Objektorientierte Programmierung WS 2013/2014 - Aufgabenblatt 2

Prof. Dr. U. Kastens
Institut für Informatik, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn

Ausgabe: 29.10.2013

Aufgabe 1 (Begriffsklärung)

Machen Sie sich den Unterschied zwischen Typen und Klassen sowie zwischen Subtyping und Subclassing nach Kim B. Bruce noch einmal klar. Entwerfen Sie in Ihrer Gruppe eine Mini-Präsentation (auf einem Din-A4-Blatt), die den Sachverhalt und die Zusammenhänge anschaulich macht.

Aufgabe 2 (Problematische Vererbung)

In einigen objektorientierten Sprachen ist es möglich, einer Unterklasse nur einen Teil der Methoden der Oberklasse zu vererben. Zum Beispiel gibt es in Eiffel dazu das `undefine`-Konstrukt:

```
class A
inherit B
   undefine m
feature -- hier kommen Attribute und Methoden von A
end
```

a) Übernehmen Sie das `undefine`-Konstrukt in die Java-Syntax und geben Sie ein Beispiel an, das die Problematik von `undefine` zeigt.
b) Eiffel fordert, dass eine Klasse, die mindestens eine Methode mit `undefine` kennzeichnet, als `deferred` (entspricht dem `abstract` in Java) definiert wird. Löst dies das Problem?

Aufgabe 3 (Kovarianz und Kontravarianz)

In der Vorlesung haben Sie die Begriffe Kovarianz und Kontravarianz kennen gelernt.

a) Erklären Sie die Begriffe.
b) In welcher Beziehung müssen die Klassen A und B sowie die Klassen C und D stehen, damit im folgenden Quelltext Subtyping vorliegt:

```
class Vater {
  function tuewas(A param) : C is
  {
    ...
  }
}

class Sohn inherits Vater modifies tuewas {
  function tuewas(B param) : D is
  {
    ...
  }
}
```

Aufgabe 4 (Java-Programm modifizieren)

Auf dem letzten Übungszettel haben Sie die Methoden und Attribute, die mit der Bewegung eines Balls zu tun haben, aus der Klasse `Ball` in eine Unterklasse `MovableBall` verschoben. Schreiben Sie nun eine Klasse `BoundedBall` als Unterklasse von `MovableBall`. Ein `BoundedBall` existiert in einem bestimmten Rechteck und ändert beim Anstoßen an den Rand seine Richtung. Verwenden Sie `BoundedBall` in einer modifizierten `BallWorld`, die nun nicht mehr prüfen muss, ob der Ball am Rande des Rahmens anstößt.

Hier sind die zum Blatt 1 erarbeiteten Lösungen:

- `Ball.java`
- `MovableBall.java`
- `BallWorld.java`

Aufgabe 5 (MultiballWorld (ggf. als Hausaufgabe))

Entwickeln Sie das Programm `BallWorld` weiter zur "MultiBallWorld": Es gibt 5 Bälle mit verschiedenen Farben und zufälligen Anfangspositionen. Berühren sich während des Spiels zwei Bälle, so fliegt jeder zurück in die entgegengesetzte Richtung.

Tipp: Das Berühren können Sie mit der Methode `intersects` der Klasse `Rectangle` prüfen.