technische universität
dortmund

Fakultät für Informatik
Lehrstuhl 14
Lars Hildebrand, Marcel Preuß,
Iman Kamehkhosh, Marc Bury, Diana Howey

Praktikum zu

Einführung in die Informatik für LogWIIngs und WiMas

Wintersemester 2015/16

Übungsblatt 10

Besprechung:
18.–22.01.16
(KW 3)

Vorbereitende Aufgaben

Diese Aufgaben dienen der Wiederholung des bisher gelernten Stoffes über die Programmstruktur von Java.

Aufgabe 10.1: Wiederholung: Klammern

Geben Sie an, wofür folgende Klammern in Java verwendet werden.

- [...]

- (...)

- { ... }

Aufgabe 10.2: Wiederholung: Funktionsparameter

Geben Sie einen geeigneten Methodenkopf für die folgenden öffentlichen, statischen Funktionen an.

- Eine Funktion **average**, die den Durchschnitt eines **double**-Arrays berechnet.

  Beispielantwort: `public static double average(double[] array)`

- Eine Funktion **plus**, die zwei rationale Zahlen miteinander addiert und die Summe zurückgibt.

- Eine Funktion **countWords**, die die Wörter in einem **String** zählt und zurückgibt.

- Eine Funktion **printMaximum**, die das Maximum eines **int**-Arrays mit `System.out.println` auf dem Bildschirm ausgibt.

- Eine Funktion **times**, die einen Integer **n** und einen Integer **x**, entgegen nimmt und ein **n** Elemente langes Array, gefüllt mit dem Wert **x** zurückgibt.

Präsenzaufgaben

Aufgabe 10.3: Bücher und (echte) Bibliotheken

In dieser Aufgabe sollen Sie objektorientiert modellieren.

In einer Stadt gibt es mehrere Bibliotheken, aufgeteilt in viele Zweigstellen, bei denen Kunden Bücher ausleihen und zurückgeben können.

a) Schreiben Sie eine Klasse **Book** zur Repräsentation von Büchern. Ein Buch hat als privates Attribut einen Titel (**title**, eine Zeichenkette). Die Klasse soll einen Konstruktor beinhalten sowie eine Methode **getTitle**, die den Titel zurückgibt.

b) Schreiben Sie eine Klasse **Customer**. Ein Kunde hat als Attribute einen Namen (**name**, eine Zeichenkette, private) und einen Zähler, wie viele Bücher er derzeit ausgeliehen hat (**rentedBooks**, ein **int**, public).

   Diese Klasse **Customer** soll enthalten:

   - einen Konstruktor, der den Namen setzt und **rentedBooks** auf 0 setzt
   - eine Methode **getName**, die den Namen zurückgibt
   - eine Methode **read**, die anzeigt, dass der Kunde ein Buch liest, indem sie eine Ausgabe macht (z. B. den Titel des Buches).

c) Schreiben Sie eine Klasse **Library**. Sie soll zwei Methoden beinhalten:

   - eine Methode **rentBook**, die als Parameter ein **Customer**-Objekt und einen **String** als Buchtitel entgegen nimmt. Sie soll modellieren, dass ein **Customer** ein Buch ausleiht und ein **Book**-Objekt erzeugen und zurückgeben.

     Dabei soll der Zähler für die ausgeliehenen Bücher vom **Customer** um eins erhöht werden.

   - eine Methode **returnBook**, deren Parameter ein **Customer**-Objekt und ein **Book**-Objekt sind. Sie repräsentiert das Zurückgeben des Buches. Hier soll der Zähler wieder um eins erniedrigt werden.

   - Überlegen Sie sich darüber hinaus eine geeignete Ausgabe in den Methoden **rentBook** und **returnBook**.

Überprüfen Sie ihre Implementierung z. B. durch folgende Testklasse:

```
1  package blatt10;
2
3  public class City {
4      public static void main(String[] args) {
5          Library firstLibrary = new Library();
6          Library secondLibrary = new Library();
7          Customer joe = new Customer("Joe");
8          Customer peter = new Customer("Peter");
9
10         Book bible = firstLibrary.rentBook(joe, "The Bible");
11         Book lotr = secondLibrary.rentBook(peter, "Lord of the Rings");
12
13         peter.read(lotr);
14         joe.read(bible);
15         // es ist egal, wo man sein Buch zurueckbringt
16         secondLibrary.returnBook(joe, lotr);
17         secondLibrary.returnBook(peter, bible);
```

```
18      }
19  }
```

Die Ausgabe könnte dann folgendermaßen aussehen:

```
Joe rented The Bible
Joe rents 1 Books
Peter rented Lord of the Rings
Peter rents 1 Books
Peter reads: Lord of the Rings
Joe reads: The Bible
Joe returns Lord of the Rings
Joe rents 0 Books
Peter returns The Bible
Peter rents 0 Books
```

Aufgabe 10.4: Sinnvolle Werte

Es wurde festgestellt: Die Kunden leihen zu viele Bücher aus und geben zu selten welche zurück. Deswegen darf jeder Kunde ab jetzt nur noch maximal 3 Bücher ausleihen.

- Passen Sie Ihre Implementierung so an, dass die Methode **rentBook** bei dem Versuch, ein weiteres Buch auszuleihen, **null** zurückgibt, wenn versucht wird, mehr als drei Bücher auszuleihen.
- Die Methoden **read** und **returnBook** müssen ab dann auch damit umgehen können, wenn sie **null** übergeben bekommen.
- Sorgen Sie des Weiteren dafür, dass die Anzahl der ausgeliehenen Bücher für einen Kunden nicht negativ werden kann.

Aufgabe 10.5: Klassenattribute

Die Bibliotheken wollen nun erfassen, wie viele Bücher ausgeliehen sind.

- Erweitern Sie die Klasse **Library** um eine Instanzvariable **rentedBooks** und eine Klassenvariable **overallRentedBooks**. Diese sollen repräsentieren, wie viele Bücher durch die einzelne Bibliothek ausgeliehen wurden und wie viele Bücher *insgesamt* ausgeliehen wurden.
- Schreiben Sie in der Klasse **Library** eine Methode **printStatistics**, die diese beiden Zähler ausgibt.

Ergänzende Aufgaben

Aufgabe 10.6: Vertrauen ist gut, Kontrolle ist besser

In dieser Aufgabe sollen Sie die übergebenen Werte von Konstruktoren und sog. Settern kontrollieren. **Setter** sind Methoden, die private Attribute auf einen Wert setzen.

Erweitern Sie Ihre geometrischen Klassen aus Übungsblatt 9 um folgende Funktionalitäten:

Wenn den Konstruktoren einer solchen Klasse ein Wert **kleiner** oder **gleich** 0 übergeben wird, so soll eine Meldung über den fehlerhaft aufgerufenen Konstruktor ausgegeben werden und anschließend das entsprechende Attribut auf den Wert 1 gesetzt werden.

Um die Objekte nach der Konstruktion noch ändern zu können sollen Sie für alle Attribute der Klassen einen **Setter** schreiben, z.B. `public void setRadius(double newRadius)` .

Kontrollieren Sie in diesen **Settern** auch, ob die übergebenen Werte auf ihre Gültigkeit. Falls ein ungültiger Wert übergeben wurde, ändern Sie den Wert stattdessen nicht und geben Sie eine Fehlermeldung aus.