Technische Universität Darmstadt
Fachbereich Informatik
Prof. Dr. Andreas Koch

Allgemeine Informatik 1 im WS 2007/08

Übungsblatt 6

Bearbeitungszeit: 03.12. bis 09.12.2007

Aufgabe 1: `while`-Schleife

Was ist – unter welchen Bedingungen – der Effekt der folgenden Anweisung?

```
while (!facingNorth()) {
  turnLeft();
}
```

Aufgabe 2: WallChecker

Schreiben Sie eine Roboterklasse **`WallChecker`** mit folgenden Methoden:

```
boolean westIsBlocked() {...}
boolean westIsBlockedAndBeeper() {...}
boolean westIsBlockedOrBeeper() {...}
```

`westIsBlocked` soll genau dann **`true`** zurückliefern, wenn sich genau einen halben Block westlich des Roboters eine Mauer befindet. Stellen Sie dabei sicher, dass der Roboter nach der Ausführung der Methode exakt an der selben Stelle wie vorher steht und dass er in die selbe Richtung blickt.

`westIsBlockedAndBeeper` soll genau dann **`true`** zurückliefern, wenn sich westlich des Roboters direkt eine Mauer befindet **und** sich auf dem aktuellen Feld mindestens ein Beeper befindet. Benutzen Sie die Methode **`westIsBlocked`**!

`westIsBlockedOrBeeper` soll genau dann **`true`** zurückliefern, wenn sich westlich des Roboters direkt eine Mauer befindet **oder** sich auf dem aktuellen Feld mindestens ein Beeper befindet. Benutzen Sie die Methode **`westIsBlocked`**!

Aufgabe 3: Polymorphie

Laden Sie sich, falls noch nicht geschehen, von der Webseite auch die Datei **uebung06-3.task** herunter. Darin sind die beiden folgenden Roboterklassen gegeben:

```
class RightTurner extends Robot {
  void turnRight() {
    turnLeft();
    turnLeft();
    turnLeft();
  }
}
```

```
class Commander extends Robot {
  void command(RightTurner robot) {
    robot.move();
    robot.turnRight();
    robot.move();
    robot.turnLeft();
    robot.move();
    robot.move();
  }
}
```

Erweitern Sie nun die vorgegebene Datei um folgende weitere Roboterklassen, die alle von **`RightTurner`** abgeleitet sein sollen (**`extends`**):

- **`FunnyTurner`**: Diese Roboterklasse bewegt sich um einen Schritt nach vorne bevor sie eine Linksdrehung ausführt. Dies wird anstelle der normalen Linksdrehung, die mit **`turnLeft`** aufgerufen wird, gemacht. Es ist also die Methode **`turnLeft`** neu zu definieren.
- **`Backwards`**: Diese Roboterklasse soll sich einen Schritt nach hinten statt nach vorne bewegen, wenn man die Methode **`move`** aufruft. Nach dem Schritt soll sie aber wieder in die gleiche Richtung schauen wie vor dem Aufruf von **`move`**. Es ist also die Methode **`move`** neu zu definieren.
- **`Runner`**: Diese Roboterklasse soll sich nicht nur einen, sondern zwei Schritte nach vorne bewegen, wenn man **`move`** aufruft. Es ist also auch hier die Methode **`move`** neu zu definieren.

Führen Sie nun folgendes Programm (in der Datei **uebung06-3.task** schon vorhanden) aus und beobachten Sie, welche Bewegungen die einzelnen Roboter machen. Beobachten Sie insbesondere die Rechtsdrehungen von **`funny`**. Obwohl nur die Methode **`turnLeft`** für **`FunnyTurner`** überschrieben wurde, hat sich auch das Verhalten von **`turnRight`** verändert. Warum ist das so?

```
task {
  RightTurner turner = new RightTurner(4, 4, 0, East);
  FunnyTurner funny = new FunnyTurner(4, 4, 0, East);
  Backwards backwards = new Backwards(4, 4, 0, East);
  Runner runner = new Runner(4, 4, 0, East);

  Commander commander = new Commander(1, 1, 0, East);

  commander.command(turner);
  commander.command(funny);
  commander.command(backwards);
  commander.command(runner);
}
```