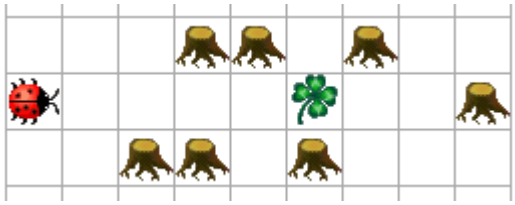
Expertin für Boole’sche Ausdrücke

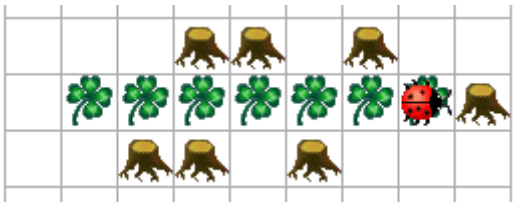
Aufgabe 1

Kara hat geradeaus vor sich einen Baum. Er möchte bis zum Baum entlang laufen und auf allen Feldern Kleeblätter deponieren.

Situation vor dem Start des Programms



Situation nachher



Das Neue

Wir brauchen eine Methode `kara.notOnLeaf()`, damit wir feststellen können, dass sich kein Kleeblatt unterhalb von Kara befindet. Diese Methode gibt es zwar nicht, aber wir haben eine Methode `kara.onLeaf()`, die uns das Gegenteil zurück gibt. Der not-Operator `!` von Java dient dazu, eine Aussage umzudrehen. Er kehrt einen wahren Wert um in einen falschen und umgekehrt.

Lösung in JavaKara

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

Bemerkung

Die Anweisung `if (!`Ausdruck`)` bedeutet: „wenn Ausdruck *nicht wahr* ist, dann mache folgendes:“

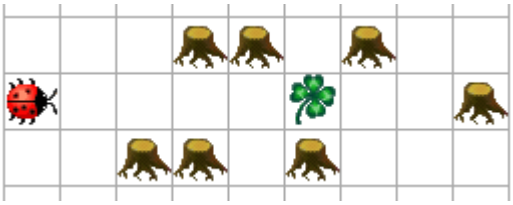
Erläuterungen

Die Methode `kara.onLeaf()` gibt einen Wahrheitswert zurück, der entweder `true` oder `false`, also wahr oder falsch, sein kann. Der `!` Operator ändert im Prinzip nur das „Vorzeichen“: Aus `true` wird `false` und umgekehrt.

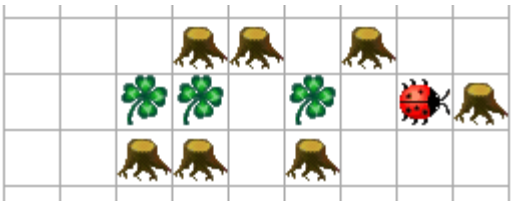
Aufgabe 2

Kara soll nun nur ein Kleeblatt legen, wenn rechts ein Baum steht.

Situation vor dem Start des Programms



Situation nachher



Das Neue

Um ein Kleeblatt zu legen, müssen nun zwei Bedingungen erfüllt sein. Immer noch soll kein Kleeblatt unterhalb von Kara liegen, zusätzlich muss aber noch rechts ein Baum stehen.

Lösung in JavaKara

```
import JavaKaraProgram;
public class LegeKleeblaetterAufWeg2 extends JavaKaraProgram
{ // Anfang von LegeKleeblaetterAufWeg2
  public void myProgram()
  { // Anfang von myProgramm
    while (!kara.treeFront())
    {
      kara.move();
      if (!kara.onLeaf() && kara.treeRight())
      {
        kara.putLeaf();
      }
    }
  } // Ende von myProgramm
} // Ende von LegeKleeblaetterAufWeg2
```

Bemerkungen

Die `&&` Operation dient dazu, zwei Bedingungen mit einem „und“ zu verknüpfen. Sowohl `!kara.onLeaf()` wie auch `kara.treeRight()` liefern `true` oder `false` zurück. Die `&&`

Operation verknüpft die beiden Bedingungen und gibt nur `true` zurück, wenn beide Bedingungen `true` sind. Alle anderen Fälle ergeben `false`.
Diese Operation wird im Fachjargon auch „logisches Und“ genannt.

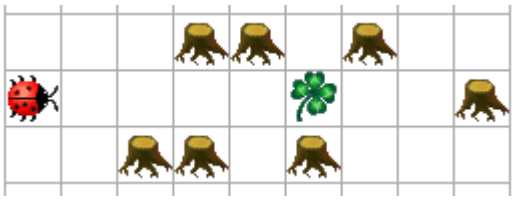
Folgendes Schema zeigt die Funktionsweise von `&&`. Wir werden ihm später wieder begegnen.

`false && false`	`false`
`false && true`	`false`
`true && false`	`false`
`true && true`	`true`

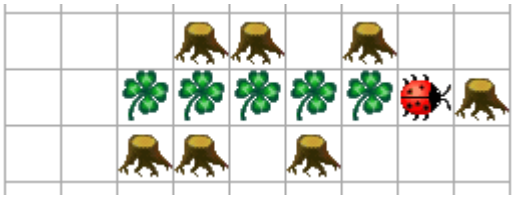
Aufgabe 3

Kara soll nun ein Kleeblatt legen, wenn links oder rechts ein Baum steht. Achtung, das bedeutet, dass auch ein Kleeblatt gelegt werden muss, falls auf beiden Seiten ein Baum steht.

Situation vor dem Start des Programms



Situation nachher



Das Neue

Um ein Kleeblatt zu legen, muss nun zusätzlich zur einen Kleeblatt-Bedingung *mindestens eine* der beiden Baum-Bedingungen `treeLeft()` und `treeRight()` zutreffen.

Lösung in JavaKara

```
import JavaKaraProgram;
public class LegeKleeblaetterAufWeg3 extends JavaKaraProgram
{ // Anfang von LegeKleeblaetterAufWeg3
  public void myProgram()
  { // Anfang von myProgramm
    while (!kara.treeFront())
    {
      kara.move();
      if ( !kara.onLeaf() &&
           ( kara.treeLeft() || kara.treeRight() ) )
      {
        kara.putLeaf();
      }
    }
```

```
  } // Ende von myProgramm
} // Ende von LegeKleeblaetterAufWeg3
```

Erläuterungen

Neu kommt ein weiterer Operator hinzu. Der `||` Operator dient dazu, zwei Bedingungen mit einem „oder“ zu verknüpfen. `||` gibt `true` zurück wenn mindestens eine der beiden Bedingungen (`kara.treeLeft()`, `kara.treeRight()`) wahr ist.
Diese Operation wird im Fachjargon „logisches Oder“ genannt.

Folgendes Schema zeigt die Funktionsweise von `||`.

`false`	`||` `false`	`false`
`false`	`||` `true`	`true`
`true`	`||` `false`	`true`
`true`	`||` `true`	`true`

Bei der Verknüpfung von mehreren Bedingungen spricht man von einem Boole’schen Ausdruck. Benannt sind die Ausdrücke nach dem englischen Mathematiker George Boole (1815-1864). Seine Theorie bildet einen Grundstein der Informatik, ist aber auch in der Mathematik von grosser Bedeutung.

Du hast gesehen, dass es teilweise notwendig ist, Boole’sche Ausdrücke mit Klammern zu ordnen. Zähle deshalb beim Schreiben von Java Programmen wenn nötig die Klammern.

Fragen

1. Wie müsstest du das dritte Programm ändern, wenn Kara nur Kleeblätter legen dürfte, wenn sich links *und* rechts ein Baumstamm befindet? Probiere deine Lösung auf dem Computer aus!

2. Das dritte Programm soll erweitert werden. Nun soll ein Kleeblatt nur gelegt werden, wenn sich entweder nur links ein Baum oder nur rechts ein Baum befindet. Wenn sich links und rechts ein Baum befindet, dann soll nichts gemacht werden.
   Probiere deine Lösung wiederum am Computer aus!
   Hinweis: Die Formulierung des Boole’schen Ausdruckes ist nicht ganz leicht. Versuche die Lösung aus Aufgabe 3 mit deiner Lösung zur vorhergehenden Frage zu kombinieren.