IBIX-JAV
Programming using Java

Assignment

Prof Fady Mohareb

Chair & Head of Bioinformatics | Course Director

www.cranfield.ac.uk

Gene model visualizer

Develop a Java program to visualize gene models using gene structures from GTF annotation and FASTA files

Software requirements (70%)

1. The program should read two file inputs from the user using a File Chooser [5 marks]

- A gene annotation file (in GTF format)
- A genome DNA sequence file (in FASTA format)

The gene models should be presented to the user (using a list or a tabular form). The sequence text from the FASTA file should also be displayed.

Software requirements (70%)

2. Basic statistics [20 marks]

Basic stats related to each file should be calculated as follows:

1. The average no. of exons per gene (GTF)
2. The longest and shortest genes models within the file (GTF)
3. The average gene length (GTF)
4. The sequence length of the fasta file being uploaded if a single sequence fasta is included, or the average length if multi-sequence fasta file is used. (FASTA)
5. The GC content (FASTA)

Software requirements (70%)

3. Exon text highlight [10 marks]

Exons should be highlighted in different colour within the sequence text

(Tip: You can use a JTextPane to use different text colour) – other approaches also exist!

e.g. `jTextPane1.setContentType("text/html");`

Nucleotide Sequence (2097 nt):

ATGAGGCTCGCCGTGGGAGCCCTGCTGGTCTGCGCCGTCCTGGGGCTGTGTCTGGCTGTCCCTGATAAAA
CTGTGAGATGGTGTGCAGTGTCGGAGCATGAGGCCACTAAGTGCCAGAGTTTCCGCGACCATATGAAAAG
CGTCATTCCATCCGATGGTCCCAGTGTTGCTTGTGTGAAGAAAGCCTCCTACCTTGATTGCATCAGGGCC
ATTGCGGCAAACGAAGCGGATGCTGTGACACTGGATGCAGGTTTGGTGTATGATGCTTACCTGGCTCCCA
ATAACCTGAAGCCTGTGGTGGCAGAGTTCTATGGGTCAAAAGAGGATCCACAGACTTTCTATTATGCTGT
TGCTGTGGTGAAGAAGGATAGTGGCTTCCAGATGAACCAGCTTCGAGGCAAGAAGTCCTGCCACACGGGT
CTAGGCAGGTCCGCTGGGTGGAACATCCCCATAGGCTTACTTTACTGTGACTTACCTGAGCCACGTAAAC
CTCTTGAGAAAGCAGTGGCCAATTTCTTCTCGGGCAGCTGTGCCCCTTGTGCGGATGGGACGGACTTCCC
CCAGCTGTGTCAACTGTGTCCAGGGTGTGGCTGCTCCACCCTTAACCAATACTTCGGCTACTCGGGAGCC
TTCAAGTGTCTGAAGGATGGTGCTGGGGATGTGGCCTTTGTCAAGCACTCGACTATATTTGAGAACTTGG
CAAACAAGGCTGACAGGGACCAGTATGAGCTGCTTTGCCTGGACAACACCCGGAAGCCGGTAGATGAATA
CAAGGACTGCCACTTGGCCCAGGTCCCTTCTCATACCGTCGTGGCCCGAAGTATGGGCGGCAAGGAGGAC
TTGATCTGGGAGCTTCTCAACCAGGCCCAGGAACATTTTGGCAAAGACAAATCAAAAGAATTCCAACTAT
TCAGCTCTCCTCATGGGAAGGACCTGCTGTTTAAGGACTCTGCCCACGGGTTTTTAAAAGTCCCCCCCAG
GATGGATGCCAAGATGTACCTGGGCTATGAGTATGTCACTGCCATCCGGAATCTACGGGAAGGCACATGC
CCAGAAGCCCCAACAGATGAATGCAAGCCTGTGAAGTGGTGTGCGCTGAGCCACCACGAGAGGCTCAAGT
GTGATGAGTGGAGTGTTAACAGTGTAGGGAAAATAGAGTGTGTATCAGCAGAGACCACCGAAGACTGCAT
CGCCAAGATCATGAATGGAGAAGCTGATGCCATGAGCTTGGATGGAGGGTTTGTCTACATAGCGGGCAAG
TGTGGTCTGGTGCCTGTCTTGGCAGAAAACTACAATAAGAGCGATAATTGTGAGGATACACCAGAGGCAG
GGTATTTTGCTATAGCAGTGGTGAAGAAATCAGCTTCTGACCTCACCTGGGACAATCTGAAAGGCAAGAA
GTCCTGCCATACGGCAGTTGGCAGAACCGCTGGCTGGAACATCCCCATGGGCCTGCTCTACAATAAGATC
AACCACTGCAGATTTGATGAATTTTTCAGTGAAGGTTGTGCCCCTGGGTCTAAGAAAGACTCCAGTCTCT
GTAAGCTGTGTATGGGCTCAGGCCTAAACCTGTGTGAACCCAACAACAAAGAGGGATACTACGGCTACAC
AGGCGCTTTCAGGTGTCTGGTTGAGAAGGGAGATGTGGCCTTTGTGAAACACCAGACTGTCCCACAGAAC
ACTGGGGGAAAAAACCCTGATCCATGGGCTAAGAATCTGAATGAAAAAGACTATGAGTTGCTGTGCCTTG
ATGGTACCAGGAAACCTGTGGAGGAGTATGCGAACTGCCACCTGGCCAGAGCCCCGAATCACGCTGTGGT
CACACGGAAAGATAAGGAAGCTTGCGTCCACAAGATATTACGTCAACAGCAGCACCTATTTGGAAGCAAC
GTAACTGACTGCTCGGGCAACTTTTGTTTGTTCCGGTCGGAAACCAAGGACCTTCTGTTCAGAGATGACA
CAGTATGTTTGGCCAAACTTCATGACAGAAACACATATGAAAAATACTTAGGAGAAGAATATGTCAAGGC
TGTTGGTAACCTGAGAAAATGCTCCACCTCATCACTCCTGGAAGCCTGCACTTTCCGTAGACCTTAA

Figures may be from made up examples for demonstration purposes only.

Software requirements (70%)

4. Exon graphical representation [15 marks]

Exons should be graphically represented as well

Example 1

Example 2

Example 3

Figures may be from made up examples for demonstration purposes only.

Software requirements (70%)

5. Clean, well commented, well abstracted code [15 marks]

Best coding practices should be followed for your program. This includes:

- Well commented methods
- Well abstracted architecture (logical abstraction into classes, methods, etc.)
- Error trapping (e.g. when the user uploads wrong file type, file is empty or doesn't include the information required for visualization)

Figures may be from made up examples for demonstration purposes only.

Documentation (30%)

5. Documentation [30 Marks]

- In Addition to the software, You must submit:

- A written report (Max 2000 words excluding source code and diagrams) describing the design of the application. This could include :
    1. A description of the design and structure of the program, including references to any imported classes used [15 marks]
    2. UML diagrams of all classes, variables and methods created [3.5marks]
    3. A Flowchart indicating the logical flow of the program [3.5 marks]
    4. A user manual describing how to setup and run the program [8 marks]

Submission

The complete Netbeans project folder + written report + Testing data as a zipped folder

- Assignment should be submitted via Canvas as per the corresponding deadlines for FT and PT students.

- You need to submit a Zipped folder containing:
    1. Netbeans project (or java source code + executable jar) as well as any required libraries – Project name should include the module code, your first name, last name, and studentID
    (eg. IBIX-JAV_John_Smith_s001010)
    2. Documentation: Technical document (incl. figures/charts) + user manual – report should also follow the same naming convention above
    3. Any additional test data (excluding the ones supplied): GTF + FASTA files, any additional data you have included for testing any extra functionalities

Marking Schema

Specification	Mark
Software (70%)	
The user can upload and display the contents of an annotation GTF file	5
The user can upload and display the content of a FASTA file	5
Basic statistics of the uploaded files are calculated	10
Exons are highlighted within the sequence text	10
Exons are highlighted visually using graphics	15
Extra functionalities	5
GUI Layout	5
Clean Code (Class Abstraction, Comments, etc)	10
Error trapping (stability)	5
Documentation (30%)	
Relevance, Conciseness, Accuracy	15
Flowchart(s) and UML(s)	7
User Manual	8
Total	**100**

Supplied files

- You are supplied with two test files to help with the implementation and testing as follows:
- **GRCh38.p12_SUBSET.gtf:** a small subset of the human genome annotation file (Release 42) – which includes ~10 genes per chromosome (not all chromosomes are included)
- **Homo_sapiens_TF_sequence.fa:** The genomic sequence for the human sero-transferrin gene (responsible for the transport of iron from sites of absorption and heme degradation to those of storage and utilization.

Additional files which may be useful for implementing extra-functionalities (http://bit.ly/3lHE1uM) :

- **Human_Gene_Sequences.fa**: genomic sequences for the entire human transcriptome (all genes – 2Gb)
- **gencode.v42.basic.annotation.gtf:** The entire human gene annotation file (1.57Gb)

A word on Plagiarism and Collusion

- Plagiarism is a serious academic offence which can result in failing the module and/or terminating your course registration

- **Don’t do it!**

- **You WILL get caught**

- **We always find out (trust me on this!)**

- **Seriously, don’t do it!**