Funktionale und objektorientierte Programmierkonzepte

Florian Kadner und Lukas Röhrig
(Gesamtleitung: Prof. Karsten Weihe)

Wintersemester 18/19
v1.0

Übungsblatt 4

Themen: KarelJ I

Relevante Folien: KarelJ

Abgabe der Hausübung: 23.11.2018 bis 23:55 Uhr

V Vorbereitende Übungen

V1 KarelJ

☆ ☆ ☆

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

> **Für alle nachfolgenden Übungen**: 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.

V2 Liegen geblieben

☆ ☆ ☆

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

```
1 Robot karel = new Robot(2,4,North,0);
2 karel.move();
3 karel.turnLeft();
4 karel.putBeeper();
```

In welcher Zeile kommt es zu einem Problem und wieso?

V3 Rechteck

★ ☆ ☆

Schreiben Sie ein Programm, welches zwei Roboter `karel` und `larel` erstellt. Dabei soll `karel` mit Beepern ein Rechteck der Höhe 5 und der Breite 3 zeichnen. Nachdem das Rechteck gezeichnet wurde, soll `larel` alle Beeper wieder einsammeln. Überlegen Sie sich, wie Sie das Programm mit nur einer Schleife pro Roboter gestalten können. Orientieren Sie sich zur Hilfe am `FillRectangle`-Beispiel (Folie 78ff) aus der Vorlesung.

V4 Bedingungen I

★ ☆ ☆

Betrachten Sie folgenden Codeausschnitt:

```
1 Robot karel = new Robot(2,4,North,1);
2 karel.move();
3 if(karel.nextToABeeper()){
4  karel.pickBeeper();
5 }
6 else{
7  karel.putBeeper();
8 }
```

Beschreiben Sie in eigenen Worten, welchem Zweck dieser Codeausschnitt dient. Erweitern Sie außerdem den Code so, dass `karel` nur einen Beeper ablegt, wenn er auch mindestens einen in seiner BeeperBag hat.

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, für was der `%` Operator verwendet werden kann.

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;
```

Was passiert beim Ausführen des Codes?

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 korrektes Codestück, um den Sachverhalt korrekt zu implementieren.

V7 Schleifen I

★ ☆ ☆

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

```
1 Robot karel = new Robot(1,1,North,1);
2 int i = 5
3 while(i < 28){
4  karel.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 karel = new Robot(1,1,North,1);
2 int i = 5;
3 while(counter < 28) karel.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 $(3, 15)$. Er schaut dabei nach Westen und hat keine Beeper in seiner Tasche. Lassen Sie den Roboter nun geradewegs auf die Stelle $(3, 1)$ zusteuern und alle Beeper auf seinem Weg aufsammeln. Liegen mehrere Beeper auf einer Stelle, so soll er alle Beeper aufsammeln. Bei jedem Aufsammeln, erhöhen Sie den Wert von `numberOfTurns` um 1. Hat er am Ende die Stelle $(3, 1)$ erreicht, soll er sich `numberOfTurns`- mal nach links drehen.

V10 Vorsicht Wand!

★ ★ ☆

Gehen Sie in dieser Aufgabe davon aus, dass Sie einen Roboter `karel` an der Position $(1, 1)$ erstellt haben und er nach Osten schaut. An der Position $(1, x)$ befindet sich eine Wand, die Avenueposition x ist allerdings unbekannt. Schreiben Sie ein kleines Programm, mit dem Sie den Roboter bis vor die Wand laufen lassen, direkt vor der Wand einen Beeper ablegen, um dann wieder an die Ausgangsposition $(1, 1)$ zurückzukehren.

Hinweis: Es gibt die Funktion `frontIsClear()`, 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 offizielle Dokumentation: https://csis.pace.edu/ bergin/KarelJava2ed/KJRdocs/index.html.

```
1  Robot robot = new Robot(1, 1, East, infinity);
2  int counter = 0;
3  while(robot.avenue() < World.numberOfAvenues()) {
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.anyBeepersInBeeperBag()) {
12   robot.putBeeper();
13  }
14  robot.move();
15 }
16
17 robot.turnLeft();
18 while(robot.frontIsClear()) {
19  robot.move();
20 }
21
22 robot.turnLeft();
23  while(!robot.areYouHere(1, 1)) {
24 robot.move();
25 }
26
27 robot.turnOff();
```

V12 Navigator ★★★

Gegeben seien vier Variablen:

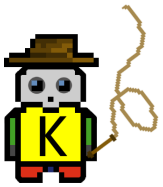
```
int startStreet        int startAvenue
int destinationStreet  int destinationAvenue
```

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

H Vierte Hausübung — Gesamt 5 Punkte

Karel der Schatzsucher

In dieser vierten Hausübung begleiten Sie den kleinen Roboter `IndianaKarel` auf seiner Schatzsuche. Helfen Sie ihm, indem Sie die drei nachfolgenden Aufgaben bearbeiten! Der Roboter `IndianaKarel` besitzt alle Fähigkeiten eines normalen Roboters und zusätzlich die `turnRight()` Methode. Alle Aufgaben sind voneinander unabhängig, sollten Sie also an einer Stelle Schwierigkeiten haben, stellt dies kein Problem für die anderen Teile dar.



Ihren Lösungscode zu den einzelnen Aufgaben schreiben Sie an die gekennzeichneten Stellen in der Klasse `IndianaKarel.java`. Um die Aufgaben auszuführen, müssen Sie die Klasse `IndianaWorld.java` ausführen und ganz oben die Variable `int exerciseNumber` auf 1,2 oder 3 setzen. **Beachten Sie auch die Hinweise ganz am Ende!**

H1 Pyramide bauen — 1 Punkt

In der ersten Aufgabe sollen Sie eine Pyramide aus Beepern zeichnen, in der `IndianaKarel` nach dem Schatz suchen möchte. Zu Beginn befindet sich `IndianaKarel` an Position $(3, 1)$ und blickt nach Osten. Die Anzahl der Beeper steht am Anfang noch auf 0 (Variable `int pyramidBeepers`) und muss von Ihnen durch einen passenden Wert ergänzt werden, sodass am Ende **keine** Beeper mehr in der BeeperBag übrig sind. Zu Beginn zeichnet der Roboter eine Grundlinie der Pyramide aus 13 Beepern in Street 4. Die darauffolgenden Ebenen werden immer mittig platziert und sind genau zwei Avenues kleiner in ihrer Breite als die vorherige Ebene. Außerdem entspricht die Anzahl an Beepern pro Feld immer der aktuellen Streetanzahl - 3. Zur Veranschaulichung betrachten Sie Abbildung 1. Ergänzen Sie den Code so, dass genau diese Pyramide gezeichnet wird. Code zu ergänzen an der Stelle: `public void buildPyramid()`

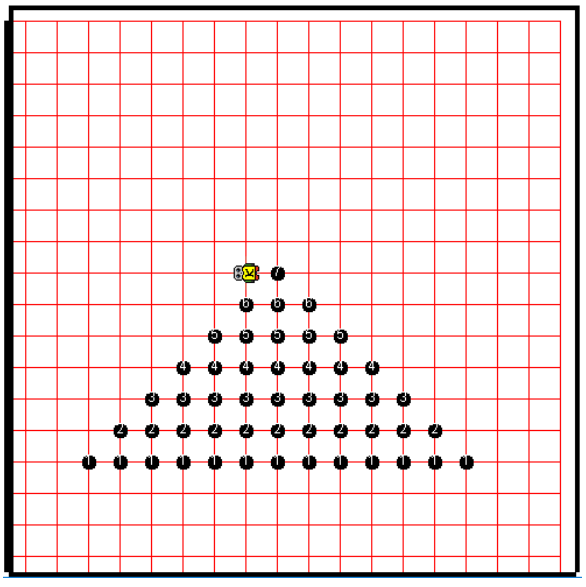


Abbildung 1: Das Bild der fertigen Pyramide

H2 Das Labyrinth 2 Punkte

Einmal die Pyramide betreten, findet sich `IndianaKarel` in einem großen Labyrinth wieder. Ihre Aufgabe ist es nun ihn aus dem Labyrinth heraus, direkt in die Schatzkammer zu führen. Das Labyrinth besteht aus vielen zufällig generierten Wänden, ein Beispiel finden Sie in Abbildung 2. Der Ausgang ist durch den einzigen Beeper im Labyrinth gekennzeichnet. Dieser soll aufgesammelt werden und `indianaKarel` soll sich danach selbst ausschalten. Der Startpunkt des Roboters ist dabei zufällig in einer der vier Eckpunkte, der Ausgang dann zufällig in einer der drei anderen Eckpunkte. Die Größe der Welt ist ebenfalls variabel und wird zufällig gesetzt. In dieser Aufgabe stehen Ihnen keine Beeper zur Verfügung.

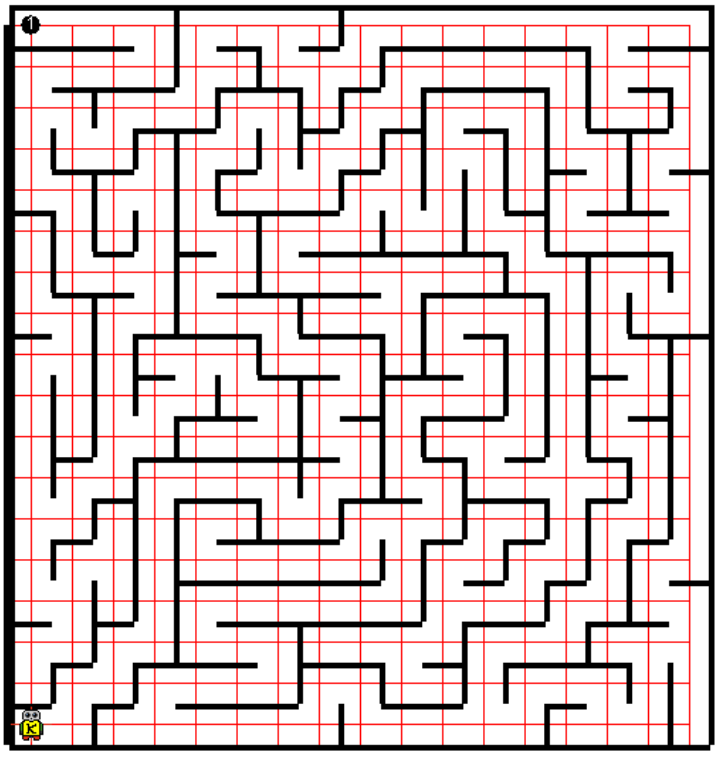


Abbildung 2: Ein zufälliges Beispiellabyrinth.

Tipp:

Überlegen Sie sich zunächst einmal, wie Sie selbst durch das Labyrinth gehen würden. Anschließend können Sie diese entwickelte Strategie dann auf `IndianaKarel` ummünzen. Das Labyrinth sieht bei jedem Start anders aus, die einzigen Anhaltspunkte die Sie haben sind die Wände, die immer wieder zufällig erzeugt werden.

Hinweise:

Sie können davon ausgehen, dass keine Säle - also größere Flächen ohne Wände - im Labyrinth vorkommen. Der Roboter kann sich dementsprechend immer nur in zwei verschiedene Himmelsrichtungen an einer Stelle bewegen.

Code zu ergänzen an der Stelle: `public void solveMaze()`

H3 Alles einsammeln 2 Punkte

Unser kleiner Held ist endlich in der Schatzkammer angekommen und möchte natürlich alle dort liegenden Beeper aufsammeln, die Welt soll also völlig leer danach sein. In dieser Aufgabe, sollen Sie nun eine Strategie für diesen Roboter umsetzen, der ihn alle Beeper in einer Welt einsammeln lässt (ähnlich des `PacmanRobot` aus der Vorlesung). Die Schwierigkeit dabei: Die Größe der Welt, die Anzahl der Beeper, die Positionen der Beeper und die Startposition des Roboters sind randomisiert (= zufällig gesetzt). Selbstverständlich können auf einem Feld auch mehrere Beeper liegen (siehe auch Abbildung 3). In diesem Fall müssen dann alle Beeper aufgenommen werden. Der Roboter schaut am Anfang immer zufällig in eine der vier Himmelsrichtungen. Code zu ergänzen an der Stelle: `public void collectAll()`

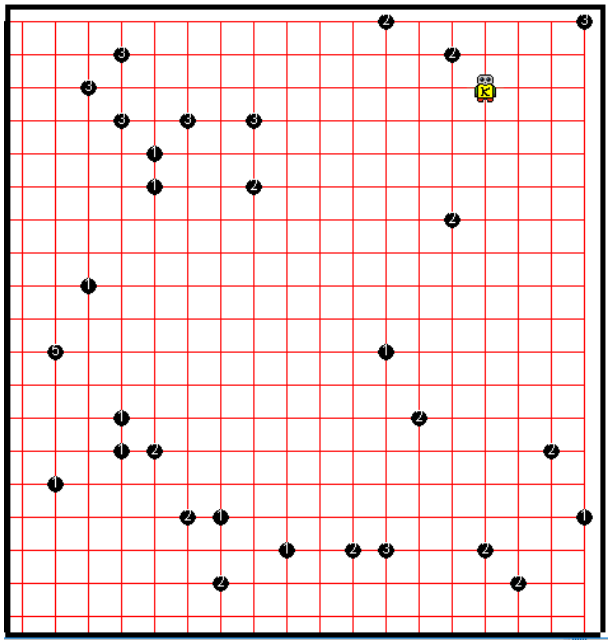


Abbildung 3: Eine zufällige Beispielwelt, in der alle Beeper eingesammelt werden sollen.

Hinweise zur Bearbeitung:

- Beachten Sie unbedingt die Hinweise zur Abgabe auf Übungsblatt 3! 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.