Institut für Informatik
Praktische Informatik und Bioinformatik
Prof. Dr. Ralf Zimmer

Wintersemester 2019/20

Übungen zum Propädeutikum Programmierung in der Bioinformatik

Blatt 8

Termin: Dienstag, 10. Dezember 2019

Achte bei den folgenden Aufgaben darauf, die Exceptions mit `throws` weiterzugeben und/oder mit einem try-catch/try-with-resources Konstrukt zu behandeln, da deine Programme sonst eventuell nicht kompilieren.

Übung 1 *Fasta einlesen mit `BufferedReader`*

Schreibe eine Klasse `Proteome` die mehrere Proteinsequenzen direkt aus einer Fasta-Datei ausliest und speichert.

Die Klasse soll folgende Elemente enthalten:

1. eine Variable `ArrayList<Protein> proteins`, in der alle Proteine aus dem Fasta abgelegt werden. Du kannst natürlich deine in den vorherigen Übungen geschriebene `Protein`-Klasse verwenden. Falls du diese nicht hast, schreibe eine kleine Klasse die nur eine Protein-ID und die Sequenz enthält.
2. einen Konstruktor `Proteome(String fastaFile)`. In diesem Konstruktor soll mit einem `BufferedReader` die Fasta-Datei eingelesen werden, und für jede Sequenz darin ein `Protein` zu `proteins` hinzugefügt werden. Dabei soll die ID des Proteins auf den Header gesetzt werden (*ohne* das `>`-Symbol am Zeilenbeginn). Zum Testen steht auf der Website `Vibrio_cholerae.fasta` zum Download bereit.
3. eine Methode `int getProteinCount()` (gibt die Anzahl der geladenen Proteine zurück)
4. eine Methode `Protein getProtein(int i)` (gibt das i-te Protein-Objekt zurück)

Übung 2 *Statistiken schreiben mit `BufferedWriter`*

Erweitere Proteome jetzt um eine Methode `void writeStatistics(String outputFile)`. Diese soll für das gesamte Proteom folgende Statistiken in eine Datei `outputFile` schreiben:

1. minimale Sequenzlänge
2. maximale Sequenzlänge
3. Mittelwert aller Sequenzlängen

Achte darauf, dass du beim Schreiben der Datei keine anderen Dateien überschreibst.

Übung 3 *Einlesen einer TSV-Datei*

Erstelle nun in `Proteome` eine Variable `HashMap<Character,Double> massMap`. Diese Variable soll von einer Methode `void readAminoAcidMassFile(String file)` befüllt werden, welche eine TSV-Datei als Input erhält die die Massen der einzelnen Aminosäuren enthält. Eine TSV-Datei ist letztendlich nur eine Tabelle, in der die Spalten durch einen Tab getrennt sind (TSV = "Tab-separated-values").

`mass.tsv` enthält in der ersten Spalte die Buchstaben für jede Aminosäure, in der zweiten die zur jeweiligen Aminosäure gehörige Masse. Folgende Informationen könnten hilfreich für das Parsen dieser Datei sein:

- Ein Tab wird in Java (und vielen anderen Kontexten) durch die Zeichenfolge `\t` beschrieben.
- Mit der Methode `String[] split(String pattern)` kann man einen String an dem angegebenen `pattern` in Einzelstrings aufspalten.
- Die Methode `String trim()` returned eine Kopie des Strings mit allem eventuell vorhandenen Whitespace um den String herum entfernt.

Nun könntest du mit Hilfe von `massMap` bequem eine Methode schreiben die die Masse einer Proteinsequenz aufsummiert.

Lösung 3

```java
import java.io.*;
import java.util.ArrayList;
import java.util.HashMap;

public class Proteome {
    // Aufgabe 1
    private ArrayList<Protein> proteins;
    private HashMap<Character,Double> massMap; // Aufgabe 3

    public Proteome(String fastaFile) {
        this.proteins = new ArrayList<>();
        this.massMap = new HashMap<>(); // Aufgabe 3
        try (BufferedReader br = new BufferedReader(new FileReader(fastaFile))) {
            String line = "";
            String id = "";
            StringBuilder seq = new StringBuilder();
            while ((line = br.readLine()) != null) {
                if (line.startsWith(">")) {
                    if (!id.isEmpty()) {
                        proteins.add(new Protein(id, seq.toString()));
                        seq = new StringBuilder();
                    }
                    id = line.substring(1);
                } else {
                    seq.append(line.strip());
                }
            }
            proteins.add(new Protein(id, seq.toString()));
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public int getProteinCount() {
        return proteins.size();
    }

    public Protein getProtein(int i) {
        return proteins.get(i);
    }

    // Aufgabe 2
    public void writeStatistics() {
        int min = Integer.MAX_VALUE;
        int max = Integer.MIN_VALUE;
        int total = 0;
        for (Protein p : proteins) {
            int length = p.getSequence().length();
            total += length;
            if (length > max) max = length;
            if (length < min) min = length;
        }
        try (BufferedWriter bw = new BufferedWriter(new FileWriter("outputFile"))) {
            bw.write(min + " " + max + " " + (total*1.0)/this.getProteinCount());
        } catch (IOException e) {
```

```
            e.printStackTrace();
        }
    }

    // Aufgabe 3
    public void readAminoAcidMassFile(String file) {
        try (BufferedReader br = new BufferedReader(new FileReader(file))) {
            String line = "";
            while ((line = br.readLine()) != null) {
                String[] stringParts = line.trim().split("\t");
                massMap.put(stringParts[0].charAt(0), Double.parseDouble(stringParts[1]));
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        Proteome p = new Proteome("Vibrio_cholerae.fasta");
        System.out.println(p.getProteinCount()); // 3504
        p.writeStatistics(); // 26 4558 325.80422374429224
        p.readAminoAcidMassFile("mass.tsv");
    }
}
o
```