Auftrag AB4

Bisher hatten wir Glück. Die Mitarbeiter konnten die Wege recht genau beschreiben. Der Roboter wusste immer, was zu tun ist. So ist das aber nicht immer. Manche Türen gehen von allein auf. Andere müssen mit einem Schlüssel aufgeschlossen werden. Oder es ist gar eine Strombarriere im Weg, die durch einen Schalter deaktiviert werden muss. Gänge knicken unvorhergesehen nach links oder rechts ab. Da muss der Roboter selbstständig **Entscheidungen** treffen können.

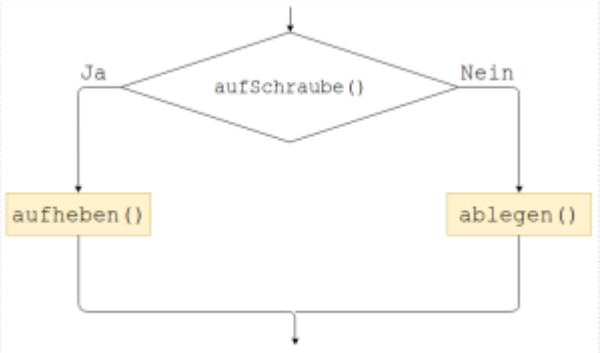
Die Roboter treffen Entscheidungen ...

Ziel

Alternative Ablaufmöglichkeiten erkennen, als **WENN-DANN-SONST**-Entscheidungen formulieren und in Programmiersprache umsetzen können.

Hintergrund: Verzweigungen

In vielen Situationen sind Anweisungen nur unter bestimmten Bedingungen auszuführen. Um z.B. ein Muster zu invertierten, muss ein Roboter dort, wo eine Schraube liegt, diese aufheben und dort, wo keine liegt, eine ablegen. Dazu muss er prüfen, ob eine Schraube da liegt oder nicht und jeweils passend handeln. Die Prüfung erfolgt in einer **Prüfbedingung**, die nur **wahr** oder **falsch** sein kann.

Normierte Sprache	Programmiersprache	Flussdiagramm
`WENN ( Schraube ist darunter ) DANN aufheben *ENDE DANN SONST ablegen *ENDE SONST`	`if (istAufGegenstand("Schraube")) { aufnehmen(); } else { ablegen("Schraube"); }`	

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

Man nennt solche Entscheidungen in der Programmierung auch **Verzweigungen**.

Aufgaben

Aufgabe 1: Invertierer

Der Roboter links unten steht in einer Reihe mit vielen Schrauben. (Zur Sicherheit üben wir erst mal wieder mit Schrauben, wer weiß, was der Roboter anstellt…). Analysiere die beiden Methoden `tausche()` und `tauscheUndVor()`: Wie verhält sich der Roboter, wenn er auf einer Schraube steht? Wie wenn er auf einem leeren Feld steht? Wozu ist in `tauscheUndVor()` die Entscheidung `if(istVorneFrei())…` verwendet worden? Wie unterscheidet sich die Form der Entscheidung in den beiden Methoden.

Aufgabe 2: Wiederholung der while-Schleife

Jedes Mal, wenn du auf „Reset“ drückst, entsteht eine neue Schraubenreihe. Implementiere eine Methode `tauscheBisWand()`, die in einer `while`-Schleife so lange `tauscheUndVor` aufruft, wie die Wand noch nicht erreicht ist. Versichere dich, dass auch direkt vor der Wand getauscht wird.

Aufgabe 3: Fleckenfrei

Ölflecken schaden den Robotern. Sie verlieren die Haftung auf dem Boden. Dadurch verlieren sie Energie. Der Roboter soll um die Ölflecken herum laufen. Mit `istVorne("Oelfleck")` kann er überprüfen, ob vor ihm ein Ölfleck ist. Implementiere eine Methode `umgeheOelfleck()`, die den Roboter einen Schritt nach vorne gehen lässt, wenn kein Ölfleck vor ihm ist. Ansonsten läuft er um den Ölfleck drumherum. Erweitere diese Methode so, dass der Roboter bis zur Wand läuft.

Aufgabe 4: Schlüssel aufheben

Schreibe eine Methode `sammleSchluessel()`, die einen Schlüssel aufnimmt, wenn der Roboter auf einem steht. Falls nicht, soll er nichts machen. Teste diese Methode an den beiden Robotern oben links.

Aufgabe 5: Sesam öffne dich

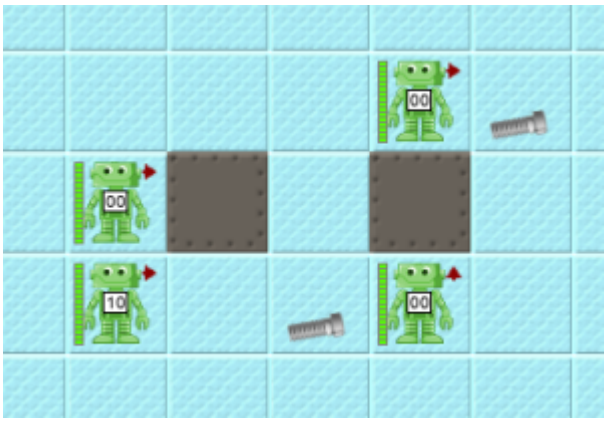
Implementiere eine Methode `oeffneTuer()`, die die beiden Roboter oben links aus ihren Gängen heraus führt. Der Roboter soll einen Schlüssel aufheben, wenn er an der Startposition liegt. Danach soll er drei Schritte vor gehen, und sich nach links drehen. Wenn er vor einem Schloss steht (`istVorne("Schloss")`), soll er den Schlüssel benutzen (`benutze("Schluessel")`), andernfalls den Schalter benutzen (`benutze("Schalter")`). Dann sollte sich die Tür bzw. die Elektrobarriere öffnen. Anschließend soll der Roboter zwei Schritte vor gehen, um hinaus zu treten.

Aufgabe 6: Fehlersuche

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.

```
if (aufSchraube() {
   schraubeAufnehmen();
}
if (istVorneFrei()); {
  einsVor();
}
```

Aufgabe 7



Zeichne für jeden der vier AB4 ein, wie sie sich bewegen, wenn sie die folgenden Anweisungen ausführen:

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

Aufgabe 8

a) Nenne zwei Bewegungsbefehle, die ein Roboter ausführen kann und zwei Fragebefehle, die er beantworten kann.

b) Gib an, welcher Ausdruck A bis E nicht als Prüfbedingung in einer Entscheidungsanweisung if (…) benutzt werden kann.

```
* A: (istVorneFrei())
* B: (x>5)
* C: (alter>=18)
* D: (anzBlaetter<=7)
* E: (einsVor())
```

Aufgabe 9: Labyrinth 1

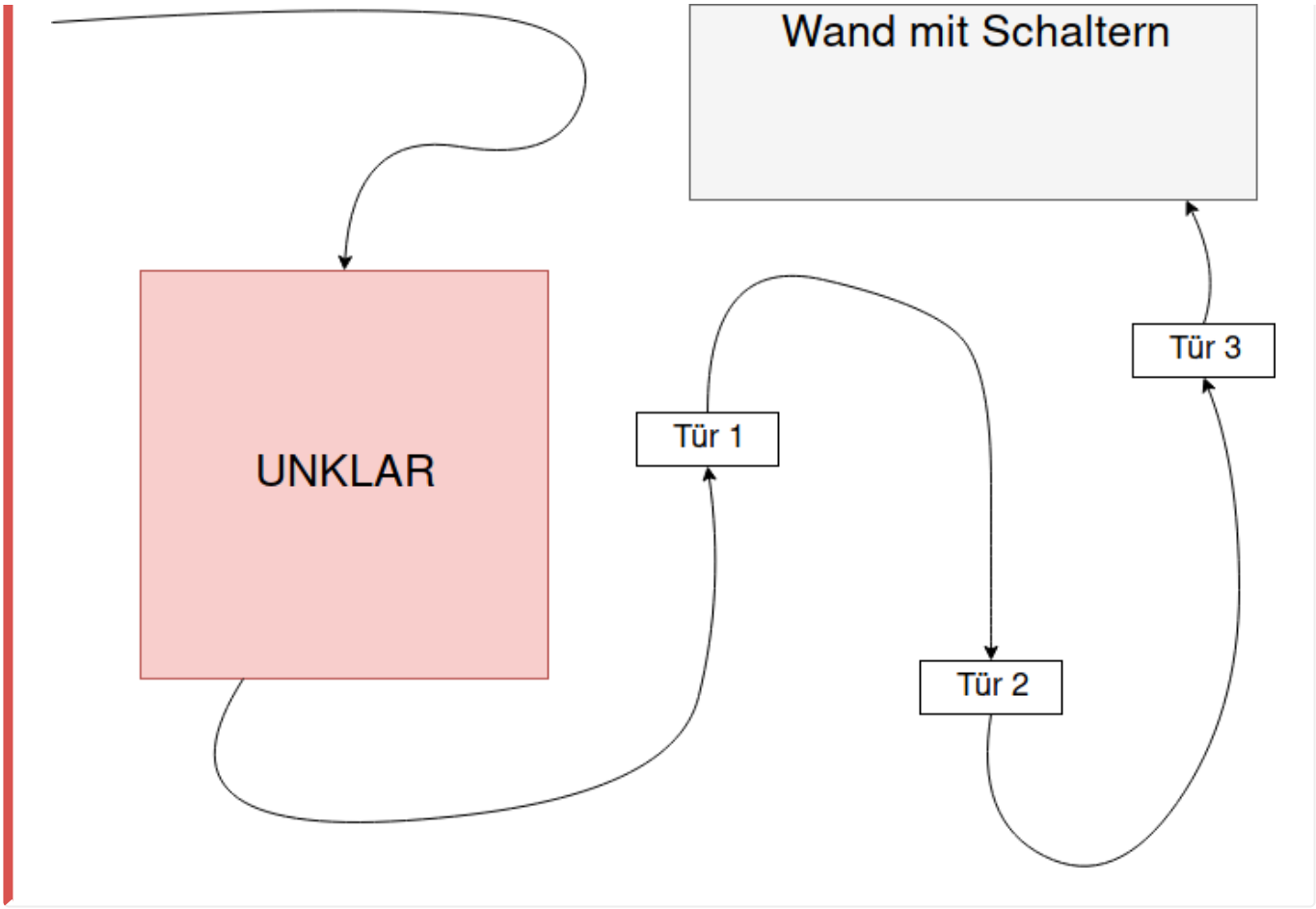
Der Roboter in der Mitte soll sich in einem Gang alleine zurecht finden. Der Gang hat keine Verzweigungen, knickt aber immer wieder nach links und nach rechts ab. Implementiere eine Methode, die den Roboter bis zur Sackgasse laufen lässt:

- Stelle den Roboter zunächst vor eine Wand. Sorge dafür, dass er sich nach links dreht, wenn links frei ist, sonst nach rechts.
- Erweitere deine Methode, indem du den Roboter an einer beliebigen Stelle vor der Wand starten lässt und dann die Methode `laufeBisWand()` benutzt.
- Lasse diese beiden Schritte solange wiederholen, wie die Sackgasse noch nicht erreicht ist.

Um eine Sackgasse zu erkennen, benötigst du komplexere Bedingungen. Du kannst mehrere Bedingungen mit "und" (in Java &&) bzw. "oder" (in Java `||`) verbinden. So kannst du beispielsweise mit `istWandVorne() && istWandLinks() && istWandRechts()` überprüfen, ob dein Roboter in einer Sackgasse steht. Das Gegenteil – also "keine Sackgasse – erreicht man, indem man die ganze Bedingung einklammert und ein "nicht" (`!`) davor setzt.

Einsatz 4: Gelangen Sie in den Kontrollraum und schalten Sie das Kernkraftwerk ab

> Die Mannschaft hat das Kernkraftwerk fluchtartig verlassen und leider die elementaren Sicherheitsvorkehrungen vernachlässigt. Sie haben vergessen, das Kernkraftwerk herunterzufahren.
>
> Dazu müssen im Kontrollraum alle Schalter umgelegt werden. Der Kontrollraum befindet sich irgendwo versteckt am Ende eines langen Ganges, der durch verschiedene Türen gesichert ist. Jede Tür ist durch einen Schalter oder ein Schloss direkt links neben der Tür zu bedienen. Hier haben Sie die nötigen Schlüssel.
>
> Im Kontrollraum gibt es eine Wand mit Schaltern. Wie viele es sind und wo genau sie sich befinden, weiß ich nicht. Wenn sie alle aus sind, haben Sie Ihren Auftrag erfüllt.



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.
- 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: Wenn NICHT vorne frei ist … oder Wenn Vorne frei falsch ist … oder Wenn 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 | **Level 4** | Weiter zu Level 5 >>>
