AB 4 – FALLUNTERSCHEIDUNGEN (IF-ELSE)

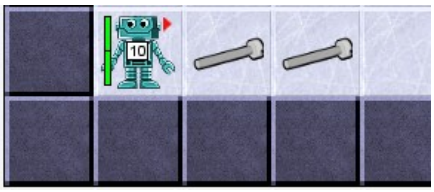
Einlasskontrollen: Bisher hatten Sie Glück. Die Mitarbeiter konnten die Wege recht genau beschreiben. Der Roboter wusste immer, was zu tun ist. So ist das aber nicht immer. Manche Türen gehen von allein auf. Andere müssen mit einem Schlüssel aufgeschlossen werden. Oder es ist gar eine Strombarriere im 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: *Alternativen in Handlungen erkennen, als FALLS-DANN-SONST-Entscheidungen formulieren und in Programmiersprache umsetzen können.*

Aufgaben:

1. **Invertierer:** Der Roboter links unten steht in einer Reihe mit vielen Schrauben. Analysiere die beiden Methoden ***tausche()*** und ***tauscheUndVor()***:



   a) Wie verhält sich der Roboter, falls er auf einer Schraube steht?
   b) Wie verhält er sich, falls er auf einem leeren Feld steht?
   c) Wozu ist in ***tauscheUndVor()*** die Entscheidung **if(istVorneFrei()){...}** verwendet worden?
   d) Wie unterscheidet sich die Art der Fallunterscheidung in den beiden Methoden?

2. **Wiederholung der while-Schleife**: Jedes Mal, wenn du auf „Reset“ drückst, entsteht eine neue Schraubenreihe. Implementiere eine Methode ***tauscheBisWand()***, die wiederholend ***tauscheUndVor()*** aufruft, so lang die Wand noch erreicht ist. Versichere dich, dass auch direkt vor der Wand getauscht wird.

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, falls kein Ölfleck vor ihm ist. Ansonsten läuft er um den Ölfleck drumherum.
   Erweitere diese Methode so, dass der Roboter bis zur Wand läuft.

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

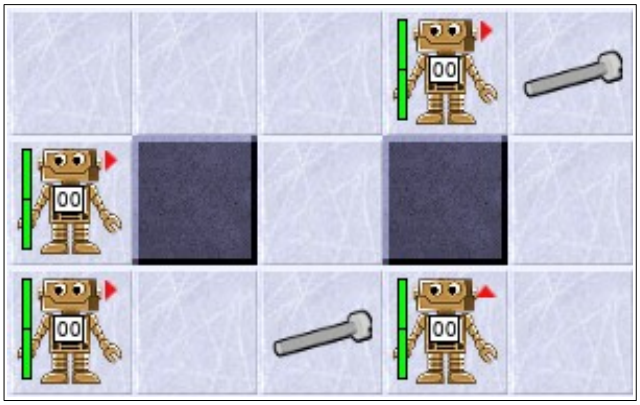
5. **Sesam öffne dich:** Implementiere eine Methode ***oeffneTuer()***, die die beiden Roboter oben links aus ihren Gängen heraus führt. Der Roboter soll einen Schlüssel aufheben, falls er an der Startposition liegt. Danach soll er drei Schritte vor gehen, und sich nach links drehen. Falls er vor einem Schloss steht (i***stVorne("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.

6. Korrigiere die beiden Schreibfehler! Dieser Quelltext wird so nicht übersetzt, sondern mit einer Fehlermeldung zurückgewiesen.

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

7. Zeichne für jeden der vier ***AB4***-Roboter auf Papier ein, wie sie sich bewegen und in welche Richtung sie am Ende schauen, wenn sie diese Anweisungen ausführen:

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



8. **a)** Nenne zwei Bewegungsbefehle, die ein Roboter ausführen kann und zwei Fragebefehle, die er beantworten kann.
   b) Gib an, welche Ausdrücke von *A* bis *E* nicht als Prüfbedingung in einer Entscheidungsanweisung **if (...)** benutzt werden kann. Dabei sind x, alter und anzBlaetter Variablen, die für Zahlen stehen.

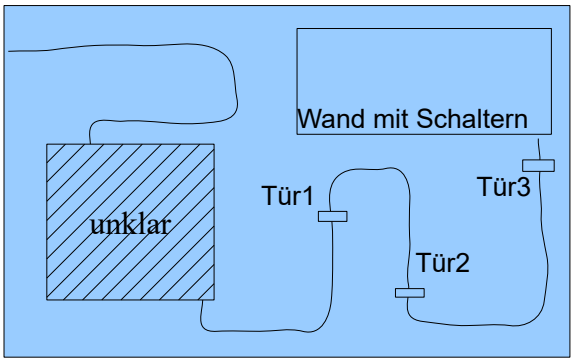
   A: if(!istVorneFrei()) B: if(x>5) C: if(alter)
   D: if(anzBlaetter<=7) E: if(einsVor())

9. **Labyrinth:** Der Roboter in der Mitte soll sich in einem Gang alleine zurecht finden. Der Gang hat keine Verzweigungen, knickt aber immer wieder nach links oder nach rechts ab. Implementiere eine Methode, die den Roboter bis zur Sackgasse laufen lässt:
   - Laufe mit dem Roboter bis vor eine Wand. Sorge dafür, dass er sich nach links dreht, falls links frei ist. Falls rechts frei ist, nach rechts. Sonst soll er einfach nichts tun/stehen bleiben.
   - Lasse diese Schritte solange wiederholen, wie die Sackgasse noch nicht erreicht ist.

   Um eine Sackgasse zu erkennen, benötigst du eine komplexere Bedingung.
   Eine Sackgasse ist erreicht, wenn **vorne** und **links** und **rechts** vom Roboter eine Wand ist. 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 drei 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 an der Wand links neben dem Eingang zahlreiche Schaltern. Wie viele es sind und wo genau sie sich befinden, weiß ich nicht. Schalten Sie alle aus, dann haben Sie Ihren Auftrag erfüllt.



Tipps:

- Der Weg zu Tür 1 sieh immer anders aus. Nutze dafür die Labyrinth-Methode.

- Es sind immer genau 3 Türe zu öffnen, aber sie gehe immer unterschiedlich per Schalter oder Schlüssel auf. Den Schlüssel hat der Roboter von Anfang an bei sich.

- Im Kontrollraum sind immer unterschiedlich viele Schalter. Drücke alle Schalter (sofern vorhanden) bis du an der Wand stehst.

Zusammenfassung: Du kannst Verzweigungen in Algorithmen nutzen, im Quelltext erkennen und formulieren. Du kennst die Schreibweise dieser Fallunterscheidung mit **if(...){...}**.

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 Fallunterscheidung, Verzweigung, Entscheidungsanweisung und auch Alternative werden gleichwertig genutzt.

Bildquellen: Die verwendeten Bilder des Roboterszenarios sind alle ohne Bildnachweis verwendbar (selbst gezeichnet, Pixabay Lizenz oder Public Domain). Genaue Nachweise: siehe *bildquellen.html*.