες, μέσα σε ένα γραφικό περιβάλλον.

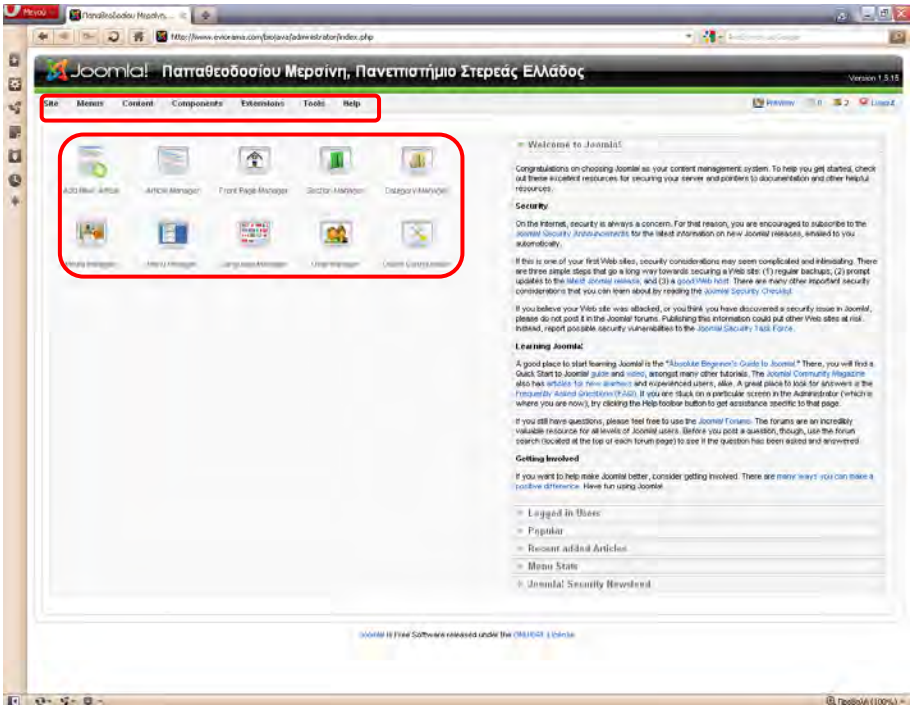
Το μενού του περιβάλλοντος διαχείρισης Joomla αποτελείται από τις εξής 7 επιλογές:

1. *Ιστοσελίδα (Site):* Δίνεται η δυνατότητα μετάβασης στη διαχείριση χρηστών (user manager), στη διαχείριση εικόνων, βίντεο και άλλων πολυμέσων, στην καθολική διαμόρφωση του ιστοχώρου (global configuration) και στην αποσύνδεση.
2. *Μενού (Menus):* Εισαγωγή στο περιβάλλον διαχείρισης των μενού, καθώς και πρόσβαση στο καθένα ξεχωριστά.
3. *Περιεχόμενο (Content):* Από αυτή την επιλογή παρέχεται πρόσβαση στο περιβάλλον διαχείρισης άρθρων, ενοτήτων, κατηγοριών καθώς και σ' αυτό της πρώτης σελίδας.
4. *Στοιχεία (Components):* Επιτρέπει τη μετάβαση στη διαχείριση επαφών, αναζητήσεων, υπερσυνδέσμων κ.ά..
5. *Επεκτάσεις (Extensions):* Προάγει την είσοδο σε σημαντικά περιβάλλοντα διαχείρισης όπως αυτά των modules των plugins, των προτύπων και της γλώσσας και παρέχει την επιλογή (Εγκατάσταση/Απεγκατάσταση) για τις αντίστοιχες λειτουργίες που αφορούν modules αλλά και ολοκληρωμένα πρότυπα.

6. *Εργαλεία (Tools):* Χρήση εργαλείων ανάγνωσης και συγγραφής μηνυμάτων, ομαδικής αλληλογραφίας, κ.ά..
7. *Βοήθεια (Help):* Κλασική επιλογή για αναζήτηση βοήθειας σε περίπτωση που χρειαστεί κατά την ενασχόληση με το Joomla.



Εικόνα 25: Προσθήκη κωδικού για είσοδο στο περιβάλλον διαχείρισης Joomla



Εικόνα 24: Αρχική σελίδα του περιβάλλοντος διαχείρησης Joomla όπου φαίνονται το βασικό μενού και ο πίνακας ελέγχου.

5.3.1.3 Photoscape

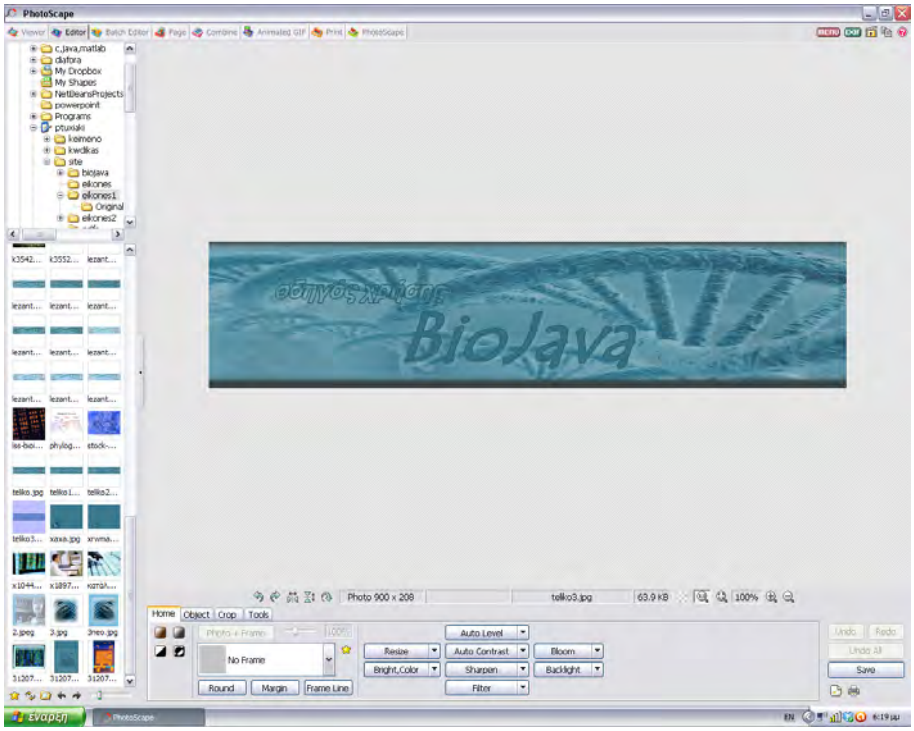
❖ ***Τι είναι;***

Είναι ένα πρόγραμμα σχεδιασμένο σχεδιασμένο να προσφέρει υπηρεσίες οργάνωσης και επεξεργασίας εικόνων. Πρόκειται για δωρεάν εφαρμογή η οποία είναι διαθέσιμη στο διαδίκτυο, στον επίσημο ιστοχώρο του Photoscape [90].

❖ ***Τι προσφέρει;***

Οι δυνατότητες που εμφανίζονται στο μενού του είναι οι εξής:

- *Επισκόπηση:* Προβολή των εικόνων που υπάρχουν στον υπολογιστή του χρήστη.
- Επεξεργασία εικόνων και φωτογραφιών: Αλλαγή μεγέθους και φωτεινότητας, αναπροσαρμογή του χρώματος, ισορροπία λευκού, προσθήκη κειμένου, πλαισίου και άλλων εφφέ.
- Συνδυασμός εικόνων.
- Μετατροπή: Μετατρέπει τη μια μορφή εικόνας σε άλλη.



Εικόνα 26: Επιλογή editor από το μενού για την επεξεργασία εικόνων

5.3.2 Διαδικασία που ακολουθήθηκε

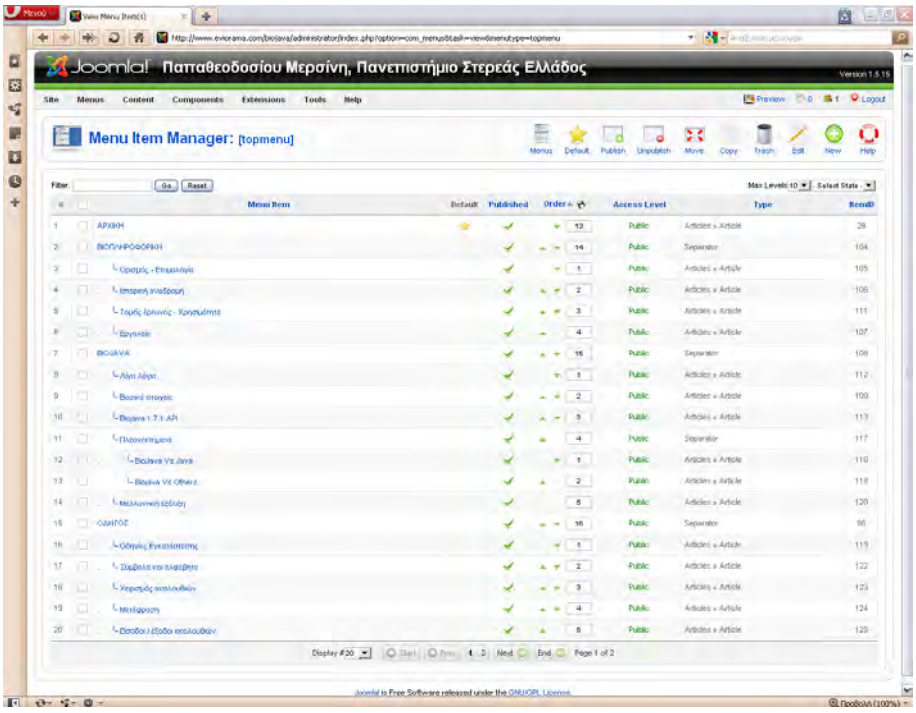
Για τη δημιουργία του ιστοχώρου για τον οδηγό χρήσης του project χρειάστηκε αρχικά να κατοχυρωθεί το όνομα. Το όνομα που αποφασίστηκε να δοθεί δεν είναι άλλο από το Biojava. Συνεπώς είναι πλέον διαθέσιμο στη διεύθυνση www.eviorama.com/biojava.

Στη συνέχεια έπρεπε να δημιουργηθεί το πρότυπο (template) με τη βοήθεια του προγράμματος Artisteer που περιγράψαμε στην ενότητα 5.3.1.3. Έγινε διαμόρφωση ενός από τα προτεινόμενα πρότυπα (συνδυασμός χρωμάτων, αλλαγή φόντου, μορφοποίηση μενού) και ρύθμιση όλων των λεπτομερειών για την εμφάνιση του κειμένου και των εικόνων, καθώς και εισαγωγή της εικόνας που δημιουργήθηκε με τη βοήθεια του προγράμματος Photoscape (εικόνα 5-8) στην τοποθεσία header. Μετά τον καθορισμό της τελικής μορφής έγινε η εξαγωγή του προτύπου σε μορφή zip αρχείου με τη βοήθεια της επιλογής export που επίσης περιγράψαμε παραπάνω (εικόνα 5-4). Το τελικό αυτό πρότυπο παρουσιάζεται στην παρακάτω εικόνα.

Εικόνα 27: Τελικό πρότυπο με το όνομα telikobiojava

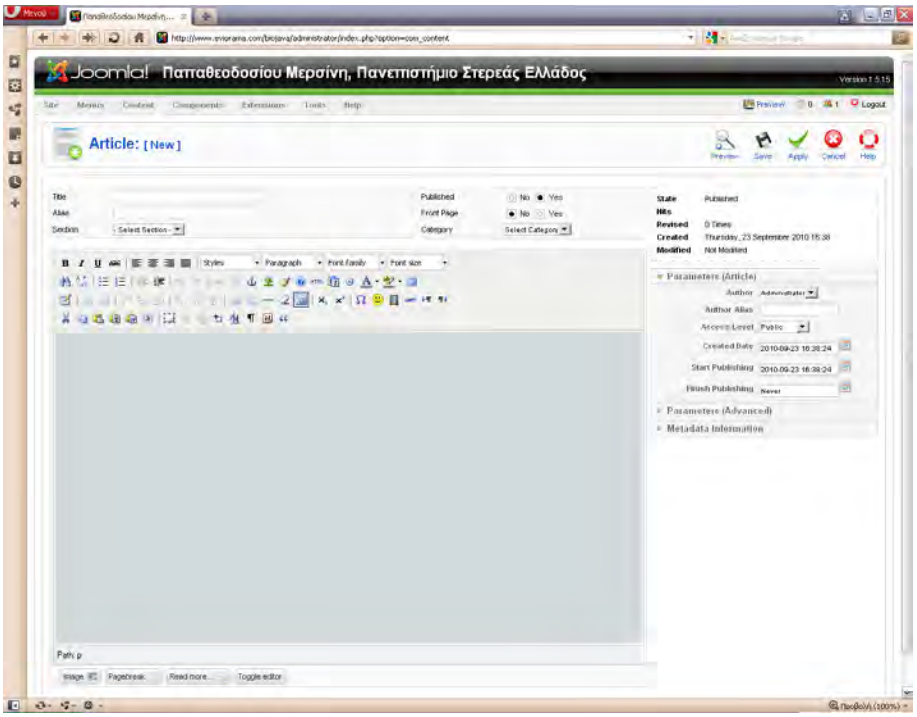
Στη συνέχεια το πρότυπο αυτό έπρεπε να εγκατασταθεί στο περιβάλλον διαχείρισης Joomla. Το μονοπάτι που ακολουθήθηκε είναι το εξής: Extensions → Install/ Uninstall → Install → Upload package file.

Μετά την εγκατάστασή του έπρεπε να διαμορφωθεί το top menu δηλαδή το βασικό οριζόντιο μενού (μαζί με τα υπομενού) που βρίσκεται στο πάνω μέρος της σελίδας. Η εργασία αυτή έγινε επιλέγοντας το topmenu από το κουμπί Menus.



Εικόνα 28: Περιβάλλον διαχείρισης Menu Item Manager

Παράλληλα δημιουργήθηκαν και τα άρθρα που θα εμφανίζονται σε κάθε επιλογή του μενού. Ακολουθώντας το μονοπάτι Content → Article Manager → New καταλήγουμε στο περιβάλλον διαχείρισης άρθρου (εικόνα 5-11), όπου μπορούμε να προσθέσουμε κείμενο, εικόνες και βίντεο.



Εικόνα 29: Περιβάλλον διαχείρισης άρθρων

Για να οδηγηθούμε στο τελικό αποτέλεσμα χρειάστηκε η ενεργοποίηση ή επιπλέον εγκατάσταση κάποιων modules,όπως αυτά για την αναζήτηση και για την παρουσίαση των επιστημονικών άρθρων σχετικών με το Biojava. Αυτό έγινε εφικτό με την επιλογή Module Manager από το κουμπί Extensions του κεντρικού μενού.

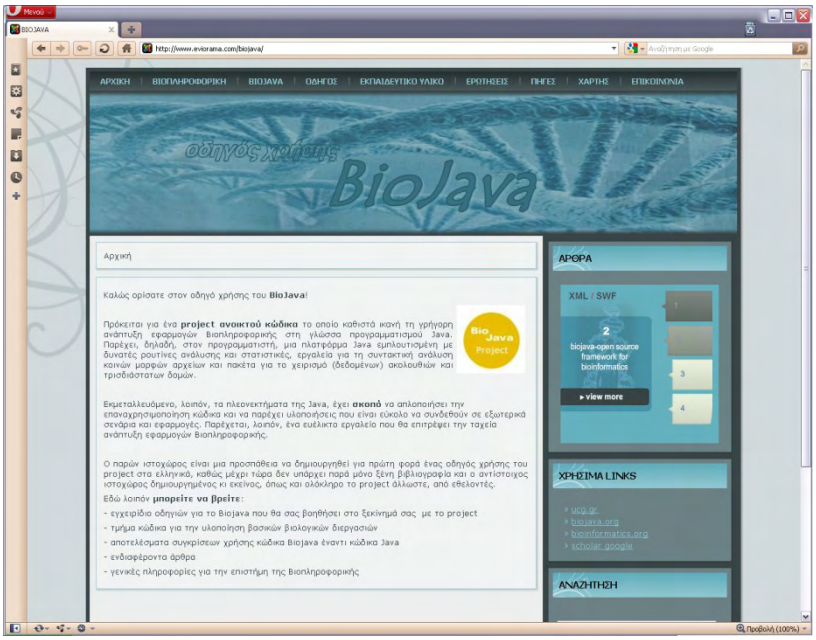
5.3.3 Παρουσίαση του ιστοχώρου

Στην προσπάθεια να οργανωθεί αποδοτικά το περιεχόμενο του οδηγού χρήσης κρίθηκε απαραίτητος ο χωρισμός του σε κατηγορίες καθεμιά από τις οποίες διαθέτει υποκατηγορίες. Δημιουργήθηκαν με αυτό τον τρόπο τέσσερις μεγάλες κατηγορίες με τα ονόματα Βιοπληροφορική, Biojava, Οδηγός και Εκπαίδευση, οι οποίες περιέχουν γενικές πληροφορίες για την επιστήμη της Βιοπληροφορικής, πληροφορίες για το project Biojava, κομμάτια κώδικα για την εκτέλεση βασικών βιολογικών διεργασιών και εκπαιδευτικό υλικό με ασκήσεις αντίστοιχα. Οι κατηγορίες αυτές αποτελούν τις τέσσερις από τις εννιά επιλογές του μενού που βρίσκεται στο πάνω μέρος της σελίδας.

Στο μενού υπάρχουν επίσης οι επιλογές Ερωτήσεις (με απαντήσεις στις πιο συχνές απορίες που μπορούν να δημιουργηθούν σε κάποιον που διαβάζει πρώτη φορά για το project), Πηγές (με τις πηγές από τις οποίες συλλέχθηκαν οι πληροφορίες), Χάρτης (με το χάρτη πλοήγησης του ιστοχώρου) και Επικοινωνία (με τη δυνατότητα απευθείας αποστολής μηνύματος ηλεκτρονικού ταχυδρομείου).

Στην πλάγια μπάρα, στο δεξί μέρος της σελίδας υπάρχουν επιλογές με τις οποίες οι χρήστες μπορούν να διαβάσουν ενδιαφέροντα άρθρα, τα οποία μάλιστα αποτέλεσαν πηγές για τη δημιουργία του οδηγού, να οδηγηθούν σε χρήσιμους ιστότοπους, να διεξάγουν αναζήτηση και να ενημερωθούν για τους επισκέπτες του ιστοχώρου.

Παρακάτω παρατίθενται κάποια στιγμιότυπα από τον ιστοχώρο με αντίστοιχο σχολιασμό.

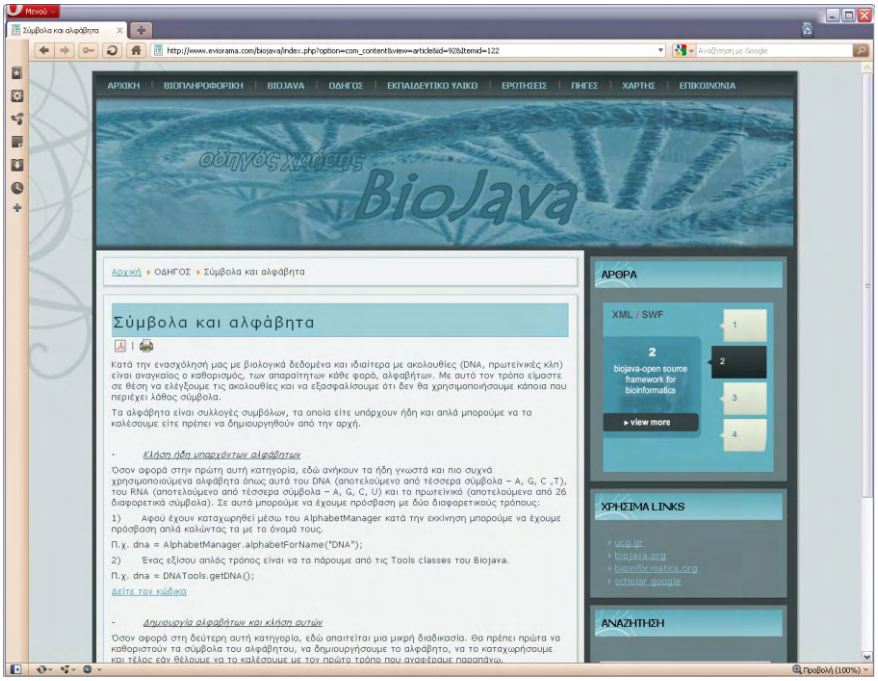


Εικόνα 30: Αρχική σελίδα του ιστοχώρου www.eviorama.com/biojava

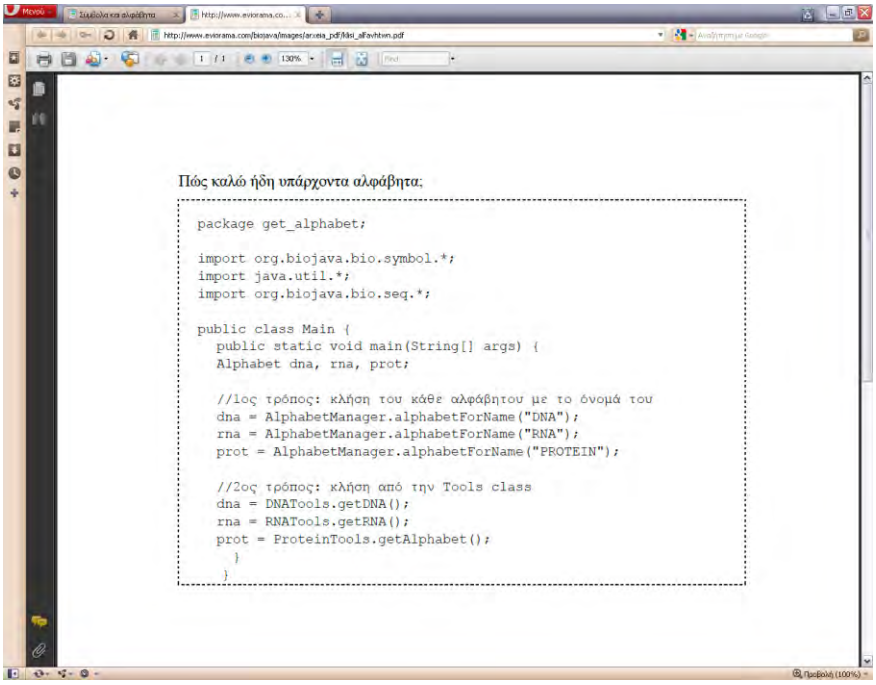
Εικόνα 32: Μενού στο πάνω μέρος της σελίδας

Εικόνα 31: Στιγμιότυπο επιλογής από το μενού

Εικόνα 33: Στιγμιότυπα από την πλάγια μπάρα



Εικόνα 35: Παράδειγμα ανάγνωσης άρθρου



Εικόνα 34: Παράδειγμα εμφάνισης κώδικα σε ξεχωριστό παράθυρο

6ο ΚΕΦΑΛΑΙΟ
ΣΥΓΚΡΙΣΗ BIOJAVA ΚΑΙ JAVA

Περίληψη
Στα πλαίσια της αξιολόγησης του project Biojava κρίθηκε απαραίτητη η διεξαγωγή συγκρίσεων του project Biojava με τη γλώσσα προγραμματισμού στην οποία βασίζεται, τη Java. Για το σκοπό αυτό δημιουργήθηκαν κώδικες σε Java οι οποίοι συγκρίθηκαν με τις αντίστοιχες πρότυπες υλοποιήσεις που παρουσιάστηκαν στο τέταρτο κεφάλαιο. Πραγματοποιήθηκαν μετρήσεις τόσο για τον χρόνο εκτέλεσης των προγραμμάτων σε κάθε περίπτωση όσο και για το ποσοστό μνήμης το οποίο απασχολούν.

Βασικές έννοιες
- Σύγκριση
- Χρόνος εκτέλεσης

6.1 Παράμετροι σύγκρισης

Για τη διεξαγωγή μιας σωστή αξιολόγησης και την απόκτηση μιας πιο ολοκληρωμένης άποψης για το Biojava, κρίθηκε απαραίτητο να γίνουν υποδειγματικές υλοποιήσεις και να μετρηθούν οι αποδόσεις του Biojava σε σχέση με τη γλώσσα προγραμματισμού Java. Οι συγκρίσεις έλαβαν χώρα σε δύο διαφορετικούς υπολογιστές, τα χαρακτηριστικά των οποίων παρουσιάζονται παρακάτω:

A. Λειτουργικό σύστημα: Windows XP Professional
   Επεξεργαστής: Intel Core 2 CPU 6400 – 2.13GHz
B. Λειτουργικό σύστημα: Windows 7 Home Premium
   Επεξεργαστής: Intel i3Core CPU M330 – 2.13GHz

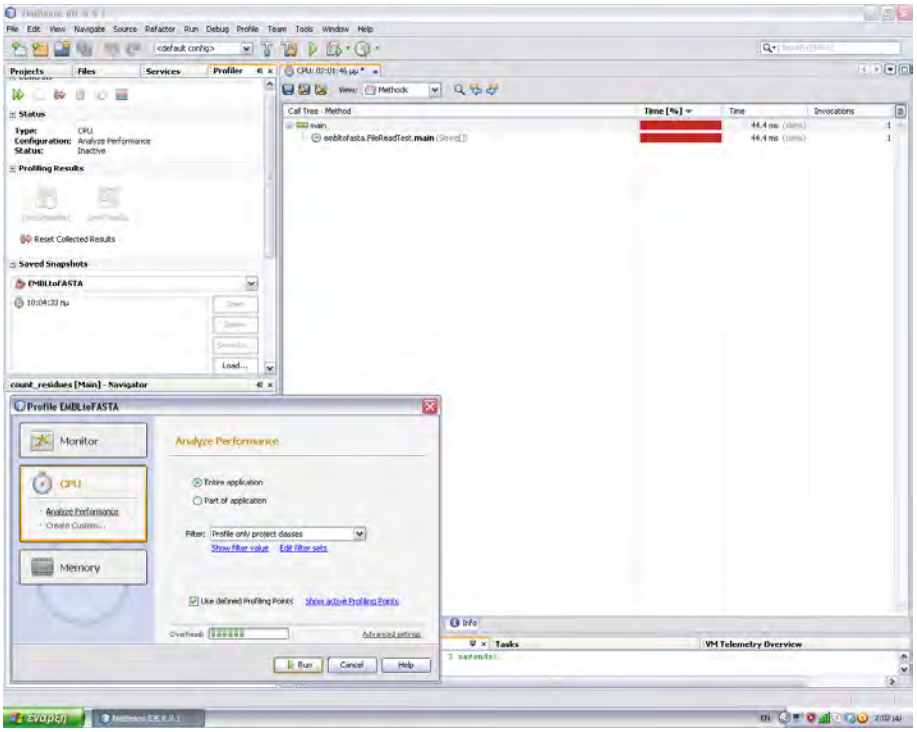
Στόχος ήταν να υπάρχει κατά το δυνατόν μεγαλύτερη ακρίβεια στις μετρήσεις, οπότε χρησιμοποιήθηκε το εργαλείο Profiler του Netbeans. Έλαβαν χώρα μετρήσεις που αφορούν τόσο τον χρόνο εκτέλεσης των προγραμμάτων όσο και τη μνήμη που δεσμεύεται στο διάστημα αυτό.

6.2 Αποτελέσματα συγκρίσεων

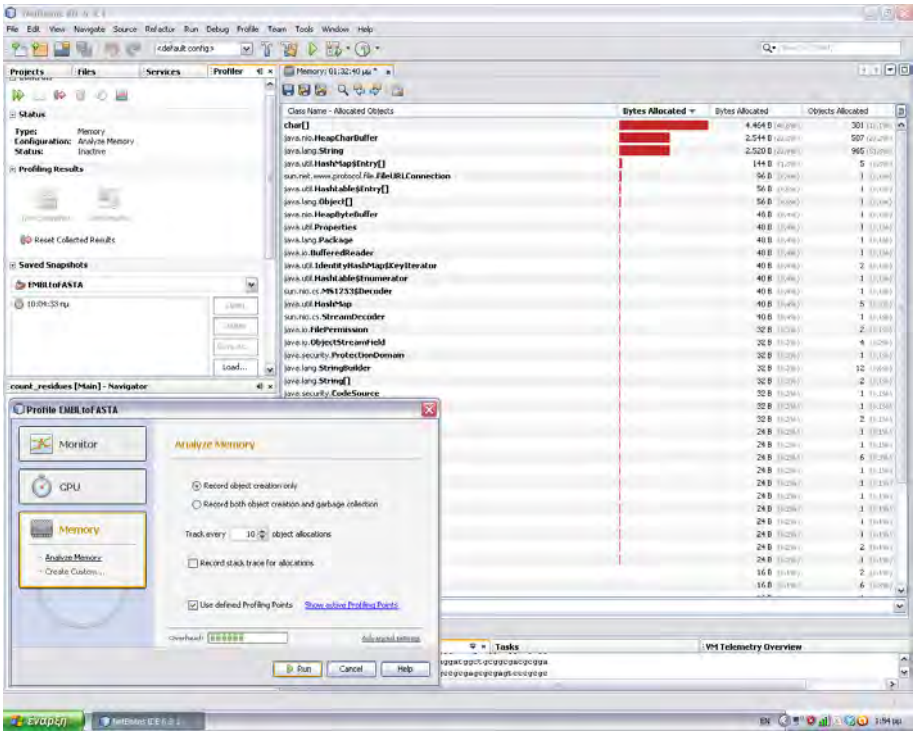
Οι μετρήσεις που ακολουθούν αφορούν τους χρόνους εκτέλεσης των προγραμμάτων (γραμμένων σε Biojava και Java) σε καθένα από τους προηγούμενους υπολογιστές και είναι μετρημένος σε χιλιοστά του δευτερολέπτου (milliseconds (ms) – $1ms=10^{-3}$ sec).

Οι μετρήσεις που αφορούν τη μνήμη δεν παρουσιάζονται εκτενώς καθώς σε όλες τις συγκρίσεις φάνηκε να χρησιμοποιείται κατά μέσο όρο το ίδιο ποσοστό μνήμης, με

τη Java να απαιτεί σχεδόν πάντα περίπου 500B παραπάνω από το αντίστοιχο πρόγραμμα γραμμένο σε Biojava.



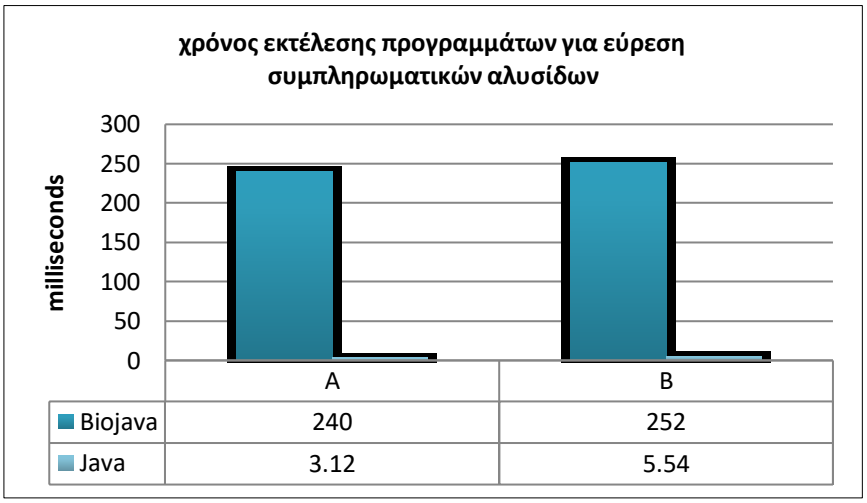
Εικόνα 36: Στιγμιότυπο από τη μέτρηση του χρόνου με το Profiler



Εικόνα 37: Στιγμιότυπο από τη μέτρηση της μνήμης με το Profiler

1η Σύγκριση

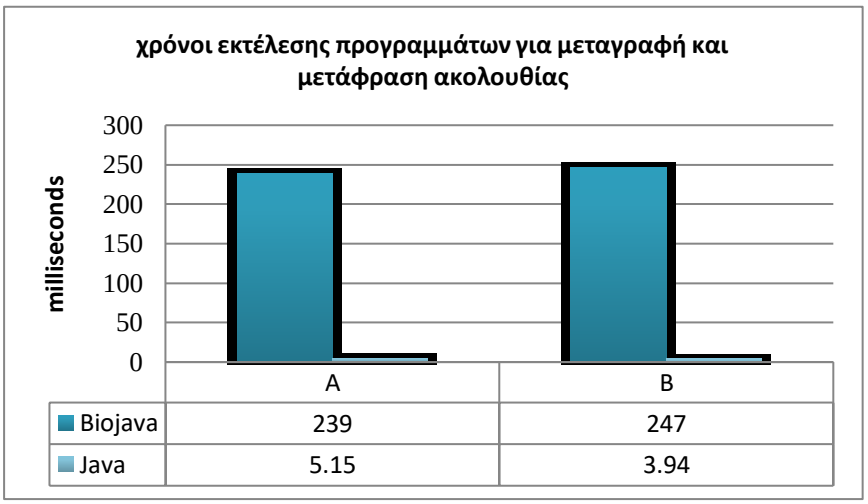
Εκτέλεση των προγραμμάτων για την εύρεση του συμπληρώματος (απλού και αντίστροφου) μιας ακολουθίας που δίνεται από το χρήστη σε Biojava και Java. (οι κώδικες παρατίθενται στα παραρτήματα 1 (κώδικας 2.4) και 3 (κώδικας 1) αντίστοιχα)



Σχήμα 15: Σύγκριση 1η

2η Σύγκριση

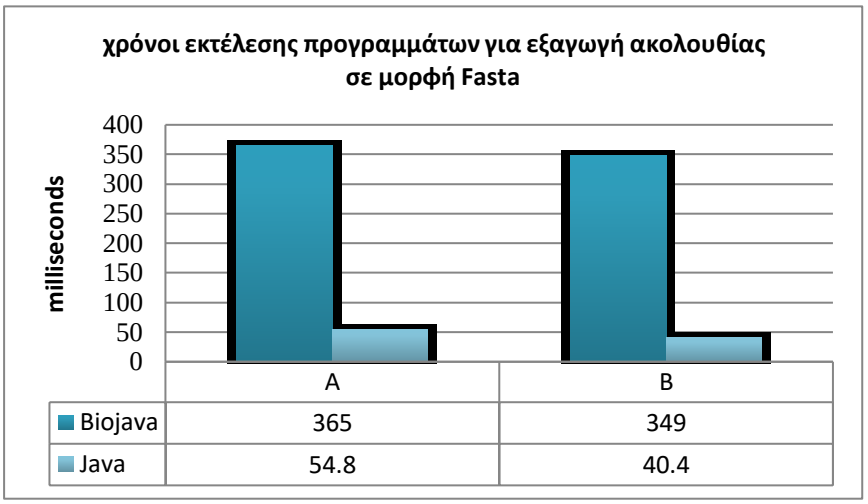
Εκτέλεση των προγραμμάτων για τη μεταγραφή και στη συνέχεια τη μετάφραση μιας τυχαίας ακολουθίας που δίνεται από το χρήστη σε Biojava και Java. (οι κώδικες παρατίθενται στα παραρτήματα 1 (κώδικας 3.1) και 3 (κώδικας 2) αντίστοιχα)



Σχήμα 16: Σύγκριση 2η

3η Σύγκριση

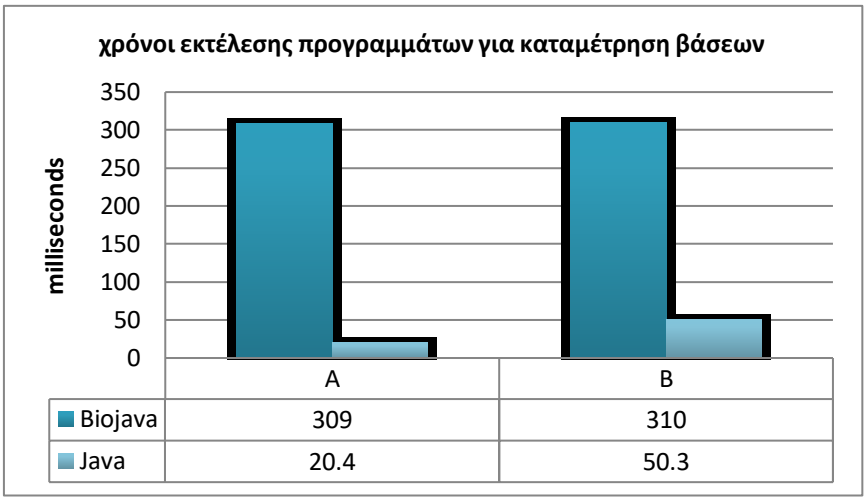
Εκτέλεση των προγραμμάτων για την ανάγνωση ενός αρχείου embl και εξαγωγή της ακολουθίας σε μορφή fasta σε Biojava και Java. (οι κώδικες παρατίθενται στα παραρτήματα 1 (κώδικας 4.3) και 3 (κώδικας 3) αντίστοιχα)



Σχήμα 17: Σύγκριση 3η

4η Σύγκριση

Εκτέλεση των προγραμμάτων για την καταμέτρηση των βάσεων σε μια ακολουθία dna που εξάγουμε από ένα αρχείο μορφής embl σε Biojava και Java. (οι κώδικες παρατίθενται στα παραρτήματα 1 (κώδικας 7.1) και 3 (κώδικας 4) αντίστοιχα)



Σχήμα 18: Σύγκριση 4η

6.3 Συμπεράσματα

Σε όλες τις παραπάνω συγκρίσεις παρατηρούμε ότι ο χρόνος εκτέλεσης των προγραμμάτων που είναι γραμμένα σε Biojava είναι μεγαλύτερος από τον αντίστοιχο των προγραμμάτων που είναι γραμμένα σε Java. Το παραπάνω αποτέλεσμα δεν ήταν αναμενόμενο, καθώς στον επίσημο ιστοχώρο δε γίνεται πουθενά αναφορά για το χρόνο εκτέλεσης των προγραμμάτων.

Γνωρίζοντας πως το Biojava είναι ένα project βασιζόμενο στη γλώσσα προγραμματισμού Java δεν θα μπορούσαμε να περιμένουμε να τη συναγωνίζεται, πόσο μάλιστα να παρουσιάζει και καλύτερες επιδόσεις. Ωστόσο, το Biojava, φαίνεται να είναι έως και 76 φορές πιο αργό από τη Java.

7ο ΚΕΦΑΛΑΙΟ
ΣΥΜΠΕΡΑΣΜΑΤΑ

Με την ολοκλήρωση της παρούσας πτυχιακής εργασίας εκπληρώθηκαν αρκετοί στόχοι, οι οποίοι είχαν τεθεί στην αρχή της εκπόνησής της.

Με το πέρας της βιβλιογραφικής μελέτης σχετικά με την επιστήμη της Βιοπληροφορικής και τα εργαλεία που χρησιμοποιούνται για την εκτέλεση βασικών βιολογικών διεργασιών κρίθηκε αναγκαίο να ταξινομηθούν και να παρουσιαστούν οι πληροφορίες που συγκεντρώθηκαν. Παρατηρήθηκε, έτσι, η μεγάλη άνθιση πάνω στον τομέα αυτό και έγινε μια προσέγγιση της ποικιλίας των ήδη υπαρχόντων γλωσσών και προγραμμάτων που χρησιμοποιούνται κατά κόρον, προκειμένου να αποκτηθεί μια γενική ιδέα για την τρέχουσα κατάσταση. Ωστόσο οι αυξημένες ανάγκες που συνεχώς προκύπτουν οδηγούν στην αναζήτηση νέων τέτοιων εργαλείων.

Το Biojava, ως ένα ώριμο πλέον project που βρίσκεται στην αιχμή του έχει αρκετά να επιδείξει. Έπειτα από ενδελεχή μελέτη του υπερσυνόλου των εντολών του Biojava πραγματοποιήθηκε μια υποδειγματική υλοποίηση τυπικών αλγορίθμων της Βιοπληροφορικής.

Η μεγάλη ευκολία στην εκμάθηση και η απλή, κοντά στο χρήστη, λογική που ακολουθεί ο τρόπος συγγραφής του κώδικα είναι τα βασικά πλεονεκτήματα που παρουσιάζει. Τα εκτενή προγράμματα που χρειάζονταν σε άλλες γλώσσες προγραμματισμού, για την υλοποίηση ακόμη και των πιο απλών αλγορίθμων, αντικαθίστανται με μικρά τμήματα κώδικα που περιέχουν την κλήση απλών μεθόδων. Εξειδικευμένες μέθοδοι και κλάσεις προσφέρονται από τις ολοκληρωμένες βιβλιοθήκες που διαθέτει.

Στα πλαίσια της παρούσας πτυχιακής εργασίας έλαβε χώρα και η σύγκριση μεταξύ του Biojava και της γλώσσας προγραμματισμού πάνω στην οποία στηρίζεται, που δεν είναι άλλη από τη γνωστή σε όλους Java. Υλοποιήθηκαν κάποια προγράμματα που αποτελούν βάση για τη Βιοπληροφορική και μετρήθηκαν οι χρόνοι εκτέλεσης και η ποσότητα μνήμης που δεσμεύεται σε κάθε περίπτωση. Με αυτό τρόπο αποκτήθηκε μια πιο τεκμηριωμένη αξιολόγηση του Biojava, η πρώτη τέτοιου τύπου, που μας βοηθά να αποκτήσουμε μια ολοκληρωμένη και σφαιρική άποψη.

Πέραν από τις διευκολύνσεις που προσφέρει, τα αποτελέσματα των συγκρίσεων που διεξήχθησαν καταδεικνύουν το μεγάλο μειονέκτημα του project. Μέχρι σήμερα πουθενά δεν υπήρχε σαφής παρουσίαση των πλεονεκτημάτων και των μειονεκτημάτων του project, αν και τα πρώτα ήταν πασιφανή. Το κύριο ελάττωμά του, δηλαδή η μικρή ταχύτητα εκτέλεσης, δεν αναφερόταν σε κάποιο σημείο. Οι μετρήσεις – χρόνοι που καταγράφονται στον επίσημο ιστοχώρο του, οι οποίες αφορούν κάποιες μικρές υλοποιήσεις, παρατίθενται χωρίς να αντιπαραβάλλονται με αντίστοιχες από άλλες γλώσσες προγραμματισμού ή

projects. Έτσι δεν είναι εύκολο να αντιληφθεί ο αναγνώστης πόσο χαμηλές ή υψηλές είναι οι ταχύτητες αυτές.

Τα αποτελέσματα δεν είναι διόλου ευκαταφρόνητα και δε μπορούν με κανένα τρόπο να παραβλεφτούν. Εν μέρει ήταν και αναμενόμενα λόγω του ότι το Biojava αποτελεί API για τη Java, συνεπώς δύσκολα θα ήταν ανταγωνιστικό ως προς την απόδοση. Υπάρχει, όμως, σημαντική διαφορά στην ταχύτητα εκτέλεσης των προγραμμάτων σε απλή γλώσσα Java και σε Biojava, πολύ μεγαλύτερη, μάλιστα, από την αναμενόμενη. Ο κώδικας Biojava παρουσιάζεται να είναι μέχρι και 76 φορές πιο αργός από τον αντίστοιχο γραμμένο σε Java. Ωστόσο, οι διαφορές αυτές, δε μπορούν να υποβιβάσουν την αξία του project και να εξαλείψουν τα πλεονεκτήματά του. Συνεχίζει να είναι ένα εύχρηστο project που μειώνει την πολυπλοκότητα, ένα API που βοηθά στη γρήγορη ανάπτυξη και προτυποποίηση αλγορίθμων που άπτονται του τομέα της Βιοπληροφορικής. Προτείνεται χωρίς επιφύλαξη η χρήση του για εκπαιδευτικούς σκοπούς αλλά όχι για επαγγελματικούς σκοπούς, όταν ο παράγοντας χρόνος δε μπορεί να παρακαμφθεί.

Η συγκέντρωση και η ανάλυση των βασικών χαρακτηριστικών, οι υλοποιήσεις των παραδειγμάτων και η αξιολόγηση έπρεπε να παρουσιαστούν συγκεντρωτικά. Το αποτέλεσμα αυτής της εργασίας δεν ήταν άλλο παρά ο οδηγός χρήσης που αναπτύχθηκε. Στα πλαίσιά του ο αναγνώστης μπορεί να βρει συγκεντρωμένες εκτενείς πληροφορίες που χρειάζεται για τα πρώτα του βήματα κατά την ενασχόλησή του με το project.

Στην προσπάθεια καλύτερης διάχυσης των πληροφοριών που προέκυψαν και λαμβάνοντας υπόψη τις ραγδαίες εξελίξεις στον τομέα της Πληροφορικής και ιδιαιτέρως την ανάπτυξη και την ολοένα και περισσότερη χρησιμοποίηση του διαδικτύου, η ανάγκη ψηφιοποίησης του οδηγού φάνηκε επιτακτική. Δημιουργήθηκε έτσι ο πρώτος ηλεκτρονικός οδηγός χρήσης του project στην ελληνική γλώσσα κάνοντας το έδαφος πρόσφορο για περεταίρω ενασχόληση με τη Biojava σε προπτυχιακό, μεταπτυχιακό ή και ερευνητικό επίπεδο. Η δημιουργία μιας εύχρηστης και φιλικής προς το χρήστη διεπαφής αποτέλεσε κατευθυντήριο άξονα κατά την ανάπτυξή του. Έγινε προσπάθεια εκπλήρωσης των απαιτήσεων και των αναγκών ευρείας κλίμακας χρηστών και ταξινόμησης και παρουσίασης του περιεχομένου με τον ευκολότερο δυνατό τρόπο.

Με την πάροδο των χρόνων δύναται να υπάρξει εξέλιξη του project. Ήδη από την αρχή της δημιουργίας του μέχρι και σήμερα έχουν λάβει χώρα αρκετές βελτιώσεις. Είναι διαθέσιμη η επέκταση BiojavaX, η οποία με τα πακέτα της ήρθε να συμπληρώσει τις λειτουργίες των μέχρι τότε διαθέσιμων. Επίσης, αξίζει να σημειωθεί πως μια νέα έκδοση με σκοπό την παροχή επιπρόσθετων διευκολύνσεων (και υψηλότερης απόδοσης) είναι ήδη υπό κατασκευή και ένα μεγάλο κομμάτι της έχει ολοκληρωθεί. Όλες αυτές οι αλλαγές περιλαμβάνονται στον ηλεκτρονικό οδηγό και στόχος είναι η ενημέρωσή του σε κάθε τέτοια εξέλιξη.

Συνοψίζοντας, μπορούμε να πούμε μετά βεβαιότητας πως μια ικανοποιητική προσέγγιση του αξιόλογου αυτού project για την ανάπτυξη εφαρμογών για τη Βιοπληροφορική έλαβε χώρα. Ο οδηγός που δημιουργήθηκε, σίγουρα δεν είναι αντίστοιχος της ολοκληρωμένης παρουσίασης του Biojava που γίνεται στον επίσημο ιστοχώρο του, ωστόσο μπορεί να αποτελέσει τη βάση για μελλοντική πληρέστερη προσέγγιση.

Οι υλοποιήσεις που παρέχονται ως υποδειγματικές αποτυπώνουν την μέθοδο με την οποία αναπτύσσει κάποιος εφαρμογές για το χώρο της Βιοπληροφορικής, και εμπλουτίζει τα υπάρχοντα παραδείγματα του Biojava. Τέλος, οι συγκρίσεις μεταξύ Java και Biojava, βάσει των υλοποιήσεων αυτών, αναδεικνύουν για πρώτη φορά και τα πεδία εφαρμογής του Biojava, που δεν είναι άλλα από την εκπαίδευση και την ανάπτυξη (γρήγορη προτυποποίηση) αλγορίθμων Βιοπληροφορικής.

ΑΝΑΦΟΡΕΣ

1. *Figure 4*. [cited 2010 26 September]; Available from: http://www.microbialcellfactories.com/content/6/1/29/figure/F4?highres=y.
2. *CLC bio: High - Performance Computing*. [cited 2010 26 August]; Available from: http://clcbio.com/index.php?id=616.
3. *Moore's law: Definitions from Answers.com*. [cited 2010 06 April]; Available from: http://www.answers.com/topic/moore-s-law.
4. *Global vs local alignment*. [cited 2010 26 September]; Available from: http://www.brc.dcs.gla.ac.uk/~drg/courses/bioinformatics_mscIT/slides/slides3/sld037.htm.
5. *File: Codons aminoacids table.png - Wikimedia Commons*. [cited 2010 26 September]; Available from: http://commons.wikimedia.org/wiki/File:Codons_aminoacids_table.png.
6. *http://checkyourhud.com/wp-content/uploads/2009/05/683px-transistor_count_and_moores_law_-_2008svg.png*. [cited 2010 31 June]; Available from: http://checkyourhud.com/wp-content/uploads/2009/05/683px-transistor_count_and_moores_law_-_2008svg.png.
7. Matthew Pocock, e.a., *Biojava: Open Sourse Components for Bioinformatics.* ACM SIGBIO Newsletter, 2000. **Volume 20**(2): p. 10-12.
8. Μπάγκος, Π. *Διάλεξη 6*. in *Διαλέξεις μαθήματος «Βιοπληροφορική Ι»*. 2008-2009. Πανεπιστήμιο Στερεάς Ελλάδος.
9. Pevsher, J., *Bioinformatics and Fuctional Genomics*. 2003: Wiley- Liss.
10. *A Complete resource site on Bioinformatics*. [cited 2010 18/03]; Available from: http://www.bioinformatics.co.nr/.
11. *Databases*. [cited 2010 18 March]; Available from: http://biophysics.biol.uoa.gr/courses/bioinformatics/DATABASES/databases.html.
12. *ExPASy – Uniprot Knowledgebases: Swiss – Prot and TrEMBL*. [cited 2010 06 April]; Available from: http://www.expasy.ch/sprot/.
13. *Welcome to PIR [Protein Information Resource]*. [cited 2010 06 April]; Available from: http://pir.georgetown.edu/.
14. *ExPASy – PROSITE*. [cited 2010 06 April]; Available from: http://www.expasy.ch/prosite/.
15. *The EMBL Nucleotide Sequence Database | EBI*. [cited 2010 06 April]; Available from: http://www.ebi.ac.uk/embl/.
16. *GenBank Overview*. [cited 2010 06 April]; Available from: http://www.ncbi.nlm.nih.gov/Genbank/index.html.
17. *DDBJ: Data Bank of Japan*. [cited 2010 06 April]; Available from: http://www.ddbj.nig.ac.jp/.
18. *RSCB Protein Data Bank*. [cited 2010 06 April]; Available from: http://www.rcsb.org/pdb/home/home.do.
19. *CATH: Protein Structure Classification Database – Prof. Orengo's Bioinformatics*. [cited 2010 06 April]; Available from: http://www.cathdb.info/.
20. *SCOP: Structural Classification of Proteins*. [cited 2010 06 April]; Available from: http://scop.mrc-lmb.cam.ac.uk/scop/index.html.
21. Μπάγκος, Π. *Διάλεξη 1*. in *Διαλέξεις μαθήματος «Βιοπληροφορική Ι»*. 2008-2009. Πανεπιστήμιο Στερεάς Ελλάδος.
22. Μπάγκος, Π. *Διάλεξη 5*. in *Διαλέξεις μαθήματος «Βιοπληροφορική Ι»*. 2008-2009. Πανεπιστήμιο Στερεάς Ελλάδος.

23. *C programming.com – Your Resource for C and C++ Programming*. [cited 2010 07 April]; Available from: http://www.cprogramming.com/.
24. H.M. Deitel, P.J.D., *C προγραμματισμός*. 2001: Μ. Γκιούρδας.
25. *Java.com: Java + You*. [cited 2010 07 April]; Available from: http://java.com/en/.
26. H.M. Deitel, P.J.D., *Java προγραμματισμός*. 2005: Μ. Γκιούρδας.
27. *The Perl Programming Language – www.perl.org*. [cited 2010 07 April]; Available from: http://www.perl.org/.
28. Randal L. Schwartz, T.C., *Μάθετε την Perl*. 1999: Κλειδάριθμος.
29. *Python Programming Language – Official Website*. [cited 2010 07 April]; Available from: http://www.python.org/.
30. *Ruby Programming Language*. [cited 2010 07 April]; Available from: http://www.ruby-lang.org/en/.
31. *BioPerl*. [cited 2010 07 April]; Available from: http://www.bioperl.org/.
32. *Biopython*. [cited 2010 07 April]; Available from: http://biopython.org/.
33. *Bioruby*. [cited 2010 07 April]; Available from: http://bioruby.org/.
34. *BioBike Documentation Index*. [cited 2010 07 April]; Available from: http://nostoc.stanford.edu/Docs/.
35. *The R project for Statistical Computing*. [cited 2010 07 April]; Available from: http://www.r-project.org/.
36. *Welcome to BioConductor – bioconductor.org*. [cited 2010 07 April]; Available from: http://www.bioconductor.org/.
37. *NEURON| for empirically- based simulations of neurons and networks of neurons*. [cited 2010 06 May]; Available from: http://www.neuron.yale.edu/neuron/.
38. *ClustalW and ClustalX Multiple Sequence Alignment*. [cited 2010 06 May]; Available from: http://www.clustal.org/.
39. *T-Coffee Home Page*. [cited 2010 06 May]; Available from: http://www.tcoffee.org/Projects_home_page/t_coffee_home_page.html.
40. *MUSCLE Multiple Accurate and Fast Sequence Comparison by Log-Expectation Tool*. [cited 2010 29 May]; Available from: http://www.ebi.ac.uk/Tools/muscle/index.html.
41. *TM4: Microarray Software Suite*. [cited 2010 29 May]; Available from: http://www.tm4.org/.
42. *GEMS: Gene Expression Model Selector*. [cited 2010 29 May]; Available from: http://www.gems-system.org/.
43. *http://www.dsl-lab.org/causal_explorer/*. [cited 2010 29 May]; Available from: http://www.dsl-lab.org/causal_explorer/.
44. *Volocity: High Performance 3D 4D imaging Software*. [cited 2010 29 May]; Available from: http://www.cellularimaging.com/products/volocity/.
45. *SVI Huygens Image Deconvolution Software from BioVision Technologies*. [cited 2010 29 May]; Available from: http://www.biovis.com/huygens.htm.
46. *AutoDock*. [cited 2010 29 May]; Available from: http://autodock.scripps.edu/.
47. *Insight II*. [cited 2010 31 June]; Available from: http://www.scripps.edu/rc/softwaredocs/msi/insight2K/insight/Intro.html.
48. *NAMD – Scalable Molecular Dynamics*. [cited 2010 29 May]; Available from: http://www.ks.uiuc.edu/Research/namd/.
49. *NMR Spectroscopy*. [cited 2010 31 June]; Available from: http://www.nmr.chem.uu.nl/education/UCU_practical/nmr_molmod_part.html.

50. *Crystallografy & NMR System*. [cited 2010 31 June]; Available from: http://cns-online.org/v1.3/.
51. *Gromacs - Fast, Flexible, Free*. [cited 2010 29 May]; Available from: http://www.gromacs.org/.
52. *Proteomics Software – ProteoIQ by BioInquire*. [cited 2010 29 May]; Available from: http://www.bioinquire.com/.
53. *ProSight PTM*. [cited 2010 31 June]; Available from: http://prosightptm2.northwestern.edu/.
54. *Matrix Science*. [cited 2010 31 June]; Available from: http://www.matrixscience.com/integra.html.
55. *Ms-lims – Project Hosting on Google Code*. [cited 2010 31 June]; Available from: http://code.google.com/p/ms-lims/.
56. *SPC Proteomics Tools*. [cited 2010 31 June]; Available from: http://tools.proteomecenter.org/software.php.
57. *OpenMS/ TOPP website*. [cited 2010 31 June]; Available from: http://open-ms.sourceforge.net/OpenMS.php.
58. *Flamel Technologies, Inc*. [cited 2010 31 June]; Available from: http://www.flamel.com/techAndProd/medusa.shtml.
59. *Cytoscape: Analyzing and Visualizing Network Data*. [cited 2010 31 June]; Available from: http://www.cytoscape.org/.
60. *BioLayout Express 3D*. [cited 2010 31 June]; Available from: http://www.biolayout.org/.
61. *OSPREY: Network Visualization System*. [cited 2010 31 June]; Available from: http://biodata.mshri.on.ca/osprey/servlet/Index.
62. *ProViz – Interactive visualization of in situ 3D protein images*. [cited 2010 31 June]; Available from: http://www.cb.uu.se/research/proviz/.
63. *Ondex: data integration and visualization*. [cited 2010 31 June]; Available from: http://www.ondex.org/.
64. *PATIKA Project Web site*. [cited 2010 31 June]; Available from: http://www.patika.org/.
65. *PIVOT 2.0 Protein Interactions VisualizatiOn Tool - Protein Analysis - Biology Software Net*. [cited 2010 31 June]; Available from: http://en.bio-soft.net/protein/PIVOT.html.
66. *Networks / Pajek*. [cited 2010 31 June]; Available from: http://vlado.fmf.uni-lj.si/pub/networks/pajek/.
67. *Dendroscope — Algorithms in Bioinformatics*. [cited 2010 31 June]; Available from: http://www-ab.informatik.uni-tuebingen.de/software/dendroscope.
68. Παυλόπουλος, κ.σ., *A reference guide for tree analysis and visualization.* BioData Mining, 2010: p. 11-13.
69. *iTOL: Interactive Tree Of Life*. [cited 2010 31 June]; Available from: http://itol.embl.de/.
70. *HyperTree hyperbolic phylogenetic tree viewer*. [cited 2010 31 June]; Available from: http://kinase.com/tools/HyperTree.html.
71. *NJPlot*. [cited 2010 31 June]; Available from: http://pbil.univ-lyon1.fr/software/njplot.html.
72. *Bioinformatics & Systems Biology*. [cited 2010 31 June]; Available from: http://bioinformatics.psb.ugent.be/hypergeny/home.php.
73. *CTree*. [cited 2010 31 June]; Available from: http://www.bioinf.manchester.ac.uk/ctree/.

74. *Phylowidget (BETA)*. [cited 2010 31 June]; Available from: http://www.phylowidget.org/.
75. *Baobab 3.31 (an editor for large phylogenetic trees written in Java)- Phylogenetic Analysis - Biology Software Net*. [cited 2010 31 June]; Available from: http://en.bio-soft.net/tree/baobab.html.
76. *MEGA 4.1b3 Molecular Evolutionary Genetics Analysis - Phylogenetic Analysis - Biology Software Net*. [cited 2010 31 June]; Available from: http://en.bio-soft.net/tree/MEGA.html.
77. *PHYLIP Home Page*. [cited 2010 31 June]; Available from: http://evolution.genetics.washington.edu/phylip.html.
78. *HCE - Hierarchical Clustering Explorer*. [cited 2010 31 June]; Available from: http://www.cs.umd.edu/hcil/hce/.
79. *Biojava*. [cited 2010 26 August]; Available from: http://biojava.org/wiki/Main_Page.
80. *http://www.oracle.com/technetwork/java/javase/downloads/index.html*. [cited 2010 07 August]; Available from: http://java.sun.com/.
81. *Welcome to NetBeans*. [cited 2010 07 August]; Available from: http://netbeans.org/.
82. *Κεντρική σελίδα ΕΛ.ΣΤΑΤ*. [cited 26 September; Available from: http://www.statistics.gr.
83. Λάλας, Χ., *Το μικρό λεξικό του Internet*: Δίαυλος.
84. Δουληγέρης, Χ., *Τεχνολογίες Διαδικτύου – αρχές λειτουργίας και προγραμματισμός εφαρμογών στο Διαδίκτυο*. 2004: Νηρηίδες.
85. *UsabilityNet: Usability resources for practitioners and managers*. [cited 2010 26 September]; Available from: http://www.usabilitynet.org/.
86. Αναγνωστόπουλος, Χ. *Διάλεξη 3*. in *Διαλέξεις μαθήματος «Επικοινωνία Ανθρώπου - Υπολογιστή»*. 2009-2010. Πανεπιστήμιο Στερεάς Ελλάδος.
87. *Artisteer – joomla template generator and wordpress theme generator*. [cited 2010 26 September]; Available from: http://www.artisteer.com/.
88. *Joomla!* [cited 2010 26 September]; Available from: http://www.joomla.org/.
89. *Joomla! Ελληνική κοινότητα υποστήριξης, .* [cited 2010 26 September]; Available from: http://www.joomla.gr/.
90. *Photoscape : Free Photo Editing Software (Photo Editor) Download*. [cited 2010 26 September]; Available from: http://www.photoscape.org/ps/main/index.php.

ΠΑΡΑΡΤΗΜΑ 1

Το παρόν παράρτημα περιέχει κώδικα γραμμένο σε Biojava για τη διεξαγωγή βασικών βιολογικών διεργασιών που παρουσιάζονται στο 4ο κεφάλαιο.

1. Τα σύμβολα και τα αλφάβητα

1.1 Πώς καλώ ήδη υπάρχοντα αλφάβητα;

```
package get_alphabet;

import org.biojava.bio.symbol.*;
import java.util.*;
import org.biojava.bio.seq.*;

public class Main {
  public static void main(String[] args) {
  Alphabet dna, rna, prot;

  //1ος τρόπος: κλήση του κάθε αλφάβητου με το όνομά του
  dna = AlphabetManager.alphabetForName("DNA");
  rna = AlphabetManager.alphabetForName("RNA");
  prot = AlphabetManager.alphabetForName("PROTEIN");

  //2ος τρόπος: κλήση από την Tools class
  dna = DNATools.getDNA();
  rna = RNATools.getRNA();
  prot = ProteinTools.getAlphabet();
    }
   }
```

1.2 Πώς δημιουργώ καινούργια αλφάβητα;

```
//Δημιουργία του δυαδικού αλφαβήτου με σύμβολα τα 0 και 1

package make_alphabet;

import org.biojava.bio.symbol.*;
import org.biojava.bio.*;
import java.util.*;

public class Main {
    public static void main(String[] args) {

    //Δημιουργία του συμβόλου zero χωρίς σχολιασμό
    Symbol zero =
        AlphabetManager.createSymbol("zero",
Annotation.EMPTY_ANNOTATION);

    //Δημιουργία του συμβόλου one χωρίς σχολιασμό
    Symbol one =
```

```
          AlphabetManager.createSymbol("one",
 Annotation.EMPTY_ANNOTATION);

     //Συγκέντρωση των συμβόλων σε μια συλλογή (set)
     Set symbols = new HashSet();
     symbols.add(zero); symbols.add(one);

     //Δημιουργία του αλφαβήτου με το όνομα Binary
     FiniteAlphabet binary = new SimpleAlphabet(symbols, "Binary");
      //Επανάληψη και εκτύπωση του αλφαβήτου για να διαπιστώσουμε αν
        δουλεύει
     System.out.println("Ta sumvola tou alphavitou einai ta
 exis:");
     for (Iterator i = binary.iterator(); i.hasNext(); ) {
       Symbol sym = (Symbol)i.next();
       System.out.println(sym.getName());
     }

     //Συνήθως καταχωρούμε τα νέα αλφάβητα με το AlphabetManager
     AlphabetManager.registerAlphabet(binary.getName(), binary);

     /*
     Με την παραπάνω εντολή καταχωρήσαμε το καινούριο αλφάβητο με
     το AlphabetManager με το όνομα "Binary". Αν ανακτήσουμε ένα
     στιγμιότυπό του χρησιμοποιώντας το όνομά του θα πρέπει να είναι
     σύμφωνο - ίδιο με το προηγούμενο στιγμιότυπο.
      */
     Alphabet alpha = AlphabetManager.alphabetForName("Binary");

     //Έλεγχος αν είναι ίδια
     System.out.print("Elegxos: ");
     System.out.println(alpha == binary);
   }
 }
```

1.3 Πώς δημιουργώ CrossProductAlphabets;

```
//Δημιουργία CrossProduct αλφαβήτων

 package crossproductalphabet;

 import java.util.*;
 import org.biojava.bio.seq.*;
 import org.biojava.bio.symbol.*;

 public class Main {
     public static void main(String[] args) {

     //Δημιουργία CrossProductAlphabet από μία λίστα
     List l = Collections.nCopies(3, DNATools.getDNA());
     Alphabet codon1 = AlphabetManager.getCrossProductAlphabet(l);
```

```
    //Δημιουργία του ίδιου αλφαβήτου βασισμένοι στο όνομα
    //(του αλφαβήτου που χρησιμοποιείται)
    Alphabet codon2 =
    AlphabetManager.generateCrossProductAlphaFromName("(DNA x DNA x
    DNA)");

    //Έλεγχος αν τα δύο αλφάβητα είναι τα ίδια
    System.out.print("Elegxos: ");
    System.out.println(codon1 == codon2);
  }
}
```

2. Χειρισμός ακολουθιών

2.1 Πώς μετατρέπω μια συμβολοσειρά σε ακολουθία (sequence) και μια ακολουθία σε συμβολοσειρά;

Μετατροπή συμβολοσειράς σε SymbolList

```
//Μετατροπή συμβολοσειράς σε SymbolList
 package string_to_symbollist;

 import org.biojava.bio.seq.*;
 import org.biojava.bio.symbol.*;

 public class Main {
     public static void main(String[] args) {
         try {
       //Δημιουργία μιας DNA SymbolList από μια συμβολοσειρά
       SymbolList dna = DNATools.createDNA("atcggtcggcttac");

       //Δημιουργία μιας RNA SymbolList από μια συμβολοσειρά
       SymbolList rna = RNATools.createRNA("auugccuacauagc");

       //Δημιουργία μιας Protein SymbolList από μια συμβολοσειρά
       SymbolList protein =
       ProteinTools.createProtein("AGFAVENDSA");
       }
        catch (IllegalSymbolException ex) {
       //Αυτό θα συμβεί εάν χρησιμοποιήσουμε στις συμβολοσειρές μας
       //κάποιον μη επιτρεπόμενο χαρακτήρα
       ex.printStackTrace();
     }
   }
 }
```

Μετατροπή συμβολοσειράς σε ακολουθία (sequence)

```
 //Μετατροπή συμβολοσειράς σε ακολουθία (sequence)
 package string_to_sequence;

 import org.biojava.bio.seq.*;
 import org.biojava.bio.symbol.*;
```

```
public class Main {
    public static void main(String[] args) {
          try {
      //Δημιουργία μιας DNA ακολουθίας με το όνομα dna_1
      Sequence dna = DNATools.createDNASequence("atgctg",
      "dna_1");

      //Δημιουργία μιας RNA ακολουθίας με το όνομα rna_1
      Sequence rna = RNATools.createRNASequence("augcug",
      "rna_1");

      //Δημιουργία μιας πρωτεϊνικής ακολουθίας με το όνομα prot_1
      Sequence prot = ProteinTools.createProteinSequence("AFHS",
      "prot_1");
    }
    catch (IllegalSymbolException ex) {
      //Θα συμβεί εάν χρησιμοποιήσουμε στις συμβολοσειρές
      //μας κάποιον μη επιτρεπόμενο χαρακτήρα
      ex.printStackTrace();
    }
   }
```

Μετατροπή SymbolList σε συμβολοσειρά

```
//Μετατροπή SymbolList σε συμβολοσειρά
package symbollist_to_string;

import org.biojava.bio.seq.*;
import org.biojava.bio.symbol.*;

public class Main {
    public static void main(String[] args) {
    SymbolList s1 = null;

        try{
        s1 = DNATools.createDNA("atcggtcggcttac");
           }catch (IllegalSymbolException ex) {
              ex.printStackTrace();
           }

    //Μετατροπή του sl σε συμβολοσειρά
    String s = s1.seqString();
    System.out.println(s);

        }
}
```

2.2 Πώς αποκτώ ένα τμήμα μιας ακολουθίας;

```
//Απόκτηση τμήματος μιας ακολουθίας
 package subsequence;
```

```
import org.biojava.bio.seq.*;
import org.biojava.bio.symbol.*;

public class Main {
    public static void main(String[] args) {
        SymbolList symL = null;

        try{
        symL = DNATools.createDNA("atcggtcggcttac");
        }catch (IllegalSymbolException ex) {
           ex.printStackTrace();
        }

//Απόκτηση του πρώτου συμβόλου
      Symbol sym = symL.symbolAt(1);
//Απόκτηση των τριών πρώτων συμβόλων
      SymbolList symL1 = symL.subList(1,3);
 //Απόκτηση των τριών τελευταίων βάσεων
      SymbolList symL2 = symL.subList(symL.length() - 3,
      symL.length());

 //Εκτύπωση του symL2
      String s = symL.subStr(symL.length() - 3, symL.length());
      System.out.print("To tria teleutaia sumvola einai ta: ");
      System.out.println(s);
    }
}
```

2.3 Πώς γίνεται η μεταγραφή DNA σε RNA;

```
//Μεταγραφή DNA σε RNA
 package transcribe_dna_to_rna;

 import org.biojava.bio.symbol.*;
 import org.biojava.bio.seq.*;

 public class Main {
     public static void main(String[] args) {

    try {
//Δημιουργία μιας DNA SymbolList
  SymbolList symL=DNATools.createDNA("atgccgaatcgtaa");

  //Εμφάνιση DNA
    System.out.print("DNA : ");
    System.out.println(symL.seqString());

  //Μεταγραφή και εμφάνιση RNA
    symL = DNATools.toRNA(symL);

    System.out.print("RNA : ");
    System.out.println(symL.seqString());
```

```
        } catch (IllegalSymbolException ex) {
          //Θα συμβεί αν χρησιμοποιήσουμε κάποιο
          //μη επιτρεπόμενο σύμβολο στη δημιουργία του DNA
          ex.printStackTrace();
        }catch (IllegalAlphabetException ex) {
          //θα συμβεί αν προσπαθήσουμε να μεταγράψουμε μια
         //SymbolList που δεν είναι DNA
          ex.printStackTrace();
       }
     }
  }
```

2.4 Πώς βρίσκω το συμπλήρωμα (αντίστροφο ή όχι) μιας αλυσίδας;

```
//Αντίστροφο συμπλήρωμα
 package reverse_complement;
 import org.biojava.bio.symbol.*;
 import org.biojava.bio.seq.*;

 public class Main {
     public static void main(String[] args) {
         SymbolList symL1,symL2;
    try {
 //Δημιουργία και εμφάνιση μιας DNA SymbolList
   SymbolList symL =   DNATools.createDNA("atgcacgggaactaa");
   System.out.print("DNA : " + symL.seqString());

 //Εύρεση του συμπληρώματος, του αντίστροφου συμπληρώματος και
 εμφάνισή τους
   symL1 = DNATools.complement(symL);
   symL2 = DNATools.reverseComplement(symL);

   System.out.print("sumplirwma : " +symL1.seqString());
  System.out.print("antistrofo sumplirwma : " + symL2.seqString());
     }
   catch (IllegalSymbolException ex) {

 //Θα συμβεί αν χρησιμοποιήσουμε κάποιο μη επιτρεπόμενο σύμβολο
 //στη δημιουργία του DNA

   ex.printStackTrace();
    }catch (IllegalAlphabetException ex) {

  //θα συμβεί αν προσπαθήσουμε να βρούμε το
  //συμπλήρωμα μιας SymbolList που δεν είναι DNA
  //(RNA)ακολουθία με το DNATools (RNATools)

  ex.printStackTrace();
         }
       }
    }
```

2.5 Πώς δημιουργώ κανονικές εκφράσεις;

```
//Δημιουργία κανονικών εκφράσεων
package regular_expression;

import org.biojava.bio.symbol.*;
import org.biojava.bio.seq.*;
import org.biojava.utils.regex.*;

public class Main {
    public static void main(String[] args) {

try {
// Απαραίτητες μεταβλητές
Matcher occurences;
FiniteAlphabet IUPAC = DNATools.getDNA();
SymbolList WorkingSequence =
DNATools.createDNA("tagagatagacgatagc");

// Δημιουργία προτύπου με το pattern factory.
Pattern pattern;
PatternFactory FACTORY =  PatternFactory.makeFactory(IUPAC);

  try{
  pattern = FACTORY.compile("wtagn");
     } catch(Exception e) {
          e.printStackTrace();
          return;}
System.out.println("Searching for:+pattern.patternAsString());

// Έλεγχος για ταιριάσματα
  try {
  occurences = pattern.matcher( WorkingSequence );
       } catch(Exception e) {
           e.printStackTrace();
           return;}

// Επανάληψη για κάθε ταίριασμα
 while( occurences.find() ) {
System.out.println("Match: " +"\t"+WorkingSequence.seqString()
+"\n"+ occurences.start() +"\t"+ occurences.group().seqString());
       }
     }catch (Exception ex) {
       ex.printStackTrace();
       System.exit(1);
    }
  }
}
```

3. Μετάφραση

3.1 Πώς γίνεται η μετάφραση DNA/ RNA SymbolList ή Sequence σε πρωτεΐνη;

```
//Μετάφραση
```

```
package translation;

import org.biojava.bio.symbol.*;
import org.biojava.bio.seq.*;

public class Main {
    public static void main(String[] args) {

try {
//Δημιουργία μιας DNA SymbolList
  SymbolList symL = DNATools.createDNA("atggccattgaatga");

//Μεταγραφή σε RNA
  symL = DNATools.toRNA(symL);

//Μετάφραση σε πρωτεΐνη
  symL = RNATools.translate(symL);

//Εμφάνιση ποτελέσματος
  System.out.println(symL.seqString());
     }catch (IllegalAlphabetException ex) {

/* Θα συμβεί αν προσπαθήσουμε να μεταγράψουμε μια ακολουθία που δεν
είναι DNA ακολουθία ή να μεταφράσουμε μια ακολουθία RNA που δεν
είναι υπό την μορφή τριπλετών/κωδικονίων*/

       ex.printStackTrace();

     }catch (IllegalSymbolException ex) {

// Θα συμβεί αν χρησιμοποιήσουμε μη επιτρεπόμενα σύμβολα για τη
//δημιουργία της DNA SymbolList
       ex.printStackTrace();
     }
   }
}
```

3.2 Πώς γίνεται η μετάφραση ενός κωδικονίου σε ένα αμινοξύ;

```
//Μετάφραση ενός κωδικονίου σε αμινοξύ
package single_translation;

import org.biojava.bio.seq.*;
import org.biojava.bio.symbol.*;

public class Main {
    public static void main(String[] args) {

//Δημιουργία ενός σύνθετου αλφαβήτου όπου τα κωδικόνια αποτελούν
σύμβολα
```

```
Alphabet a = AlphabetManager.alphabetForName("(RNA x RNA x RNA)");

//Απόκτηση του πίνακα μετάφρασης με χρήση ενός από τα ονόματα

//από την κλάση TranslationTable

 TranslationTable table =
 RNATools.getGeneticCode(TranslationTable.UNIVERSAL);

  try {

 //Δημιουργία ενός κωδικονίου

    SymbolList codon = RNATools.createRNA("UUG");

 //Αναπαράσταση του κωδικονίου σαν σύμβολο

    Symbol sym = a.getSymbol(codon.toList());


 //Μετάφραση σε αμινοξύ
    Symbol aminoAcid = table.translate(sym);

 ProteinTools.getTAlphabet().validate(aminoAcid);

 //Εμφάνιση του αμινοξέος
   System.out.println(aminoAcid.getName());

     }
     catch (IllegalSymbolException ex) {
       ex.printStackTrace();
     }
   }
 }
```

4. Είσοδοι / έξοδοι ακολουθιών

4.1 Πώς γράφω - εκτυπώνω ακολουθίες σε μορφή FASTA;

```
//Ανάγνωση/Εκτύπωση σε μορφή FASTA
package write_print_fasta;

import org.biojava.bio.seq.*;
import org.biojava.bio.seq.db.*;
import org.biojavax.bio.seq.*;

public class Main {
    public static void main(String[] args) {

SequenceDB db = new HashSequenceDB();
Sequence dna1;
Sequence dna2;

 try {
dna1 = DNATools.createDNASequence("atgctgtgg", "dna_1");
dna2 = DNATools.createDNASequence("atgctgctt", "dna_2");
```

```
  db.addSequence(dna1);
  db.addSequence(dna2);
  RichSequence.IOTools.writeFasta(System.out, db.sequenceIterator(),
  null);

       } catch (Exception e) {

           e.printStackTrace();

      }
    }
  }
```

4.2 Πώς διαβάζω ακολουθίες από ένα αρχείο FASTA;

```
//Ανάγνωση αρχείου σε μορφή FASTA
  package read_fasta;

  import java.io.*;
  import java.util.*;
  import org.biojava.bio.*;
  import org.biojava.bio.seq.db.*;
  import org.biojava.bio.seq.io.*;
  import org.biojava.bio.symbol.*;

  public class Main {
      public static void main(String[] args) {

   try {
   //Καθορισμός του προς ανάγνωση αρχείου
     String filename = args[0];
     BufferedInputStream is =
     new BufferedInputStream(new FileInputStream(filename));

   //Απόκτηση του κατάλληλου Alphabet
     Alphabet alpha = AlphabetManager.alphabetForName(args[1]);

   //Απόκτηση ενός SequenceDB όλων των ακολουθιών του αρχείου
     SequenceDB db = SeqIOTools.readFasta(is, alpha);
      }
      catch (BioException ex) {
  //αν δεν είναι σε fasta format ή λάθος αλφάβητο
        ex.printStackTrace();
      }catch (NoSuchElementException ex) {
  //αν δεν υπάρχουν fasta ακολουθίες στο φάκελο
        ex.printStackTrace();
      }catch (FileNotFoundException ex) {
  //αν υπάρχει πρόβλημα με την ανάγνωση του αρχείου
        ex.printStackTrace();
      }
    }
  }
  /*Το πρόγραμμα παίρνει 2 παραμέτρους (arguments).
  Σαν πρώτη παράμετρο args[0] παίρνει το όνομα του αρχείου που έχει
  τη fasta ακολουθία. Σαν δεύτερη παράμετρο args[1] παίρνει το όνομα
  του αλφάβητου (αποδεκτά ονόματα είναι τα DNA, RNA και PROTEIN).
  Επισήμανση: η fasta ακολουθία μπορεί να αποθηκευτεί απλά σε ένα
  txt αρχείο-αυτό που πρέπει να προσέξουμε είναι να μην υπάρχουν
```

```
κενά ανάμεσα στα σύμβολα της ακολουθίας και να αποθηκεύσουμε το
αρχείο στο directory που δουλεύουμε.
 */
```

4.3 Πώς εξάγω ακολουθίες από τις GenBank/ EMBL/ SwissProt και τις γράφω σε μορφή FASTA;

```
//Ανάγνωση αρχείων από βάσεις δεδομένων
package general_reader;

import java.io.*;
import org.biojava.bio.*;
import org.biojava.bio.seq.*;
import org.biojava.bio.seq.io.*;

public class Main {
    public static void main(String[] args) {

  try {
//Καθορισμός του προς ανάγνωση αρχείου
    BufferedReader br = new BufferedReader(new
    FileReader(args[0]));

//Δήλωση του είδους (format) αρχείου
    String format = args[1];

//Καθορισμός του αλφάβητου
    String alpha = args[2];

   SequenceIterator iter =

(SequenceIterator)SeqIOTools.fileToBiojava(format, alpha, br);


SeqIOTools.writeFasta(System.out, iter);
       }
       catch (FileNotFoundException ex) {
//Αν δεν βρίσκει το αρχείο που καθορίζεται στο args[0]
         ex.printStackTrace();
       }catch (BioException ex) {
//Αν δεν είναι έγκυρο το όνομα του είδους (format) αρχείου
         ex.printStackTrace();
       }catch (IOException ex){
//Αν παρουσιαστεί πρόβλαμα κατά την εγγραφή σε fasta
         ex.printStackTrace();
       }
    }
}
/*Το πρόγραμμα παίρνει 3 παραμέτρους (arguments).
Σαν πρώτη παράμετρο args[0] παίρνει το όνομα του αρχείου που είναι
προς ανάγνωση. Σαν δεύτερη παράμετρο args[1] παίρνει το είδος του
αρχείου. Σαν τρίτη παράμετρο args[2] παίρνει το είδος του
αλφαβήτου που θα χρησιμοποιηθεί. Προσοχή: οι δηλώσεις των
παραμέτρων θα πρέπει να συμβαδίζουν μεταξύ τους – απαγορεύονται μη
έγκυροι συνδυασμοί όπως Swissprot και DNA.
Οι επιτρεπόμενοι τύποι (formats) αρχείων είναι:
 * FASTA
 * EMBL
 * GENBANK
 * SWISSPROT (or swiss)
```

```
   * GENPEPT
  Οι επιτρεπόμενοι τύποι ακολουθιών – αλφαβήτων είναι:
   * DNA
   * AA (or Protein)
   * RNA
  */
```

5. Σχολιασμοί

5.1 Πώς καταγράφω τα σχόλια σε μια ακολουθία;

```
//Καταγραφή σχολιασμού
 package annotation;

 import java.io.*;
 import java.util.*;
 import org.biojava.bio.*;
 import org.biojava.bio.seq.*;
 import org.biojava.bio.seq.io.*;

 public class Main {
     public static void main(String[] args) {

     try {
 //Ανάγνωση ενός αρχείου - καθορισμός του ως παράμετρος
  BufferedReader br = new  BufferedReader(new FileReader(args[0]));

 //Για κάθε ακολουθία καταγραφή του σχολιασμού
  for(SequenceIterator seqs = SeqIOTools.readEmbl(br);
  seqs.hasNext(); ){
  Annotation anno = seqs.nextSequence().getAnnotation();

//Εμφάνιση
  for (Iterator i = anno.keys().iterator(); i.hasNext(); ) {
       Object key = i.next();
       System.out.println(key +" : "+ anno.getProperty(key));
          }
       }
     }
     catch (Exception ex) {
       ex.printStackTrace();
     }
   }
 }
```

6. Features

6.1 Πώς δημιουργώ ένα feature;

```
 //Δημιουργία feature
 package make_feature;

 import org.biojava.bio.*;
 import org.biojava.bio.seq.*;
 import org.biojava.bio.symbol.*;
 import org.biojava.utils.*;

 public class Main {
     public static void main(String[] args) {
```

```
//Χρήση του feature template για τη δημιουργία ενός StrandedFeature
      StrandedFeature.Template templ = new
      StrandedFeature.Template();

//Συμπλήρωση του template
     templ.annotation = Annotation.EMPTY_ANNOTATION;
     templ.location = new RangeLocation(3,6);
     templ.source = "my feature";
     templ.strand = StrandedFeature.POSITIVE;
     templ.type = "interesting motif";

     try {
//Καθορισμός της ακολουθίας
   Sequence seq = DNATools.createDNASequence("atgcgcttaag","seq1");
    System.out.println(seq.getName()+" contains
   "+seq.countFeatures()+" features");


    System.out.println("adding new feature...");

//Βλέπουμε που βρίσκεται το feature στην ακολουθία και βάζουμε σε
αυτή
//ένα δείκτη (pointer) έτσι ώστε να μπορούμε να φτιάξουμε κι άλλο
       Feature f = seq.createFeature(templ);
       System.out.println(seq.getName()+" contains
       "+seq.countFeatures()+" features");

//Χρήση ενός πανομοιότυπου template με αυτό που χρησιμοποιήθηκε για
το f
       templ = (StrandedFeature.Template)f.makeTemplate();

//Δίνουμε διαφορετική θέση και τύπο
       templ.location = new PointLocation(4);
       templ.type = "point mutation";

       System.out.println("adding nested feature...");
//Βλέπουμε το νέο feature σαν ένα εμφωλευμένο feature του f
       f.createFeature(templ);

//Η μέθοδος countFeatures() μετρά μόνο τα features του πρώτου
επιπέδου
       System.out.println(seq.getName()+" contains
       "+seq.countFeatures()+" features");
       System.out.println(f.getSource()+" contains
       "+seq.countFeatures()+" features");
     }
     catch (Exception ex) {
       ex.printStackTrace();
     }
   }
 }
```

6.2 Πώς αφαιρώ ένα feature;

```
 //Αφαίρεση ενός feature
 package remove_feature;

 import java.io.*;
 import java.util.*;
 import org.biojava.bio.*;
 import org.biojava.bio.seq.*;
```

```
import org.biojava.bio.seq.io.*;

public class Main {
    public static void main(String[] args)throws Exception {

 //Ανάγνωση ενός αρχείου EMBL
  BufferedReader br = new BufferedReader(new FileReader(args[0]));
  SequenceIterator seqI = SeqIOTools.readEmbl(br);

     while (seqI.hasNext())
        {
          Sequence seq = seqI.nextSequence();

  //Όλα τα features στο θετικό σκέλος
    FeatureHolder fh =seq.filter(new
FeatureFilter.StrandFilter(StrandedFeature.POSITIVE));

 //Επανάληψη ανάμεσα στα features
     for (Iterator i = fh.features(); i.hasNext();)
        {
 //και αφαιρείται ένα-ένα
         seq.removeFeature((Feature) i.next());
            }

 //Εμφάνιση της επεξεργασμένης ακολουθίας
         SeqIOTools.writeEmbl(System.out, seq);
        }
    }
 }
```

7. Μετρήσεις και κατανομές (counts and distributions)

7.1 Πώς μετράω τις βάσεις σε μια ακολουθία;

```
//Καταμέτρηση βάσεων
package count_residues;

import java.io.*;
import java.util.*;
import org.biojava.bio.dist.*;
import org.biojava.bio.seq.*;
import org.biojava.bio.seq.io.*;
import org.biojava.bio.symbol.*;

public class Main {
    public static void main(String[] args) {
//Αντικείμενο που κρατά τις μετρήσεις
    Count counts = null;

    try {
//Άνοιγμα του αρχείου που περιέχει την ακολουθία
  BufferedReader br = new BufferedReader(new FileReader(args[0]));

//Παίρνουμε έναν SequenceIterator για τις ακολουθίες του αρχείου
   SequenceIterator iter =
   (SequenceIterator)SeqIOTools.fileToBiojava(args[1],args[2],b r);

//Επανάληψη για κάθε ακολουθία
   while(iter.hasNext()){
```

```
          Sequence seq = iter.nextSequence();

  //Αν είναι απαραίτητο αρχικοποίηση του counts
     if(counts == null){
           counts = new
      IndexedCount((FiniteAlphabet)seq.getAlphabet());
          }

  //Επανάληψη στα σύμβολα της ακολουθίας
     for (Iterator i = seq.iterator(); i.hasNext(); ) {
          AtomicSymbol sym = (AtomicSymbol)i.next();
          counts.increaseCount(sym,1.0);
          }
        }

   //Εμφάνιση των αποτελεσμάτων
     for (Iterator i =
    ((FiniteAlphabet)counts.getAlphabet()).iterator();i.hasNext();){

       AtomicSymbol sym = (AtomicSymbol)i.next();
       System.out.println(sym.getName()+" : "+counts.getCount(sym));
        }
      }
      catch (Exception ex) {
        ex.printStackTrace();
      }
   }
}
```

7.2 Πώς υπολογίζω τη συχνότητα εμφάνισης ενός συμβόλου σε μια ακολουθία;

```
  //Υπολογισμός συχνότητας εμφάνισης μιας ακολουθίας
  package frequency;

  import org.biojava.bio.seq.*;
  import org.biojava.bio.symbol.*;
  import org.biojava.bio.dist.*;
  import org.biojava.utils.*;
  import java.util.*;

  public class Main {
      public static void main(String[] args) {

     try {
  //Δημιουργία μιας DNA SymbolList
    SymbolList dna = DNATools.createDNA("atcgctagcgtyagcntatsggca");

  //Δημιουργία μιας RNA SymbolList
    SymbolList rna = RNATools.createRNA("aucgcuaucccaggga");

  //Δημιουργία μιας πρωτεϊνικής SymbolList
     SymbolList protein =
     ProteinTools.createProtein("asrvgchvhilmkapqrt");

    SymbolList[] sla = {dna, rna, protein};

  //Κλήση του DistributionTrainerContext
    DistributionTrainerContext dtc = new
     SimpleDistributionTrainerContext();

 //Δημιουργία τριών κατανομών
```

```
Distribution dnaDist =
DistributionFactory.DEFAULT.createDistribution(dna.getAlphabet());

Distribution rnaDist =
DistributionFactory.DEFAULT.createDistribution(rna.getAlphabet());

Distribution proteinDist =
DistributionFactory.DEFAULT.createDistribution(protein.getAlphabet()
);

       Distribution[] da = {dnaDist, rnaDist, proteinDist};

  //Καταγραφή των κατανομών με το trainer
       dtc.registerDistribution(dnaDist);
       dtc.registerDistribution(rnaDist);
       dtc.registerDistribution(proteinDist);

  //Επανάληψη για κάθε ακολουθία
       for (int i = 0; i < sla.length; i++) {
  //Μέτρηση του κάθε συμβόλου στην κατάλληλη κατανομή
         for(int j = 1; j <= sla[i].length(); j++){

             dtc.addCount(da[i], sla[i].symbolAt(j), 1.0);
         }
       }

  //"Εκπαίδευση" των κατανομών
       dtc.train();

  //Εμφάνιση των αποτελεσμάτων
       for (int i = 0; i < da.length; i++) {
         for (Iterator iter =
             ((FiniteAlphabet)da[i].getAlphabet()).iterator();
              iter.hasNext(); ) {

    Symbol sym = (Symbol)iter.next();
    System.out.println(sym.getName()+" : "+da[i].getWeight(sym));
         }
         System.out.println("\n");
       }

     }
     catch (Exception ex) {
       ex.printStackTrace();
     }
   }
  }
```

7.3 Πώς μπορώ να μετατρέψω μια μέτρηση σε κατανομή;

```
  //Μετατροπή μέτρησης σε κατανομή
  package count_to_distribution;

  import org.biojava.bio.dist.*;
  import org.biojava.bio.seq.*;
  import org.biojava.bio.symbol.*;

  public class Main {
```

```
    public static void main(String[] args) {

FiniteAlphabet alpha = RNATools.getRNA();
AlphabetIndex index = AlphabetManager.getAlphabetIndex(alpha);

    try {
//Δημιουργία μιας μέτρησης
    Count c = new IndexedCount(alpha);
    c.increaseCount(RNATools.a(),35.0);
    c.increaseCount(RNATools.c(),44.0);
    c.increaseCount(RNATools.g(),68.0);
    c.increaseCount(RNATools.u(),34.0);

    System.out.println("COUNT (METRHSH)");
      for (int i = 0; i < alpha.size(); i++) {
        AtomicSymbol s = (AtomicSymbol)index.symbolForIndex(i);
        System.out.println(s.getName()+" : "+c.getCount(s));
      }

//Μετατροπή σε κατανομή
    Distribution d = DistributionTools.countToDistribution(c);

     System.out.println("\nDISTRIBUTION (KATANOMH)");
      for (int i = 0; i < alpha.size(); i++) {
        Symbol s = index.symbolForIndex(i);
        System.out.println(s.getName()+" : "+d.getWeight(s));
      }
    }
    catch (Exception ex) {
      ex.printStackTrace();
    }
  }
}
```

7.4 Πώς μπορώ να δημιουργήσω μια τυχαία ακολουθία από μια κατανομή;

```
//Δημιουργία τυχαίας ακολουθίας με βάση κάποια κατανομή
package random_sequence;

import org.biojava.bio.dist.*;
import org.biojava.bio.seq.*;
import org.biojava.bio.seq.io.*;
import java.io.*;

public class Main {
    public static void main(String[] args) {

//Δημιουργία ομοιόμορφης κατανομής του DNA αλφαβήτου
  Distribution dist = new UniformDistribution(DNATools.getDNA());

//Παραγωγή μιας τυχαίας ακολουθίας 700 βάσεων
  Sequence seq = DistributionTools.generateSequence("random seq",
   dist, 700);

    try {
 //Εμφάνιση αποτελέσματος
      SeqIOTools.writeFasta(System.out, seq);
    }
      catch (IOException ex) {
 //Αν υπάρχει λάθος στην είσοδο ή στην έξοδο
      ex.printStackTrace();
```

```
      }
      }
  }
```

7.5 Πώς βρίσκω το ποσό πληροφορίας ή την εντροπία σε μια κατανομή;

```
  //Υπολογισμός της εντροπίας και του ποσού πληροφορίας
  package entropyamount_of_information;

  import java.util.*;
  import org.biojava.bio.dist.*;
  import org.biojava.bio.seq.*;
  import org.biojava.bio.symbol.*;

  public class Main {
      public static void main(String[] args) {
      Distribution dist = null;

      try {
   //Δημιουργία της επιθυμητής κατανομής
     dist =
     DistributionFactory.DEFAULT.createDistribution(DNATools.getDNA(
     ));

  //Ορισμός του βάρους του a στο 0.97
      dist.setWeight(DNATools.a(), 0.97);

  //Ορισμός των υπολοίπων στο 0.01
      dist.setWeight(DNATools.c(), 0.01);
      dist.setWeight(DNATools.g(), 0.01);
      dist.setWeight(DNATools.t(), 0.01);
      }
      catch (Exception ex) {
        ex.printStackTrace();
        System.exit(-1);
      }

//Υπολογισμός του ποσού πληροφορίας

      double info = DistributionTools.bitsOfInformation(dist);
      System.out.println("information = "+info+" bits");
      System.out.print("\n");


//Υπολογισμός της εντροπίας (με χρήση λογαρίθμου με βάση το 2)

      HashMap entropy = DistributionTools.shannonEntropy(dist, 2.0);

//Εμφάνιση της εντροπίας κάθε βάσης

      System.out.println("Symbol\tEntropy");
      for (Iterator i = entropy.entrySet().iterator(); i.hasNext();
        ) {
        Map.Entry entry = (Map.Entry)i.next();
        Symbol sym = (Symbol)entry.getKey();
        Double val = (Double)entry.getValue();
        System.out.println(sym.getName()+ "\t" +val);
      }
    }
```

```
}
```

8. Πίνακες βαρών και δυναμικός προγραμματισμός

8.1 Πώς χρησιμοποιώ ένα πίνακα βαρών για να βρω ένα μοτίβο;

```
//Πίνακες βαρών
package weight_matrix;

import java.util.*;
import org.biojava.bio.dist.*;
import org.biojava.bio.dp.*;
import org.biojava.bio.seq.*;
import org.biojava.bio.symbol.*;

public class Main {
    public static void main(String[] args) throws Exception{

//Δημιουργία μιας στοίχισης ενός μοτίβου
    Map map = new HashMap();
    map.put("seq0", DNATools.createDNA("aggag"));
    map.put("seq1", DNATools.createDNA("aggaa"));
    map.put("seq2", DNATools.createDNA("aggag"));
    map.put("seq3", DNATools.createDNA("aagag"));
    Alignment align = new SimpleAlignment(map);

//Δημιουργία ενός πίνακα με τις κατανομές του μοτίβου
    Distribution[] dists =
    DistributionTools.distOverAlignment(align, false, 0.01);

//Δημιουργία ενός πίνακα βαρών
    WeightMatrix matrix = new SimpleWeightMatrix(dists);

//Δημιοργία ακολουθίας
     Sequence seq =
     DNATools.createDNASequence("aaagcctaggaagaggagctgat","seq");

//Σχολιασμός της ακολουθίας βάση του πίνακα βαρών με χρήση
//χαμηλού κατωφλίου(0.1)
     WeightMatrixAnnotator wma = new WeightMatrixAnnotator(matrix,
     0.1);
     seq = wma.annotate(seq);

//Εμφάνιση αποτελεσμάτων του matching
    for (Iterator it = seq.features(); it.hasNext(); ) {

      Feature f = (Feature)it.next();
      Location loc = f.getLocation();

      System.out.println("Match at " + loc.getMin()+"-
      "+loc.getMax());
      System.out.println("\tscore :
      "+f.getAnnotation().getProperty("score"));
    }
  }
}
```

8.2 Πώς κάνω ολική ή τοπική στοίχιση με βάση τον αλγόριθμο Needleman – Wunsch ή τον Smith – Waterman;

Ολική στοίχιση

```
//Ολική στοίχιση με τον αλγόριθμο Needleman Wunsch
package global_alignment_needleman_wunsch;

import java.io.File;
import org.biojava.bio.alignment.NeedlemanWunsch;
import org.biojava.bio.alignment.SequenceAlignment;
import org.biojava.bio.alignment.SubstitutionMatrix;
import org.biojava.bio.seq.DNATools;
import org.biojava.bio.seq.Sequence;
import org.biojava.bio.symbol.AlphabetManager;
import org.biojava.bio.symbol.FiniteAlphabet;

public class Main {
    public static void main(String[] args) {

    if (args.length < 3)
       throw new Error("Usage: DeterministicAlignmentDemo " +
             "querySeq targetSeq substitutionMatrixFile");
    try {
//Επιλογή του αλφαβήτου για τις ακολουθίες - εδώ επιλέγουμε το DNA
      FiniteAlphabet alphabet = (FiniteAlphabet)
      AlphabetManager.alphabetForName("DNA");
// Ανάγνωση του αρχείου με τον πίνακα substitution
// Γι αυτό το παράδειγμα ο πίνακας NUC.4.4 είναι κατάλληλος
      SubstitutionMatrix matrix = new SubstitutionMatrix(alphabet,
      new File(args[2]));
// Καθορίζουμε τα προεπιλεγμένα (default) κόστη για το χειρισμό
//των ακολουθιών για την ολική στοίχιση
      SequenceAlignment aligner = new NeedlemanWunsch(
      (short) 0,   // match - ταίριασμα
      (short) 3,   // replace - αντικατάσταση
      (short) 2,   // insert - εισαγωγή
      (short) 2,   // delete - διαγραφή
      (short) 1,   // gapExtend
       matrix      // SubstitutionMatrix
      );

  Sequence query  = DNATools.createDNASequence(args[0], "query");
  Sequence target = DNATools.createDNASequence(args[1], "target");

//Εκτέλεση στοίχισης και αποθήκευση των αποτελεσμάτων
      aligner.pairwiseAlignment(
        query, // πρώτη ακολουθία
        target // δεύτερη ακολουθία
      );
// Εμφάνιση της στοίχισης
      System.out.println("Global alignment with Needleman-
      Wunsch:\n" + aligner.getAlignmentString());

      } catch (Exception exc) {
      exc.printStackTrace();
    }
  }
}
/*Το πρόγραμμα παίρνει 3 παραμέτρους. Οι πρώτες 2 είναι οι
ακολουθίες οι οποίες θα υποβληθούν στη στοίχιση και η 3η είναι το
αρχείο που περιέχει τον πίνακα substitution. Τέτοιοι πίνακες
υπάρχουν στο ftp://ftp.ncbi.nlm.nih.gov/blast/matrices/, ώστε να
χρησιμοποιείται ο κατάλληλος κάθε φορά.
```

```
*/
```

Τοπική στοίχιση

```
//Τοπική στοίχιση με τον αλγόριθμο Smith Waterman
package local_alignment_smith_waterman;

import java.io.File;
import org.biojava.bio.alignment.SequenceAlignment;
import org.biojava.bio.alignment.SmithWaterman;
import org.biojava.bio.alignment.SubstitutionMatrix;
import org.biojava.bio.seq.DNATools;
import org.biojava.bio.seq.Sequence;
import org.biojava.bio.symbol.AlphabetManager;
import org.biojava.bio.symbol.FiniteAlphabet;

public class Main {
    public static void main(String[] args) {

      if (args.length < 3)
       throw new Error("Usage: DeterministicAlignmentDemo " +
             "querySeq targetSeq substitutionMatrixFile");
     try {
//Επιλογή του αλφαβήτου για τις ακολουθίες - εδώ επιλέγουμε το DNA
      FiniteAlphabet alphabet = (FiniteAlphabet)
      AlphabetManager.alphabetForName("DNA");
// Ανάγνωση του αρχείου με τον πίνακα substitution matrix
// Γι αυτό το παράδειγμα ο πίνακας NUC.4.4 είναι κατάλληλος
      SubstitutionMatrix matrix = new SubstitutionMatrix(alphabet,
      new File(args[2]));
// Καθορίζουμε τα προεπιλεγμένα (default) κάστη για το χειρισμό
//των ακολουθιών για την ολική στοίχιση
      SequenceAlignment aligner =aligner = new SmithWaterman(
        (short) -1,     // match - ταίριασμα
        (short) 3,      // replace - αντικατάσταση
        (short) 2,      // insert - εισαγωγή
        (short) 2,      // delete - διαγραφή
        (short) 1,      // gapExtend
        matrix  // SubstitutionMatrix
      );
  Sequence query  = DNATools.createDNASequence(args[0], "query");
  Sequence target = DNATools.createDNASequence(args[1], "target");

 // Εκτέλεση της στοίχισης
      aligner.pairwiseAlignment(query, target);
 //Εμφάνιση της στοίχισης
      System.out.println("\nlocal alignment with SmithWaterman:\n"
      + aligner.getAlignmentString());
    } catch (Exception exc) {
      exc.printStackTrace();
    }
 }
}
/*Το πρόγραμμα παίρνει 3 παραμέτρους. Οι πρώτες 2 είναι οι
ακολουθίες οι οποίες θα υποβληθούν στη στοίχιση και η 3η είναι το
αρχείο που περιέχει τον πίνακα substitution. Τέτοιοι πίνακες
υπάρχουν στο ftp://ftp.ncbi.nlm.nih.gov/blast/matrices/, ώστε να
χρησιμοποιείται ο κατάλληλος κάθε φορά.
*/
```

ΠΑΡΑΡΤΗΜΑ 2

Το παρόν παράρτημα περιέχει κώδικα γραμμένο σε Java που αφορά βασικές βιολογικές διεργασίες και χρησιμοποιήθηκε για τη διενέργεια των συγκρίσεων στα πλαίσια της 5ης ενότητας του 3ου κεφαλαίου.

1. Σύγκριση 1

Εύρεση συμπληρώματος (απλού και αντίστροφου) μιας ακολουθίας DNA.

Κώδικας Java – κλάση Main

```
package sugkrisi_1;

public class Main {
     public static void main(String[] args) {

         long start = System.currentTimeMillis();
         int i;
                 char dna[]=new char[0];

    utils test=new utils();
    String sequence="atgcacgggaactaatacaagcg";

    dna=sequence.toCharArray();

    System.out.print("DNA :                  ");
    for( i=0; i< dna.length; i++)
           System.out.print(" "+ dna[i] );

    char [] dna_c = new char [dna.length];
    char [] dna_rc = new char [dna.length];

              for( i=0; i< dna_c.length; i++){
                   dna_c[i]='0';
                   dna_rc[i]='0';
                 }

    test.complement(dna, dna_c);
    test.reverse_complement(dna, dna_rc);

    System.out.println();

 long end = System.currentTimeMillis() - start;

System.out.println("Sunolikos xronos :"+ end +" ms.");

    }

}
```

Κώδικας Java – κλάση utils

```
package sugkrisi_1;

public class utils {
  int i;

     public void complement ( char dna [], char dna_c [])
```

```
    {
         for( i=0; i< dna.length; i++)

                switch(dna[i])
            {
                case 'T': dna_c[i]='A';
                            break;
                case 't': dna_c[i]='a';
                            break;
                case 'A': dna_c[i]='T';
                            break;
                case 'a': dna_c[i]='t';
                            break;
                case 'C': dna_c[i]='G';
                            break;
                case 'c': dna_c[i]='g';
                            break;
                case 'G': dna_c[i]='C';
                            break;
                case 'g': dna_c[i]='c';
                            break;
                default: dna_c[i]=dna[i];
                            break;
            }

     System.out.println();
     System.out.print("Sumplirwma    :        ");
     for( i=0; i< dna_c.length; i++)
            System.out.print(" "+ dna_c[i] );
     }

    public void reverse_complement ( char dna [], char dna_c [])
    {
     for( i=0; i< dna.length; i++)

                switch(dna[i])
            {
                case 'T': dna_c[i]='A';
                            break;
                case 't': dna_c[i]='a';
                            break;
                case 'A': dna_c[i]='T';
                            break;
                case 'a': dna_c[i]='t';
                            break;
                case 'C': dna_c[i]='G';
                            break;
                case 'c': dna_c[i]='g';
                            break;
                case 'G': dna_c[i]='C';
                            break;
                case 'g': dna_c[i]='c';
                            break;
                default: dna_c[i]=dna[i];
                            break;
            }

      System.out.println();
       System.out.print("Antistrofo sumplirwma :");
```

```
          for( i=0; i< dna_c.length; i++)

              System.out.print (" " +dna_c[ (dna_c.length -1 -i) ] );
    }
}
```

2. Σύγκριση 2

Ανάγνωση μιας ακολουθίας DNA, μεταγραφή της σε RNA και μετάφραση σε πρωτεΐνη.

Κώδικας Java – κλάση Main

```
package sugkrisi_2;

import java.io.*;

public class Main {
    public static void main(String[] args)throws IOException {

      int i, size;

      char [] dna = new char [0];
      char trans_rna[]=new char[1000];

      String sequence="atggccattgaatga";

      dna=sequence.toCharArray();

      RNA element = new RNA ("test",dna.length);
      element.transcribe(dna, element.rna);

      System.out.print("DNA :  ");
      for( i=0; i< dna.length; i++)
                  System.out.print(" "+ dna[i] );

      System.out.println();

      element.printRNA();

      System.out.print( "METAFRASI :" );

      size=element.translate (  element.rna ,  trans_rna );

      for( i=0; i< size; i++)
            System.out.print(" "+ trans_rna[i] );

      System.out.println();
    }
}
```

Κώδικας Java – κλάση RNA

```
package sugkrisi_2;
public class RNA {
    public String label;
    public char rna[];
    public int size;

// μέθοδος δημιουργός  αρχικοποιεί ένα αντικείμενο RNA με αυτές τις
τιμές
```

```
    public RNA (   String label, int size)
        {
        int i;
            this.label=label;
            this.size=size;

            rna=new char[size];

        for( i=0; i< size; i++)
                    rna[i]='0';
        }
    void transcribe ( char dna [] ,  char rna[] )
    {
        int i;

        for( i=0; i< dna.length; i++)

            switch(dna[i])
            {
                case 'T': rna[i]='U';
                            break;
                case 't': rna[i]='U';
                            break;
                case 'a':rna[i]='A';
                            break;
                case 'c':rna[i]='C';
                            break;
                case 'g':rna[i]='G';
                            break;
                case 'u':rna[i]='U';
                            break;
                default: rna[i]=dna[i];
                            break;
            }
    }
    void printRNA()
    {
    int i;

   // System.out.println("RNA name : " +label);

      System.out.print("RNA :  ");

       for( i=0; i< rna.length; i++)
            System.out.print(" "+ rna[i] );
      System.out.println();
    }
     int translate ( char rna [] ,  char trans_rna[] )
    {
       int i=0, k=0;
       char [] temp=new char[3];
```

```
       while( i< rna.length){
             temp[k++]=rna[i];
             if(k==3 && i < rna.length){
                  if( temp[0]=='A'&&temp[1]=='U'&&temp[2]=='G')
                      {
                        i=i-2;
                        utils neo=new utils();
                        return neo.help_translate(rna, i, trans_rna);
                      }
                 else
                        k=0;
                        i=i-2;
                       }
                }
                 i++;
       }
        return 0;
     }
   }
```

Κώδικας Java – κλάση utils

```
package sugkrisi_2;
public class utils {


    int help_translate( char rna[],int start,char trans_rna[])
    {
         int i, j=0, k=0;

          char [] temp=new char[3];

         for( i =start; i< rna.length; i++)
         {
             temp[k++]=rna[i];

               if(k==3 && i < rna.length){

 // System.out.println("Checking :" +temp[0]+""+temp[1]+""+temp[2]);

   if( temp[0]=='A'&&temp[1]=='U'&&temp[2]=='G')
        trans_rna[j++]='M';

   if(temp[0] == 'U' && temp[1] == 'U' && temp[2] == 'U' )
        trans_rna[j++]='F';
   if(temp[0] == 'U' && temp[1] == 'U' && temp[2] == 'C' )
        trans_rna[j++]='F';

   if(temp[0] == 'U' && temp[1] == 'U' && temp[2] == 'G' )
        trans_rna[j++]='L';
   if(temp[0] == 'U' && temp[1] == 'U' && temp[2] == 'A' )
        trans_rna[j++]='L';
   if(temp[0] == 'C' && temp[1] == 'U' && temp[2] == 'U' )
        trans_rna[j++]='L';
   if(temp[0] == 'C' && temp[1] == 'U' && temp[2] == 'C' )
        trans_rna[j++]='L';
   if(temp[0] == 'C' && temp[1] == 'U' && temp[2] == 'A' )
        trans_rna[j++]='L';
   if(temp[0] == 'C' && temp[1] == 'U' && temp[2] == 'G' )
```

```
        trans_rna[j++]='L';
    if(temp[0] == 'A' && temp[1] == 'U' && temp[2] == 'U')
        trans_rna[j++]='I';
    if(temp[0] == 'A' && temp[1] == 'U' && temp[2] == 'C')
        trans_rna[j++]='I';
    if(temp[0] == 'A' && temp[1] == 'U' && temp[2] == 'A')
        trans_rna[j++]='I';

    if(temp[0] == 'G' && temp[1] == 'U' && temp[2] == 'U')
        trans_rna[j++]='V';
    if(temp[0] == 'G' && temp[1] == 'U' && temp[2] == 'C')
        trans_rna[j++]='V';
    if(temp[0] == 'G' && temp[1] == 'U' && temp[2] == 'A')
        trans_rna[j++]='V';
    if(temp[0] == 'G' && temp[1] == 'U' && temp[2] == 'G')
        trans_rna[j++]='V';

    if(temp[0] == 'U' && temp[1] == 'C' && temp[2] == 'U')
        trans_rna[j++]='S';
    if(temp[0] == 'U' && temp[1] == 'C' && temp[2] == 'C')
        trans_rna[j++]='S';
    if(temp[0] == 'U' && temp[1] == 'C' && temp[2] == 'A')
        trans_rna[j++]='S';
    if(temp[0] == 'U' && temp[1] == 'C' && temp[2] == 'G')
        trans_rna[j++]='S';
    if(temp[0] == 'A' && temp[1] == 'G' && temp[2] == 'U')
        trans_rna[j++]='S';
    if(temp[0] == 'A' && temp[1] == 'G' && temp[2] == 'C')
        trans_rna[j++]='S';

    if(temp[0] == 'C' && temp[1] == 'C' && temp[2] == 'U')
        trans_rna[j++]='P';
    if(temp[0] == 'C' && temp[1] == 'C' && temp[2] == 'C')
        trans_rna[j++]='P';
    if(temp[0] == 'C' && temp[1] == 'C' && temp[2] == 'A')
        trans_rna[j++]='P';
    if(temp[0] == 'C' && temp[1] == 'C' && temp[2] == 'G')
        trans_rna[j++]='P';

    if(temp[0] == 'A' && temp[1] == 'C' && temp[2] == 'U')
        trans_rna[j++]='T';
    if(temp[0] == 'A' && temp[1] == 'C' && temp[2] == 'C')
        trans_rna[j++]='T';
    if(temp[0] == 'A' && temp[1] == 'C' && temp[2] == 'A')
        trans_rna[j++]='T';
    if(temp[0] == 'A' && temp[1] == 'C' && temp[2] == 'G')
        trans_rna[j++]='T';

    if(temp[0] == 'G' && temp[1] == 'C' && temp[2] == 'U')
        trans_rna[j++]='A';
    if(temp[0] == 'G' && temp[1] == 'C' && temp[2] == 'C')
        trans_rna[j++]='A';
    if(temp[0] == 'G' && temp[1] == 'C' && temp[2] == 'A')
        trans_rna[j++]='A';
    if(temp[0] == 'G' && temp[1] == 'C' && temp[2] == 'G')
        trans_rna[j++]='A';

    if(temp[0] == 'U' && temp[1] == 'A' && temp[2] == 'U')
        trans_rna[j++]='Y';
```

```
    if(temp[0] == 'U' && temp[1] == 'A' && temp[2] == 'C')
         trans_rna[j++]='Y';

    if(temp[0] == 'C' && temp[1] == 'A' && temp[2] == 'U')
         trans_rna[j++]='H';
    if(temp[0] == 'C' && temp[1] == 'A' && temp[2] == 'C')
         trans_rna[j++]='H';

    if(temp[0] == 'C' && temp[1] == 'A' && temp[2] == 'A')
         trans_rna[j++]='Q';
    if(temp[0] == 'C' && temp[1] == 'A' && temp[2] == 'G')
         trans_rna[j++]='Q';

    if(temp[0] == 'A' && temp[1] == 'A' && temp[2] == 'U')
         trans_rna[j++]='N';
    if(temp[0] == 'A' && temp[1] == 'A' && temp[2] == 'C')
         trans_rna[j++]='N';

    if(temp[0] == 'A' && temp[1] == 'A' && temp[2] == 'A')
         trans_rna[j++]='K';
    if(temp[0] == 'A' && temp[1] == 'A' && temp[2] == 'G')
         trans_rna[j++]='K';

    if(temp[0] == 'G' && temp[1] == 'A' && temp[2] == 'U')
         trans_rna[j++]='D';
    if(temp[0] == 'G' && temp[1] == 'A' && temp[2] == 'C')
         trans_rna[j++]='D';

    if(temp[0] == 'G' && temp[1] == 'A' && temp[2] == 'A')
         trans_rna[j++]='E';
    if(temp[0] == 'G' && temp[1] == 'A' && temp[2] == 'G')
         trans_rna[j++]='E';

    if(temp[0] == 'U' && temp[1] == 'G' && temp[2] == 'U')
         trans_rna[j++]='C';
    if(temp[0] == 'U' && temp[1] == 'G' && temp[2] == 'C')
         trans_rna[j++]='C';

    if(temp[0] == 'U' && temp[1] == 'G' && temp[2] == 'G')
         trans_rna[j++]='W';

    if(temp[0] == 'C' && temp[1] == 'G' && temp[2] == 'U')
         trans_rna[j++]='R';
    if(temp[0] == 'C' && temp[1] == 'G' && temp[2] == 'C')
         trans_rna[j++]='R';
    if(temp[0] == 'C' && temp[1] == 'G' && temp[2] == 'A')
         trans_rna[j++]='R';
    if(temp[0] == 'C' && temp[1] == 'G' && temp[2] == 'G')
         trans_rna[j++]='R';
    if(temp[0] == 'A' && temp[1] == 'G' && temp[2] == 'A')
         trans_rna[j++]='R';
    if(temp[0] == 'A' && temp[1] == 'G' && temp[2] == 'G')
         trans_rna[j++]='R';

    if(temp[0] == 'G' && temp[1] == 'G' && temp[2] == 'U')
         trans_rna[j++]='G';
    if(temp[0] == 'G' && temp[1] == 'G' && temp[2] == 'C')
         trans_rna[j++]='G';
    if(temp[0] == 'G' && temp[1] == 'G' && temp[2] == 'A')
         trans_rna[j++]='G';
    if(temp[0] == 'G' && temp[1] == 'G' && temp[2] == 'G')
```

```
        trans_rna[j++]='G';

   if(temp[0] == 'U' && temp[1] == 'A' && temp[2] == 'A')
                {
                  trans_rna[j++] = '*';
                  return j;
                           }
   if(temp[0] == 'U' && temp[1] == 'A' && temp[2] == 'G')
                {
                  trans_rna[j++] = '*';
                  return j;
                           }
   if (temp[0] == 'U' && temp[1] == 'G' && temp[2] == 'A')
                {
                  trans_rna[j++] = '*';
                  return j;
                           }
                  k=0;
                }
            }
             return j;
    }
}
```

3. Σύγκριση 3

Ανάγνωση ενός αρχείου embl και εξαγωγή της ακολουθίας σε μορφή fasta.

Κώδικας Java – κλάση FileReadText

```
package sugkrisi_3;

import java.io.*;

  class FileReadTest {
     public static void main (String[] args) {

        String record = null, roar =null, id=null;
        int recCount = 0, flag=0, mode=0;
        String filename="embl.txt";

        mode=args.length;
   if (mode==1){
    System.out.println("You provided " + args.length + " arguments");
    System.out.println("Opening file " + args[0] );
    filename=args[0];
                }
   else
    System.out.println("Opening file " + filename+" \n The FASTA ARE
    : \n");

        try {
           FileReader fr = new FileReader(filename);
           BufferedReader br = new BufferedReader(fr);

           record = new String();
           roar = new String();
           id= new String();

           System.out.print(">");
```

```
            while ((record = br.readLine()) != null) {
               recCount++;

               id=record.substring(0,2);
               if(id.equals("AC") || id.equals("DE") )
               {
                     roar = record.substring(6);
                     System.out.print(roar);
                     System.out.print(" ");
               }
               if(flag==1){
                              roar = record.substring(5,15 );
                                System.out.print(roar);
                              roar = record.substring(16,26 );
                                System.out.print(roar);
                              roar = record.substring(27,37 );
                                System.out.print(roar);
                              roar = record.substring(38,48 );
                                System.out.print(roar);
                              roar = record.substring(49,59 );
                                System.out.print(roar);
                    System.out.println();
               }
                if(id.equals("SQ")){
                   System.out.println(" ");
                   flag=1;
               }
            }
         } catch (IOException e) {
            // catch possible io errors from readLine()
            System.out.println("Uh oh, got an IOException error!");
            e.printStackTrace();
         }
      }

      void readMyFile() {

         String record = null, roar =null, id=null;
         int recCount = 0, flag=0;

         try {
            FileReader fr = new FileReader("embl.txt");
            BufferedReader br = new BufferedReader(fr);

            record = new String();
            roar = new String();
            id= new String();

            System.out.print(">");
            while ((record = br.readLine()) != null) {
               recCount++;

               id=record.substring(0,2);
               if(id.equals("AC") || id.equals("DE") )
               {
                     roar = record.substring(6);
                     System.out.print(roar);
                     System.out.print(" ");
               }
               if(flag==1){
                              roar = record.substring(5,59 );
```

```
                                System.out.println(roar);
                }
                 if(id.equals("SQ")){
                    System.out.println(" ciao");
                    flag=1;
                }
             }
          } catch (IOException e) {
             // catch possible io errors from readLine()
             System.out.println("Uh oh, got an IOException error!");
             e.printStackTrace();
          }
       }
    }
```

4. Σύγκριση 4

Ανάγνωση αρχείου embl και καταμέτρηση των βάσεων της ακολουθίας.

```
package embltofasta;

import java.io.*;

  class FileReadTest {

     public static void main (String[] args) {

        String record = null, roar =null, id=null, str=null;
        int recCount = 0, flag=0, mode=0, i;
        char [] c_ar = new char [0];
        String filename="embl.txt";

     mode=args.length;
     if (mode==1){
     filename=args[0];
                 }
     else
System.out.println("Opening file " + filename+" \n The FASTA IS :
\n");

        try {
           FileReader fr = new FileReader(filename);
           BufferedReader br = new BufferedReader(fr);

           record = new String();
           roar = new String();
           id= new String();
           str= new String();

           System.out.print(">");
           while ((record = br.readLine()) != null) {
              recCount++;

              id=record.substring(0,2);
              if(id.equals("AC") || id.equals("DE") )
              {
                    roar = record.substring(5);
              }
```

```
            if(flag==1){
                  roar = record.substring(5,15 );
                  str+=roar;
                  roar = record.substring(16,26 );
                  str+=roar;
                  roar = record.substring(27,37 );
                  str+=roar;
                  roar = record.substring(38,48 );
                  str+=roar;
                  roar = record.substring(49,59 );
                  str+=roar;
                  roar = record.substring(60,70 );
                  str+=roar;
            }
             if(id.equals("SQ")){
                flag=1;
            }

          }

      } catch (IOException e) {
      // catch possible io errors from readLine()
         System.out.println("IOException error!");
         e.printStackTrace();
      }

      c_ar=str.toCharArray();

        for( i=0; i< c_ar.length; i++){
           switch(c_ar[i])
          {

              case 't': c_ar[i]='T';
                        break;
              case 'a': c_ar[i]='A';
                        break;
              case 'c':c_ar[i]='C';
                        break;
              case 'g':c_ar[i]='G';
                        break;

          }
        }

   float numA=0, numC=0, numG=0, numT=0;
   for( i=0; i<c_ar.length; i++)
       switch (c_ar[i] )
                        {
       case 'A' : numA++;
            break;
       case 'C' : numC++;
            break;
       case 'G' : numG++;
            break;
       case 'T' : numT++;
            break;
                       }

        System.out.println();

             System.out.println("Adenine :"+  numA );
```

```
                System.out.println("Cytosine :"+  numC );
                System.out.println("Guanine :"+  numG );
                System.out.println("Thymine :"+  numT );
}

} // end of class
```

ΠΑΡΑΡΤΗΜΑ 3

Στο παρόν παράρτημα καταγράφονται τα ολοκληρωμένα παραδείγματα - προβλήματα που παρουσιάζονται στην 3η ενότητα του 4ου κεφαλαίου. Παρατίθενται ο πηγαίος κώδικας, οι είσοδοι όπου αυτές αποτελούν ξεχωριστά αρχεία και τα αποτελέσματα.

1. Πρόβλημα 1

Ανάγνωση μιας ακολουθίας σε μορφή fasta και εύρεση του αντίστροφου συμπληρώματος. (παράρτημα)

Πηγαίος κώδικας

```
package problem1;

import org.biojava.bio.seq.impl.RevCompSequence;
import org.biojavax.bio.seq.RichSequence;
import org.biojavax.bio.seq.RichSequenceIterator;
import org.biojavax.bio.seq.io.FastaFormat;
import org.biojavax.bio.seq.io.FastaHeader;

import java.io.BufferedReader;
import java.io.FileReader;


public class Main {
    public static void main(String[] args)throws Exception {

 String fastaformat= args[0];


FastaHeader fastaHeader = new FastaHeader();
            fastaHeader.setShowAccession(false);
            fastaHeader.setShowDescription(false);
            fastaHeader.setShowIdentifier(false);
            fastaHeader.setShowName(false);
            fastaHeader.setShowNamespace(false);
            fastaHeader.setShowVersion(false);

            FastaFormat fastaFormat = new FastaFormat();
            fastaFormat.setHeader(fastaHeader);
            fastaFormat.setLineWidth(60);


 BufferedReader br = new BufferedReader(new FileReader(fastaformat));
 RichSequenceIterator iter = RichSequence.IOTools.readFastaDNA(br ,
null);

 while(iter.hasNext()) {

      RichSequence seq = iter.nextRichSequence();
      System.out.println("Akolouthia dna tou "+seq.getDescription());
      fastaFormat.writeSequence(seq, System.out);

      RevCompSequence rev = new RevCompSequence(seq);
      System.out.println();
```

```
      System.out.println("Antistrofo sumplirwma");
      fastaFormat.writeSequence(rev, System.out);
    }
  }
}
```

Είσοδος (τμήμα ακολουθίας σε μορφή fasta)

```
>gi|50302071|gb|AY675564.1| Mus musculus molossinus mitochondrion,
complete genome
GTTAATGTAGCTTAATAACAAAGCAAAGCACTGAAAATGCTTAGATGGATAATTTTATCC
CATAAACACAAAGGTTTGGTCCTGGCCTTATAATTAATTAGAGGTAAAGTTACACATGCA
AACCTCCATAGACCGGTGTAAAATCCCTTAAACATTTGCTTAAACTTTAAGGAGAGGGTA
TCAAGCACATTAAAATAGCTTAAGACACCTTGCCTAGCCACACCCCCACGGGACTCAGCA
GTGATAAATATTAAGCAATAAACGAAAGTTTGACTAAGTTATACCTCTTAGGGTTGGTAA
ATTTCGTGCCAGCCACCGCGGTCATACGATTAACCCAAACTAATTATTTTCGGCGTAAAA
CGTGTCAACTATAAATAAATAAATAGAATTAAAATCCAACTTATATGTGAAAATTCATTG
TTAGGACCTAAACTCAATAACGAAAGTAATTCTAATCATTTATAATACACGACAGCTAAG
ACCCAAACTGGGATTAGATACCCCACTATGCTTAGCCATAAACCTAAATAATTAAATTTA
ACAAAACTATTTGCCAGAGAACTACTAGCCATAGCTTAAAACTCAAAGGACTTGGCGGTA
CTTTATATCCATCTAGAGGAGCCTGTTCTATAATCGATAAACCCCGCTCTACCTCACCAT
CTCTTGCTAATTCAGCCTATATACCGCCATCTTCAGCAAACCCTAAAAAGGTATTTAAGT
AAGCAAAAGAATCAAACATAAAAACGTTAGGTCAAGGTGTAGCCAATGAA
```

Αποτέλεσμα – Έξοδος

```
Akolouthia dna tou Mus musculus molossinus mitochondrion, complete
genome
>
gttaatgtagcttaataacaaagcaaagcactgaaaatgcttagatggataattttatcc
cataaacacaaaggtttggtcctggccttataattaattagaggtaaagttacacatgca
aacctccatagaccggtgtaaaatcccttaaacatttgcttaaactttaaggagagggta
tcaagcacattaaaatagcttaagacaccttgcctagccacacccccacgggactcagca
gtgataaatattaagcaataaacgaaagtttgactaagttatacctcttagggttggtaa
atttcgtgccagccaccgcggtcatacgattaacccaaactaattattttcggcgtaaaa
cgtgtcaactataaataaataaatagaattaaaatccaacttatatgtgaaaattcattg
ttaggacctaaactcaataacgaaagtaattctaatcatttataatacacgacagctaag
acccaaactgggattagataccccactatgcttagccataaacctaaataattaaattta
acaaaactatttgccagagaactactagccatagcttaaaactcaaaggacttggcggta
ctttatatccatctagaggagcctgttctataatcgataaaccccgctctacctcaccat
ctcttgctaattcagcctatataccgccatcttcagcaaaccctaaaaaggtatttaagt
aagcaaaagaatcaaacataaaaacgttaggtcaaggtgtagccaatgaa

Antistrofo sumplirwma
>
ttcattggctacaccttgacctaacgttttatgtttgattcttttgcttacttaaatac
ctttttagggtttgctgaagatggcggtatataggctgaattagcaagagatggtgaggt
agagcggggtttatcgattatagaacaggctcctctagatggatataaagtaccgccaag
tcctttgagttttaagctatggctagtagttctctggcaaatagttttgttaaatttaat
tatttaggtttatggctaagcatagtggggtatctaatcccagtttgggtcttagctgtc
gtgtattataaatgattagaattactttcgttattgagtttaggtcctaacaatgaattt
tcacatataagttggattttaattctatttatttatttatagttgacacgttttacgccg
aaaataattagtttgggttaatcgtatgaccgcggtggctggcacgaaatttaccaaccc
taagaggtataacttagtcaaactttcgtttattgcttaatatttatcactgctgagtcc
cgtgggggtgtggctaggcaaggtgtcttaagctattttaatgtgcttgataccctctcc
ttaaagtttaagcaaatgtttaagggattttacaccggtctatggaggtttgcatgtgta
actttacctctaattaattataaggccaggaccaaacctttgtgtttatgggataaaatt
atccatctaagcattttcagtgctttgctttgttattaagctacattaac
```

2. Πρόβλημα 2

Παρουσίαση της δυνατότητας απευθείας πρόσβασης σε βάσεις δεδομένων.

Πηγαίος κώδικας

```
package problem2;

import org.biojava.bio.BioException;
import org.biojava.bio.symbol.SymbolList;
import org.biojavax.bio.db.ncbi.GenbankRichSequenceDB;
import org.biojavax.bio.db.ncbi.GenpeptRichSequenceDB;
import org.biojavax.bio.seq.RichSequence;

public class Main {

    public static void main(String[] args) {

  //πρόσβαση στη Genbank
      System.out.println("Prosvasi sti Genbank");
        RichSequence rs1 = null;

      GenbankRichSequenceDB geng = new GenbankRichSequenceDB();
      try{
      // κλήση με τη βοήθεια του αριθμού πρόσβασης (accession number)
      rs1 = geng.getRichSequence("M98343");
      System.out.println(rs1.getName()+" | "+rs1.getDescription());
      SymbolList sl = rs1.getInternalSymbolList();
      System.out.println(sl.seqString());

      // κλήση με τη βοήθεια του αριθμού gi
      rs1 = geng.getRichSequence("182086");
      System.out.println(rs1.getName()+" | "+rs1.getDescription());
      sl = rs1.getInternalSymbolList();
      System.out.println(sl.seqString());

      }
      catch(BioException be){
      be.printStackTrace();
      System.exit(-1);
      }
      System.out.println();

  //πρόσβαση στη Genpept
      System.out.println("Prosvasi sti Genpet");
      RichSequence rs2 = null;

      GenpeptRichSequenceDB genp = new GenpeptRichSequenceDB();
      try{
      // κλήση με τη βοήθεια του αριθμού πρόσβασης (accession number)
         rs2 = genp.getRichSequence("AAD51968");
      System.out.println(rs2.getName()+" | "+rs2.getDescription());
      SymbolList s2 = rs2.getInternalSymbolList();
      System.out.println(s2.seqString());

      // κλήση με τη βοήθεια του αριθμού πρόσβασης gi
      rs2 = genp.getRichSequence("5805369");
      System.out.println(rs2.getName()+" | "+rs2.getDescription());
      s2 = rs2.getInternalSymbolList();
      System.out.println(s2.seqString());
```

```
      }
      catch(BioException be){
      be.printStackTrace();
      System.exit(-1);
      }
    }
  }
```

Αποτέλεσμα – Έξοδος

```
Prosvasi sti Genbank
HUMEMS | Homo sapiens amplaxin (EMS1) mRNA, complete cds.
gcctggtgcctgggagcggctggcgcggcggaatccagggccgacccgggccggaccgaccccaggcgg
cgacggaatcagtccccaatgcctggaaattcctcattggattactgtgttttaaacagaatttcgtga
acagccttttatctccaagcggaaagaaagatgtggaaagcttcagcaggccacgctgtgtccatcgcc
caggatgacgcgggggccgatgactgggagaccgaccctgattttgtgaatgatgtgagtgagaaggag
caaagatggggtgccaagacggtgcagggctccgggcaccaggagcatatcaacatacacaagctgagg
gagaatgtctttcaagagcatcagacccttaaggagaaggaacttgaaacaggaccaaaagcttcccat
ggctatggagggaaatttggtgtggaacaagaccgaatggataagtcagctgtcggccacgaatatcag
tcgaaactttccaagcactgctcgcaggtggactcggtccgtggcttcggaggcaagtttggtgtccag
atggacagagttgatcagtctgctgtaggctttgaataccaggggaagactgagaagcatgcctcccag
aaagactactccagtggttttggcggcaagtatggcgtgcaggccgaccgagtagacaagagcgcggtg
ggcttcgactaccagggcaagacggagaagcacgagtcacagagagattactccaaaggtttcggcggc
aaatacggtatcgacaaggacaaagtggataagagcgccgttggctttgagtatcaaggcaaaacggag
aagcacgagtcccagaaagactatgtgaaagggtttggaggaaaatttggtgtgcagacagacagacaa
gacaaatgtgcccttggctgggatcaccaggagaaattgcagctgcatgaatcccaaaaagattataag
actggttttggaggcaaattcggtgttcagtcggagaggcaggactccgctgctgtggggtttgattac
aaggagaagctggccaagcacgagtcccagcaagactactccaaaggattcggcgggaagtatggggtg
cagaaggatcggatggataagaatgcgtcaacctttgaggatgtcacccaggtgtcctctgcctaccag
aagacagtacctgtcgaagctgtgaccagcaaaacaagtaacatcagagctaactttgaaaacctcgct
aaggagaaagagcaggaggacaggcggaaggcggaggcggagagagcccagcggatggccaaggagcgg
caggagcaggaagaggccaggaggaagctggaggagcaagccagagccaaaacgcaaacgccccctgtg
tcgcccgcacctcagccaaccgaggagaggctgccctcgagccccgtctatgaggatgcggcttccttc
aaggcagagctgagctacagaggccctgtgagtgggacggagccggagcccgtgtacagcatggaggcc
gctgactaccgagaggccagcagccagcagggcctggcctatgccacagaggctgtctatgaaagcgca
gaggccccgggccactatcccgcagaggacagcacctacgatgagtacgagaacgatctggggtacaca
gccgtcgccctgtacgactaccaggctgcgggcgatgatgagatctcatttgaccctgatgacatcatc
accaacatcgagatgattgacgacggctggtggcgcggggtgtgcaagggccggtacgggctcttccca
gccaactatgtggagctgcggcagtagggccccccagccccccccccggagctggcgccctggatcctcac
actacagatcaggccttctttggttcttgggtggttttgggtttttttctgttttttttttttttttttt
ttttttttttgaaggtggggaggggaatatacacattgcttttatatttaatactttgctgatgctt
ttgaaaatgtttatgccacagaatttgctaatatattgtaatcacattccttaggaggactttggtaat
tggttttatgcattgatggtttttttttctttttgccaaattgactgtcacgcggcagcttcaggga
gctcgcattctcttgtgttcgtgttgccctcgtgcccatcaagtgcagtcgggacctcccaggacaagc
acgagcctcaggtcggccctgtggcgggtaggcaggaaggactgtcccagacgaggggcttcctctaga
gtctcactgctggggaggagaggactgggcctgatggaagttaacccggagctaagtcacccagagcac
aggagctgccatgtcagatgggaaatctgcctatgtcataccgtgacagcccgcaggatcaggtgactt
ctagcagagaccctggtttttttcctgtgcccactccggcttgtcctcatctctacccatcccctgatg
cccaggtcaccgggagggctgctgggagcctctcctgtccccgccggcagtgtcactgagtccttgaaa
tcctcccctgcccgcgggtctctggattgggacgcacagtgcagttgaggtctgcgtcgggcttggctt
ttcacaaaggctgatgtcttaactgtcacccatatggtccctgggccaccgggcagcctggggcggtgt
gtgtgccatgtcacagcatggcctctcggccttgggaaggaaggcagtgtgcctgctctgctgtgagcc
gccaggaaccctcctcctgtcaatgggggtgtagtatttttgccaaaatatcatgttcaatttcagtag
tttgatcagttgaaggctagaagtgtgaagtgcagatgagtgtgtgttcttccccaaggtccccccaca
gctccaggacaccgctgtcctggcatttgtggccactcactttgtaggaaactcatctccttcctgagg
agccgggaggctggaccagtcccgtcgtgcagtcaggtgggcggtgtgtctttccagaaggtcacgtgg
aaatgtctcgggacttgggtcccggagtgcccgtgaagcgtgtttttgctcctgaggtgcattttctca
tcatccttgctttaccacaatgagcaatgaggtcgggttttatatgcaacttattgtatctgaattcct
gtaggcacaccctccatagggtatgatttttttaaattaaagaattcagaataaacatttttgatcc
```

aaaaa
HUMEMS | Homo sapiens amplaxin (EMS1) mRNA, complete cds.
gcctggtgcctgggagcggctggcgcggcggaatccagggccgacccgggccggaccgaccccaggcgg
cgacggaatcagtccccaatgcctggaaattcctcattggattactgtgttttaaacagaatttcgtga
acagccttttatctccaagcggaaagaaagatgtggaaagcttcagcaggccacgctgtgtccatcgcc
caggatgacgcgggggccgatgactgggagaccgaccctgattttgtgaatgatgtgagtgagaaggag
caaagatggggtgccaagacggtgcagggctccgggcaccaggagcatatcaacatacacaagctgagg
gagaatgtctttcaagagcatcagacccttaaggagaaggaacttgaaacaggaccaaaagcttcccat
ggctatggagggaaatttggtgtggaacaagaccgaatggataagtcagctgtcggccacgaatatcag
tcgaaactttccaagcactgctcgcaggtggactcggtccgtggcttcggaggcaagtttggtgtccag
atggacagagttgatcagtctgctgtaggctttgaataccaggggaagactgagaagcatgcctcccag
aaagactactccagtggttttggcggcaagtatggcgtgcaggccgaccgagtagacaagagcgcggtg
ggcttcgactaccagggcaagacggagaagcacgagtcacagagagattactccaaaggtttcggcggc
aaatacggtatcgacaaggacaaagtggataagagcgccgttggctttgagtatcaaggcaaaacggag
aagcacgagtcccagaaagactatgtgaaagggtttggaggaaaatttggtgtgcagacagacagacaa
gacaaatgtgcccttggctgggatcaccaggagaaattgcagctgcatgaatcccaaaaagattataag
actggttttggaggcaaattcggtgttcagtcggagaggcaggactccgctgctgtggggtttgattac
aaggagaagctggccaagcacgagtcccagcaagactactccaaaggattcggcgggaagtatggggtg
cagaaggatcggatggataagaatgcgtcaacctttgaggatgtcacccaggtgtcctctgcctaccag
aagacagtacctgtcgaagctgtgaccagcaaaacaagtaacatcagagctaactttgaaaacctcgct
aaggagaaagagcaggaggacaggcggaaggcggaggcggagagagcccagcggatggccaaggagcgg
caggagcaggaagaggccaggaggaagctggaggagcaagccagagccaaaacgcaaacgccccctgtg
tcgcccgcacctcagccaaccgaggagaggctgccctcgagccccgtctatgaggatgcggcttccttc
aaggcagagctgagctacagaggccctgtgagtgggacggagccggagcccgtgtacagcatggaggcc
gctgactaccgagaggccagcagccagcagggcctggcctatgccacagaggctgtctatgaaagcgca
gaggccccgggccactatcccgcagaggacagcacctacgatgagtacgagaacgatctggggtacaca
gccgtcgccctgtacgactaccaggctgcgggcgatgatgagatctcatttgaccctgatgacatcatc
accaacatcgagatgattgacgacggctggtggcgcggggtgtgcaagggccggtacgggctcttccca
gccaactatgtggagctgcggcagtagggcccccagccccccccccggagctggcgccctggatcctcac
actacagatcaggccttctttggttcttgggtggttttgggtttttttctgttttttttttttttttttt
ttttttttttgaaggtggggaggggaatatacacattgcttttatatttaatacttttgctgatgctt
ttgaaaatgtttatgccacagaatttgctaatatattgtaatcacattccttaggaggactttggtaat
tggttttatgcattgatggttttttttttctttttgccaaattgactgtcacgcggcagcttcaggga
gctcgcattctcttgtgttcgtgttgccctcgtgcccatcaagtgcagtcgggacctcccaggacaagc
acgagcctcaggtcggccctgtggcgggtaggcaggaaggactgtcccagacgaggggcttcctctaga
gtctcactgctggggaggagaggactgggcctgatggaagttaacccggagctaagtcacccagagcac
aggagctgccatgtcagatgggaaatctgcctatgtcataccgtgacagcccgcaggatcaggtgactt
ctagcagagaccctggtttttttcctgtgcccactccggcttgtcctcatctctacccatcccctgatg
cccaggtcaccgggagggctgctgggagcctctcctgtccccgccggcagtgtcactgagtccttgaaa
tcctcccctgcccgcgggtctctggattgggacgcacagtgcagttgaggtctgcgtcgggcttggctt
ttcacaaaggctgatgtcttaactgtcacccatatggtccctgggccaccgggcagcctggggcggtgt
gtgtgccatgtcacagcatggcctctcggccttgggaaggaaggcagtgtgcctgctctgctgtgagcc
gccaggaaccctcctcctgtcaatgggggtgtagtatttttgccaaaatatcatgttcaatttcagtag
tttgatcagttgaaggctagaagtgtgaagtgcagatgagtgtgtgttcttccccaaggtccccccaca
gctccaggacaccgctgtcctggcatttgtggccactcactttgtaggaaactcatctccttcctgagg
agccgggaggctggaccagtcccgtcgtgcagtcaggtgggcggtgtgtctttccagaaggtcacgtgg
aaatgtctcgggacttgggtcccggagtgcccgtgaagcgtgtttttgctcctgaggtgcattttctca
tcatccttgctttaccacaatgagcaatgaggtcgggttttatatgcaacttattgtatctgaattcct
gtaggcacaccctccatagggtatgatttttttaaattaaagaattcagaataaacatttttgatcc
aaaaa

Prosvasi sti Genpet
AF171097_1 | transcriptional regulator RovA [Yersinia nterocolitica].
MESTLGSDLARLVRVWRALIDHRLKPLELTQTHWVTLHNINRLPPEQSQIQLAKAIGIEQPSLVRTLDQ
LEEKGLITRHTCANDRRAKRIKLTEQSSPIIEQVDGVICSTRKEILGGISPDEIELLSGLIDKLERNII
QLQSK
AF171097_1 | transcriptional regulator RovA [Yersinia nterocolitica].
MESTLGSDLARLVRVWRALIDHRLKPLELTQTHWVTLHNINRLPPEQSQIQLAKAIGIEQPSLVRTLDQ
LEEKGLITRHTCANDRRAKRIKLTEQSSPIIEQVDGVICSTRKEILGGISPDEIELLSGLIDKLERNII

```
QLQSK
```

3. Πρόβλημα 3

Ανάγνωση ενός αρχείου τύπου embl, εξαγωγή της ακολουθίας dna σε μορφή fasta, μεταγραφή σε rna και μετάφραση.

Πηγαίος κώδικας

```
package problem3;

import java.io.*;
import org.biojava.bio.seq.*;
import org.biojava.bio.symbol.*;
import org.biojavax.bio.seq.RichSequence;
import org.biojavax.bio.seq.RichSequenceIterator;
import org.biojavax.bio.seq.io.*;

//EMBLFormat
public class Main {
    public static void main(String[] args) throws Exception{

        String file = args[0];

            FastaHeader fastaHeader = new FastaHeader();
            fastaHeader.setShowAccession(true);
            fastaHeader.setShowDescription(false);
            fastaHeader.setShowIdentifier(false);
            fastaHeader.setShowName(false);
            fastaHeader.setShowNamespace(false);
            fastaHeader.setShowVersion(false);

            FastaFormat fastaFormat = new FastaFormat();
            fastaFormat.setHeader(fastaHeader);
            fastaFormat.setLineWidth(60);

  BufferedReader br = new BufferedReader(new FileReader(file));
  RichSequenceIterator iter = RichSequence.IOTools.readEMBLDNA(br,
   null);

      while(iter.hasNext()) {

      RichSequence seq = iter.nextRichSequence();
      System.out.println("Akolouthia dna tou "+seq.getDescription());
      fastaFormat.writeSequence(seq, System.out);

      SymbolList s= seq.subList(1, seq.length());

      s=DNATools.toRNA(s);

      String str=s.seqString();
      System.out.println("Metagrafi se RNA:");
      System.out.println(s.seqString());

       int i=1, arxi=0,telos=0;
       int mikos=s.length();

      while (i<  mikos-2)
          {

            if (s.subStr(i, i+2).equals("aug"))
```

```
                    {    arxi=i;
                         break;
                     }
              else
                  i=i+1;

                     }

        if (arxi!=0)
        {
            while (arxi < mikos - 2)
                  {
                     telos=i;
                     if (s.subStr(i, i + 2).equals("uaa"))
                         break;
                      else if (s.subStr (i, i + 2).equals("uag"))
                         break;
                      else if (s.subStr(i, i + 2).equals("uga"))
                         break;
                     else
                         i=i+3;
                 }
           }

     System.out.println("Metafrasi se proteini: ");
                      s = s.subList(arxi, telos+2);
                      s=RNATools.translate(s);
                      System.out.print(s.seqString());

         }
    }
}
```

Είσοδος (αρχείο μορφής embl)

```
ID   AB030203; SV 1; linear; mRNA; STD; MUS; 1622 BP.
XX
AC   AB030203;
XX
DT   16-MAR-2000 (Rel. 63, Created)
DT   23-SEP-2008 (Rel. 97, Last updated, Version 3)
XX
DE   Mus musculus mRNA for E25B protein, complete cds, clone:2-76.
XX
KW   E25B protein; Mus musculus E25B protein.
XX
OS   Mus musculus (house mouse)
OC   Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia;
OC   Eutheria; Euarchontoglires; Glires; Rodentia; Sciurognathi; Muroidea;
OC   Muridae; Murinae; Mus; Mus.
XX
RN   [1]
RP   1-1622
RA   Inoue S., Ohta M.;
RT   ;
RL   Submitted (16-JUL-1999) to the EMBL/GenBank/DDBJ databases.
RL   Shinichi Inoue, Tsukuba Research Institute, Banyu Pharmaceutical Co., Ltd.,
RL   Biomedical Research Laboratories; Okubo 3, Tsukuba, Ibaragi 300-2611, Japan
RL   (E-mail:inouesi@banyu.co.jp, URL:http://www.banyu.co.jp/,
RL   Tel:81-298-77-2000(ex.2251), Fax:81-298-77-2027)
XX
RN   [2]
RX   DOI; 10.1006/bbrc.2000.2170.
RX   PUBMED; 10679242.
```

```
RA   Inoue S., Sano H., Ohta M.;
RT   "Growth suppression of Escherichia coli by induction of expression of
RT   mammalian genes with transmembrane or ATPase domains";
RL   Biochem. Biophys. Res. Commun. 268(2):553-561(2000).
XX
DR   Ensembl-Gn; ENSMUSG00000022108; Mus_musculus.
DR   Ensembl-Tr; ENSMUST00000022704; Mus_musculus.
XX
CC   Sequence updated (28-Jul-1999)
XX
FH   Key             Location/Qualifiers
FH
FT   source          1..1622
FT                   /organism="Mus musculus"
FT                   /mol_type="mRNA"
FT                   /clone="2-76"
FT                   /tissue_lib="whole brain"
FT                   /tissue_type="whole brain"
FT                   /db_xref="taxon:10090"
FT   CDS             166..966
FT                   /codon_start=1
FT                   /transl_table=1
FT                   /product="E25B protein"
FT                   /function="expression of this gene product inhibits the
FT                   growth of E. coli"
FT                   /note="a altenative splicing form of MMU76253 Mus musculus
FT                   E25B protein mRNA"
FT                   /note="contains transmembrane (TM) region and ATP binding
FT                   region"
FT                   /db_xref="GOA:O89051"
FT                   /db_xref="InterPro:IPR007084"
FT                   /db_xref="MGI:1309517"
FT                   /db_xref="UniProtKB/Swiss-Prot:O89051"
FT                   /protein_id="BAA92766.1"
FT                   /translation="MVKVTFNSALAQKEAKKDEPKSSEEALIVPPDAVAVDCKDPGDVV
FT                   PVGQRRAWCWCMCFGLAFMLAGVILGGAYLYKYFALQPDDVYYCGLKYIKDDVILNEPS
FT                   ADAPAARYQTIEENIKIFEEDAVEFISVPVPEFADSDPANIVHDFNKKLTAYLDLNLDK
FT                   CYVIPLNTSIVMPPKNLLELLINIKAGTYLPQSYLIHEHMVITDRIENVDNLGFFIYRL
FT                   CHDKETYKLQRRETIRGIQKREASNCFTIRHFENKFAVETLICS"
XX
SQ   Sequence 1622 BP; 424 A; 387 C; 379 G; 432 T; 0 other;
     gggaaaccgc gctgcactga gccgctgctg ctgtcgcgca gtccgctcct ccgctgcaga        60
     gtcgtgccct gagctcggcc gacaaggctg ccttcgcagc cggggatcct gccagccgcg       120
     accccagcct tcgccgtcgc cgcctagggc gccccaggcc gcaccatggt gaaggtgacg       180
     ttcaactcgg cgctggccca gaaggaggcc aagaaggacg agcccaagag cagcgaggag       240
     gcgctcatcg tccctccgga tgccgtggcg gtggattgca aggacccggg tgacgtggtt       300
     ccggttggac agaggagagc gtggtgttgg tgcatgtgtt tcggactggc cttcatgctt       360
     gctggcgtca tcctcggagg ggcgtacctg tacaagtatt ttgctcttca gccagatgat       420
     gtgtactact gtggactaaa gtacatcaaa gatgacgtca tcctgaacga gccttctgcg       480
     gatgccccag ctgctcgcta ccagacaatt gaagagaaca ttaagatctt tgaggaagac       540
     gcagtggaat tcatcagtgt gcctgtacca gagtttgcgg acagcgatcc tgccaacatt       600
     gtgcacgact tcaacaagaa actcactgct tatttggacc ttaacctgga caagtgctac       660
     gtgattcctc tgaacacttc catcgttatg ccgcccaaaa acctgctgga gctccttatt       720
     aacattaagg ccgggaccta cctgcctcag tcctacctta tccatgagca catggtgatc       780
     accgaccgca tcgagaacgt ggacaacctg ggcttcttca tctaccgact gtgtcacgac       840
     aaggagacct acaaactgca gcgccgggaa acaattagag gtattcagaa gcgggaagcc       900
     agtaactgtt tcaccattcg gcattttgag aacaaatttg ctgtggagac tttaatttgt       960
     tcttgagaag tcaagaaaaa acgtggggag gaattcaatg ccacagcata ccctgcccct      1020
     ttgtattttg tgcagtgatt gtttttttaaa atcttctttt catgtaagta gcaaacaggg      1080
     cttcactgtc tcttcatctc aataactcaa ttaaaaacca ttatcttaaa aaaaagaaaa      1140
     caaaaccttt ctttttttcta agtgtggtgt ctttgatgtt tgaattagca aatgtgcagg      1200
     ttcctagata agattcgctt ctccttagag cttacctact aggaagaatc taaattgctt      1260
     ggaaatcact aatctggatt tttgtgttaa ttctgcactt ccatgaggga aagatgccta      1320
     aagaatagtc attcgcatat gttaaaggga ccacagtgac ttgcttgtag atgctagccc      1380
     tgctacctag tctgttagca tttgaagtca ccttctcata ctactttaat taaaatgtgc      1440
```

```
     cgtatcttca atgttgcttt aactacttta gaggatttca gccttgatgt tttaatatcc      1500
     taggcctctg ctgtaataag attttagaca aatgtttgga atttaagaag caactcatgt      1560
     tactaatttg tatagcccat atctgtggaa tggaatataa atatcacaaa gccaaaaaaa      1620
     aa                                                                    1622
//
```

Αποτέλεσμα – Έξοδος

```
Akolouthia dna tou Mus musculus mRNA for E25B protein, complete cds,
clone:2-76.

>AB030203
gggaaaccgcgctgcactgagccgctgctgctgtcgcgcagtccgctcctccgctgcaga
gtcgtgccctgagctcggccgacaaggctgccttcgcagccggggatcctgccagccgcg
accccagccttcgccgtcgccgcctagggcgccccaggccgcaccatggtgaaggtgacg
ttcaactcggcgctggcccagaaggaggccaagaaggacgagcccaagagcagcgaggag
gcgctcatcgtccctccggatgccgtggcggtggattgcaaggacccgggtgacgtggtt
ccggttggacagaggagagcgtggtgttggtgcatgtgtttcggactggccttcatgctt
gctggcgtcatcctcggaggggcgtacctgtacaagtattttgctcttcagccagatgat
gtgtactactgtggactaaagtacatcaaagatgacgtcatcctgaacgagccttctgcg
gatgccccagctgctcgctaccagacaattgaagagaacattaagatctttgaggaagac
gcagtggaattcatcagtgtgcctgtaccagagtttgcggacagcgatcctgccaacatt
gtgcacgacttcaacaagaaactcactgcttatttggaccttaacctggacaagtgctac
gtgattcctctgaacacttccatcgttatgccgcccaaaaacctgctggagctccttatt
aacattaaggccgggacctacctgcctcagtcctaccttatccatgagcacatggtgatc
accgaccgcatcgagaacgtggacaacctgggcttcttcatctaccgactgtgtcacgac
aaggagacctacaaactgcagcgccgggaaacaattagaggtattcagaagcgggaagcc
agtaactgtttcaccattcggcattttgagaacaaatttgctgtggagactttaatttgt
tcttgagaagtcaagaaaaaacgtggggaggaattcaatgccacagcataccctgcccct
ttgtattttgtgcagtgattgtttttaaaatcttctttcatgtaagtagcaaacaggg
cttcactgtctcttcatctcaataactcaattaaaaaccattatcttaaaaaaaagaaaa
caaaaccttttttttctaagtgtggtgtctttgatgtttgaattagcaaatgtgcagg
ttcctagataagattcgcttctccttagagcttacctactaggaagaatctaaattgctt
ggaaatcactaatctggatttttgtgttaattctgcacttccatgagggaaagatgccta
aagaatagtcattcgcatatgttaaagggaccacagtgacttgcttgtagatgctagccc
tgctacctagtctgttagcatttgaagtcaccttctcatactactttaattaaaatgtgc
cgtatcttcaatgttgctttaactactttagaggatttcagccttgatgttttaatatcc
taggcctctgctgtaataagattttagacaaatgtttggaatttaagaagcaactcatgt
tactaatttgtatagcccatatctgtggaatggaatataaatatcacaaagccaaaaaaa

aa

Metagrafi se RNA:

gggaaaccgcgcugcacugagccgcugcugcugucgcgcaguccgcuccuccgcugcagagucgugccc
ugagcucggccgacaaggcugccuucgcagccggggauccugccagccgcgacccc agccuucgccguc
gccgccuagggcgccccaggccgcaccaugguga aggugacguucaacucggcgcuggcccagaaggag
gccaagaaggacgagcccaagagcagcgaggaggcgcucaucgucccuccggaugccguggcgguggau
ugcaaggacccgggugacgugguuccgguuggacagaggagagcgugguguuggugcauguguuucgga
cuggccuucaugcuugcuggcgucauccucggaggggcguaccuguacaaguauuuugcucuucagcca
gaugauguguacuacuguggacuaaaguacaucaaagaugacgucauccugaacgagccuucugcggau
gccccagcugcucgcuaccagacaauugaagagaacauuaagaucuuugaggaagacgcaguggaauuc
aucagugugccuguaccagaguuugcggacagcgauccugccaacauugugcacgacuucaacaagaaa
cucacugcuuauuuggaccuuaaccuggacaagugcuacgugauuccucugaacacuuccaucguuaug
ccgcccaaaaaccugcuggagcuccuuauuaacauuaaggccgggaccuaccugccucaguccuaccuu
auccaugagcacauggugaucaccgaccgcaucgagaacguggacaaccugggcuucuucaucuaccga
cugugucacgacaaggagaccuacaaacugcagcgccgggaaacaauuagagguauucagaagc gggaa
gccaguaacuguuucaccauucggcauuuugagaacaaauuugcuguggagacuuuaauuuguucuuga
gaagucaagaaaaaacgugggggaggaauucaaugccacagcauacccugcccc uuuguauuuugugcag
ugauuguuuuuuaaaaucuucuuuucauguaaguagcaaacagggcuucacugucucuucaucucaaua
```

```
acucaauuaaaaaccauuaucuuaaaaaaaagaaaacaaaaccuuucuuuuuucuaaguguggugucuu
ugauguuugaauuagcaaaugugcagguuccuagauaagauucgcuucuccuuagagcuuaccuacuag
gaagaaucuaaaauugcuuggaaaucacuaaucuggauuuuuguguuaauucugcacuuccaugagggaa
agaugccuaaagaauagucauucgcauauguuaaagggaccacagugacuugcuuguagaugcuagccc
ugcuaccuagucuguuagcauuugaagucaccuucucauacuacuuuaauuaaaaugugccguaucuuc
aauguugcuuuaacuacuuuagaggauuucagccuugauguuuuaauauccuaggccucugcuguaaua
agauuuuagacaaauguuuggaauuuaagaagcaacucauguuacuaauuuguauagcccauaucugug
gaauggaauauaaauaucacaaagccaaaaaaaaa
```

```
Metafrasi se proteini:
```

```
MVKVTFNSALAQKEAKKDEPKSSEEALIVPPDAVAVDCKDPGDVVPVGQRRAWCWCMCFGLAFMLAGVI
LGGAYLYKYFALQPDDVYYCGLKYIKDDVILNEPSADAPAARYQTIEENIKIFEEDAVEFISVPVPEFA
DSDPANIVHDFNKKLTAYLDLNLDKCYVIPLNTSIVMPPKNLLELLINIKAGTYLPQSYLIHEHMVITD
RIENVDNLGFFIYRLCHDKETYKLQRRETIRGIQKREASNCFTIRHFENKFAVETLICS*
```

4. Πρόβλημα 4

Ανάγνωση ενός αρχείου τύπου embl, μέτρηση των βάσεων των ακολουθιών που περιέχει, εξαγωγή κατανομής/ών και δημιουργία τυχαίας ακολουθίας βάση της κατανομής αυτής.

Πηγαίος κώδικας

```
package problem4;

import java.io.*;
import java.util.*;
import org.biojava.bio.dist.*;
import org.biojava.bio.seq.*;
import org.biojava.bio.seq.io.*;
import org.biojava.bio.symbol.*;

public class Main {
    public static void main(String[] args) {

        FiniteAlphabet alpha = DNATools.getDNA();
        AlphabetIndex index =
AlphabetManager.getAlphabetIndex(alpha);

 //Αντικείμενο που κρατά τις μετρήσεις
   Count counts = null;

    try {
 //Άνοιγμα του αρχείου που περιέχει την ακολουθία
      BufferedReader br = new BufferedReader(new
      FileReader(args[0]));

 //Παίρνουμε έναν SequenceIterator για τις ακολουθίες του αρχείου
      SequenceIterator iter =
      (SequenceIterator)SeqIOTools.fileToBiojava(args[1],args[2],br);

 //Επανάληψη για κάθε ακολουθία
      while(iter.hasNext()){
        Sequence seq = iter.nextSequence();

 //Αν είναι απαραίτητο αρχικοποίηση του counts
```

```
        if(counts == null){
          counts = new
          IndexedCount((FiniteAlphabet)seq.getAlphabet());
        }

        //Επανάληψη στα σύμβολα της ακολουθίας
        for (Iterator i = seq.iterator(); i.hasNext(); ) {
          AtomicSymbol sym = (AtomicSymbol)i.next();
          counts.increaseCount(sym,1.0);
        }
      }

      //Εμφάνιση των αποτελεσμάτων
      System.out.println("\nCOUNT (METRHSH))");
      for (Iterator i =
    ((FiniteAlphabet)counts.getAlphabet()).iterator();i.hasNext(); )
    {
        AtomicSymbol sym = (AtomicSymbol)i.next();
        System.out.println(sym.getName()+" : "+counts.getCount(sym));
      }

      //Μετατροπή σε κατανομή
      Distribution d = DistributionTools.countToDistribution(counts);

      System.out.println("\nDISTRIBUTION (KATANOMH)");
      for (int i = 0; i < alpha.size(); i++) {
        Symbol s = index.symbolForIndex(i);
        System.out.println(s.getName()+" : "+d.getWeight(s));
      }

 //Δημιουργία μιας τυχαίας συμβολοσειράς βάση της κατανομής
   SymbolList s= DistributionTools.generateSymbolList(d,100);
   System.out.println("\nTYXAIA SYMBOLOSEIRA BASH THS KATANOMHS: ");
   System.out.println(s.seqString());

    }
    catch (Exception ex) {
      ex.printStackTrace();
    }
  }
}
```

Είσοδος (αρχείο μορφής embl)

Το αρχείο παρατίθεται στην είσοδο του προηγούμενου κώδικα (πρόβλημα 3).

Αποτέλεσμα – Έξοδος

```
COUNT (METRHSH))
thymine : 691.0
guanine : 1693.0
adenine : 769.0
cytosine : 1717.0

DISTRIBUTION (KATANOMH)
thymine : 0.14188911704312115
guanine : 0.34763860369609856
adenine : 0.15790554414784394
cytosine : 0.35256673511293635
```

```
TYXAIA SYMBOLOSEIRA BASH THS KATANOMHS:
Cgcgtacgtactggtcccgacgtgtgccgaagccccgtcagagcgccgcaatccgagtgccccccccctg
gacagcttgcctacgaagaaggcatctcagc
```

5. Πρόβλημα 5

Ολική στοίχιση δύο ακολουθιών.

Πηγαίος κώδικας

```
package problem6;

import java.io.File;
import java.io.BufferedReader;
import java.io.FileReader;
import org.biojava.bio.alignment.NeedlemanWunsch;
import org.biojava.bio.alignment.SequenceAlignment;
import org.biojava.bio.alignment.SubstitutionMatrix;
import org.biojava.bio.seq.Sequence;
import org.biojava.bio.symbol.AlphabetManager;
import org.biojava.bio.symbol.FiniteAlphabet;
import org.biojavax.bio.seq.RichSequence;
import org.biojavax.bio.seq.RichSequenceIterator;

public class Main {
    public static void main(String[] args) throws Exception {
      if (args.length < 3)
        throw new Error("Usage: DeterministicAlignmentDemo " +
                        "querySeq targetSeq substitutionMatrixFile");

    Sequence query= null;
    Sequence target= null;

//Ανάγνωση των αρχείων fasta που περιέχουν τις ακολουθίες
  BufferedReader br1 = new BufferedReader(new FileReader(args[0]));
  RichSequenceIterator iter  = RichSequence.IOTools.readFastaDNA(br1,
  null);
      while(iter.hasNext()) {
      RichSequence seq = iter.nextRichSequence();
      query= seq;
        }

 BufferedReader br2 = new BufferedReader(new FileReader(args[1]));
 RichSequenceIterator iter2  = RichSequence.IOTools.readFastaDNA(br2,
 null);
      while(iter2.hasNext()) {
      RichSequence seq2 = iter2.nextRichSequence();
      target= seq2;
        }

 //Επιλογή του αλφαβήτου για τις ακολουθίες - εδώ επιλέγουμε το DNA
      FiniteAlphabet        alphabet        =        (FiniteAlphabet)
      AlphabetManager.alphabetForName("DNA");
 // Ανάγνωση του αρχείου με τον πίνακα substitution matrix
 // Γι αυτό το παράδειγμα ο πίνακας NUC.4.4 είναι κατάλληλος
      SubstitutionMatrix  matrix  =  new  SubstitutionMatrix(alphabet,
      new File(args[2]));
 // Καθορίζουμε τα προεπιλεγμένα (default) κάστη για το χειρισμό
```

```
 //των ακολουθιών για την ολική στοίχιση
      SequenceAlignment aligner = new NeedlemanWunsch(
        (short) 0,      // match - ταίριασμα
        (short) 3,      // replace - αντικατάσταση
        (short) 2,      // insert - εισαγωγή
        (short) 2,      // delete - διαγραφή
        (short) 1,      // gapExtend
        matrix    // SubstitutionMatrix
      );

 //Εκτέλεση στοίχισης και αποθήκευση των αποτελεσμάτων
      aligner.pairwiseAlignment(
        query, // πρώτη ακολουθία
        target // δεύτερη ακολουθία
      );
 // Εμφάνιση της στοίχισης
    System.out.println("Global alignment with Needleman-Wunsch:\n" +
aligner.getAlignmentString());
  }
}
```

Είσοδοι (2 αρχεία μορφής fasta και ο κατάλληλος πίνακας υποκαταστάσεων)

```
>gi|288900259|gb|GW425382.1|GW425382 UMC-bplad35_0B02-004-h12 Day 35
Placenta bplad35 Bos taurus cDNA 3', mRNA sequence
AGGAGATGGAATTGTTTTCTATATTTTTATGGTGATGATGGTTGTACTATTATAAGGAATTATCAAACAC
ATTTAAATGGTACATTTAGTACTGATAAATTATACTCTGTGTAAATAATAACAAAAAAAGAAATTTTTTT
TAAGTATCATGCTACATTTAATCTTTGGGACTAAACATTTAATATTTTATTGCTAAGAATCATTCATATT
GTTATGTGTCACTGTAGTTCACTTGTTTCGAAACCAGTATAATATTCCACTGTGTGAACATTCCAGTTTA
TTTGTCCACTCTCTTGCTCATGGTATTTGGGTTTGCTTTTGTGACCAGGGCTGCTATGAAAACATTCTTG
TATGTATCTCCTGATGTACATGTAAGAGTTTCTCTTGGATTTATCCCTAATATTGGAATAAGTGAGTTGT
AGGTAAACGTTCAATTATAGGAGATAAACTATTTTCCAAAGCAGATCTCTGCTTTTCTTATCTGCGTCTC
TCTGTCCCCAAATAGACTTCGACCCCAAGTGCAGGAGATGCCGGCGGACGCGTGGG

>gi|288900258|gb|GW425381.1|GW425381 UMC-bplad35_0B02-004-h11 Day 35
Placenta bplad35 Bos taurus cDNA 3', mRNA sequence
CAGCACCATTAAGAGTGAAACCATTTATTCTTTATTGAGAATAGGGTGTAAAGTGAAAGTATAAGTGAAG
TTGCTCAGTTATGCCCGACTCTTAGAAACCCCATGGACTGGAGCCCACCAGGCTCCTCTGTCCATGTGAT
TTTCCAGGCAAGAGTACTGGAGTGGGGTGCCATTGCCTTCTCCAGAGAATAGGGTGTACAGACCACCAAA
TACAGTTCACCAGCATCCTGGGCAGGAGTGCCCAAAGTGTGAGTGAGTGTGTGTTAGGAGTGGCCTTACT
GATTCCTGAACCACTCCAAGCATTCACACTGCCTGTGCCAGGCCAATTCTGTCCTTTCTTCGATCAAAGA
CTGAGAAATACAGCCTCAGAAAGACGTCACCCAGGATCCACATCTCTGTAGATGAACTCTGTTGATGCCC
TTTAAAGGCAGTAAAGCAGCGGCCTCTAGAATCCTTGAGGATGTAGGCTTGACCTGGCACTCGGTAGTTG
TTGCTTTTGATGGTGAAGATAATAGAGGGTAGGCTATTGACCGCAGAACATGAAACGTAGTGCTCAGAAC
CCTGTGGCGTGGCACCGATGAGCTTCCAGATGTTATTGACCAGTGTACTTGGGCCTACGATATCTGATGA
CCCCGTGTCCACAAGGGCCTTGCAGCCGCCAGAGCAAGCAATAACCTTTCTTTTCATGGAGATGCGGTCC
ATGTGTACACTCCAGTCGCCTGCTTTCATCAATGGTATCCAGTTGAGCTCTCCCTTGT


# This matrix was created by Todd Lowe   12/10/92
# Uses ambiguous nucleotide codes, probabilities rounded to
#  nearest integer
# Lowest score = -4, Highest score = 5
    A   T   G   C   S   W   R   Y   K   M   B   V   H   D   N
A   5  -4  -4  -4  -4   1   1  -4  -4   1  -4  -1  -1  -1  -2
T  -4   5  -4  -4  -4   1  -4   1   1  -4  -1  -4  -1  -1  -2
G  -4  -4   5  -4   1  -4   1  -4   1  -4  -1  -1  -4  -1  -2
C  -4  -4  -4   5   1  -4  -4   1  -4   1  -1  -1  -1  -4  -2
S  -4  -4   1   1  -1  -4  -2  -2  -2  -2  -1  -1  -3  -3  -1
W   1   1  -4  -4  -4  -1  -2  -2  -2  -2  -3  -3  -1  -1  -1
R   1  -4   1  -4  -2  -2  -1  -4  -2  -2  -3  -1  -3  -1  -1
```

```
Y  -4   1  -4   1  -2  -2  -4  -1  -2  -2  -1  -3  -1  -3  -1
K  -4   1   1  -4  -2  -2  -2  -2  -1  -4  -1  -3  -3  -1  -1
M   1  -4  -4   1  -2  -2  -2  -2  -4  -1  -3  -1  -1  -3  -1
B  -4  -1  -1  -1  -1  -3  -3  -1  -1  -3  -1  -2  -2  -2  -1
V  -1  -4  -1  -1  -1  -3  -1  -3  -3  -1  -2  -1  -2  -2  -1
H  -1  -1  -4  -1  -3  -1  -3  -1  -3  -1  -2  -2  -1  -2  -1
D  -1  -1  -1  -4  -3  -1  -1  -3  -1  -3  -2  -2  -2  -1  -1
N  -2  -2  -2  -2  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1
```

Αποτέλεσμα – Έξοδος

```
Global alignment with Needleman-Wunsch:
 Time (ms):    172
 Length:       871
 Score:        1057
 Query:        GW425382,     Length:        546
 Target:       GW425381,     Length:        758

Query:    1 ~ag---------gagatgga---attgttttctatatt-tttatggtgatg-at-ggttg 44
             ||         ||| || |   |||        |||| |||||  ||| | || || ||
Target:   1 cagcaccattaagag-tgaaaccatt--------tattctttat--tga-gaatagggtg 48

Query:   45 tacta--t-----tataag-gaa-tt-atca---a---ac-acat-tt---aa----atg 79
            ||  |  |     |||||| ||| ||  |||   |    | || | ||   ||    |||
Target:  49 ta--aagtgaaagtataagtgaagttgctcagttatgcccgac-tcttagaaaccccatg 105

Query:   80 g--t-------a-ca---t--t-tagtac---tgat---------aa-a-t--t--a--- 102
            |  |       | ||   |  | | || |   ||||         || | |  |  |
Target: 106 gactggagcccaccaggctcctct-gtccatgtgattttccaggcaagagtactggagtg 164

Query:  103 ---t---a-------ctc-------t---gtgta------a--ataata-a---ca--a- 124
               |   |       |||       |   |||||      |  | |||| |   ||  |
Target: 165 gggtgccattgccttctccagagaatagggtgtacagaccacca-aatacagttcaccag 223

Query:  125 -a--------aaaag-----aaa-t-t---t---t-t-t-tta--agt----------a- 146
             |        |  ||     ||| | |   |   | | | |||  |||          |
Target: 224 catcctgggcaggagtgcccaaagtgtgagtgagtgtgtgttaggagtggccttactgat 283

Query:  147 tcatg-----ct---a-cattta-a-t--ctttg---ggactaaac-att---taatatt 186
            || ||     ||   | |||| | | |  || ||   || |    | |||   |  | ||
Target: 284 tcctgaaccactccaagcattcacactgcctgtgccagg-c----caattctgtcct-tt 337
```

```
Query:  187 -tt--at---tg-ctaag-aat-ca--ttcatattgttatgt-------g-t---ca--- 221
             ||  ||    | || || ||| ||   ||| |  |  | ||       | |   ||
Target: 338 cttcgatcaaagactgagaaatacagcctcagaaag--acgtcacccaggatccacatct 395

Query:  222 ctgtag-ttcact-tgtt--tcg------aaa-ccagtataatattcca-ctg--tgt-g 266
            |||||| |  ||| ||||  | |     |||  ||||| ||     || | |  | | |
Target: 396 ctgtagatgaactctgttgat-gccctttaaaggcagta-aa----gcagcggcctctag 449

Query:  267 aacattccagttt----atttgtccactct--cttg--ct--ca-t-ggtatttgggttt 314
            ||   |||   ||    |  |||  |      ||||  ||  || | |||| |||   ||
Target: 450 aa---tcc---ttgagga--tgt--a----ggcttgacctggcactcggtagttg---tt 492

Query:  315 gctttt----gtgaccag--------ggct--gcta-tga----a-aacat---tcttgt 351
            ||||||    ||||  ||        || |  |||| |||    | |||||    |  ||
Target: 493 gcttttgatggtga--agataatagagggtaggctattgaccgcagaacatgaaac--gt 548

Query:  352 a-tg-t---atctcctgat---gt---ac--atgtaagagtttctcttggat-ttat--- 394
            | || |   | | |||| |   ||   ||  ||    ||| ||| |   ||| ||||
Target: 549 agtgctcagaac-cctg-tggcgtggcaccgat----gagcttc-c--agatgttattga 599

Query:  395 ccctaa---ta-ttgg----a--ataagtgagttgtaggtaaa---cgt-t-ca-attat 438
            ||   |   || ||||    |  |||  |||  ||       |   ||| | || |
Target: 600 cc---agtgtacttgggcctacgatatctga--tg-------accccgtgtccaca---- 643

Query:  439 aggagataaactatttt-------cca----aagcagat---ctctgcttttcttat--- 481
            ||| |     |    ||       |||    ||||| ||   || | ||||||  ||
Target: 644 agg-g-----c---cttgcagccgccagagcaagca-ataacct-ttcttttc--atgga 690

Query:  482 -ctgc-gtctc-tctgt-c-c-ccaaatagacttcgacc------c--caagtg-----c 522
             ||| ||| | | ||| | | ||    ||   ||| ||      |  ||| ||     |
Target: 691 gatgcggtc-catgtgtacactcc----ag---tcg-cctgctttcatcaa-tggtatcc 740

Query:  523 ag--gagatgc-cggcggacgc-g-t-g-gg 546
            ||  ||| | | |      | |
Target: 741 agttgagct-ctc------c-ct-t-g-t—758
```