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

Wintersemester 2019/20

Übungen zum Propädeutikum
Programmierung in der Bioinformatik

Blatt 4

Termin: Dienstag, 12. November 2019

Hinweis: Denkt daran, dass einige der in den Aufgaben verlangten Dinge bereits als vorgefertigte Methoden existieren. Im Zweifel also erst mal in der Java API nachsehen, ob die gewollte Funktion bereits existiert.

Übung 1 *ArrayList*

1. Erstelle eine ArrayList mit Strings und füge ihr fünf Elemente hinzu. Gib danach den Inhalt der ArrayList mit Hilfe eines for-each Loops auf der Konsole aus.
2. Entferne das dritte Elemente aus der ArrayList.
3. Vertausche das zweite mit dem ersten Element der ArrayList.
4. Sortiere die ArrayList.
5. Kehre die Reihenfolge der Elemente in der ArrayList um.

Lösung 1

```java
//1
ArrayList<String> cars = new ArrayList<>();

cars.add("Volvo");
cars.add("Ford");
cars.add("Mazda");
cars.add("BMW");

for (String car : cars)
    System.out.println(car);

//2
cars.remove(2);

//3
String tmp = cars.get(0);
cars.set(0, cars.get(1));
cars.set(1, tmp);


//4
Collections.sort(cars);

//5
Collections.reverse(cars);
```

Übung 2 *HashSet*

1. Erstelle ein HashSet mit Integers und füge ihm fünf Elemente hinzu. Gib danach den Inhalt des HashSets mit Hilfe eines for-each Loops auf der Konsole aus.

2. Lass dir die Mächtigkeit des Sets auf der Konsole ausgeben.
3. Wandele das HashSet in einen Array um.
4. Entferne alle Elemente aus dem HashSet.

Lösung 2

```java
//1
HashSet<String> cars2 = new HashSet<String>();

cars2.add("Volvo");
cars2.add("BMW");
cars2.add("Ford");
cars2.add("Mazda");

System.out.println(cars);

for (String car : cars){
    System.out.println(car);
}

//2
System.out.println(cars.size());

//3
Object[] carsArray = cars2.toArray();

//4
cars2.clear();
```

Übung 3 *HashMap*

1. Erstelle eine HashMap mit Datentyp String für die Schlüssel und Datentyp Integer für die Werte.
2. Füge der HashMap fünf *mappings* hinzu. Die Schlüssel sollen dabei die fünf bevölkerungsreichsten Städte Deutschlands sein und die Werte die dazugehörigen Einwohnerzahlen.
3. Gib die Einwohnerzahl von Berlin auf der Konsole aus.

Lösung 3

```java
//1
HashMap<String, Integer> einwohnerZahlen = new HashMap<>();

//2
einwohnerZahlen.put("Berlin", 3644826);
einwohnerZahlen.put("Hamburg", 1841179);
einwohnerZahlen.put("München", 1471508);
einwohnerZahlen.put("Köln", 1085664);
einwohnerZahlen.put("Frankfurt am Main", 753056);

//3
System.out.println(einwohnerZahlen.get("Berlin"));
```

Übung 4 *Einfaches Einlesen einer Datei & relative Häufigkeit*

In dieser Aufgabe soll ein FASTA-File eingelesen werden. Das FASTA-Format[1] wird genutzt um Nukleotid- oder Aminosäuresequenzen zu speichern. Dabei steht immer ein *Header*, also eine Kopfzeile, vor jeder Sequenz der mit einem `>` beginnt und den Namen und andere Informationen zur nachfolgenden Sequenz enthält. Nach einem Header folgen eine oder mehrere Zeilen der *Sequenz*, die durch den Ein-Buchstaben-Code für Nukleotide oder Aminosäuren dargestellt wird. Hier beispielsweise eine Sequenz des PLS-Gens aus *Arabidopsis thaliana*:

```
>NC_003075.7:c18329699-18329094 Arabidopsis thaliana chromosome 4 sequence
GTATCGCATTTGTTTCAAGTTTTTTTTTCTATAATGTTTCTTCGAAATCCATGATCATATAGTATATAAG
AAGCATGTATTTATAATGTTCCACTTAATATATTAGTATTGGAGACTAAAGCGAACATATAAAACCCAAA
TAAACCTTTCTTTAAGTTTTATTAAAAGTCTAAACACTTGATTTGTGTTTTAGTTTGGGTAGTAGTGAGA
AAAGAAAAATAAATAATCAAAAAGATTAAAGAAGAAAGAATTTGAAAGCAAGGAACACGAAATCCGAAGA
GCGAGGGGAGCGAAGACAGTCCACGTAGCTGCAGAGAGAAAGAGAAGAGCACGTGAGGCACACGTTCCTT
GTGTAAGACTTGTTGTGGTGATGTTGGCGCAGTGTCTCACTGAAACATGAATGAAACCCAGACTTTGTTT
TAATTTCAGGCGAAGGTCCATTTCTCCATGTTATATATCAATCTCTTATTTATTAGTAGCAAAATTGTTT
AAACTTTTTAAAATCCATTGATCACCCTATCATTTTCAATATCTACATACAATCTTATGTCTCGATAAAG
GTTTATCTTTATCTTATTATGCAATACATATCCCTCCCATTTCTAT
```

Erstelle nun eine Klasse `Sequence`. Die Klasse soll folgende Anforderungen erfüllen:

1. Die Klasse soll eine FASTA-Datei einlesen und die Nukleotidsequenz als String speichern. In *dieser* Aufgabe gehen wir der Einfachheit halber davon aus, dass in der Datei nur ein Header und eine Sequenz vorhanden sind. Die FASTA-Datei steht auf der Propädeutikums-Website zum Download bereit.
2. Die Klasse soll die relativen Häufigkeiten der Nukleotide in der Sequenz berechnen und auf der Konsole ausgeben. Die korrekten Werte für die angegebene Datei sind:

```
A=0.3415841584158416; C=0.16831683168316833; G=0.1551155115511551; T=0.334983498349835
```

Lösung 4

```java
import java.io.IOException;
import java.nio.file.FileSystems;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.ArrayList;

public class RelativeFrequency {
    public static String readSingleSequenceFasta(ArrayList<String> lines) {
        String sequence = "";
        for (String line : lines) {
            if (!line.startsWith(">")) {
                sequence += line;
            }
        }
        return sequence;
    }

    public static double[] computeRelativeFrequency(String sequence) {
        int a = 0;
        int c = 0;
        int g = 0;
        int t = 0;
        for (char nucleotide : sequence.toCharArray()) {
            if (nucleotide == 'A') {
                a++;
```

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

```
            } else if (nucleotide == 'G') {
                g++;
            } else if (nucleotide == 'C') {
                c++;
            } else if (nucleotide == 'T') {
                t++;
            }
        }
        double[] frequencies = new double[4];
        frequencies[0] = a*1.0/sequence.length();
        frequencies[1] = g*1.0/sequence.length();
        frequencies[2] = c*1.0/sequence.length();
        frequencies[3] = t*1.0/sequence.length();
        return frequencies;
    }

    public static void main(String[] args) throws IOException {
        Path path = FileSystems.getDefault().getPath("test.fasta");
        ArrayList<String> lines = (ArrayList<String>) Files.readAllLines(path);
        String seq = readSingleSequenceFasta(lines);
        double[] freqs = computeRelativeFrequency(seq);
        System.out.println("A=" + freqs[0] + "; C=" + freqs[1] + ";
            G=" + freqs[2] + "; T=" + freqs[3]);
    }
}
```

Übung 5 *Knobelaufgabe: Computing GC-Content*

Bearbeite die Aufgabe Computing GC-Content in Rosalind. Hier ist zu beachten, dass in der FASTA-Datei nicht mehr nur eine, sondern mehrere Header und Sequenzen stehen.

Lösung 5

```
import java.io.IOException;
import java.nio.file.FileSystems;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.HashMap;

public class GcContent {
    public static HashMap<String,String> createSequenceHashMap(ArrayList<String> lines) {
        HashMap<String, String> sequences = new HashMap<>();
        String header = "";
        String sequence = "";
        boolean isFirstLine = true;
        for (String line : lines) {
            if (line.startsWith(">")) {
                if (isFirstLine) {
                    header = line.substring(1);
                    isFirstLine = false;
                } else {
                    sequences.put(header, sequence);
                    sequence = "";
                    header = line.substring(1);
                }
            } else {
```

```
                sequence += line;
            }
        }
        sequences.put(header, sequence);
        return sequences;
    }

    public static double computeGcContent(String sequence) {
        int totalGC = 0;
        for (char nucleotide : sequence.toCharArray()) {
            if (nucleotide == 'G' || nucleotide == 'C') totalGC++;
        }
        return totalGC * 1.0 / sequence.length();
    }

    public static void printHighestGcContent(HashMap<String,String> sequences) {
        String maxHeader = "";
        double maxValue = -1;
        for (HashMap.Entry<String, String> entry : sequences.entrySet()) {
            String currentHeader = entry.getKey();
            String currentSequence = entry.getValue();
            double currentGcContent = computeGcContent(currentSequence);
            if (computeGcContent(currentSequence) > maxValue) {
                maxHeader = currentHeader;
                maxValue = currentGcContent;
            }
        }
        System.out.println(maxHeader + '\n' + maxValue);
    }

    public static void main(String[] args) throws IOException {
        Path path = FileSystems.getDefault().getPath("test.fasta");
        ArrayList<String> lines = (ArrayList<String>) Files.readAllLines(path);
        HashMap<String,String> sequences = createSequenceHashMap(lines);
        printHighestGcContent(sequences);
    }
}
```