Dozent
Prof. Dr. Thomas Vetter
Departement
Mathematik und Informatik
Spiegelgasse 1
CH – 4051 Basel

Assistenten
Bernhard Egger
Andreas Forster

Tutoren
Marvin Buff
Sein Coray
Eddie Joseph
Loris Sauter
Linard Schwendener
Florian Spiess

Webseite
http://informatik.unibas.ch/hs2017/uebung-erweiterte-grundlagen-der-programmierung/

Erweiterte Grundlagen der Programmierung (45398-01)

Blatt 6 [8 Punkte]

Vorbesprechung	30. Okt - 03. Nov
Abgabe	6. - 10. Nov (vor dem Tutorat)

Wir empfehlen Ihnen, dass Sie im Buch "Sprechen Sie Java" bis und mit Kapitel 13 lesen, bevor Sie beginnen die Übungen zu lösen.

Aufgabe 1 - Evolution [4 Punkte]

Wir wollen in dieser Aufgabe eine Art Evolution simulieren. Dies anhand eines vereinfachten Models, welches nicht den Anspruch erhebt, im biologischen Sinne korrekt zu sein. Wir nehmen an, dass jedes Lebewesen durch eine Gen-Sequenz beschrieben werden kann. Eine Gen-Sequenz soll eine Folge der Buchstaben 'A', 'C', 'G' und 'T' sein. Diese Gen-Sequenzen verändern sich mit der Zeit. Dies wollen wir nun simulieren. In der Aufgabe müssen Sie dazu drei Klassen vervollständigen.

Vorbereitung

- Laden Sie sich als erstes den Code von der Übungswebseite herunter und entpacken Sie diesen.
- Kompilieren Sie die Klasse *TestGeneticEvolution* und führen Sie diese aus.

Genom initialisieren

- Schauen Sie sich nun die Klasse *Genom* an. Überlegen Sie sich, wie Sie die Sequenz der Buchstaben speichern wollen, und definieren Sie die nötigen Felder. Wir empfehlen ein *Array* von *char*s.
- Implementieren Sie nun anhand der Kommentare den Konstruktor *Genom*, sowie die Methoden *getChar*, *randomize* und *toString*.
- Testen Sie Ihre Implementierung, indem Sie die Test-Klasse *TestGeneticEvolution* kompilieren und ausführen.

GenPool initialisieren

- Schauen Sie sich nun die Klasse *GenPool* an. Überlegen Sie sich, wie Sie mehrere Instanzen der Klasse Genom verwalten wollen. Definieren Sie die nötigen Klassen-Attribute.
- Implementieren Sie nun anhand der Kommentare den Konstruktor *GenPool* sowie die Methoden *randomize* und *toString*.
- Testen Sie ihre Implementierung, indem Sie die Test-Klasse *TestGeneticEvolution* kompilieren und ausführen.

Evolution

- Implementieren Sie die nun die in der Klasse *Genom* die Methode *isEqual*, sowie in der Klasse *GenPool* die Methode *contains*.
- Vervollständigen Sie nun in der Klasse *Genom* die Mutationen *pointMutation*, *insertion* und *deletion*.
- In der Klasse GenPool müssen Sie nun noch die Methode *mutate* implementieren.
- Vervollständigen Sie die Simulation in dem Sie in der Klasse *TestGeneticEvolution* die Methode *mutateTo* implementieren.
- Testen Sie ihre Implementierung, indem Sie die Test-Klasse *TestGeneticEvolution* kompilieren und ausführen.

Aufgabe 2 - Kassenbon [4 Punkte]

In dieser Aufgabe werden Sie ein Programm schreiben, welches für einen Einkauf einen Kassenzettel erstellt. Die Hauptklasse `Kasse` in der Datei `Kasse.java` steht Ihnen als Grundgerüst zur Verfügung - daran sollen Sie nichts ändern.

```
1 public class Kasse {
2   public static void main(String[] args) {
3   Kassenbon b = new Kassenbon( new Adresse("Herbstmesse Basel", "Uni Basel", "Petersplatz",
        "1", "4001", "Basel"));
4   b.add( new Artikel("Marroni",2,5.40));
5   b.add( new Artikel("Magebrot",5,1.10));
6   b.add( new Artikel("Glühwein",2,6));
7   b.print();
8   }
9 }
```

Ziel ist es nun, die benötigen Klassen `Kassenbon`, `Artikel` und `Adresse` zu erstellen, um folgende Ausgabe zu erhalten:

```
|==========================|
|    Herbstmesse Basel     |
|        Uni Basel         |
|      Petersplatz 1       |
|        4001 Basel        |
|==========================|

Marroni       2 x  5.40
                        10.80
Magebrot      5 x  1.10
                         5.50
Glühwein      2 x  6.00
                        12.00

----------------------------
Total                   28.30
============================
```

Wenn sie bereits Programmiererfahrung haben, können Sie versuchen diese Aufgabe ohne die weiteren Hilfestellungen zu lösen – die Hilfestellung dient dann zum Vergleich der verschiedenen Lösungsansätze. Ansonsten folgen Sie einfach den folgenden Hinweisen. Das Erscheinungsbild des Kassenzettels steht dabei im Hintergrund.

- Erstellen Sie die Dateien für die Klassen `Kassenbon`, `Artikel` und `Adresse`.
- Leiten Sie aus der Hauptklasse ab, welche Felder Sie in den jeweiligen Klassen benötigen.
- Schreiben Sie in jeder Klasse einen Konstruktor, der diese mit den übergebenen Werten füllt.
- Fügen Sie der Klasse `Kassenbon` eine Liste hinzu, die Artikel halten kann: `ArrayList<Artikel> artikelliste = new ArrayList()` (die Bedeutung der eckigen Klammern lernen Sie später in der Vorlesung kennen, Sie erhalten eine

Liste die mit Objekten des Typs `Artikel` gefüllt werden kann). Sie müssen dazu folgenden Teil der API importieren: `import java.util.ArrayList;`.

- Fügen Sie der Klasse `Kassenbon` eine Methode `add` hinzu, um der `artikelliste` einen zusätzlichen `Artikel` hinzuzufügen. Suchen Sie die benötigte Methode, um diesen `Artikel` in der `ArrayList` hinzuzufügen, in der API-Dokumentation.
- Fügen Sie den Klassen `Kassenbon`, `Artikel` und `Adresse` je eine Methode `print` hinzu, diese darf noch leer sein.
- Ihr Gesamt-Programm sollte nun bereits kompilieren, sie müssen dazu nur das Kompilieren der Hauptklasse ausführen.
- Fügen Sie der Klasse `Artikel` eine Methode `getPrice` hinzu, welche den Preis dieses Postens zurückgibt.
- Schreiben Sie nun die `print`-Methoden für `Artikel` und `Adresse` - achten Sie in einem ersten Schritt nur auf den Inhalt, die Formatierung erfolgt später.
- Schreiben Sie nun auch die `print`-Methode für `Kassenbon` diese soll die `print`-Methoden für `Artikel` und `Adresse` aufrufen und das Total berechnen.
- Die Ausgabe sollte nun inhaltlich gleich sein wie die obige Vorlage.
- Versuchen Sie nun die Formatierung der Vorlage anzupassen. Hilfreich ist dazu die Funktion `String.format`.