Schleifen-Expertin

Aufgabe

Kara soll geradeaus laufen, bis er vor einem Baum steht.

Situation vor dem Start des Programms

Situation nachher

Das Neue

Mit einer `while` Schleife können Befehle oder Befehlsblöcke wiederholt werden, solange eine Bedingung erfüllt ist.

Lösung in JavaKara

```
import JavaKaraProgram;
public class FindeBaum extends JavaKaraProgram
{ // Anfang von FindeBaum
  public void myProgram()
  { // Anfang von myProgramm
    while (!kara.treeFront())
    {
      kara.move();
    }
  } // Ende von myProgramm
} // Ende von FindeBaum
```

Bemerkung

Das Ausrufezeichen bei `!kara.treeFront()` ist der not-Operator. Er kehrt die Aussage von `kara.treeFront()` um. Also wenn ein Baum vor Kara ist, wird der Ausdruck *falsch* anstatt *wahr*. Entsprechend, wenn Kara nicht vor einem Baum steht, *wahr* anstelle von *falsch*.

Erläuterungen

1. **Schleifen: grundlegende Ablaufstruktur beim Programmieren**
   Der Computer eignet sich zum wiederholten Ausführen von Instruktionen. Durch diese Schleifen im Programmablauf kann man viele Aufgaben automatisieren. Roboter können zum Beispiel ein Auto zusammenbauen oder andere „Fliessbandarbeiten“ übernehmen.

2. **Syntax einer `while` Schleife**

   `while (`Schleifenbedingung`)`

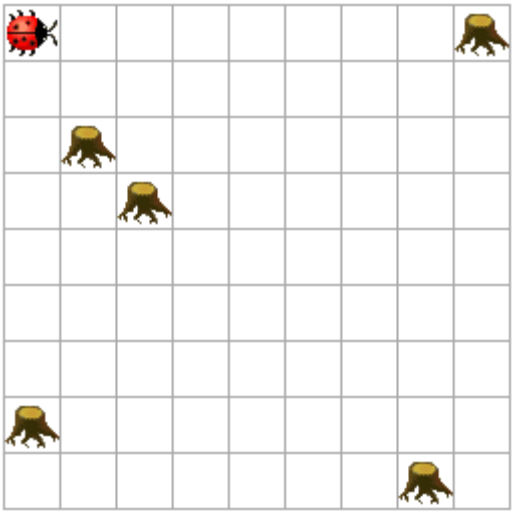
```
{
   Anweisung1
   Anweisung2
   Anweisung3
}
```

Solange die Bedingung erfüllt ist, werden die verschiedenen Anweisungen der Reihe nach abgearbeitet. Dabei wird stets *vor* dem Abarbeiten der ersten Anweisung geprüft, ob der Anweisungsblock überhaupt noch ausgeführt werden muss.

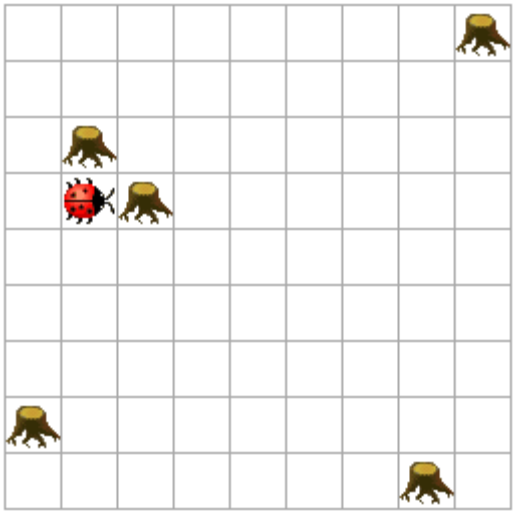
Aufgabe

Nun soll Kara beim Erreichen des Baumstammes eine Rechtsdrehung machen und Weiterlaufen bis zum nächsten Baumstamm. Das soll er so lange machen, bis er nach der Rechtsdrehung unmittelbar wieder vor einem Baum steht.

Situation vor dem Start des Programms



Situation nachher



Das Neue

`while` Schleifen müssen teilweise auch verschachtelt angewandt werden.

Lösung in JavaKara

```
import JavaKaraProgram;
public class ImKreisHerum extends JavaKaraProgram
{ // Anfang von ImKreisHerum
  public void myProgram()
  { // Anfang von myProgramm
    while (!kara.treeFront())
    {
      while (!kara.treeFront())
      {
        kara.move();
      }
      kara.turnRight();
    }
  } // Ende von myProgramm
```

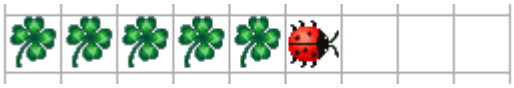
```
} // Ende von ImKreisHerum
```

Aufgabe

Kara möchte 5 Kleeblätter in die leere Welt setzen, die alle hintereinander angeordnet sein sollen.

Situation vor dem Start des Programms

Situation nachher



Das Neue

Mit einer `for` Schleife kann ein Block von Anweisungen eine bestimmte Anzahl mal durchlaufen werden.

Lösung in JavaKara

```
import JavaKaraProgram;
public class FuenfKleeblaetter extends JavaKaraProgram
{ // Anfang von FuenfKleeblaetter
  public void myProgram()
  { // Anfang von myProgramm
    for (int i=1; i<=5; i++)
    {
      kara.putLeaf();
      kara.move();
    }
  } // Ende von myProgramm
} // Ende von FuenfKleeblaetter
```

Erläuterungen

1. **Die Syntax einer `for` Schleife:**

   ```
   for (Initialisierung; Bedingung; Aktualisierung)
   {
     Anweisung1
     Anweisung2
     Anweisung3
   }
   ```

   Die Initialisierungs-Anweisung (im obigen Beispiel `int i=1`) wird vor Beginn der Schleife einmal ausgeführt. Sie wird oft dazu verwendet, Zählervariablen auf Anfangswerte zu setzen.
   Die Bedingung (im obigen Beispiel `i<=5`) wird genau wie bei der `while` Schleife vor der ersten Anweisung getestet. Trifft sie nicht (mehr) zu, dann wird die Schleifen-ausführung gestoppt und mit dem restlichen Programmablauf fortgefahren.
   Am Ende eines Schleifendurchlaufes wird – bevor die Bedingung neu getestet wird –

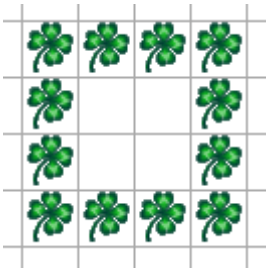
die Aktualisierungs-Anweisung (im obigen Beispiel `i++`) ausgeführt. Meistens wird sie dazu verwendet, um die Zählvariable zu erhöhen.

2. Den Ausdruck `int` braucht es in der Initialisierung, damit Java weiss, dass unsere Zählvariable `i` eine ganze Zahl darstellt.

3. Die Aktualisierungs-Anweisung `i++` erhöht die Zählvariable um eins. Man hätte auch `i=i+1` schreiben können. Entsprechend kann man auch `i--` anstelle von `i=i-1` schreiben.

Fragen

1. Kara möchte ein Rechteck zeichnen, indem er den Rand einer 4x4 Felder grossen Fläche mit Kleeblättern markiert. Überlege dir, ob sich wieder eine `while` oder eine `for` Schleife eignet und schreibe das entsprechende Programm. Teste es anschliessend am Computer.



2. Neben der Methode `kara.putLeaf()` gibt es bei JavaKara auch die Methode `world.setLeaf(int x, int y, boolean putLeaf)`. Der Vorteil dabei ist, dass wir Kara gar nicht mehr brauchen und trotzdem Kleeblätter in der Welt legen können. Das Beispiel `world.setLeaf(0,2,true)` setzt ein Kleeblatt an die Koordinate (0,2).
   Fülle mit dieser Methode eine Fläche von 5x5 Feldern in der Kara-Welt.
   Zusatzaufgabe: Versuche herauszufinden, was das `true`/`false` bei der Methode `world.setLeaf` bedeutet. Benutze dazu deine Kursunterlagen.

3. Kara soll ebenfalls eine 5x5 Felder grosse Fläche mit Kleeblätter bedecken. Entwerfe die Programme zuerst auf Papier, teste anschliessend am Computer.
   a. Lass deinen Kara immer 5 Kleeblätter horizontal legen und anschliessend die 5 Felder zurück laufen, bevor er sich auf die nächste Zeile wagt.
   b. Nun kann Kara den Weg noch optimieren. Er kann bereits beim zurück spazieren wieder 5 Kleeblätter auf der nächsten Zeile legen.
      Hinweis: Diese Teilaufgabe ist einiges schwieriger als die Teilaufgabe a.