Wiederholungsklausur zur Vorlesung: Einführung in die objektorientierte Programmierung mit Java im Wintersemester 2017/18

Alexander Bazo 26. März 2018

Allgemeine Hinweise

1. Die Bearbeitungszeit beträgt 60 Minuten. Sie können 50 Punkte erreichen.

2. 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.

3. Schreiben Sie Ihren Namen, Vornamen, Studiengang und Ihre Matrikelnummer leserlich unten auf jedes Angabenblatt 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.

4. Benutzen Sie keine Bleistifte, keine rotschreibenden Stifte und kein TippEx (oder ähnliche Produkte).

5. 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.

6. 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.

7. 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.

Teil 1: Theorie

1) Datentypen (3 Punkte)

Nennen Sie jeweils einen geeigneten Datentyp zur Speicherung von a) dem Namen, b) der Matrikelnummer und c) der aktuellen Durchschnittsnote eines Studierenden.

2) Schleifen (2 Punkte)

Nennen Sie zwei Schleifentypen. Nennen Sie für jeden Schleifen-Typ einen typischen Anwendungsfall.

3) Klassen (3 Punkte)

Können in einer Java-Klasse zwei Methoden mit dem identischen *Namen* existieren? Begründen Sie Ihre Entscheidung.

4) Schlüsselwörter (2 Punkte)

Kreuzen Sie alle Begriffe an, bei denen es sich nicht um Schlüsselwörter (*keywords*) der Programmiersprache Java handelt.

- ☐ new
- ☐ that
- ☐ stop
- ☐ super

Teil 2: Codeverständnis

5) Codeanalyse (5 Punkte)

Geben 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 static int doStuff(int[] l) {
2      int x = l[0];
3      if(l.length == 1) {
4          return x;
5      }
6      for(int i = 1; i < l.length; i++) {
7          int y = l[i];
8          if(y > x) {
9                  x = y;
10         }
11     }
12     return x;
13 }
```


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 CharReplacer {
2
3      private static final char defaultReplacementChar = '_';
4      private char replacementChar;
5
6      public CharReplacer() {
7          this.replacementChar = defaultReplacementChar;
8      }
9
10     public CharReplacer(char replacementChar) {
11         replacementChar = replacementChar;
12     }
13
14     public String replaceCharIn(String string; char
           charToReplace) {
15         String result = "";
16         for(char c: string.toCharArray()) {
17             if(c == charToReplace) {
18                 result += replacementChar;
19             } otherwise {
20                 result += c;
21             }
22         }
23         return true;
24     }
25 }
```


Name, Studiengang & Matrikelnummer:

Teil 3: Code implementieren

7) Bibliothek (15 Punkte)

In dieser Aufgabe implementieren Sie zwei Klassen einer Software zur Verwaltung des Buchbestands einer Bibliothek. Vorgegeben ist dabei die Klasse `Media`. Diese Klasse beschreibt alle Eigenschaften, die sich alle Medien der Bibliothek teilen. Die konkreten Exemplare im Bestand (z.B. Bücher oder Zeitschriften) werden durch Klassen repräsentiert, die von dieser Basisklasse erben. Für die Suche im Bestand können alle Medien mit einer beliebigen Anzahl an Schlagwörtern (*Tags*) ausgezeichnet werden. *Lagern Sie Ihren Code wenn nötig in weitere Methoden aus. Verwenden Sie sinnvolle Bezeichner für Methoden und Variablen und wählen Sie, wo nötig, passende Sichtbarkeitsbereiche. Implementieren Sie, wenn nicht anders verlangt, vollständige Klassen.*

```
1  public abstract class Media {
2
3      private String id;
4      private String title;
5      private ArrayList<String> tags;
6
7      public Media(String id, String title, ArrayList<String>
           tags) {
8          this.id = id;
9          this.title = title;
10         this.tags = tags;
11     }
12
13     public String getID() {
14         return id;
15     }
16
17     public String getTitle() {
18         return title;
19     }
20
21     // Gibt true zurück, wenn die ArrayList tags den String tag
           beeinhaltet
22     public boolean hasTag(String tag) {
23         return tags.contains(tag);
24     }
25
26 }
```

7a) Implementieren Sie eine Klasse `Book`. Diese repräsentiert ein einzelnes Buch der Bibliothek. Die Klasse erbt von `Media` und verfügt über zwei zusätzliche nicht-öffentliche Eigenschaften zum Speichern des Authoren bzw. der Authorin und des Erscheinungsjahr. Beide Eigenschaften werden über den Konsruktor gesetzt und sind über öffentliche *Getter*-Methoden auslesbar.

7b) Implementieren Sie eine zusätzliche Klasse `Library`, die eine unbestimmte Menge an Büchern verwalten kann. Die Klasse verfügt dazu über eine nicht-öffentliche Variable zum Speichern von Büchern (Instanzen der `Book`-Klasse) in einer *ArrayList*. Implementieren Sie eine öffenliche Methode `addBook`, die alle nötigen Informationen zum Erstellen eines neuen Buch-Objektes übergeben bekommt, dann ein solches Objekt instanziiert und anschließend in der *ArrayList* speichert.

7c) Ergänzen Sie in der Klasse `Library` eine zusätzliche öffentliche Methode `findBooks`, die einen *Tag* sowie ein zeitliches Intervall in Form zweier Jahreszahlen übergeben bekommt. Die Methode soll eine *ArrayList* mit allen Büchern zurückgeben, die von der Klasse `Library` verwaltet werden, innerhalb des angegebenen Zeitraums erschienen und mit dem angegebenen Tag ausgezeichnet sind.

Name, Studiengang & Matrikelnummer:

8) Bildverarbeitung (15 Punkte)

In dieser Aufgabe implementieren Sie verschiedene Schritte zur Manipulation von Rastergrafiken. Ergänzen Sie dazu die vorgegebenen Methoden in der Klasse `ImageProcessor`. Die zu verarbeitenden Grafiken werden jeweils durch ein zweidimensionalen *Array* vom Typen `Pixel` abgebildet. Jeder Eintrag in dem Array steht für einen Pixel des Bildes. Die Klasse `Pixel` bildet einen einzelnen Farbwert im RGB-Raum ab. *Lagern Sie Ihren Code wenn nötig in weitere Methoden aus. Verwenden Sie sinnvolle Bezeichner für Methoden und Variablen und wählen Sie, wo nötig, passenden Sichtbarkeitsbereiche.*

```
1  public class ImageProcessor {
2      private static Pixel convertToGrayscale(Pixel pixel) {
3          // Ergänze Sie hier Ihren Code
4      }
5
6      public static Pixel[][] getThresholdImage(Pixel[][] image,
           int grayscaleThreshold) {
7          // Ergänzen Sie hier Ihren Code
8      }
9  }
10
11 public class Pixel {
12     privat int r;
13     privat int g
14     privat int b;
15
16     public Pixel(int r, int g, int b) {
17         this.r = r;
18         this.g = g;
19         this.b = b;
20     }
21
22     public int getRed() {
23         return r;
24     }
25
26     public int getGreen() {
27         return g;
28     }
29
30     public int getBlue() {
31         return b;
32     }
33 }
```


Name, Studiengang & Matrikelnummer:

8a) Vervollständigen Sie die Methode `convertToGrayscale` der Klasse `ImageProcessor`. Diese erhält einen RGB-Farbwert in Form einer Instanz der `Pixel`-Klasse übergeben und gibt eine entsprechende Graustufen-Repräsentation der Farbe zurück. Zur Berechenung einer einfachen Graustufe aus einer RGB-Farbe können Sie das ungewichtete Mittel der drei Farbkanäle bilden und mit diesem Wert alle drei Kanäle einer neuen RGB-Farbe befüllen.

> **Beispiel**: Die Originalfarbe wird im RGB-Raum durch die Werte `R = 200`, `G = 30` und `B = 70` repräsentiert. Der Graustufenwert berechnet sich aus `(200+30+70)/3` und entspricht *100*. Die errechnete Graustufe würde im RGB-Raum also durch die Werte `R = 100`, `G = 100` und `B = 100` repräsentiert werden.

8b) Vervollständigen Sie die Methode `getThresholdImage` der Klasse `ImageProcessor`. Die Methode bekommt eine Rastergrafik als zwei-dimensionales Pixel-*Array* übergeben. Zurückgegeben wird ein neues Pixel-*Array* mit identischen Abmessungen zum originalen Bild. Jeder Pixel der übergebenen Grafik wird untersucht. Anhand der Graustufenrepräsentation wird festgestellt, ob ein Schwellwert (Parameter `grayscaleThreshold`) überschritten wird. Ist dies der Fall, wird an der korrespondierenden Stelle im neuen Pixel-*Array* die Farbe Weiß (`R: 255, G: 255, B: 255`) eingetragen. Wird der Schwellwert nicht überschritten, wird die Farbe Schwarz (`R: 0, G: 0, B: 0`) eingetragen. Das zurückgegeben Bild stellt also eine auf zwei Farben reduzierte Repräsentation des Originals dar.

Beispiel für die Transformation eines Originals (links) in eine Threshold-Darstellung (rechts). *Bildquelle: Ștefan Jurcă, "Regensburg - Dom", Creative Commons 2.0*

Name, Studiengang & Matrikelnummer: