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

Aufgabenblatt 7

— Veröffentlicht am 15.06.2010 —

Abgabe bis 22.06.2010, 13:00 per SVN

Aufgabe 1 *(Vererbung - 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	
		Unterklassen können öffentliche Methoden einer Oberklasse überschreiben, indem in der Unterklasse Methoden mit der gleichen Signatur und leerem Methodenrumpf definiert werden.
		Konstruktoren werden automatisch von der Oberklasse geerbt.
		Alle Klassen in Java haben eine gemeinsame Oberklasse.
		Alle Interfaces in Java haben eine gemeinsame Oberklasse.

Aufgabe 2 *(Begriffsdschungel - 7 Punkte)*

Schreiben Sie die nach Ihrem Verständnis jeweils synonymen Begriffe in je eine Zeile und überlegen Sie sich, wie Sie den Tutoren gegenüber mündlich verwandte aber nicht wirklich identische Bedeutungen von einander abgrenzen würden. Greifen Sie bei Bedarf auf Sekundärliteratur zurück:

statische Variable	Vererbung	Methode	abgeleitete Klasse
Paket	Objekt	Klassenmethode	Subklasse
Feld	Superklasse	Instanzvariable	Ableitung
Operation	Datenelement	Prozedur	Eigenschaft
Instanz	Routine	Member	Klassenvariable
Namensraum	Argument	Oberklasse	Attribut
Unterklasse	statische Methode	Basisklasse	Exemplar

Aufgabe 3 *(Interfaces - 3 Punkte)*

Betrachten Sie folgendes Programm:

```
public interface I { }

public class testI {
  public void test() { I i;  i.toString(); }
}
```

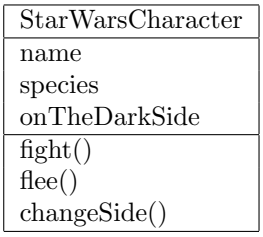
a) Überlegen Sie, ob der Aufruf von `toString()` in obigem Code akzeptiert wird oder nicht? Wird es eine Fehlermeldung geben (außer der Warnung, dass der Aufruf von `toString()` auf einer nicht initialisierten Variablen stattfindet)? Begründen Sie Ihre Antwort.

b) Probieren Sie aus, was der Java-Compiler wirklich tut. Wie erklären Sie sich das Verhalten?

Aufgabe 4 *(Wiederholung - 10 Punkte)*

In dieser Aufgabe geht es darum, verschiedene Konzepte der Objektorientierung, die sowohl in der Vorlesung als auch in der Übung bereits behandelt wurden, zu wiederholen.

a) Auf der Folie *2-13* des Foliensatzes *oose-02-oop-basics.pdf* wird die UML-Notation einer einfachen Java-Klasse dargestellt. Implementieren Sie eine Klasse `StarWarsCharacter` gemäß dem folgenden UML-Diagramm:



Die Variable `onTheDarkSide` soll dabei vom Typ `boolean` sein, `name` und `species` sollen durch `Strings` repräsentiert werden. Der Aufruf der Funktion `changeSide()` soll den internen Zustand `onTheDarkSide` ändern. Beim Aufruf von `fight()` bzw. `flee()` soll eine Ausgabe gemacht werden, die neben der Information *fleeing* oder *fighting* auch Name und Rasse der Instanz enthält.

b) Leiten Sie von der Klasse `StarWarsCharacter` die Klasse `Jedi` ab. Diese soll als zusätzliche `String`-Variable das Feld `rank`[1] besitzen, als zusätzliche Operationen die Methoden `increaseRank()` und `decreaseRank()`, die den internen Zustand `rank` entsprechend modifizieren. Erweitern Sie die Funktionen `fight()` und `flee()` dahingehend, dass auch der Rang eines Jedis ausgegeben wird (Die Implementierung der Oberklasse soll weiterhin aufgerufen werden!).

c) Da Flucht für einen Jedi, der mindestens den Rang *Knight* trägt, ausgeschlossen ist, muss die Methode `flee()` in der Klasse `Jedi` überschrieben werden: Während Jedis der beiden niedrigsten Ränge beim Aufruf von `flee()` wie gehabt die Flucht ergreifen sollen (greifen Sie hierzu auf die Implementierung der Oberklasse zurück), soll fortgeschritteneren Jedis die Fluchtmöglichkeit verwehrt bleiben: Hier soll stattdessen die Funktion `fight()` aufgerufen werden.

d) Erstellen Sie in den beiden Klassen `StarWarsCharacter` und `Jedi` Standardkonstruktoren, die alle Variablen nach Ihren Wünschen vorinitialisieren[2]. Erstellen Sie in der Klasse `Jedi` zusätzlich einen Konstruktor, der es erlaubt, den Rang zu setzen. Testen Sie Ihre Implementierung, indem Sie eine neue Klasse `Demo` erstellen, in deren `main`-Funktion Sie zunächst ein Jedi-Objekt mit dem Rang *Younglin* erschaffen, einmal fliehen, den Rang zweimal erhöhen und anschließend einen weiteren Fluchtversuch unternehmen!

[1] Wie Sie sicherlich wissen, existieren die Jedi-Ränge *Younglin*, *Padawan*, *Knight*, *Master*, *Council Member* und *Grand Master*.

[2] Hier finden Sie Anregungen für die Wahl der `species`: `http://en.wikipedia.org/wiki/List_of_Star_Wars_races`