Funktionale und objektorientierte Programmierkonzepte

Alexander Mainka und Lukas Röhrig
(Gesamtleitung: Prof. Karsten Weihe)

Wintersemester 19/20
v1.0

Übungsblatt 1

Themen:	FopBot
Relevante Folien:	01a bis 01c
Abgabe der Hausübung:	01.11.2019 bis 23:55 Uhr

V Vorbereitende Übungen

V1 FopBot

☆ ☆ ☆

Beschreiben Sie kurz in ihren eigenen Worten worum es sich bei FopBot handelt, wie die Welt aufgebaut ist und welche Grundfunktionen jeder Roboter beherrscht.

> **Für alle Aufgaben auf diesem und allen weiteren Übungsblättern**: In der Abschlussklausur werden Sie keine Hilfsmittel zur Verfügung haben. Üben Sie also schon zu Beginn auch ohne Entwicklungsumgebung und nur mit Stift auf einem Blatt Papier zu programmieren. Abschließend können Sie dann ihre vollständige Lösung in die Entwicklungsumgebung übertragen und überprüfen.

V2 Liegen geblieben

☆ ☆ ☆

Betrachten Sie den folgenden Codeausschnitt/führen Sie ihn selbst einmal aus:

```
1 Robot bot = new Robot(0,2,UP,0);
2 bot.move();
3 bot.turnLeft();
4 bot.putCoin();
```

In welcher Zeile kommt es zu einem Problem und wieso?

V3 Rechteck

★ ☆ ☆

Schreiben Sie ein Programm, welches zwei Roboter `putbot` und `pickbot` erstellt. Dabei soll `putbot` mit Coins ein Rechteck der Höhe 5 und der Breite 3 zeichnen. Es sollen nur die Seiten des Rechtecks gezeichnet werden, die restlichen innenliegenden Felder des Rechtecks bleiben unberührt. Nachdem das Rechteck gezeichnet wurde, soll `pickbot` alle Coins wieder einsammeln. Überlegen Sie sich, wie Sie das Programm mit nur einer Schleife pro Roboter gestalten können.

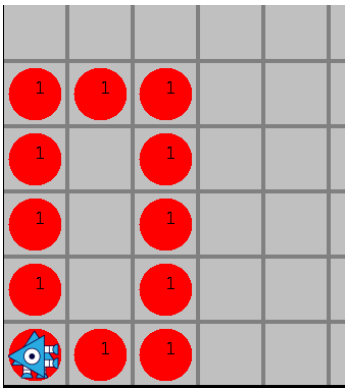

Abbildung 1: Fertiggestelltes Rechteck durch putbot

V4 Bedingungen I

★ ☆ ☆

Betrachten Sie folgenden Codeausschnitt:

```
1 Robot bot1 = new Robot(3,1,UP,1);
2 bot1.move();
3 if(bot1.isNextToACoin()){
4  bot1.pickCoin();
5 }
6 else{
7  bot1.putCoin();
8 }
```

Beschreiben Sie in eigenen Worten, welchem Zweck dieser Codeausschnitt dient. Erweitern Sie außerdem den Code so, dass `bot1` nur einen Coin ablegt, wenn er auch mindestens einen besitzt.

V5 Variablen

★ ☆ ☆

Legen Sie eine Variable `int a` an und setzen Sie ihren Wert auf 127. Jetzt legen Sie eine weitere Variable `int b` an und setzen Ihren Wert auf 42. Was gibt nun der Ausdruck `int c = a % b` wieder? Beschreiben Sie in Ihren eigenen Worten, welche Berechnung mit dem `%` Operator durchgeführt wird.

V6 Bedingungen II

★ ☆ ☆

Ihr Kommilitone ist etwas tippfaul und lässt deswegen gerne einmal Klammern weg, um sich Arbeit zu sparen. Er hat in seinem Code eine Variable `int number` angelegt, in der er eine Zahl speichert. Ist diese Zahl kleiner als 0, so möchte er das Vorzeichen der Zahl umdrehen und sie anschließend um 1 erhöhen. Ist die Zahl hingegen größer als 0, so möchte er die Zahl verdoppeln. Dazu schreibt er folgenden Code:

```
1 if(number < 0) number = -number;
2 number = number + 1;
3 else number = number * 2;
```

Kann der Code so ausgeführt werden? Beschreiben Sie den Fehler, den ihr Kommilitone begangen hat.

Nachdem Sie ihren Kommilitonen auf den obigen Fehler hingewiesen haben, überarbeitet er seinen Versuch. Wie sieht es mit folgender Variante aus?

```
1 if(counter > 0) counter = counter * 2;
2 if(counter < 0) counter = -counter;
3 counter = counter + 1;
```

Da müssen Sie wohl selbst ran. Erstellen Sie ein Codestück, um den Sachverhalt korrekt zu implementieren.

V7 Schleifen I

★ ☆ ☆

Schreiben Sie den folgenden Ausdruck mithilfe einer `for`-Schleife:

```
1 Robot bot = new Robot(0,0,UP,1);
2 int i = 3;
3 while(i < 9){
4  bot.move();
5  i = i + 1;
6 }
```

V8 Schleifen II

★ ☆ ☆

Ihr klammerfauler Kommilitone hat auch diesmal wieder zugeschlagen und versucht den Code aus vorheriger Aufgabe kürzer zu schreiben. Was sagen Sie dazu?

```
1 Robot bot = new Robot(0,0,UP,1);
2 int i = 3;
3 while(counter < 9) bot.move(); i = i + 1;
```

V9 Anzahl an Umdrehungen ★★☆

Legen Sie eine Variable `int numberOfTurns` an und setzen Sie ihren Wert zu Beginn auf 0. Erstellen Sie dann einen neuen Roboter und platzieren Sie ihn an der Stelle $(8, 2)$. Er schaut dabei nach links und besitzt keine Coins. Lassen Sie den Roboter nun geradewegs auf die Stelle $(0, 2)$ zusteuern und alle Coins auf seinem Weg aufsammeln. Liegen mehrere Coins auf einer Stelle, so soll er alle Coins aufsammeln. Bei jedem Aufsammeln, erhöhen Sie den Wert von `numberOfTurns` um 1. Hat er am Ende die Stelle $(0, 2)$ erreicht, soll er sich `numberOfTurns`-mal nach links drehen.

V10 Vorsicht Wand! ★★☆

Gehen Sie in dieser Aufgabe davon aus, dass Sie einen Roboter `wally` an der Position $(0, 0)$ erstellt haben und er nach rechts schaut. An der Position $(x, 0)$ befindet sich eine vertikale Wand die den Weg nach $(x + 1, 0)$ versperrt, die x-Koordinate ist allerdings unbekannt. Schreiben Sie ein kleines Programm, mit dem Sie den Roboter bis vor die Wand laufen lassen, direkt vor der Wand einen Coin ablegen, um dann wieder an die Ausgangsposition $(0, 0)$ zurückzukehren.

Hinweis: Es gibt die Funktion `isFrontClear()`, mit der getestet werden kann, ob sich in der Blickrichtung des Roboters direkt eine Wand befindet.

V11 Codeverständnis ★★☆

Beschreiben Sie ausführlich, welches Verhalten der nachfolgende Code umsetzt. Bei Fragen zur Funktionalität einzelner Methoden, werfen Sie einen Blick in die entsprechenden Vorlesungsfolien.

```
1  Robot robot = new Robot(0, 0, RIGHT, 99999);
2  int counter = 0;
3  while(robot.getX() < World.getWidth() - 1) {
4   robot.move();
5   counter = counter + 1;
6  }
7
8  robot.turnLeft();
9
10 for(int i = 0; i < counter; i++) {
11  if(i % 2 == 0 && robot.hasAnyCoins()) {
12   robot.putCoin();
13  }
14  robot.move();
15 }
16
17 robot.turnLeft();
18 while(robot.isFrontClear()) {
19  robot.move();
20 }
21
22 robot.turnLeft();
23  while(robot.getX() != 0 || robot.getY() != 0) {
24 robot.move();
25 }
26
27 robot.turnOff();
```

V12 Navigator ★★★

Gegeben seien vier Variablen:

```
int startX          int startY
int destinationX    int destinationY
```

Ihr Roboter befindet sich zu Beginn an der Position (`startX`, `startY`) und schaut in eine beliebige Richtung. Schreiben Sie ein Programm, das ihn von dieser Position auf die Position (`destinationX`, `destinationY`) laufen lässt.

H Erste Hausübung — Gesamt 11 Punkte

Pac-Man

In dieser Hausübung implementieren Sie Teile einer leicht vereinfachten Version des Arcade-Spiel-Klassikers Pac-Man unter Verwendung des FopBot-Frameworks. Das Spielprinzip ist leicht erklärt, die Spielfigur Pac-Man muss Punkte in einem Labyrinth fressen, während sie von Gespenstern verfolgt wird. Wird Pac-Man von einem Gespenst gefangen, so ist das Spiel verloren. Sammelt Pac-Man hingegen alle Punkte ein, ohne von einem Gespenst gefangen zu werden, so hat man das Spiel gewonnen. Anstelle von Punkten werden wir die bereits vorhandenen Coins aus dem FopBot-Framework verwenden. Pac-Man und die Geister werden wir als Roboter (Klasse `Robot`) realisieren. Das Grundgerüst des Spiels ist Ihnen bereits durch die Vorlage gegeben. Sie müssen lediglich, an den von uns gekennzeichneten Stellen ihren eigenen Code ergänzen. Abbildung 2 zeigt die Spielumgebung.[1]

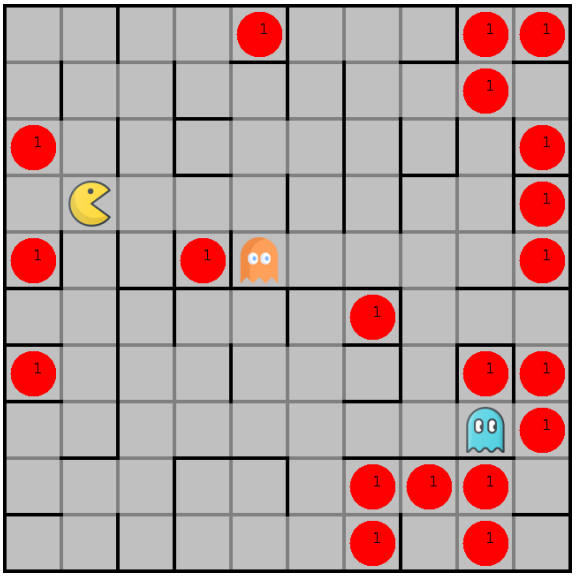

Abbildung 2: Pac-Man Spiel

Hinweise zur Bearbeitung:

- Beachten Sie unbedingt die Hinweise zur Abgabe auf Übungsblatt 0! Insbesondere sind die Erläuterungen zum korrekten Importieren und Exportieren, sowie zum Plagiarismus wichtig!
- Ergänzen Sie nur den Code an den angegebenen Stellen. Ändern Sie auf keinen Fall etwas an anderen Stellen ab.

[1] Quellen für die Bilder der Spielfiguren:
Pacman von Those Icons auf www.flaticon.com
Blaues Gespenst von Those Icons auf www.flaticon.com
Oranges Gespenst von Freepik auf www.flaticon.com

H1 Steuerung des Pac-Man-Roboters 4 Punkte

Zunächst wollen wir die Steuerung des Pac-Man-Roboters implementieren. Der Roboter soll über die auf der Tastatur befindlichen Pfeiltasten gesteuert werden. Wird die Pfeiltaste nach oben gedrückt, so soll sich der Roboter einen Schritt nach oben bewegen, wird die Pfeiltaste nach rechts gedrückt, so soll sich der Roboter einen Schritt nach rechts bewegen usw. Die Tastatureingaben werden schon automatisch von der Vorlage abgefangen, Sie müssen diese selbstverständlich nicht selbst implementieren. Ihre Aufgabe ist es lediglich eine Methode zu vervollständigen, die dann die Steuerung des Roboters möglich macht.

Vervollständigen Sie die Methode `handleKeyInput(int k)` der Klasse `Pacman` (Datei Pacman.java). Diese bekommt einen ganze Zahl übergeben (Parameter `k`), die für eine vom Spieler gedrückte Taste auf der Tastatur steht. Sie können den Parameter `k`, wie eine in der Methode definierte Variable (`int k`) behandeln, nur das ihnen der Wert unbekannt ist. Die Pfeiltaste nach links hat den Wert 0, die Pfeiltaste nach oben den Wert 1, die Pfeiltaste nach rechts den Wert 2 und die Pfeiltaste nach unten den Wert 3. Sollte der Methode ein Wert übergeben werden, der nicht zwischen 0 und 3 liegt (beide inklusive), so soll die Methode `handleKeyInput(int k)` gar nichts machen. Liegt der Wert im vorher genannten Bereich, so soll der Roboter zunächst in die Richtung blicken, die der gedrückten Pfeiltaste entspricht. Also `UP` bei gedrückter Pfeiltaste nach oben usw. Nachdem der Roboter in die entsprechende Richtung blickt, soll er einen Schritt in eben diese Richtung gehen, sofern der Weg nicht durch eine Wand versperrt ist. Ist der Weg nicht versperrt und befinden sich Coins auf dem soeben neu betretenen Feld, so soll der Roboter alle Coins auf diesem Feld aufheben. Ist der Weg versperrt, so verbleibt der Roboter an seiner momentanten Position.

Haben Sie die oben genannte Methode korrekt vervollständigt, so müsste es Ihnen nun bei der Ausführung des Programms möglich sein, den Pac-Man-Roboter über die Pfeiltaste ihrer Tastatur zu steuern. Wenn Sie mehrere Pfeiltasten gleichzeitig drücken, bekommt die Methode `handleKeyInput(int k)` den Wert -1 übergeben, demnach darf sich der Roboter nicht drehen und nicht bewegen.

H2 Gespenster 7 Punkte

Nun wollen wir die Gespenster der Pac-Man-Welt implementieren. In unserer leicht vereinfachten Pac-Man-Version gibt es nur 2 verschiedene Arten von Gespenstern, diese verfolgen beide unterschiedliche Taktiken um unseren Pac-Man-Roboter zu fangen.

H2.1 Blaue Gespenster 4 Punkte

Blaue Gespenster verfolgen eine primitive Taktik. Vor jedem Schritt prüfen sie, in welche Richtungen ihnen der Weg nicht versperrt ist. Sind mehrere Wege frei, so wird einer dieser Wege zufällig gewählt. Ist nur ein Weg frei, so wird eben ein Schritt in genau diese Richtung gegangen.

Vervollständigen Sie nun die Methode `doMove()` der Klasse `BlueGhost` (Datei BlueGhost.java). Die Methode soll zunächst überprüfen, in welche der vier Richtungen (`UP`, `DOWN`, `LEFT` und `RIGHT`) sich der Roboter bewegen kann, ohne dabei gegen eine Wand zu laufen. Sind alle freien Richtungen ausgemacht, wird eine dieser Richtungen zufällig gewählt, gefolgt von einem Schritt in eben genau diese Richtung. Zur Generierung von Zufallszahlen benutzen Sie die Methode `Util.getRandomInteger(int min, int max)`, diese bekommt zwei ganze Zahlen `min` und `max` übergeben und gibt eine zufällig gewählte Zahl zwischen `min` und `max` (beide inklusive) zurück. Alle freien Richtungen sollen mit der gleichen Wahrscheinlichkeit ausgewählt werden.

H2.2 Orange Gespenster 3 Punkte

Die orangen Gespenster verfolgen eine andere Taktik. Sie gehen folgendermaßen vor:

1. Vor jedem Schritt wird sich die momentane Blickrichtung des Roboters gemerkt.
2. Dann wird eine Rechtsdrehung ausgeführt.
   - 2.1. Ist der Weg nicht versperrt, wird ein Schritt in die momentane Richtung gegangen. Danach ist nichts mehr zu machen.
   - 2.2. Ist der Weg hingegen versperrt, wird die gemerkte Blickrichtung aus 1. wiederhergestellt.
     - 2.2.1. Ist der Weg nun nicht mehr versperrt, wird ein Schritt in die momentane Richtung gegangen. Danach ist nichts mehr zu machen.
     - 2.2.2. Ist der Weg hingegen immer noch versperrt, werden so lange Linksdrehungen ausgeführt, bis der Weg frei wird. Ist der Weg frei, wird ein Schritt in eben diese Richtung gegangen. Danach ist nichts mehr zu machen.

Vervollständigen Sie nun die Methode `doMove()` der Klasse `OrangeGhost` (Datei OrangeGhost.java), sodass die oben beschriebene Taktik umgesetzt wird.