technische universität
dortmund

Bearbeitet von:
Sebastian Hauer
Sergej Karsten

Fakultät für Informatik
Lehrstuhl 1
Dr. Lars Hildebrand
Iman Kamehkhosh, Marcel Preuß, Henning Timm,
Alexander Lochmann, Bartho Rudak, Thomas Schmitz

Praktikum zu

Einführung in die Informatik für LogWings und WiMas

Wintersemester 2014/15

Übungsblatt 10

Bearbeitungszeit:
12.01.15 -
16.01.15

Aufgabe 10.1 – Objektorientierung

Bearbeitungszeit: 15 Minuten

In den bisherigen Übungsblättern haben wir Klassen genutzt, um Programme und Unterprogramme zu definieren und diese aus dem statischen Kontext der **main**-Methode aufgerufen. Wofür dienen Klassen in der objektorientierten Programmierung?

Welcher Teil einer Klassendefinition wird verwendet, um ein Objekt zu instantiieren?

Wie sieht die Syntax für diese Deklaration dieser Teile aus? Gehen Sie z.B. von einer Klasse mit dem Namen `Student` aus.

Wie erzeugen Sie aus der Klasse Student nun ein neues Objekt mit dem Namen „meinStudent“?

Wie können Sie Methoden auf dem Objekt „meinStudent“ aufrufen? Zum Vergleich: Statische Methoden einer Klasse haben wir mit **Klassenname.methode()** aufgerufen. Gehen Sie z.B. von einer Methode **public int getMatrikelNr(){...}** in der Klasse Student aus.

Wie können Sie innerhalb einer Objektmethode speziell auf Werte und Methoden des Objektes verweisen? — Um z.B. durch einen Block überdeckte Attributen trotzdem verwenden zu können.

Aufgabe 10.2 – Referenzen und Nullpointer

Bearbeitungszeit: 25 Minuten

Wenn Sie eine nicht-primitive Variable deklarieren, so wird in dieser nur eine Referenz auf ein Objekt des entsprechenden Typs vermerkt. Wenn Sie einer solchen Variablen keine Referenz zuweisen, so hat die Variable den besonderen Wert **null**. Jede Variable eines nicht-primitiven Datentyps kann diesen Wert annehmen und steht für „Hier ist nichts!“. Sie können auch jede nicht-primitive Variable auf diesen Wert setzen, um die Referenz lokal zu *löschen*. Existieren keine Referenzen mehr auf ein Objekt, so wird dieses auch aus dem Speicher vom sog. „Garbage Collector“ entfernt.

Implementieren Sie die **main**-Methode einer Klasse `ReferenceTest` wie folgt:

- Deklarieren Sie zwei Variablen `a` und `b` vom Typ `String`.
- Initialisieren Sie beide Strings mit dem Ausdruck `null`.
- Geben Sie das Ergebnis des Ausdrucks `a == b` aus.
- Weisen Sie `a` einen neuen String mit der Anweisung `new String(„Hallo“)` zu.
- Geben Sie das Ergebnis des Ausdrucks `a == b` aus.
- Weisen Sie `b` einen neuen String mit der Anweisung `new String(„Hallo“)` zu.
- Geben Sie das Ergebnis des Ausdrucks `a == b` aus.
- Geben Sie das Ergebnis des Ausdrucks `a.equals(b)` aus.
- Weisen Sie `b` einen neuen String mit der Anweisung `new String(„World“)` zu.
- Geben Sie das Ergebnis des Ausdrucks `b.equals(a)` aus.
- Geben Sie das Ergebnis des Ausdrucks `a.equals(b)` aus. (Symmetrie)
- Weisen Sie der Variablen `b` den Wert von `a` zu.
- Geben Sie das Ergebnis des Ausdrucks `a == b` aus.
- Geben Sie das Ergebnis des Ausdrucks `a.equals(b)` aus.
- Weisen Sie `a` den Wert **null** zu.
- Geben Sie das Ergebnis des Ausdrucks `b.equals(a)` aus.
- Geben Sie das Ergebnis des Ausdrucks `b == a` aus.
- Geben Sie das Ergebnis des Ausdrucks `a == b` aus.
- Geben Sie das Ergebnis des Ausdrucks `a.equals(b)` aus.

Worin liegt der Unterschied zwischen `equals` und `==`?

Was geschieht in der letzten Zeile und wie ist es sich zu erklären?

Aufgabe 10.3 – Eigene Objekte: Die Kiste - Attribute

Im Folgenden wollen wir einfache Kisten modellieren. Eine Kiste ist geometrisch ein Quader und besitzt Höhe, Breite und Länge. Schreiben Sie eine Klasse `Box`, welche diese drei Attribute in Form von privaten `double`-Attributen vermerkt.

Aufgabe 10.4 – Eigene Objekte: Die Kiste - Konstruktor

Schreiben Sie einen Konstruktor für Ihre Klasse `Box`, welcher die drei Größen einer Kiste entgegennimmt und die entsprechenden Attribute der Kiste setzt.

Aufgabe 10.5 – Eigene Objekte: Die Kiste - Methoden

Die Attribute der Kiste können Sie zwar innerhalb der Methoden der Kiste erreichen und abfragen, jedoch nicht von außerhalb der Klasse. Woran liegt das?

Schreiben Sie Methoden, welche die 3 Attribute einer Kiste nach außen Sichtbar und Veränderbar machen: getter und setter.

- **getWidth()**, **setWidth(double width)**
- **getHeight()**, **setHeight(double height)**
- **getLength()**, **setLength(double length)**

Schreiben Sie zwei Methoden, **getSurfaceArea()** und **getVolume()**, welche die Oberfläche und Volumen einer Kiste berechnen und zurückgeben.

Schreiben Sie eine Methoden, **isCube()**, welche überprüft, ob die Kiste ein Würfel ist.

Aufgabe 10.6 – Eigene Objekte: Die Kiste - Testen

Schreiben Sie eine Klasse `BoxTest`, welche eine **main**-Methode besitzt und ihre Kistenklasse testen soll. Testen Sie ihre Kisten, indem Sie Objekte vom Typ `Box` instantiieren und alle von Ihnen implementierten Methoden auf ihnen aufrufen.

Gehen Sie zum Beispiel wie folgt vor:

- Erstellen Sie eine neue Kiste der Maße 3x4x5 und geben Sie die Länge, Breite und Höhe der Kiste auf der Konsole aus.
- Geben Sie die Oberfläche und das Volumen auf der Konsole aus.
- Ändern Sie die Attribute der Kiste mit ihren settern.
- Rufen Sie erneut die Methoden zum Berechnen der Oberfläche und des Volumens auf und geben Sie die Werte aus.
- Erstellen Sie eine neue, quadratische Kiste und überprüfen Sie ihre alte und die neue Kiste mit der Methode **isCube()** und geben Sie das Ergebnis auf der Konsole aus.
- Überprüfen Sie, ob Ihre Ausgabe mit den erwarteten Werten übereinstimmt.

Funktioniert alles, so wie Sie erwarten?

Ergänzende Aufgaben

Aufgabe 10.7 – Eigene Objekte: Die Kiste - Vergleichen

Schreiben Sie eine Methode **public int compare(Box other)** in Ihrer Klasse `Box`, welche eine andere Box als Parameter entgegennimmt und die Kiste, auf der **compare()** aufgerufen wird, mit der übergebenen Kiste vergleicht.

Die Methode soll einen negativen Wert zurückgeben, wenn die übergebene Kiste größer ist, einen positiven Wert zurückgeben, wenn die übergebene Kiste kleiner ist und 0, falls die übergebene Kiste gleich groß ist. Das Maß für die Größe einer Kiste ist das Volumen.

Testen Sie ihre Methode mit einer kleineren, größeren und gleich großen Kiste mit unterschiedlichen Maßen.