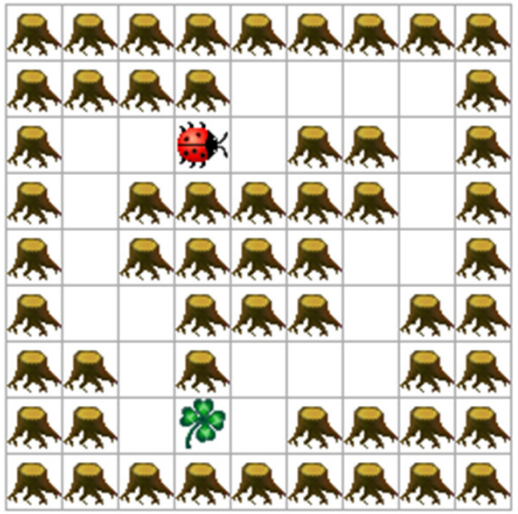
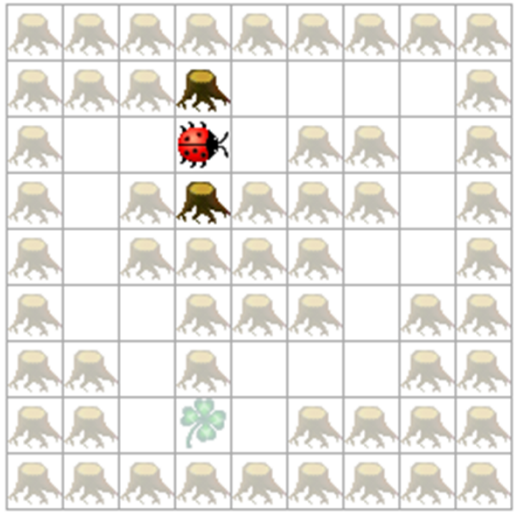
Wenn Programme Entscheidungen fällen müssen, dann …

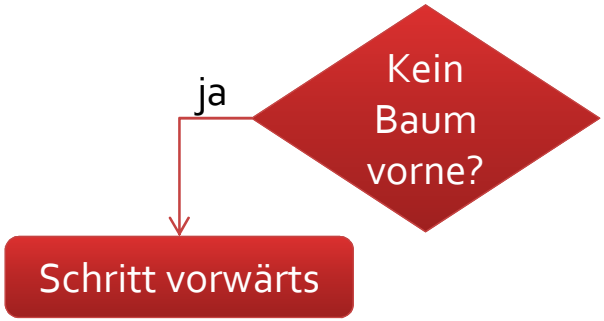
JavaKara programmieren: Verzweigungen

Kara soll durch diesen «Wald» laufen, bis er beim Kleeblatt ist

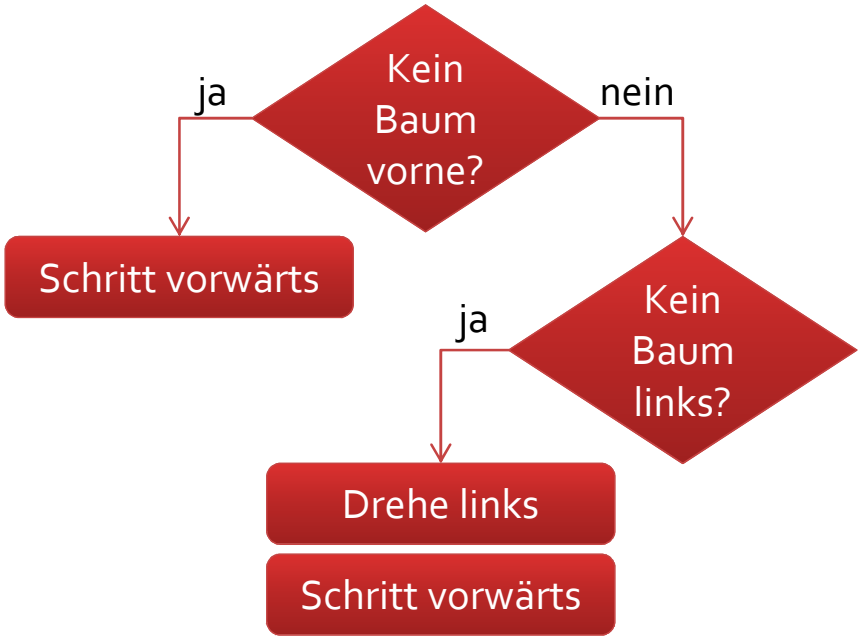


Kara soll durch diesen «Wald» laufen, bis er beim Kleeblatt ist

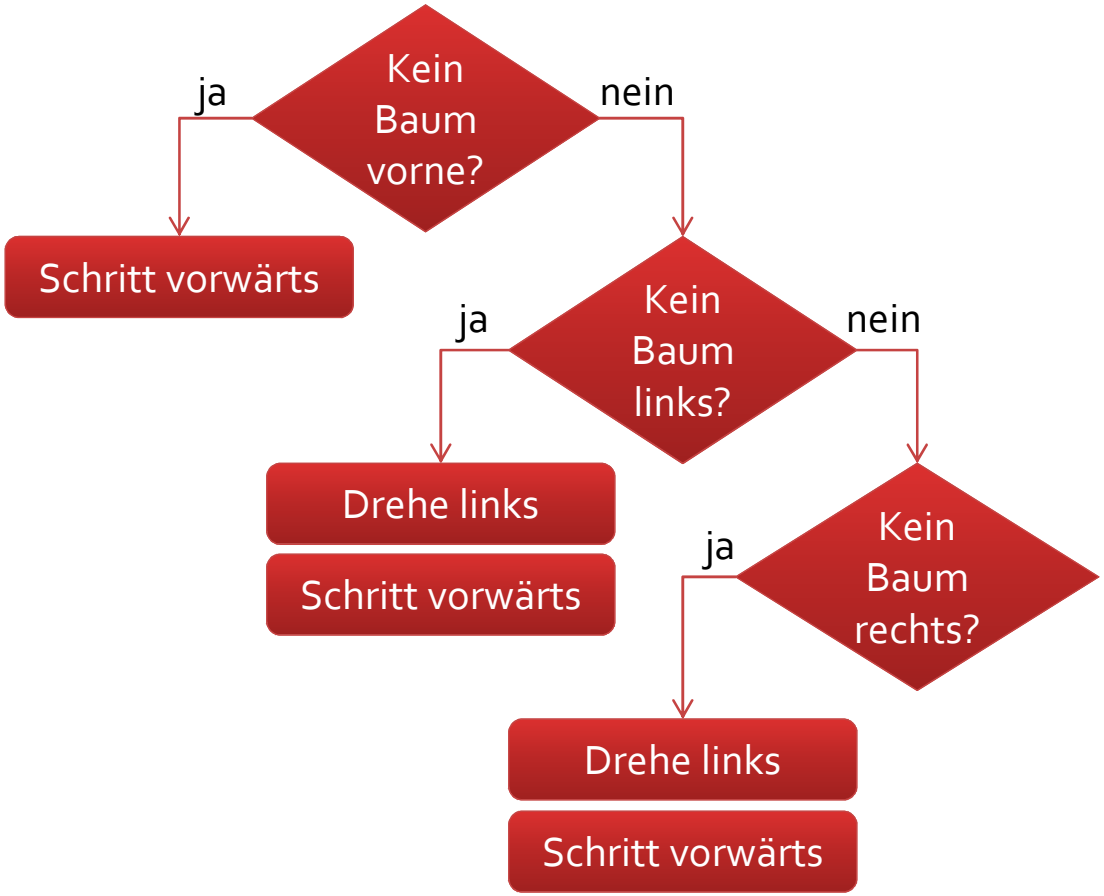




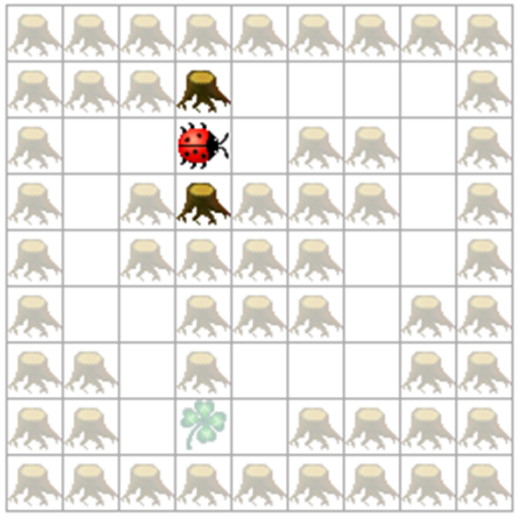
Kara soll durch diesen «Wald» laufen, bis er beim Kleeblatt ist



Kara soll durch diesen «Wald» laufen, bis er beim Kleeblatt ist



Kleeblattsuche in Java-Code



```
public void myProgram() {
  while (! kara.onLeaf()) {
    if (!kara.treeFront()) {
      kara.move();
    }
    else if (!kara.treeLeft()) {
      kara.turnLeft();
      kara.move();
    }
    else if (!kara.treeRight()) {
        kara.turnRight();
        kara.move();
    }
  }
  kara.removeLeaf();
}
```

Kleeblattsuche in Java-Code

```
public void myProgram() {
  while (! kara.onLeaf()) {
    if (!kara.treeFront()) {
      kara.move();
    }
    else if (!kara.treeLeft()) {
      kara.turnLeft();
      kara.move();
    }
    else if (!kara.treeRight()) {
        kara.turnRight();
        kara.move();
    }
  }
  kara.removeLeaf();
}
```

Kleeblattsuche in Java-Code

```
public void myProgram() {
  while (! kara.onLeaf()) {
    if (!kara.treeFront()) {
      kara.move();
    }
    else if (!kara.treeLeft()) {
      kara.turnLeft();
      kara.move();
    }
    else if (!kara.treeRight()) {
       kara.turnRight();
       kara.move();
    }
  }
  kara.removeLeaf();
}
```

Eine einzelne Verzweigung

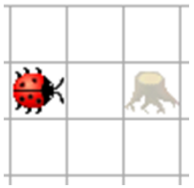
```
if («Boole'scher Ausdruck») {
  // falls wahr («true»), dann führe diese Befehle aus
}
else {
  // falls nicht wahr («false»), dann führe diese Befehle aus
}
```

Ein Boole'scher Ausdruck ist eine Formel,
deren Resultat ein Wahrheitswert ist.

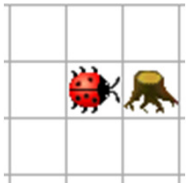
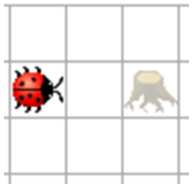
Der Aufruf von kara.treeFront() ist eine solcher Ausdruck:
Die Methode gibt true oder false zurück.

Boole'sche Ausdrücke: Ein einzelner Wahrheitswert und seine Negation

Boole'scher Ausdruck:
kara.treeFront()

«Input»	Resultat
	**false**
	**true**

Negation des Ausdrucks:
!kara.treeFront()

«Input»	Resultat
	**false**
	**true**

Boole'sche Ausdrücke: Zwei Werte verknüpfen

UND-Verknüpfung:

kara.treeLeft() && kara.treeRight()

«Input»	Resultat
	false
	false
	false
	true

ODER-Verknüpfung:

kara.treeLeft() || kara.treeRight()

«Input»	Resultat
	false
	true
	true
	true

Boole'scher Ausdruck: Beliebige Kombinationen

Operator-Präzedenz und Klammerung

kara.treeLeft() && kara.treeRight() || !kara.onLeaf()
entspricht
(kara.treeLeft() && kara.treeRight()) || !kara.onLeaf()
und nicht
kara.treeLeft() && (kara.treeRight() || !kara.onLeaf())

! «bindet» am stärksten (analog -5 in Mathematik)
&& «bindet» am zweitstärksten (analog -5*4)
|| «bindet» am drittstärksten (analog -5*4+3)
() Klammerungen ermöglichen eigene Reihenfolgen

Weitere Boole'sche Ausdrücke

Test auf Gleichheit
kara.onLeaf() == true
kürzer geschrieben als kara.onLeaf()

Test auf grösser, grösser gleich, kleiner, kleiner gleich
5 < 4 == false
5 <= 4 == false
5 <= 5 == true

5 > 4 == true
5 >= 4 == true
5 >= 5 == true

Logische Formulierungen im Alltag und in Java

Kara zum «Tunneleingang» (Feld 2a) laufen und dort stehen bleiben.

Alltagssprache: «Kara soll laufen, ***bis*** (er links einen Baum und rechts einen Baum hat)». Wir geben die **Abbruchbedingung** an: Ist das Ziel erreicht?

Java: «***Solange*** **nicht** (Kara links einen Baum und rechts einen Baum hat), soll Kara laufen.» Hier muss die **Ausführungs-bedingung** angegeben werden: Müssen die Befehle noch ausgeführt werden?

```
while (!(kara.treeLeft() && kara.treeRight())) { kara.move(); }
```

Logische Formulieungen: De Morgan'sche Gesetze

Kara zum «Tunneleingang» (Feld 2a) laufen und dort stehen bleiben.

Ausführungsbedingung «auf (mind.) einer Seite kein Baum»:
«nicht (links Baum und rechts Baum)»
!(kara.treeLeft() && kara.treeRight())

ist gleich zu
«nicht links Baum oder nicht rechts Baum»
!kara.treeLeft() || !kara.treeRight()

Allgemein formuliert ist das eines der De Morgan'schen Gesetze: **!(a && b) == !a || !b**

Logische Formulieungen: De Morgan'sche Gesetze

Kara zur «Gallerie» (Feld 1) laufen und dort stehen bleiben.

Ausführungsbedingung «auf keiner Seite ein Baum»:
«nicht (links Baum oder rechts Baum)»
!(kara.treeLeft() || kara.treeRight())

ist gleich zu «nicht links Baum und nicht rechts Baum»
!kara.treeLeft() && !kara.treeRight()

Allgemein formuliert ist das eines der De Morgan'schen Gesetze: **!(a || b) == !a && !b**