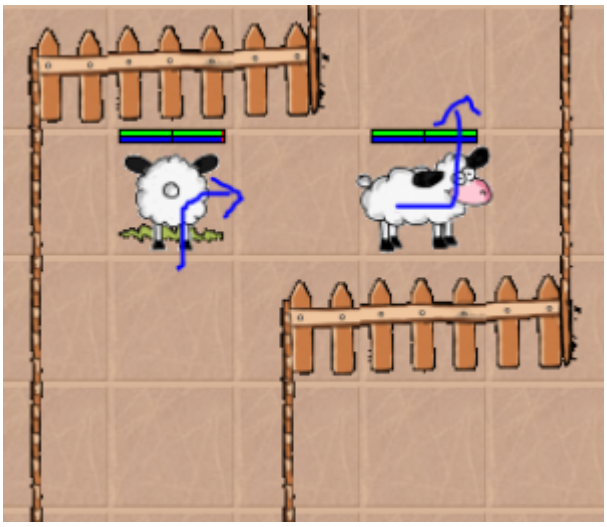
AB04 - Verzweigungen: Die Figuren treffen Entscheidungen...

Bisher war es einfach. Alles Grüne konnte man fressen. Es gibt aber auch besondere Leckerbissen - Blümchen. Aber aufgepasst: Die blauen sind giftig! Da muss sich ein Schaf schon überlegen, ob es die einfach so fressen möchte.

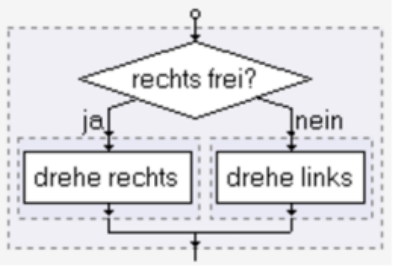
> **Ziel:** Alternativen in Handlungen erkennen, als FALLS-DANN-SONST-Entscheidungen formulieren und in Programmiersprache umsetzen können.

In vielen Situationen sind Anweisungen nur unter bestimmten Bedingungen auszuführen. Folgt ein Schaf einem Gang und trifft auf einen Zaun vor ihm, muss es entscheiden, ob es sich nach links oder rechts drehen soll. Dazu muss es prüfen, ob rechts frei ist. Wenn nicht, muss es links weitergehen.

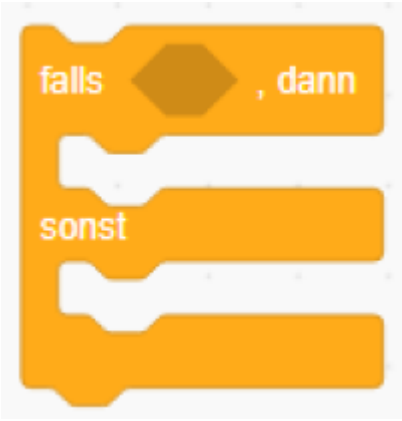


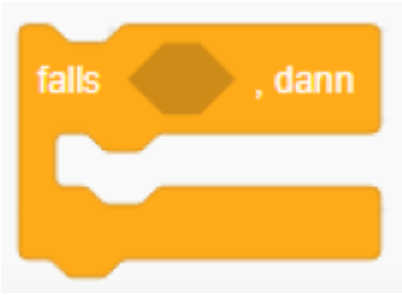
Derartige Entscheidungen trifft man anhand von Prüfbedingungen, die wie bei den while-Schleifen wahr oder falsch sein können.

-- *normierte Sprache*	-- *Programmiersprache*	

FALLS (rechts frei ist)
DANN
  drehe nach rechts
*ENDE DANN
SONST
  drehe nach links
*ENDE SONST
	`if(istRechtsFrei())`
	`{`
	`  dreheRechts();`
	`}`
	`else {`
	`  dreheLinks();`
	`}`
		

Vorsicht:
statt ~~`if (Prüfbedingung) then {…}`~~
steht nur `if (Prüfbedingung) {…}`





Man nennt solche Entscheidungen in der Programmierung auch Verzweigungen. Du kennst sie schon von Scratch (siehe rechts).

Genauso wie bei Scratch, kann auch in Java der else-Teil weggelassen werden.

(A1) Drehe richtigrum

Implementiere die Methode `dreheRichtigrum()`, die das Schaf nach rechts drehen lässt, wenn dort frei ist. Andernfalls soll es sich nach links drehen.

Teste die Methode an dem Schaf links unten und am Schaf vor dem Teich.

(A2) Gerade aus oder drehen

Implementiere eine Methode, die das Schaf einen Schritt nach vorne machen lässt, wenn dies möglich ist. Andernfalls soll es sich in die freie Richtung drehen.

Hinweis: Du kannst entweder zwei Verzweigungen ineinander verschachteln oder die Methode von Aufgabe 1 nutzen.

(A3) Zurück in den Stall

Die Schafe sollen gemäß den Regeln von Aufgabe 2 laufen, bis sie im Stall ankommen. Sie sind solange nicht im Stall, wie die x-Koordinate nicht 7 oder die y-Koordinate nicht 3 ist. Teste diese Methode an allen vier Schafen. Eines kommt nicht an. Warum?

Hinweis: Zusammengesetzte Bedingungen kann man mit && (= und) bzw. || (=oder) formulieren.

(A4) Zurück in den Stall 2

Ergänze die Methode von Aufgabe 3 so, dass die Schafe alles Gras fressen, was unterwegs wächst.

(A5) Lecker Blümchen

Implementiere eine Methode, die das Schaf bis zum nächsten Zaun laufen lässt und dabei alle leckeren Blümchen unterwegs frisst.

Achtung: die blauen sind giftig.
Hinweis: Rufe `pruefe("Blume")` direkt am Schaf auf, um herauszufinden, wie man giftige Blumen erkennen kann..

(A6) Lecker Blümchen

```
if (istAuf("Gras") {
   fresse();
}
if (istVorneFrei());{
  einsVor();
}
```

Korrigiere die beiden Schreibfehler! Dieser Quelltext wird so nicht übersetzt, sondern mit einer Fehlermeldung zurückgewiesen. Der zweite Fehler führt zwar nicht zu einer Fehlermeldung, ist daher aber noch schwieriger zu finden.

(A7)

```
if (!istVorneFrei()) {
    dreheLinks();
    einsVor();
    dreheRechts();
}
else {
    einsVor();
}
dreheRechts();
```

Zeichne für jedes der vier AB4-Schafe im Bild rechts ein, wie sie sich bewegen, wenn sie diese Anweisungen ausführen:

(A8)

Gib an, welche Ausdrücke von A bis E **nicht** als Prüfbedingung in einer Verzweigungsanweisung `if (…)` oder in einer `while`-Schleife benutzt werden können.

a) `(istVorneFrei())`	b) `(getHunger > 5)`	c) `(getX())`
d) `(istAuf("Gras") && (getHunger() > 0))`	e) `(einsVor())`	

Leveltest 4: Durchs Labyrinth

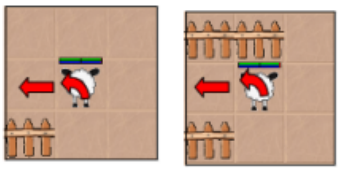
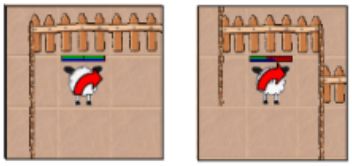
Aufgabe

Das eine Schaf kommt mit der Strategie von den Aufgaben 1-3 nicht in den Stall, sondern läuft immer im Kreis. Eine Strategie den Ausgang in einem Irrgarten zu finden, ist die linke Hand an eine Wand zu legen und dann immer an dieser Wand so entlang zu laufen, dass die Hand an der Wand bleibt. Ist der Irrgarten so gebaut, dass man nicht im Kreis laufen kann, so findet man auf diese Weise garantiert den Ausgang. Man kann dabei in die folgenden drei Situationen gelangen. Überlege dir, wie man erkennen kann, in welcher der drei Situationen das Schaf ist. Es gibt neben `istVorneFrei()` auch die Methoden

`istRechtsFrei()` und `istLinksFrei()`.

Hinweis 1: Am Anfang muss man natürlich erst mal noch einen Zaun finden, also so lange nach vorne laufen, bis man einen gefunden hat und sich dann so drehen, dass er auf der linken Seite ist.

Hinweis 2: Und unterwegs sollte man genügend fressen und trinken.

<table>
<tr><td>links ist frei</td><td colspan="2">links nicht frei</td></tr>
<tr><td></td><td>vorne ist frei</td><td>vorne ist nicht frei</td></tr>
<tr><td>  </td><td> </td><td>  </td></tr>
<tr><td>Man hat die Wand verloren, da sie geendet hat. Dann biegt man links ab und geht eins vor. Damit steht man wieder neben der Wand.</td><td colspan="2">Wenn aktuell links eine Wand ist, geht man entweder nach vorne oder muss rechts abbiegen, weil vorne kein Platz ist. Beachte, dass das Schaf nach dem Rechts-Abbiegen keinen Schritt gehen darf, da es sich manchmal gleich nochmal drehen muss</td></tr>
</table>

Zusammenfassung

Du kannst Verzweigungen in Algorithmen nutzen, im Quelltext erkennen und formulieren. Du kennst die Schreibweise dieser Entscheidungs-Anweisung mit dem Schlüsselwort `if` und der Prüfbedingung im runden Klammerpaar dahinter.

```
if ( Prüfbedingung ) {
  // DANN-Teil
  // Anweisungen, die ausgeführt werden
  // falls die Prüfbedingung stimmt.
}
else {
  // SONST-Teil
  // Anweisungen, die ausgeführt werden
  // falls die Prüfbedingung NICHT stimmt.
}
```

Manchmal ist im SONST-Fall nichts zu tun. Dann entfällt der Teil ab `else`. Du kannst die Antworten der Ja/Nein-Abfragen wie `istVorneFrei()` als Prüfbedingung in einer Entscheidung nutzen, auch in ihrer negierten Form wie bei: `if (!istVorneFrei()) {…}`. Dies wird gelesen als: „Falls NICHT vorne frei ist…“ oder „Falls vorne frei falsch ist…“ oder „Falls vorne nicht frei ist…“. Die Verneinung NICHT wird durch das Ausrufezeichen `!` geschrieben. Die Begriffe Verzweigung, Entscheidungsanweisung und auch Alternative werden gleichwertig genutzt.

<<< Zurück zu Level 3 **AB04** Weiter zu Level 5 >>>

From:
https://www.info-bw.de/ -

Permanent link:
https://www.info-bw.de/faecher:informatik:mittelstufe:bauernhof:ab4:start?rev=1718186073

Last update: **12.06.2024 09:54**