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

Alexander Bazo Januar 2018

Allgemeine Hinweise

1. Die Bearbeitungszeit beträgt 60 Minuten. Sie können 60 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) Primitive Datentypen (2 Punkte)

Nennen Sie vier primitive Datentypen, die in Java verwendet werden können.

2) Datenstrukturen (2 Punkte)

Bei welchen der folgenden Begriffe handelt es sich NICHT um eine der aus dem Kurs bekannten Datenstrukturen:

- ☐ ArrayList
- ☐ ArrayHolder
- ☐ HashMap
- ☐ DataChain

3) Stackoverflow (3 Punkte)

Beschreiben Sie, in welcher Situation es beim Ausführen einer JAVA-Anwendung zu einem Stackoverflow-Fehler kommt.

4) Polymorphie (3 Punkte)

Erläutern Sie den Begriff der Polymorphie in Java anhand des Beispiels der Verwendung von Interfaces.

Teil 2: Codeverständnis

5) Codeanalyse (5 Punkte)

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

```
1  public ArrayList<String> doStuff(ArrayList<String> l1, char c1)
       {
2          ArrayList<String> result = new ArrayList<String>();
3          for(String s1: l1) {
4                  char c2 = s1.charAt(0);
5                  if(c2 == c1) {
6                          result.add(s1);
7                  }
8          }
9          return result;
10 }
```


Name, Studiengang & Matrikelnummer:

6) Fehler finden (5 Punkte)

Finden Sie die Konventionsverstöße, Syntaxfehler und Bugs in der folgenden Methode. 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 IdChecker {
2          private int[] validIds;
3
4          public IdChecker(int[] validIds) {
5                  validIds = validIds;
6          }
7
8          public String checkIfIdIsValid(int id) {
9                  for(int i=0; i < validIds.length(); i++) {
10                         int CURRENT_ID = validIds[i];
11                         if(CURRENT_ID === id) {
12                                 return true;
13                         }
14                 }
15                 return false;
16         }
17 }
```


Name, Studiengang & Matrikelnummer:

Teil 3: Code implementieren

7) Minesweeper (15 Punkte)

Der Parameter `map` der Methode `markMinesOnMap` repräsentiert eine Karte des Spiels *Minesweepers*. Jedes Feld enthält entweder den Wert `0` oder den Wert `1`. Eine `1` drückt dabei aus, dass auf dem Feld eine Mine liegt - eine `0` drückt aus, dass auf dem Feld keine Mine liegt. Erweitern Sie die Methode `markMinesOnMap` so, dass ein zweidimensionales `int`-Array zurückgegeben wird, dass in seiner Größe dem übergebenen Parameter entspricht. Für jedes Feld der ursprünglichen Karte enthält das zurückgegebene Array an der entsprechende Position die Anzahl der Minen, die sich auf den acht angrenzenden Feldern befinden. Felder auf denen sich Minen befinden werden mit dem Wert `-1` gekennzeichnet. Eine exemplarische Darstellung der Ein- und Ausgabewerte der Methode können Sie dieser Grafik entnehmen:

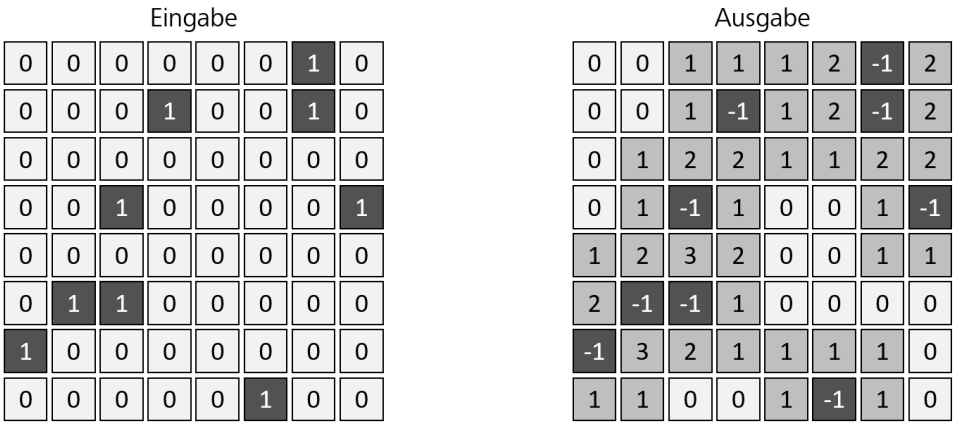

Abbildung 1: Graphische Darstellung der Array-Inhalte für eine exemplarische Ein- und Ausgabesituation beim bzw. nach dem Aufruf der Methode.

```
1 public int[][] markMinesOnMap(int[][] map) {
2         int[][] result = new int[map.length][map[0].length];
3         // Ergänzen Sie hier Ihren Code
4 }
```

8) Radioplaylist (25 Punkte)

In dieser Aufgabe implementieren Sie einige Klassen einer Anwendung, die zur automatischen Generierung von *Playlists* für ein Radioprogramm genutzt werden kann. Die *Playlists* enthalten neben Musiktiteln auch Radiowerbung, die automatisch bei der Generierung der *Playlist* eingefügt wird. Achten Sie beim Erstellen von Subklassen auf Sichtbarkeitsbereiche und Eigenschaften der Superklasse. Fügen Sie alle nötigen Methoden und Instanzvariablen hinzu und implementieren Sie, falls nicht explizit anders angegeben, komplette Klassen. Lagern Sie Teile Ihres Code – falls nötig – in zusätzliche Methoden aus. Gehen Sie davon aus, dass der gegebene Code korrekt ist.

```
public class AudioTrack {
        private String title;
        private File audioFile;
        private double durationInMinutes;
        public AudioTrack(String title, File audioFile, double
           durationInMinutes) {
                this.title = title;
                this.audioFile = audioFile;
                this.durationInMinutes = durationInMinutes;
        }
        public String getTitle() {
                return title;
        }
        public File getFile() {
                return audioFile;
        }
        public double getDurationInMinutes() {
                return durationInMinutes;
        }
}
```

```
public class PlaylistManager {
    private ArrayList<AdTrack> adTracks;
    private ArrayList<AudioTrack> playlist;
    public PlaylistManager(ArrayList<AdTrack> adTracks) {
        this.adTracks = adTracks;
        this.playlist = new ArrayList<AudioTrack>();
    }
    public void addMusicTrack(MusicTrack track) {
        playlist.add(track);
        // Ergänzen Sie hier Ihren Code
    }
}
```


Name, Studiengang & Matrikelnummer:

8a) Erstellen Sie eine Klasse `MusicTrack`, die von `AudioTrack` erbt und zusätzlich über private Instanzvariablen für den Namen des Künstlers sowie das Musikgenre verfügt. Die Eigenschaften werden über den Konstruktor gesetzt und über öffentliche *Getter*-Methoden zurückgegeben.

8b) Erstellen Sie eine Klasse `AdTrack`, die von `AudioTrack` erbt und zusätzlich über private Instanzvariablen für den Namen des jeweiligen Kunden sowie die vereinbarten Werbekosten in Euro pro Minute verfügt. Die Eigenschaften werden über den Konstruktor gesetzt und über öffentliche *Getter*-Methoden zurückgegeben. Ergänzen Sie eine zusätzliche öffentliche Methode, die die Gesamteinnahmen zurückgibt, die erwirtschaftet werden, wenn die Werbung vollständig abgespielt wird.

8c) Erweitern Sie die Methode `addMusicTrack` der vorgegebenen Manager-Klasse: Nach jedem dritten hinzugefügten Titel wird zufällig einer der Werbetracks in die *Playlist* eingefügt. Dabei dürfen sich Werbetracks vom gleichen Kunden nicht wiederholen. Sie können annehmen, dass die Liste der Werbtetracks Inhalte von mindestens zwei unterschiedlichen Kunden enthält. Sie müssen bei der zufälligen Auswahl des Tracks keine besonderen Anforderungen hinsichtlich der Performanz Ihrer Lösung beachten.

> Für die Auswahl des Titels können Sie annehmen, dass der `RandomGenerator` der `GraphicsApp`-Umgebung zur Verfügung steht (Import-Anweisungen sind nicht notwendig). Die Klasse `RandomGenerator` verfügt über eine öffentliche, statische Methode `getInstance`, die eine Instanz des Zufallsgenerators zurück gibt. Die Methode `nextInt(int low, int heigh` Klasse erlaubt die Generierung einer Zufallszahl innerhalb eines gegebenen Wertebereichs (`low` ist inklusive, `heigh` ist exklusive).

8d) Ergänzen Sie eine weitere Methode im Manager. Diese gibt die Gesamteinnahmen zurück, die beim vollständigen Abspielen der *Playlist* bzw. der dort enthaltenen Werbung erwirtschaftet werden würde.

> Mit dem Schlüsselwort `instanceof` prüfen Sie, ob ein bestimmtes Objekt eine Instanz einer bestimmten Klasse ist. Die Anweisung `myObject instanceof MyClass` prüft z.B. ob das Objekt, das in der Variable `myObject` gespeichert ist, eine Instanz der Klasse `MyClass` ist.

Name, Studiengang & Matrikelnummer: