Aufgabe: KARA steht vor einem Baum oder nicht. Wenn nicht, soll er einfach einen Schritt weitergehen oder ansonsten um den einzelnen Baum herumgehen.

Das Neue: KARA benutzt jetzt seine Sensoren. Hier braucht er nur seinen Sensor, der ihm mitteilt, ob vor ihm ein Baum steht (`kara.treeFront()`).

```
1  import javakara.JavaKaraProgram;
2
3  public class KaraSensor extends JavaKaraProgram
4  {  // Anfang von KaraSensor1
5
6    void geheUmBaumHerum()
7    {
8      kara.turnLeft();
9      ...                  // wie vorher, Kara hat es gelernt.
10   }
11
12   public void myProgram()
13   {  // Anfang von myProgram
14     if (kara.treeFront())
15           { this.geheUmBaumHerum(); }
16     else { kara.move(); }
17
18   }  // Ende von myProgram
19
20 }  // Ende von KaraSensor
```

Erläuterungen:

1. Die Syntax der Verzweigung mit `if`:

```
if ( Bedingung )
     { Auszufuehrende Methoden, wenn ja   }
else { Auszufuehrende Methoden, wenn nein }
```

   Die Bedingung wird in `( )` eingeschlossen, die auszuführenden Methoden in `{ }`.

2. Ist jeweils nur eine Methode auszuführen, dann können die geschweiften Klammern auch weggelassen werden.

```
if (kara.treeFront())
     this.geheUmBaumHerum();
else kara.move();
```

   Es empfiehlt sich aber, im Sinne eines konsequenten und übersichtlichen Programmierstils die geschweiften Klammern immer zu setzen.

3. Der `else`-Zweig kann auch entfallen. Beispiel:

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

4. Eine verneinte Bedingung wird mit einem Ausrufezeichen gekennzeichnet:

```
if (!kara.treeFront())
     { kara.move(); }
else { this.geheUmBaumHerum(); }
```

5. Mehrere Bedingungen können durch ODER bzw. UND verknüpft werden.

Verknüpfung	Zeichen	Beispiel
ODER	`||`	`if (kara.treeFront() || kara.onLeaf()) ...`
UND	`&&`	`if (kara.treeFront() && kara.onLeaf()) ...`

   Das sieht etwas merkwürdig aus, aber man kann sich daran gewöhnen. Mir persönlich wären Schlüsselwörter wie NOT, OR und AND lieber gewesen, aber mich hat ja niemand gefragt, als man die Programmiersprache Java entwickelte.

6. Kompliziertere Bedingungen mit mehreren UNDs, ODERs und NOTs muss man durch Klammerpaare `( )` schachteln.