

Algorithmen

Computerprogramme sind eine Abfolge von Befehlen. Das ist so und das bleibt auch in Zukunft so. Wir erweitern dies Prinzip aber jetzt. Software muss auf unterschiedliche Situationen unterschiedlich reagieren können, damit sie flexibel ist.

Auch unser Roboter ist in der Lage Entscheidungen zu treffen. Er kann an einer Weggabelung abbiegen, oder einen Akku aufnehmen, wenn er auf dem gleichen Feld ist. Dazu benötigen wir eine bedingte Anweisung.

Aber auch diese bedingte Anweisung macht aus der Informatik noch keine Wissenschaft, erst die kreative Arbeit, Probleme zu lösen, ist der eigentliche Clou. Software-Entwicklung hat also immer das Ziel, ein Lösungsverfahren – einen Algorithmus – zu entwickeln, der das Problem löst.

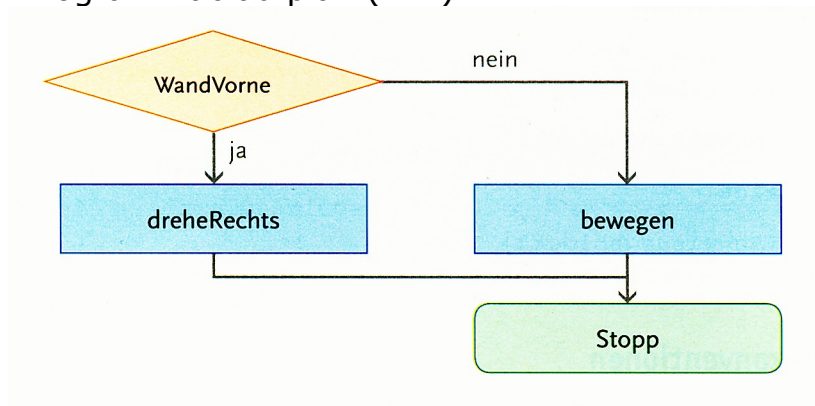
Wie arbeitet die bedingte Anweisung?

Nehmen wir mal an, Robby steht vor einer Wand. Wenn er vorwärts kommen will, muss er sich drehen, um die Wand zu umgehen. Ist vor ihm keine Wand, kann er geradeaus laufen.

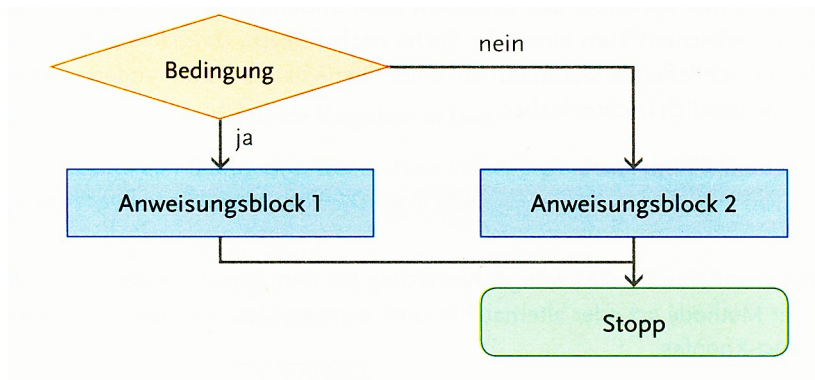
Umgangssprachlich kann man das so ausdrücken:

WENN roboterVorWandSteht
DANN roboterDrehen
SONST roboterBewegen

Darstellung im Programmablaufplan (PAP)



Allgemeiner PAP



Schreibweise in Java

```
if (wandVorne()) // Bedingung
{
    dreheRechts(); // Anweisungsblock 1 (bei true)
}
else
{
    bewegen(); // Anweisungsblock 2 (bei false)
}
```

Auf geht's mit ein paar Aufgaben.

1. Implementiere die bedingte Anweisung aus dem Beispiel in der Klasse Robby in der Methode act() oder in einer eigenen Methode. Teste mithilfe des Act-Knopfes.

2. Beschreibe genau, was folgende bedingte Anweisungen bewirken. Notiere jeweils als PAP-Diagramm. Teste in Greenfoot.

```
a) if (akkuAufFeld()) {
    akkuAufnehmen();
    rechtsDrehen();
} else {
    bewegen();
}

b) if (wandRechts()) {
    bewegen();
    dreheRechts();
    bewegen();
    dreheRechts();
    bewegen();
    bewegen();
    dreheRechts();
    bewegen();
    bewegen();
    dreheRechts();
} else {
    schraubeAblegen();
}

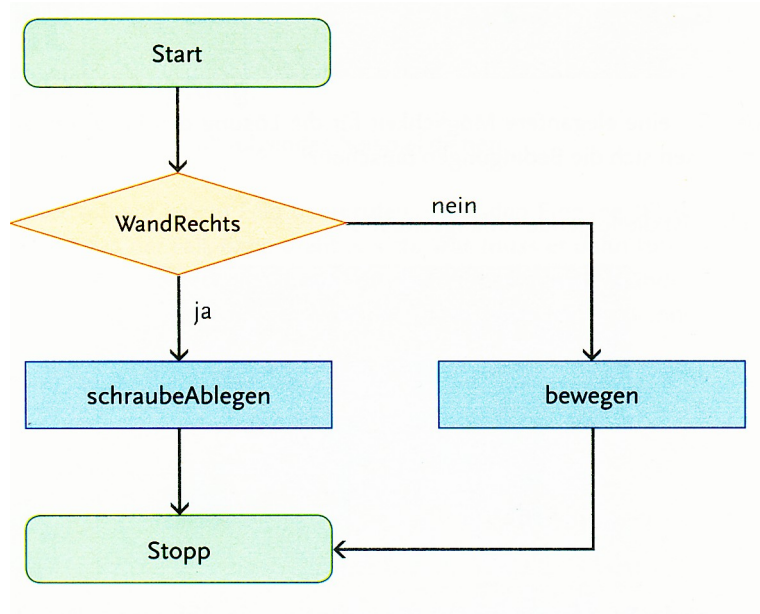
c) if (aufAkku()) {
    bewegen();
} else {
}
```

3. Programmiere die folgende Vorgabe in Java.

a) Ist rechts von Robby eine Wand, dann soll er sich rechts um 90 Grad drehen, sonst soll er sich links um 90 Grad drehen.

b) WENN RoboterRechtsWand
DANN RoboterBewegen
SONST RoboterMachtNichts

c)

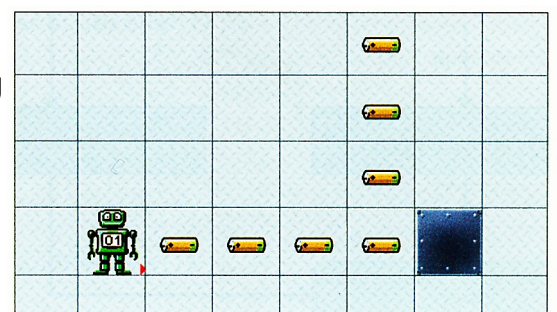


4. Bedingte Anweisungen können auch ineinander verschachtelt werden. Soll z.B. der Akku nur dann aufgesammelt werden, wenn eine Mauer links neben ihm liegt, so bietet sich folgende Lösung an:

Anfangssituation:	Endsituation:
<pre> Wenn wandLinksIst Dann Wenn akkuAufFeldIst Dann nehmeAkku macheSchritt Sonst macheSchritt Sonst macheSchritt </pre>	<pre> public void act() { if(wandLinks()) { if(akkuAufFeld()) { akkuAufnehmen(); bewegen(); }else{ bewegen(); } }else{ bewegen(); } } </pre>

5. Gib die Änderungen im Quellcode an, wenn der Roboter die Akkus nur dann aufsammeln soll wenn keine Mauer neben ihm ist. Gib eine elegantere Möglichkeit für die Lösung an – vielleicht lassen sich die Bedingungen tauschen?

6. Folgende Situation:
 Robby soll der Spur folgen, die beliebig lang sein kann, Abbiegen soll er allerdings nur durch Linksknicks jeweils vor einer Mauer. Entwickle zuerst einen PAP, anschließend ein Java-Programm.



7. Finde die Fehler im folgenden Java-Code:

```
a) if (wandVorne ())
    {
        bewegen ();
        dreheRechts ();
        dreheRechts ();
        bewegen ();

    }
else
    {
        schraubeAblegen;
    }
```

- b) Implementiere selbst einen kurzen Algorithmus, der fünf Fehler enthält. Tausche den Algorithmus mit dem deines Nachbarn und finde die Fehler.