Programmierung III

Wintersemester 2021/22

Prof. Dr.-Ing. Marco Block-Berlitz
Prof. Dr.-Ing. Dietrich Kammer
Dipl.-Ing. Jan Roeper

HOCHSCHULE FÜR
TECHNIK UND WIRTSCHAFT
DRESDEN
UNIVERSITY OF APPLIED SCIENCES

Objektorientiertes Design

E01
Erläutern Sie das Grundkonzept der Objektorientierung.
Was bedeuten dabei die Begriffe: Klasse, Objekt, Vererbung und Interface?

E02
Überlegen Sie sich analog zu `Spieler`, `Trainer` und `Person` eine Klassenstruktur, die `Studenten` (`name`, `vorname`, `wohnort`, `matrikelnummer`, …) und `Professoren` (`name`, …, `gehalt`, `publikationen`, …) modelliert. Erzeugen Sie Exemplare beider Klassen und experimentieren Sie damit!

E03
Das Ihnen bereits bekannte Spiel Conway's Game of Life sollen Sie objektorientiert programmieren. Es gibt dabei die Klassen `Zelle` und `Petrischale`. Finden Sie geeignete Methoden und erläutern Sie Ihr Konzept.

Fußballmanagerspiel

E04

Erweitern Sie je nach Motivation das Fußballmanagerspiel um folgende Eigenschaften:

a) Auswechslungen sind möglich. Spieler können sich verletzen.
b) Während des Spieles sollten Aktionen, wie vergebene oder erfolgreiche Torschüsse Einfluss auf die Motivation der Spieler haben.
c) Führen Sie als Aktionen gelbe und rote Karten ein. Sollte ein Spieler beispielsweise eine gelbe Karte erhalten haben, so sinkt seine Motivation
d) Erweitern Sie den Fußballmanager um einen größeren Kader, bei denen Stamm- und Ersatzspieler verwaltet werden können.
e) Verbessern Sie das Zeitmanagement (z.B. Halbzeiten), in denen der Trainer die Spieler wieder neu motivieren kann.
f) Schreiben Sie den Turniermodus Ligapokal und lassen verschiedene Mannschaften um den Pokal spielen

Fußballmanagerspiel Fortsetzung

E05

Bei der Implementierung der Klasse `Person` sind wir von einem konkreten Alter in Jahren ausgegangen. Nun ließe sich eine Fußballsimulation in einer Ligasimulation aber über mehrere Jahre spielen. Erweitern Sie die Klasse `Person` um die Methode `setGeburtsdatum`. Ändern Sie gegebenenfalls Attribute der Klasse und entsprechend den Aufruf der Methode `getAlter`. Überlegen Sie sich weiterhin, an welche Stelle der aktuelle Tag mit entsprechender Organisation gehört.

Objekte und Referenzen

E06

Betrachten Sie das folgende Code-Fragment:

```
int[][] a = {{2,4,6,8}, {1,2,3,}, {3,4,5}};
int[][] b = a;
int[][] c = (int[][]) a.clone();
c[2]    = a[1];
c[2][1] = 6;
b[2][2] = 7;

for (int i=0; i < a.length; i++)
   a[i][i]++;
```

Welche Werte haben die Ausdrücke:

`a[1] == c[1]`,
`b[2] == c[2]`,
`a == c`,
`b[2][2]`,
`c[1][1]`,
`c[2][2]`,

nach der Ausführung und warum?

Haustiere

E07

Wir haben ein Interface `Haustier` gegeben:

```
public interface Haustier {
   public String getName();
   public int getAlter();
   public String getBezeichnung();
   public String getTierstimme();
}
```

und eine Klasse `Hund`, die folgende Methoden implementiert:

```
public class Hund implements Haustier {
   private String name;
   private int alter;

   public Hund(String name, int alter) {
      this.name = name;
      this.alter = alter;
   }
   public String getName() {
      return name;
   }
   public int getAlter() {
      return alter;
   }
   public String getBezeichnung() {
      return "Hund";
   }
   public String getTierstimme() {
      return "wuff";
   }
}
```

Haustiere (Fortsetzung)

Ein `Haustierhalter` kennt nur das Interface `Haustier`. Ein kleines Testprogramm soll den Zusammenhang beider Klassen zeigen. Machen Sie sich mit den Klassen vertraut und erweitern Sie dieses Projekt um die folgende Funktionalität:

a) Ein Haustierhalter darf mehrere Haustiere halten. Ein neues Haustier soll dabei über die Methode `neuesHaustier(Haustier h)` hinzugefügt werden können.
b) Erweitern Sie die Menge der Haustiere um Ihre drei Lieblingshaustiere.
c) Haustiere leben leider nicht ewig, mal abgesehen von Tamagotchis. Erweitern Sie den Haustierhalter in der Art, dass er auch Haustiere wieder abgeben kann.

```
public class Haustierhalter {
   private Haustier meinHaustier;

   public Haustierhalter() {
      meinHaustier = null;
   }
   public void neuesHaustier(Haustier haustier) {
      meinHaustier = haustier;
   }
   public String getHaustierBezeichnung() {
      return meinHaustier.getBezeichnung();
   }
}
```

```
public class HaustierHaushalt {
   public static void main(String[] args) {
      Haustierhalter heinz = new Haustierhalter();
      Hund rambo = new Hund("Rambo", 3);
      heinz.neuesHaustier(rambo);
      System.out.println("Haustier von Heinz: " +
         heinz.getHaustierBezeichnung());
   }
}
```

Stack und Queue

E08

Arbeiten Sie sich in die Themen **Stack** und **Warteschlange** in Java ein. Sie können beispielsweise folgende Literatur studieren.

Versuchen Sie nach dieser Anleitung einen Stack und eine Warteschlange als Array zu realisieren (kleiner Hinweis: bei der Implementierung der Warteschlange sollten Sie ein zyklisches Array simulieren). Sie können dabei voraussetzen, dass die Einträge in die Datenstrukturen vom Datentyp `int` sind.