Lehrstuhl für Medieninformatik
Institut für Information und Medien, Sprache und Kultur
Universität Regensburg

Klausur zur Vorlesung: Einführung in die objektorientierte Programmierung mit Java im Wintersemester 2019/20

Alexander Bazo 18. Februar 2020

Allgemeine Hinweise

1. Die Bearbeitungszeit beträgt 60 Minuten. Sie können 50 Punkte erreichen.
2. Schreiben Sie Ihren Namen, Vornamen, Studiengang und Ihre Matrikelnummer leserlich unten auf jedes der 10 Blätter bevor Sie mit der Bearbeitung beginnen! Blätter ohne diese Angaben werden nicht gewertet. Wenn Sie die Rückseite eines Blattes verwenden, notieren Sie dies bitte auf der Vorderseite. Kennzeichnen Sie eindeutig, zu welcher Aufgabe eine Lösung gehört.
3. Verwenden Sie für das Lösen der Aufgaben die Programmiersprache Java. Alle gegebenen Code-Fragmente sind ebenfalls in Java verfasst. Halten Sie sich an die bekannten Regeln zur Codequalität.
4. Falls nicht anders angegeben gilt: Sie können Begriffe in englischer oder deutscher Sprache verwenden. Antworten können in Stichworten gegeben werden.
5. Benutzen Sie keine Bleistifte, keine rotschreibenden Stifte und kein TippEx (oder ähnliche Produkte).
6. Die Klausur ist als *Closed Book*-Klausur angelegt. Sie dürfen keine mitgebrachten Quellen oder Notizen zur Bearbeitung der Aufgaben verwenden. Technische Hilfsmittel sind ebenfalls nicht erlaubt.
7. Wenden Sie sich bei Unklarheiten in den Aufgabenstellungen immer an die Klausuraufsicht (Hand heben). Hinweise und Hilfestellungen werden dann, falls erforderlich, offiziell für den gesamten Hörsaal durchgegeben. Aussagen unter vier Augen sind ohne Gewähr.
8. Geben Sie keine mehrdeutigen (oder mehrere) Lösungen an. In solchen Fällen wird stets die Lösung mit der geringeren Punktzahl gewertet. Eine richtige und eine falsche Lösung zu einer Aufgabe ergeben also null Punkte. Das gilt auch für die *Multiple Choice*-Fragen. Kreuzen Sie bei einer Frage zwei richtige und zwei falschen Möglichkeiten an, ergibt das in der Summe null Punkte.
9. Falls der Platz für Ihre Lösungen nicht ausreichen sollte, benutzen Sie bitte die leeren Seiten zwischen den Teilen bzw. die Rückseiten der Angabenblätter. Verweisen Sie an gegebener Stelle auf die Fortsetzung Ihrer Lösung.
10. Schreiben Sie leserlich. Wenn wir es nicht entziffern können, können wir Ihnen keine Punkte geben.

Name, Studiengang & Matrikelnummer:

Teil 1: Theorie

Antworten können in Stichworten gegeben werden. Begründungen sind nur dort notwendig, wo auch nach einer Beschreibung gefragt wird.

1) Variablen und Konstanten (2 Punkte)

Nennen Sie einen wesentlichen Unterschied zwischen *Variablen* und *Konstanten* in Java. Beschreiben Sie einen konkreten Anwendungsfall, in dem eine *Konstante* einer *Variable* vorzuziehen ist.

2) Vererbung (2 Punkte)

Welche der folgenden Aussagen trifft auf das Vererbungskonzept in der Programmiersprache Java zu?

- □ Eine Unterklasse kann auf alle geerbten Eigenschaften der Superklasse direkt zugreifen.
- □ Eine Klasse kann von mehreren Superklassen erben, aber nur ein *Interface* implementieren.
- □ Durch das Erstellen einer Methode mit gleicher Signatur kann eine geerbte Methode in der Unterklasse überschrieben werden.
- □ Auf die Konstruktoren der Superklasse kann mit dem Schlüsselwort `super` zugegriffen werden.

3) Interfaces (2 Punkte)

Nennen Sie einen zentralen Bestandteil von *Interfaces* in Java. Beschreiben Sie die Konsequenzen, die sich aus dem Implementieren eines *Interfaces* für eine Klasse ergeben.

4) Event-basierte Programmierung (4 Punkte)

Beschreiben Sie wann und warum beim Ausführen einer Java-Anwendung ein *Stackoverflow*-Fehler auftreten kann.

Teil 2: Codeverständnis

5) Codeanalyse (5 Punkte)

Gegeben ist die Methode `doStuff`. Beschreiben Sie kurz Aufbau und Ablauf der Methode und nennen Sie anschließend den wahrscheinlichen Einsatzzweck. Gehen Sie davon aus, dass der gegebene Code korrekt ist.

```
1  public String doStuff(String[] x) {
2    if(x.length == 0) {
3      return null;
4    }
5    String y = x[0];
6    for(int i = 1; i < x.length; i++) {
7      if(x[i].length() > y.length()) {
8            y = x[i];
9          }
10   }
11   return y;
12 }
```


Name, Studiengang & Matrikelnummer:

6) Fehler finden (5 Punkte)

Finden Sie die **Konventionsverstöße, Syntaxfehler, Bugs und Logikfehler** in dem folgenden Code-Abschnitt. Markieren Sie die fehlerhaften Stellen im Code und beschreiben Sie den jeweiligen Fehler kurz unter Angabe der Zeilennummer. Schlecht gewählte Bezeichner für Methoden und Variablen sowie fehlerhafte Einrückungen zählen nicht als Fehler. Im Code sind fünf Fehler zu finden.

```
1  public class BagOfNumbers {
2    private ArrayList<Double> NUMBERS;
3
4    public BagOfNumbers() {
5      numbers = new ArrayList<Double>;
6    }
7
8    public void addNumber(double number) {
9      numbers.add(number);
10   }
11
12   public double getAverage() {
13     double average = 0;
14     if(numbers.size() = 0) {
15         return average;
16     }
17     for(Double value: numbers) {
18         average = value;
19     }
20     return  average / numbers.length();
21   }
22 }
```


Name, Studiengang & Matrikelnummer:

Teil 3: Code implementieren

7) Baumschule (20 Punkte)

In dieser Aufgabe implementieren Sie zwei Klassen einer fiktiven Software für die Pflanzenzucht. Implementieren Sie - falls nicht anders angegeben - vollständige Klassen und achten Sie bei der Implementierung Ihres eigenen Codes auf sinnvolle Modifikatoren für die Sichtbarkeit der Klassenbestandteile, korrekte Datentypen und verständliche Bezeichner.

7a) Implementieren Sie eine Klasse `Plant`, die eine einzelne Pflanze repräsentiert. Jede Instanz verfügt über private Eigenschaften für die aktuelle Wuchshöhe und den Wasserstand, der zwischen `0` und `10 Liter` liegen kann. Diese Eigenschaften können über entsprechende, öffentliche Methoden ausgelesen werden. Wuchshöhe und Wasserstand werden im Konstruktor mit für alle Pflanzen gleichen, zentral festgelegten, Startwerten initialisiert.

7b) Implementieren Sie für die Klasse `Plant` zwei öffentliche, parameterlose Methoden `water` und `grow`. Durch den einmaligen Aufruf von `water` wird der Wasserstand der Pflanze um `1 Liter` erhöht, falls der maximale Wasserstand von `10 Liter` noch nicht erreicht ist. Bei jedem Aufruf der Methode `grow` wächst die Pflanze um `4 cm` und verbraucht dazu `1 Liter` des verfügbaren Wassers. **Diese Werte gelten für alle Pflanzen und werden an entsprechender Stelle der Klasse abgebildet.** Die Pflanze wächst nur, wenn ausreichend Wasser verfügbar ist.

7c) Implementieren Sie außerhalb der erstellten Klassen eine öffentliche, statische Methode `getWateringList`. Die Methode erhält eine *ArrayList* mit `Plkant`-Objekten übergeben und gibt eine ebensolche *ArrayList* aus *Plant*-Objekten zurück. Die zurückgegebene Liste enthält all diejenigen Pflanzen, deren Wasserstand einen bestimmten Grenzwert unterschreitet. Der Grenzwert wird über einen zusätzlichen Parameter der Methode bestimmt.

7d) Implementieren Sie eine Klasse `FancyPlant` die von `Plant` erbt und über zwei zusätzliche Eigenschaften für den aktuellen und den optimalen Standort verfügt. Die Standorte werden in Textform gespeichert und als Parameter an den Konstruktor übergeben. Optimaler und aktueller Standort können über entsprechende öffentliche Methoden ausgelesen werden. Der aktuelle Standort kann über eine entsprechende öffentliche Methode angepasst werden. Das Wachstum von `FancyPlant`-Pflanzen ist um die Hälfte reduziert, wenn der aktuelle Standort nicht mit dem optimalen Standort übereinstimmt.

Name, Studiengang & Matrikelnummer:

8) Motion Detector (10 Punkte)

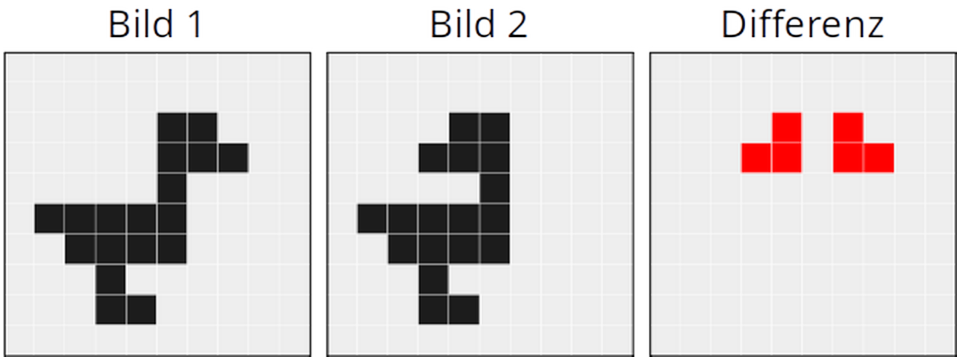

Abbildung 1: Vereinfachte Darstellung: Die erkannten, farblichen Unterschiede zwischen den Bildern 1 und 2 werden in der dritten Grafik dargestellt.

Vervollständigen Sie den gegebenen Code der `MotionDetector`-Klasse (siehe Seite 10) um einen einfachen Bewegungserkenner zu implementieren. Das Programm vergleicht zwei Bilder (Instanzen der `Image`-Klasse) pixelweise und stellt die Anzahl farblich unterschiedlicher Pixel an gleichen Positionen fest. Übersteigt dieser Wert einen festgelegten Prozentsatz, erkennt das Programm dies als Bewegung. Gehen Sie davon aus, das die beiden verwendeten Bilder über die gleichen Dimensionen (Höhe und Breite) verfügen und dass der gegebene Code korrekt ist. **Gehen Sie bei Ihrer Implementierung von der Existenz der wie folgt dokumentieren Methoden aus:**

Methoden der Klasse: `Image` (*Repräsentiert eine Rastergrafik*)

`public Pixel[][] getPixels()`: Gibt das repräsentierte Bild als *zwei-dimensionales* Array aus `Pixel` zurück. Die Länge des äußeren Arrays entspricht der Höhe, die Länge der inneren Arrays entspricht der Breite des Bildes in Pixel.

Methoden der Klasse: `Pixel` (*Repräsentiert einen Pixel als RGB-Farbe*)

`public boolean differsFrom(Pixel pixel, int threshold)`: Gibt `true` zurück, wenn die Graustufenrepräsentation des Pixels stärker als angegeben (Parameter `threshold`) vom ebenfalls als Parameter übergebenen Pixel `pixel` abweicht.

Aufgabe: Ergänzen Sie den Rumpf der Methode `differenceInPercent` in der Klasse `MotionDetector` (siehe Seite 10). Die Methode gibt den prozentualen Anteil an unterschiedlichen Pixel in den als Arrays übergebenen Bildern zurück. Als unterschiedlich gelten Pixel die in beiden Arrays an der selben Stelle (z.B. `[0,0]`) liegen aber über zu starke Farbabweichungen verfügen. Verwenden Sie zum Feststellen der Abweichung die auf Seite 9 beschriebene, öffentliche Methode `differsFrom` der Klasse `Pixel`.

```
1  public class MotionDetector {
2    // Grenzwerte für Farb- und Pixel-Abweichungen
3    private static final int PIXEL_COLOR_THRESHOLD = 10;
4    private static final double PIXEL_COUNT_THRESHOLD = 10;
5
6    private double differenceInPercent(Pixel[][] p1, Pixel[][] p2)
         {
7      // Ergänzen Sie hier Ihren Code
8    }
9
10   public boolean hasMotion(Image i1, Image i2) {
11     Pixel[][] p1 = i1.getPixels();
12     Pixel[][] p2 = i2.getPixels();
13     double difference = differenceInPercent(p1, p2);
14     if(difference >= PIXEL_COUNT_THRESHOLD) {
15         return true;
16     }
17     return false;
18   }
19 }
```


Name, Studiengang & Matrikelnummer: