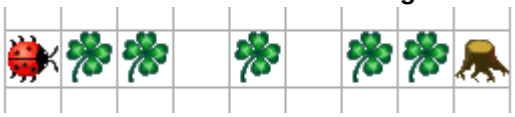
Verzweigungs-Expertin

Aufgabe

Vor Kara hat es Felder, die entweder leer oder mit einem Kleeblatt belegt sind. Diese „Spur“ endet vor einem Baum. Kara soll alle Kleeblätter aufheben und bei allen Lücken ein Kleeblatt legen.

Situation vor dem Start des Programms



Situation nachher

Das Neue

Kara muss 2 Fälle unterscheiden:

- Ist ein Kleeblatt unterhalb von Kara, so muss er dieses aufheben.
- Ist noch kein Kleeblatt unter Kara, dann legt er ein Kleeblatt hin.

Lösung in JavaKara

```
import JavaKaraProgram;
public class InvertiereKleeblaetter extends JavaKaraProgram
{ // Anfang von InvertiereKleeblaetter
  public void myProgram()
  { // Anfang von myProgramm
    while (!kara.treeFront())
    {
      kara.move();
      if (kara.onLeaf())
      {
        kara.removeLeaf();
      }
      else
      {
        kara.putLeaf();
      }
    }
  } // Ende von myProgramm
} // Ende von InvertiereKleeblaetter
```

Bemerkung:

Die Zeile `while (!kara.treeFront())` heisst soviel wie „solange kein Baum vor Kara ist, mache folgendes:“

Erläuterungen

1. **Verzweigungen: grundlegende Ablaufstruktur beim Programmieren**
   Das Prinzip von Verzweigungen gibt es nicht nur bei Kara, sondern überall wo programmiert wird. Stell dir zum Beispiel eine Maschine vor, die abschalten muss, wenn das Kühlsystem defekt ist. Oder einen Getränkeautomaten, der anders reagieren soll, wenn man ihn mit thailändischen 10-Baht-Münzstücken füttert anstelle der gleich schweren und gleich grossen 2-Euro-Münzen. Oder bei einem Kalender: Je nach Monat hat es eine unterschiedliche Anzahl Tage.

2. **Die Syntax einer Verzweigung**

   ```
   if (Bedingung) { Anweisungen1 } else { Anweisungen2 }
   ```

   oder

   ```
   if (Bedingung) { Anweisungen }
   ```

   Beim zweiten Fall werden die Anweisungen nur abgearbeitet, wenn die Bedingung zutrifft. Wenn die Bedingung nicht zutrifft, wird das Programm einfach nach den geschweiften Klammern fortgesetzt.

   Beispiel:
   ```
   if (kara.onLeaf()) { kara.removeLeaf(); }
   ```

3. **Mögliche Darstellungen**
   Es ist nicht übersichtlich, mehrere Anweisungen auf eine Zeile zu schreiben. Deshalb macht es Sinn, die geschweiften Klammern auf eine neue Zeile zu setzen und die Befehle für die Fallunterscheidung einzurücken.

   ```
   if (Bedingung)
   {
     Anweisung1;
     Anweisung2;
     Anweisung3;
   }
   ```

   Man nennt das `if` mit der Bedingung und allen Anweisungen (inklusive des `else` Teils) einen `if`-Block.

4. **Geschachtelte `if`-Anweisungen**
   In den Anweisungen eines `if`-Blockes können natürlich weitere `if`-Anweisungen stehen.

   ```
   if (Bedingung1)
   {
     if (Bedingung2)
     {
       Anweisung1;
       Anweisung2;
     }
   ```

```
  else
  {
    Anweisung3;
  }
}
else
{
  Anweisung4;
  Anweisung5;
}
```

Beachte, dass Einrückungen den Programmablauf nicht beeinflussen, aber die Lesbarkeit wesentlich verbessern.

Fragen

1. Betrachten wir untenstehendes Programm

```
(...)
while (!kara.treeFront())
{
  if (kara.onLeaf())
  {
    kara.removeLeaf();
  }
  else
  {
    kara.putLeaf();
  }
  kara.move();
}
(...)
```

   Im Gegensatz zur Lösung 1 wurde die Anweisung `kara.move();` hinter den `if`-Block gestellt. Ändert sich dadurch die Funktionalität des Programmes?

2. Das Programm auf der ersten Seite arbeitet nicht in allen Situationen korrekt.
   a. Versuche den Spezialfall zu finden, bei welchem das Programm nicht richtig arbeitet.
   b. Entwirf auf Papier eine verbesserte Version des Programms, die auch den Spezialfall berücksichtigt.
   c. Teste am Computer, ob deine Lösung funktioniert.

   Hinweis: Teste dein Programm für Spezialfälle: Kara steht zu Beginn auf einem Kleeblatt, oder vor dem Baum liegt ein Kleeblatt, etc.

3. Kara steht in einer Welt ohne Bäume. Er soll nun einen Schritt nach vorne machen. Liegt dort ein Kleeblatt, so soll Kara es aufnehmen, sich nach links drehen und einen Schritt nach vorne machen. Ist auf dem Feld kein Kleeblatt, so soll Kara ein Kleeblatt hinlegen, sich nach rechts drehen und einen Schritt nach vorne machen.

   Schreibe ein JavaKara Programm, das obige Aufgabe erfüllt und teste es anschliessend

am Computer.

4. Wozu könnte folgendes Programmschema gut sein?
   Hinweis: Stell dir vor, du musst dir die Ampelfarbe einer Lichtsignalanlage anschauen.

```
(...)
if (Bedingung1) { Anweisungen1 }
else {
  if (Bedingung2) { Anweisungen2 }
  else {
    if (Bedingung3) { Anweisungen3 }
    else { Anweisungen4 }
  }
(...)
```

5. *Knifflige Zusatzaufgabe:*

   Kara soll folgendes tun, wenn er auf einem Kleeblatt steht:
   - Zuerst soll er das Kleeblatt aufnehmen.
   - Ist vor ihm ein Baum, dann soll Kara eine Drehung um 180° machen.
   - Ist vorne kein Baum, dann soll Kara ein Feld nach vorne schreiten.

   Steht Kara auf keinem Kleeblatt, soll er nichts tun.

   Obige Aufgabe wurde bereits programmiert. Leider befindet sich ein Fehler im Programm.

```
 (...)
if (kara.onLeaf()) {
  // Kara ist auf Kleeblatt: Also aufnehmen
  kara.removeLeaf();
  if (kara.treeFront()) {
    // Kara steht vor Baum, drehe um
    kara.turnRight();
    kara.turnRight(); }
  }
  else {
    // Kara steht nicht vor einem Baum
    kara.move();
}
// und sonst mache gar nichts
(...)
```

   Finde den Fehler und verbessere das Programm.