Einführung in die Programmierung

WS 2019/2020, Blatt 04

Bauhaus-Universität Weimar

Prof. Norbert Siegmund
André Karge

Ausgabe: 25.11.2019
Abgabetermin: Montag, 02.12.2019, 11:00
Besprechung: 05.12.2019

Bitte lesen Sie die folgenden Informationen zum Übungsablauf **sorgfältig** durch.

Grundsätzlich – wenn nicht anders angegeben – sind die Lösungen zu den Übungen zu Einführung in die Programmierung jeden **Montag bis spätestens 11:00 Uhr** an André Karge per E-Mail zu schicken.

Schreiben Sie bitte im Betreff Ihrer E-Mail Ihre **Teamnummer** sowie die Nummer des Übungsblattes. In der E-Mail schreiben Sie bitte zusätzlich Ihren **Namen** und **Matrikelnummer**. Die Lösungen für Sie bitte als Java Dateien als Anlage hinzu. Es werden **keine** kompilierten Dateien, wie *.class oder *.jar angenommen.

Übungen müssen von **minimal zwei** und **maximal drei** Studierenden aus derselben Übungsgruppe in einem festen Team bearbeitet werden (Ausnahmen nur auf Anfrage beim Übungsleiter). Pro Team soll die Lösung nur einmal abgegeben werden. Aufgaben sollen **im Team gelöst** und nicht nur vom Team abgegeben werden. Sie müssen mindestens **50%** dieser Punkte für eine Zulassung zur Prüfung erreichen. Das **Abschreiben** identischer Lösungen wird jeweils mit 0 Punkten bewertet.

Bei Fragen oder Unklarheiten wenden Sie sich bitte **vor der Abgabe** des Übungsblattes an den Übungsleiter (per E-Mail oder persönlich). Es soll nie jemand sagen müssen: „Wir haben die Aufgabe nicht verstanden und konnten sie daher nicht bearbeiten."
Weitere Informationen, wie aktuelle Ankündigungen, finden Sie online (`https://www.uni-weimar.de/de/medien/professuren/intelligente-softwaresysteme/lehre/`) unter Einführung in die Programmierung

Aufgabe 1 Advent Advent (7 Punkte)

(a) Implementieren Sie eine Klasse Adventskalender.

- Die 24 Türchen (Array) sind mit Objekten vom Typ `Sweetie` gefüllt.
- `Sweetie` ist ein Aufzählungstyp. Es gibt: Lebkuchen, Plaetzchen, Praline, Schoko-Nikolaus, Gummibaerchen und Spekulatius.
- Der Konstruktor initialisiert die Türchen per Zufallsgenerator.
- Stellen Sie sicher, dass auf die Türchen nicht direkt von außerhalb der Klasse zugegriffen werden kann.
- Die Methode `public String openDoor(int day)` öffnet das entsprechende Türchen, falls `day` heute ist oder in der Vergangenheit liegt, sonst erfolgt die Rückgabe: Heute ist noch nicht der `day`. Dezember.(Hinweis: (new java.util.GregorianCalendar()).get(java.util.Calendar.DAY_OF_MONTH) ist der aktuelle Tag). Verbirgt sich hinter dem Türchen ein Schoko-Nikolaus wird zusätzlich ein Sound abgespielt (Verwenden Sie die in von der Vorlesungsseite herunterladbare Datei MyAudio.java und die x-mas.wav. Der Aufruf erfolgt, z.B. `MyAudio.play(/x-mas.wav");`)

(b) Schreiben Sie eine main-Methode, zu Interaktion mit einem Benutzer. Es wird abgefragt, welches Türchen er öffnen will. Der Inhalt wird angezeigt. Und danach ob er noch weitere Türchen öffnen will, bis er dies verneint und das Programm so beendet.

Aufgabe 2 Array Referenzen (4 Punkte)

Es sollen von sechs Primzahlen die Wurzel und das Quadrat berechnet werden und jeweils in einem Array gespeichert werden. Dazu wurde folgender Code implementiert:

```
double[] primeNumbers = {3, 5, 7, 11, 13, 17};

double[] sqrtOfPrimeNumbers = primeNumbers;
for (int i = 0; i < 6; ++i)
{
  sqrtOfPrimeNumbers[i] = Math.sqrt(sqrtOfPrimeNumbers[i]);
}

double[] squaredPrimeNumbers = primeNumbers;
for (int i=0; i<6; ++i)
{
```

```
  squaredPrimeNumbers[i] = squaredPrimeNumbers[i] * squaredPrimeNumbers[i];
}
```

Lassen Sie sich die Ergebnisse des obigen Codes ausgeben um deren Korrektheit zu überprüfen. Erklären Sie warum die Ergebnisse (nicht) korrekt sind und implementieren Sie gegegebenenfalls eine korrekte Lösung.

Aufgabe 3 Stack / Heap (14,5 Punkte)

Gegeben sind folgende Klassen in JAVA:

```
public class Person {
  private String prename;
  private String surname;

  public Person(String prename,
                String surname) {
    this.prename = prename;
    this.surname = surname;
  }

  public void putOneVeg(Grill grill) {
    grill.putOneVeg();
  }

  public void putOneSteak(Grill grill) {
    grill.putOneSteak();
  }

  public void fillGrillWithSteak(Grill grill) {
    grill.fillSteak();
  }
}
```

```
public class Steak {
  private String id;

  public Steak(String id) {
  this.id = id;
  }
}
```

```
public class Vegetable {
  private String id;

  public Vegetable(String id) {
  this.id = id;
  }
}
```

```
public class GrillParty {
  public static void main(String[] args) {
    Person willy = new Person("Willy", "Grilly");

    Grill g1 = new Grill("red", willy, 3, 2);

    Person frank = new Person("Frank", "Fire");

    Grill g2 = new Grill("blue", frank, 4, 2);

    willy.putOneVeg(g1);
    willy.putOneSteak(g2);
    frank.fillGrillWithSteak(g1);
    g2.assignOwner(willy);
  }
}
```

```
public class Grill {
  private String colour;
  private Person owner;
  private Steak[] steakSpace;
  private int steakIndex;
  private Vegetable[] vegSpace;
  private int vegIndex;

  public Grill(String colour,
               Person owner,
               int steakSize,
               int vegSize) {
    this.colour = colour;
    this.owner = owner;
    this.steakSpace = new Steak[steakSize];
    this.steakIndex = steakSize - 1;
    this.vegSpace = new Vegetable[vegSize];
    this.vegIndex = vegSize - 1;
  }

  public void assignOwner(Person owner) {
    this.owner = owner;
  }

  public void putOneSteak() {
    if (this.steakIndex >= 0) {
      String steakName = "s" + steakIndex;
      this.steakSpace[steakIndex] = new Steak(steakName);
      --this.steakIndex;
    }
  }

  public void putOneVeg() {
    if (this.vegIndex >= 0) {
      String vegName = "v" + vegIndex;
      this.vegSpace[vegIndex] = new Vegetable(vegName);
      --this.vegIndex;
    }
  }


  public void fillSteak() {
    while (this.steakIndex >= 0) {
      this.putOneSteak();
    }
    //Hier Heap und Stack skizzieren
  }
}
```

- Stellen Sie graphisch alle Objekte sowie die Beziehungen zwischen den Objekten am Ende der `main`-Methode dar. Dabei soll jedes Objekt durch ein Rechteck repräsentiert werden, das den Namen der Klasse, von der das Objekt eine Instanz ist, und alle Attribute zusammen mit ihren Werten enthält.

- Zeichnen Sie Referenzen als Pfeile von der Stelle, die die Referenz enthält, zu dem Objekt, das referenziert wird.
- Fügen Sie auch die Variablen der `main`-Methode in Ihre Zeichnung ein. Schreiben Sie dazu den Namen jeder Variablen in die Zeichnung und fügen Sie entsprechende Pfeile jeweils von dort aus ein.
- Entwickeln Sie Ihr Diagramm Zeile für Zeile. Sollten Werte überschrieben oder geändert werden, so soll dies durch ein (lesbares) „Durchstreichen“ in der Zeichnung erkennbar sein. Sollten Sie eine Referenz abändern müssen, so kennzeichnen Sie die alte Referenz (also die gelöschte Referenz) durch einen Kringel um die Pfeilspitze und einen Kringel um den Startpunkt des Pfeils.
- Sie können `String`s als primitive Datentypen interpretieren, dafür müssen also keine zusätzlichen Objekte gezeichnet werden.
- Attribute, die Arrays sind, können Sie ebenfalls innerhalb der zugehörigen Klasse als Rechteck mit Feldunterteilung zeichnen.
- Skizzieren Sie zusätzlich den Inhalt von Heap und Stack während des Programmablaufs am Ende von `fillSteak()` (siehe Kommentar), kommend aus der Main-Methode durch den Aufruf `frank.fillGrillWithSteak(g1)`.