Check und Prepare

Lösungsvorschläge

TECHNISCHE UNIVERSITÄT DARMSTADT

Autoren: Nhan Huynh und Darya Nikitina
Fachbereich: Informatik
Übungsblatt: 01
Version: 20. November 2021
Semesterübergreifend

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 und Papier zu programmieren. Abschließend können Sie dann ihre vollständige Lösung in die Entwicklungsumgebung übertragen und überprüfen.

Lösungsvorschlag:

FopBot ist ein Package mit mehreren Klassen. Man kann damit eine beliebige Welt (mit Wänden) generieren und beliebig viele Objekte vom Typ Robot rein setzen, die jeweils beliebig viele Coins haben können. In der Klasse Robot sind mehrere Methoden, die es dem Robot erlauben mit der Welt zu interagieren wie bspw. move(), die den Robot in eine Richtung laufen lässt (eine Richtung ist mithilfe von Directions als Enumeration umgesetzt).

Information: „Ein Package ist ein Namespace mithilfe dessen eine Reihe verwandter Klassen und Interfaces organisiert werden können.“(`https://docs.oracle.com/javase/tutorial/java/concepts/package.html`)

Als Beispiel dazu das Package `java.awt`. Dieses Package enthält Klassen zum Erstellen von Benutzeroberflächen und zum Zeichnen von Grafiken und Bildern.

Mithilfe des folgenden Verweises können Sie sich einen Überblick über Packages verschaffen:

`https://docs.oracle.com/en/java/javase/11/docs/api/allpackages-index.html`

Mehr Informationen zu FopBot finden Sie unter den folgenden Verweisen:

- Sourcecode: `https://github.com/FOP-2022/FOPBot`
- Dokumentation: `https://fop-2022.github.io/FOPBot/fopbot/package-summary.html`

Übungsblatt 01: Lösungsvorschläge von Nhan Huynh und Darya Nikitina

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 ();
```

Lösungsvorschlag:

Es kommt in der Zeile 4 zu einem Fehler, weil der Roboter keinen Coin hat, den er ablegen kann.

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 innen liegenden 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.

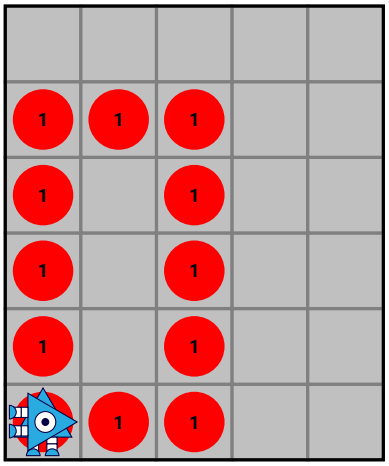

Abbildung 1: Fertiggestelltes Rechteck durch `putbot`

Übungsblatt 01: Lösungsvorschläge von Nhan Huynh und Darya Nikitina

Lösungsvorschlag:

```
int width = 3;
int height = 5;

// Number of total coins to place - 4 for edges (put coins only once on every edge)
int coins = 2 * width + 2 * height - 4;

Robot putbot = new Robot(0, 0, Direction.RIGHT, coins);
Robot pickbot = new Robot(0, 0, Direction.RIGHT, 0);

// Marker to change directions
boolean swap = false;
int i = 1;

// Put coins until we placed all of them
while (putbot.hasAnyCoins()) {
  putbot.putCoin();

  // If we reached a corner, we change the direction of the bot
  if (!swap && i == width || swap && i == height) {
    swap = !swap;
    i = 1;
    putbot.turnLeft();
  }

  putbot.move();
  i++;
}

// Reset variables
swap = false;
i = 1;
// Counter for coins
int counter = 0;

// Pick up coins until we picked up all (we can have a variable amount of coins) of them
while (counter < coins) {
  pickbot.pickCoin();
  counter++;

  // If we reached a corner, we change the direction of the bot
  if (!swap && i == width || swap && i == height) {
    swap = !swap;
    i = 1;
    pickbot.turnLeft();
  }

  pickbot.move();
  i++;
}
```

Information: Der Lösungsvorschlag verwendet in Zeile 19 und 41 „Bedingte Operatoren“. In diesem Fall ist es ein logisches oder, d.h. wenn Bedingung 1 oder 2 zutrifft, dann führen wir die folgenden Anweisungen aus.

Erste Bedingung	Zweite Bedingung	Ausführung der Anweisungen?
`false`	`alse`	Nein
`false`	`true`	Ja
`true`	`false`	Ja
`true`	`true`	Ja

Tabelle 1: Wahrheitstabelle zum logischen oder

Als alternative zum logischen oder könnte man zwei unabhängige `if`-Abfragen verwenden:

```
1 if (!swap&&i==width) {
2   swap=!swap;
3   i=1;
4   pickbot.turnLeft();
5 }
6
7 if (swap&&i==height) {
8   swap=!swap;
9   i=1;
10  pickbot.turnLeft();
11 }
```

Mehr Informationen zu bedingten Operatoren kann man unter folgendem Verweis finden:

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/op2.html

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 } else {
6   bot1.putCoin();
7 }
```

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.

Übungsblatt 01: Lösungsvorschläge von Nhan Huynh und Darya Nikitina

Lösungsvorschlag:

Es wird ein Roboter an der Position `(3, 1)` mit der Blickrichtung nach oben platziert. Dieser Roboter hat eine Münze und führt einen Schritt nach vorne (in Blickrichtung) aus. Falls sich unter ihm eine Münze befindet, so hebt er diese auf. Ansonsten legt er eine Münze ab (Erweiterung: Sofern er mindestens eine Münze besitzt).

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

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.

Lösungsvorschlag:

Der Ausdruck gibt `1` zurück, also den Rest der ganzzahligen Division von `a` und `b`. Beim `%` handelt sich um den Modulo-Operator, welcher den Rest einer ganzzahligen Division zurückgibt.

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.

Übungsblatt 01: Lösungsvorschläge von Nhan Huynh und Darya Nikitina

Lösungsvorschlag:

1. Zwischen einer `if`- und `else`-Anweisung darf keine weitere Anweisung nach der Ersten zwischen dem `if` und `else` stehen. Der Komilitone müsste die erste und zweite Anweisung nach dem `if` in Klammern schreiben.
2. Hier fehlen die Klammern bei der zweiten Anweisung nach dem `if`, da der `counter` ansonsten immer um `1` inkrementiert wird.

   Der verbesserte Code lautet:

```
1 if (counter > 0)
2   counter = counter * 2;
3 if (counter < 0) {
4   counter = -counter;
5   counter = counter + 1;
6 }
```

Übungsblatt 01: Lösungsvorschläge von Nhan Huynh und Darya Nikitina

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 }
```

Lösungsvorschlag:

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

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;
```

Lösungsvorschlag:

Der Roboter würde in diesem Falle unendlich weiterlaufen, da die Inkrementation der Variable `i` außerhalb der `while`-Schleife ist.

> **Information:** Inkrementation stammt aus dem lateinischen Wort „incrementare“ (deutsch: vergrößern) und beschreibt im Bezug zum oberen Codeauschnitt die schrittweise Vergrößerung einer Variable.

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.

Übungsblatt 01: Lösungsvorschläge von Nhan Huynh und Darya Nikitina

Lösungsvorschlag:

```
Robot bob = new Robot(8, 2, LEFT, 0);
int numberOfTurns = 0;

while (bob.getX() != 0) {
  while (bob.isNextToACoin()) {
    bob.pickCoin();
    numberOfTurns++;
  }
  bob.move();
}

for (int i = 0; i < numberOfTurns; i++) {
  bob.turnLeft();
}
```

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 `(0, 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 Blickrichtung des Roboters vor ihm direkt eine Wand befindet.

Lösungsvorschlag:

```
Robot wally = new Robot(0, 0, RIGHT, 1);

while (wally.isFrontClear()) {
  wally.move();
}

wally.putCoin();
wally.turnLeft();
wally.turnLeft();

while (wally.getX() != 0 || wally.getY() != 0) {
  wally.move();
}
```

Übungsblatt 01: Lösungsvorschläge von Nhan Huynh und Darya Nikitina

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

Übungsblatt 01: Lösungsvorschläge von Nhan Huynh und Darya Nikitina

Lösungsvorschlag:

```
// Create a robot at position (0,0), looks to the right and has 99999
// coins
Robot robot = new Robot(0, 0, RIGHT, 99999);
// Move the robot to the end of the world and store the counter in a variable
int counter = 0;
while (robot.getX() < World.getWidth() - 1) {
  robot.move();
  counter = counter + 1;
}

// Turn the robot to the left (Direction.UP)
robot.turnLeft();

// The robot moves in the y-Direction and places a coin at every even
// position. The robot does this until the y-coordinate is greater than or equal
// to the size of the width of the world
for (int i = 0; i < counter; i++) {
  if (i % 2 == 0 && robot.hasAnyCoins()) {
    robot.putCoin();
  }
  robot.move();
}

// Turn the robot to the left (Direction.LEFT)
robot.turnLeft();
// Move the robot until it hits a wall
while (robot.isFrontClear()) {
  robot.move();
}

// Turn the robot to the left (Direction.DOWN)
robot.turnLeft();
// Move back to the starting point
while (robot.getX() != 0 || robot.getY() != 0) {
  robot.move();
}

// Turn the robot off
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.

Übungsblatt 01: Lösungsvorschläge von Nhan Huynh und Darya Nikitina

Lösungsvorschlag:

```
// Suppose the variable for the robot is called bot

// Move to x-coordinate
if (startX != destinationX) {
  // Adjust the direction of the robot so that it looks left
  if (startX - destinationX > 0) {
    while (!bot.isFacingLeft()) {
      bot.turnLeft();
    }
  } else {
    // Adjust the direction of the robot so that it looks right
    while (!bot.isFacingRight()) {
      bot.turnLeft();
    }
  }
  while (bot.getX() != destinationX)
    bot.move();
}
// Move to y-coordinate
if (startY != destinationY) {
  // Adjust direction of the robot so that it looks down
  if (startY - destinationY > 0) {
    while (!bot.isFacingDown()) {
      bot.turnLeft();
    }
  } else {
    // Adjust direction of the robot so that it looks up
    while (!bot.isFacingUp()) {
      bot.turnLeft();
    }
  }
  while (bot.getY() != destinationY) {
    bot.move();
  }
}
```