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

Wintersemester 2018/19

Übungen zum Bioinformatik-Tutorium

Blatt 7

Termin: Dienstag, 04.12.2018, 11 Uhr

1. Fasta Das FASTA-Format ist ein textbasiertes Format zur Darstellung und Speicherung der Sequenz von Nukleinsäuren (Nukleinsäuresequenz) und Proteinen (Proteinsequenz) in der Bioinformatik. Die Nukleinbasen bzw. Aminosäuren werden durch einen Ein-Buchstaben-Code dargestellt. Das Format erlaubt es, den Sequenzen einen Namen und Kommentare voranzustellen[1].

Schreibe eine Klasse `Proteome` die eine Menge von Proteinen speichern kann. Die Klasse soll folgende Elemente enthalten:

(a) eine private Instanzvariable `ArrayList<Protein> proteins;` (speichert alle Protein-Objekte aus der Fasta-Datei)

(b) einen Konstruktor `Proteome(File fastaFile)` (liest Fasta-Datei als Protein-Objekte ein). Die Klasse `StringBuilder` könnte hier hilfreich sein.

(c) eine Methode `getProteinCount()` (gibt die Anzahl der geladenen Proteine zurück)

(d) eine Methode `getProtein(int i)` (gibt das i-te Protein-Objekt zurück)

Hinweise: Eine ArrayList funktioniert ähnlich wie ein Array, allerdings muss die Länge nicht vorher bekannt sein. Die spitzen Klammern deuten darauf hin, dass ArrayList eine sogenannte generische Klasse ist und in diesem Fall nur Protein-Objekte speichern kann (siehe später).

- Man kann bequem Objekte über die `add(Protein p)` Methode ans Ende hinzufügen.
- Die Methode `size()` liefert die Anzahl der enthaltenen Objekte
- Über die `get(int i)` Methode kann man sich einzelne Objekte zurückgeben lassen (beginnt bei 0 zu zählen)
- Ein Blick in das javadoc der Java API kann hilfreich sein:
  `https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html`

[1] `https://en.wikipedia.org/wiki/FASTA_format`

```
import java.io.*;
import java.util.ArrayList;
public class Proteome{
    ArrayList<Protein> proteins;

    public Proteome(File fastaFile){
        proteins = new ArrayList<>();
        try{
            BufferedReader br = new BufferedReader(new FileReader(fastaFile));
            String line;
            StringBuilder sb = new StringBuilder();
            String id = "";
            while ((line=br.readLine())!=null){
                if (line.startsWith(">")){
                    if (sb.length()>0){
                        proteins.add(new Protein(id,sb.toString()));
                    }
                    id = line.substring(1);
                    sb = new StringBuilder();}
                else {
                    sb.append(line);
                }
            }
            br.close();
                if(id.length()>0){
                        proteins.add(new Protein(id, sb.toString()));
                }
        }
        catch(IOException e){
            e.printStackTrace();
        }
    }
    public int getProteinCount(){
        return proteins.size();
    }

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

2. Proteom-Statistiken

Erweitere die Proteome-Klasse um eine Methode `void writeStatistics(File output)`, die folgende Statistiken der geladenen Proteine in die gegebene Datei in einem selbst gewählten Format schreibt:

(a) Minimale Sequenzlänge

(b) Maximale Sequenzlänge

(c) Mittelwert der Sequenzlänge

```
public void writeStatistics(File output){
    int minLength = Integer.MAX_VALUE;
    int maxLength = Integer.MIN_VALUE;
    int sumLengths = 0;
    for(int i=0; i<proteins.size(); i++){
        Protein p = getProtein(i);
        sumLengths += p.length();
        if(p.length()<minLength){
           minLength = p.length();
        }
        if(p.length()>maxLength){
            maxLength = p.length();
        }
    }
    double avrgLength = (double)sumLengths/getProteinCount();
    try{
        PrintWriter pw = new PrintWriter(new FileWriter(output));
        pw.println("minimal length: "+minLength);
        pw.println("maximal length: "+maxLength);
        pw.println("average length: "+avrgLength);
        pw.println();
        pw.close();
    }
    catch(IOException e){
        System.err.println("unable to write statistics to "+output.getAbsolutePath
        e.printStackTrace();
    }
}
```

3. Aminosäure-Statistiken

Als zusätzliche Statistik soll jetzt auch noch die Anzahl jeder Aminosäure mit ausgegeben werden. Erweitere dazu die Protein-Klasse um die Methode `int countAminoAcids(char aa)`, die die Anzahl der gegebenen Aminosäure in der Sequenz des Proteins zurückgibt. Erweitere dann die Proteome-Klasse um die Methode `int countAminoAcids(char aa)`, die `countAminoAcids` für jedes Protein aufruft und die Gesamtzahl zurückgibt.

Die Methode `void writeStatistics(File output)` aus der Proteome-Klasse soll dann diese Methode für jede Aminosäure (aus `Protein.aminoAcids`) aufrufen und die Ergebnisse mit in die Ausgabe-Datei schreiben.

Hinweis: Dass beide geforderten Methoden `countAminoAcids` heißen, ist natürlich kein Problem, da sie zu verschiedenen Klassen gehören.

```
//in Proteome:
    public int countAminoAcids(char c){
        int count = 0;
        for(int i=0; i<proteins.size(); i++){
            count += getProtein(i).countAminoAcids(c);
        }
        return count;
    }

//in der writeStatistics Methode in Proteome vor dem pw.close() Statement:
    for(int i=0; i<Protein.aminoAcids.length(); i++){
        char c = Protein.aminoAcids.charAt(i);
        pw.println(c+": "+countAminoAcids(c));
    }
```

```
//in der Protein Klasse:
        public int countAminoAcids(char a){
            int count=0;
        for(int i=0; i<sequence.length(); i++)
        {
            char c = sequence.charAt(i);
            if(c==a)
            {
                count++;
            }
        }
        return count;
    }
```