Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik III
Prof. Dr. A.B. Cremers

Vorlesung Objektorientierte Softwareentwicklung
Sommersemester 2010
Dr. Günter Kniesel

Aufgabenblatt 5

— Veröffentlicht am 01.06.2010 —

Abgabe bis 08.06.2010, 13:00 per SVN

Aufgabe 1 *(Interfaces, Überladen, Objektidentität - 2 Punkte)*

Überlegen Sie sich für jede Aussage ein Argument, mit dem Sie eindeutig bestimmen können, ob die Aussage wahr oder falsch ist. Für jede richtig angekreuzte Antwort werden 0,5 Punkte angerechnet, für jede falsch angekreuzte Antwort 0,5 Punkte abgezogen. Nicht angekreuzte Antworten geben keinen Punktabzug.

wahr	falsch	
		Ein Interface kann nur öffentliche Methoden definieren.
		„Objekt“ und „Klasse“ sind zwei verschiedene Bezeichnungen für das selbe Architekturelement.
		~~Eine Java-Klasse kann von mehreren abstrakten Klassen gleichzeitig erben.~~
		Private Felder eines Java-Objektes können nur von Methoden desselben Objektes gelesen oder geschrieben werden.

Aufgabe 2 *(Vererbung und Objektidentität - 2 Punkte)*

Überlegen Sie sich für jede Aussage ein Argument, mit dem Sie eindeutig bestimmen können, ob die Aussage wahr oder falsch ist. Für jede richtig angekreuzte Antwort werden 0,5 Punkte angerechnet, für jede falsch angekreuzte Antwort 0,5 Punkte abgezogen. Nicht angekreuzte Antworten geben keinen Punktabzug.

wahr	falsch	
		In Java gilt: Objekte der Unterklassen können stets an die Stelle von Objekten der Oberklasse treten.
		Man kann in Java verhindern, dass Methoden oder Felder einer Klasse in Unterklassen überschrieben werden.
		Wenn `o1` und `o2` beides Java-Objekte sind, dann liefern die folgenden Ausdrücke immer das gleiche Ergebnis: `o1 == o2` und `o1.equals(o2)`
		~~Wenn für zwei beliebige Java-Objekte `o1` und `o2` gilt: `o1 == o2`, dann gilt auch immer `o1.equals(o2)`~~

Aufgabe 3 *(Bindungen - 5 Punkte)*

Betrachten Sie folgendes Codefragment:

```
interface Food {
  public String getMeal();
}
class Cauliflower implements Food {
  public String getMeal() {
    return "Blumenkohl an Gorgonzolasoße";
  }
}
class Spaghetti implements Food {
  public String getMeal() {
    return "Spaghetti Bologneser Art";
  }
}
```

Welche Ausgabe erzeugt das folgende Programm?

```
1:  Food food;
2:  food  = new Cauliflower();
3:  System.out.println(food.getMeal());
4:  food  = new Spaghetti();
5:  System.out.println(food.getMeal());
```

Warum unterscheiden sich die Ausgaben von Zeile 3 und Zeile 5, obwohl beide Zeilen identisch sind? Benennen und erklären Sie das dahinter stehende Prinzip. Programmieren Sie selbst eine Klasse, die das Interface `Food` implementiert und demonstrieren Sie das Prinzip.

Aufgabe 4 *(Überladen von Methoden - 5 Punkte)*

Zu den Klassen aus Aufgabe 3 wird folgende Klasse hinzugefügt und ein Objekt der Klasse in der Variable `student` gespeichert:

```
class Student {
  public void eat(Food food) {
    System.out.println("I like " + food.getMeal());
  }
  public void eat(Cauliflower Cauliflower) {
    System.out.println("I don't like " + Cauliflower.getMeal());
  }
}
```

a) Welche Ausgabe erzeugt der folgende Aufruf? Warum?

```
Cauliflower essen1 = new Cauliflower();
student.eat(essen1);
```

b) Welche Ausgabe erzeugt der folgende Aufruf? Warum?

```
Spaghetti essen2 = new Spaghetti();
student.eat(essen2);
```

c) Welche Ausgabe erzeugt der folgende Aufruf? Warum?

```
Food essen = new Cauliflower();
student.eat(essen);
```

~~**Aufgabe 5** *(Overloading - Overriding - 4 Punkte)*~~

~~Erklären Sie den Unterschied zwischen *Überladen (Overloading)* und *Überschreiben (Overriding)*. Welches Verfahren führt zur Auswahl der richtigen Methode?~~

Aufgabe 6 *(Weltobjekte - Teil 2 - 7 Punkte)*

a) Betrachten Sie die Weltobjekte *Kirschbaum*, *Apfelbaum* und *Feigenbaum*. Benennen Sie in Java-Code ein gemeinsames *Interface* und beschreiben Sie zwei Methoden, die dieses Interface anbietet.

b) Wählen Sie eine Methode aus Aufgabenteil a) und definieren Sie zwei weitere Methoden, die Ihre Standardimplementierung überladen.

~~**c**) Ersetzen Sie nun das Interface aus Aufgabenteil a) durch eine *abstrakte Klasse*. Welche Ihrer Methoden können Sie sinnvollerweise bereits in der abstrakten Klasse implementieren, welche nicht? Warum?~~

d) Wählen Sie eine der Methoden, für die Sie in Aufgabenteil c) eine Implementierung in der gemeinsamen Oberklasse vorgeschlagen haben. Argumentieren Sie, in welchem Fall ein *Überschreiben* der Methode in einer Unterklasse sinnvoll ist und benennen Sie auch die entsprechende Unterklasse.