# Auftrag AB5

Puh, das war knapp! Gut, dass die Roboter schon recht selbständig Ihren Weg finden. Der kommende Einsatz wird ein wenig „anstrengender“. Ein Upgrade wird benötigt …

# Neue Sensoren …

**Ziel:** Methoden mit boolschen Rückgabewerten erstellen können.

Unsere Roboter haben nur eine beschränkte Anzahl von Methoden, die die Umwelt des Roboters wahrnehmen (z.B. `istVorneFrei()`, `istVorne("Schraube“))`. In diesem Arbeitsblatt lernst du neue (komplexere) Sensoren selbst zu erstellen.

Die Roboter aus dem ersten Übungsblatt hatten Sensoren wie `aufSchraube()`, die die darauffolgenden Roboter nicht mehr hatten, da man das Gleiche auch mit `aufGegenstand("Schraube")` erreichen kann. Trotzdem kann es sinnvoll sein, aus den vorhandenen Sensoren neue zu erschaffen. Schaue dir dazu die Methode `istFassVorne()` des AB5-Roboters an. Im Gegensatz zu den Methoden, die du bisher implementiert hast, gibt diese eine Antwort zurück.

# Aufgaben

### Aufgabe 1

Teste die Methode `istFassVorne()` in Greenfoot. Wie beantwortet ein Roboter diese Anfrage? Welche Antwortmöglichkeiten gibt es?

### Aufgabe 2

Analysiere nun den dazugehörigen Quelltext: Wo tauchen diese Antwortmöglichkeiten im Quelltext auf? Welcher Befehl sorgt dafür, dass eine Antwort zurückgegeben wird?

## Methoden mit Rückgabewert

Alle bisherigen Methoden wurden mit `public void methodenName() {…}` programmiert. Void steht dabei für "leer/nichts". Void steht an der Stelle, an der der sogenannte Rückgabetyp steht. Es wird also nichts zurückgegeben. Möchte man nun mit true/false antworten, muss man den Rückgabewert als `boolean` deklarieren:

```
public boolean istFassVorne() {
```

```
        if (istVorne("Fass")) {
            return true;
        } else {
            return false;
        }
    }
```

Mit return true/false; kann man dann den gewünschten Wert zurückgeben. `return` beendet außerdem das Unterprogramm. Es werden also keine Befehle mehr nach dem `return` ausgeführt.

## Aufgabe 3

Füge die Anweisung `dreheUm();` in die Methode `istFassVorne()` ein:

a) Vor der if-Anweisung.
b) Jeweils vor dem Return.
c) Nach der if-else-Anweisung / vor der letzten Klammer.

## Aufgabe 4: Rückspiegel

Vervollständige die Methode `istFassHinten()`. Diese soll testen, ob hinter dem Roboter ein Fass steht. Dazu muss er sich natürlich kurzfristig umdrehen. Am Ende soll er aber wieder so stehen, wie am Anfang.

## Aufgabe 5: Out of Power

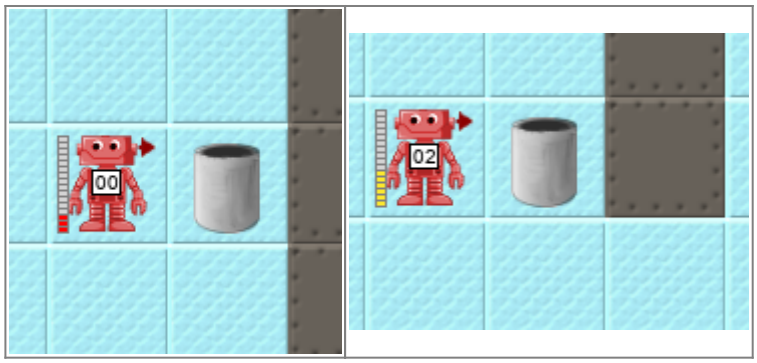
Implementiere eine Methode `istEnergieSchwach()`. Diese soll `true` zurückgeben, wenn der Roboter nur noch weniger als 40 Energiepunkte hat (Überprüfung mit `getEnergie()<40`). Ansonsten gibt sie `false` zurück. Teste deine Methode an den drei Robotern im Raum unten links (nur der unterste hat einen schwachen Ladezustand).

## Aufgabe 6: Heavy Duty

Implementiere eine Methode `istSchwerBeladen()`, die testet, ob der Roboter fünf oder mehr Gegenstände herumträgt (benutze `getAnzahl()`). Teste deine Methode an verschiedenen Robotern.

## Aufgabe 7: Look Ahead

Implementiere eine Methode `istVorFassFrei()`. Diese soll `false` zurückgeben, wenn vor dem Fass eine Wand ist. Dabei können die folgenden beiden Fälle auftreten:



Der Roboter muss rechts um das Fass herumlaufen (dort ist immer Platz) und nachschauen, ob vor dem Fass frei ist. Wenn ja, gibt er `true` zurück, sonst `false`. Trifft er schon vorher auf die Wand (Fall 1), dann gibt er auch `false` zurück. In allen Fällen ist er am Ende wieder an seinen Ausgangspunkt.

### Aufgabe 8: Aufräumen

Implementiere eine Methode `schiebeFassBisWand()`, die ein Fass bis zur nächsten Wand schiebt. Teste deine Methode an dem Roboter rechts oben, nachdem du zunächst mit der Maus das Fass vor der Wand aus dem Weg geschoben hast.

## Logische Verknüpfungen

Geschickt ist es auch, wenn man zwei Sensoren A und B miteinander verbindet. In Java gibt es dazu die Operatoren

```
A || B  →  A oder B
A && B  →  A und B
!(A)    →  nicht A
```

Man kann also beispielsweise mit `(!(istWandVorne()) && !(istWandLinks()) && !(istWandRechts()))` testen, ob in allen drei Richtung keine Wand ist.

### Aufgabe 9: Führerschein

Implementiere eine Methode `istKreuzung()`, die testet, ob der Roboter auf einer Kreuzung steht, d.h. ob vorne, links und rechts frei ist. Implementiere eine Methode `geheBisKreuzung()` und teste sie am Roboter oben links.

### Aufgabe 10: Upgrade 1

Erweitere die Methode `istFassVorne()` so, dass nicht nur die Übungsfässer erkannt werden, sondern auch Atommüllfässer. (Benutze `istVorne("Atommuell")`.) Überlege dir, welche Auswirkungen diese Änderungen für die Methode `istVorFassFrei()` haben. Funktioniert diese Methode jetzt auch mit Atommüllfässern?

### Aufgabe 11: Upgrade 2

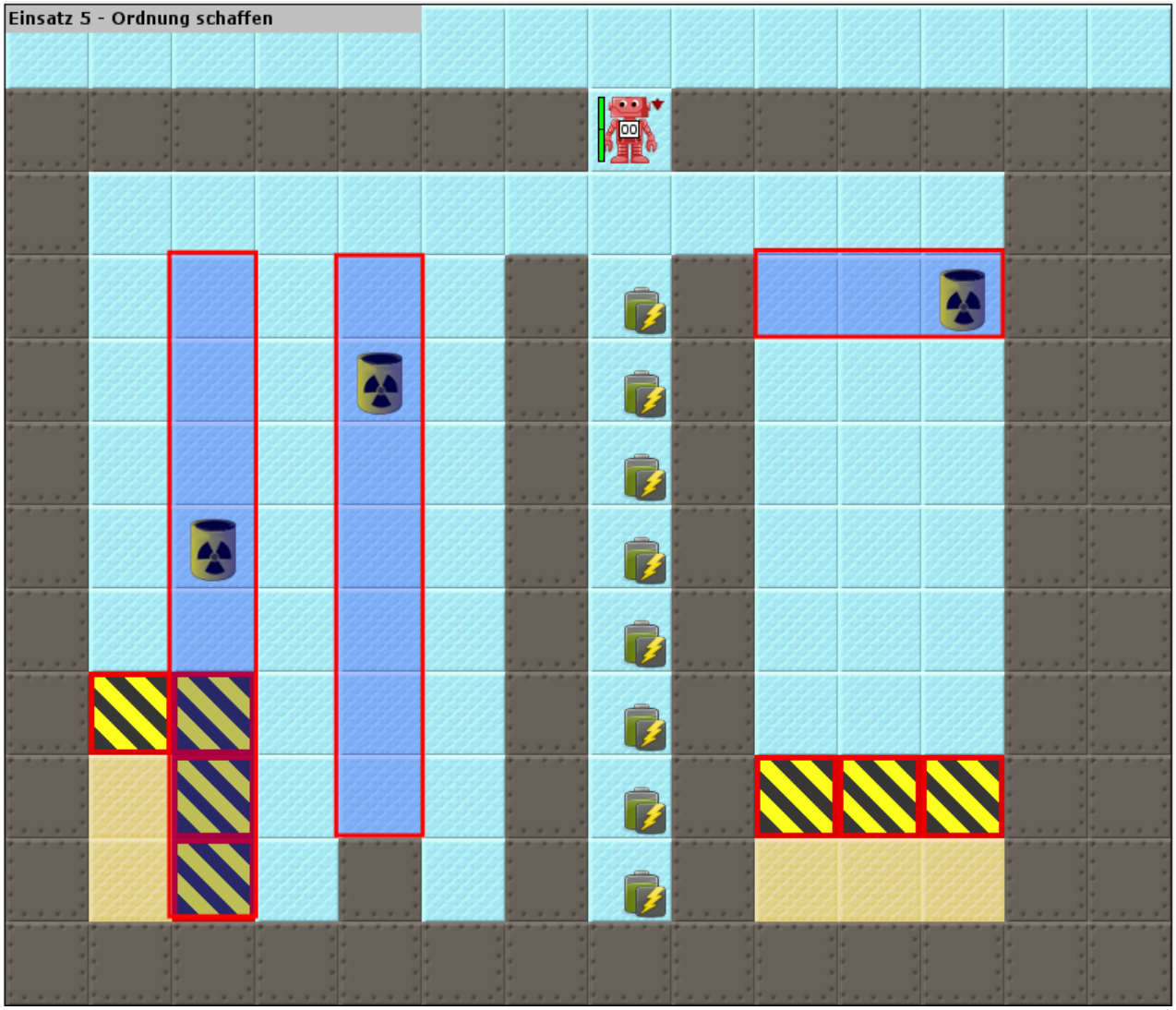
Erweitere die Methode `istVorFassFrei()` so, dass nicht nur Wände erkannt werden, die vor dem Fass sind, sondern auch andere Fässer (normale und Atommüll). Jetzt müsste `schiebeFassBisWand()` auch für den Roboter rechts oben funktioniere, ohne dass du das Fass vor der Wand zuerst wegschiebst.

### Aufgabe 12: Aufräumen

Implementiere eine Methode, die den Roboter rechts unten das Fass auf das gelbe Feld in der Ecke schieben lässt. Verwende einen sinnvollen Methodennamen.

# Einsatz 5: Schaffe Ordnung im Atommüllzwischenlager

Die Arbeiter haben die Atommüllfässer einfach willkürlich in die zwei Räume gestellt. Der Eingang zu den Räumen ist an der Kreuzung. Geht man geradeaus weiter, liegt dort eine Reihe von Akkus. Im Raum links stehen zwei Fässer. Ich weiß nicht genau, wie viele Fässer im anderen Raum stehen. Die Fässer können an den eingerahmten Stellen stehen. Sorge für Ordnung. Schiebe dazu die Fässer in die gekennzeichneten gelben Bereiche




- **Tipp 1:** Kopiere die Methode `laufeBisWand()` von AB3. Sie kann dir gute Dienste leisten.
- **Tipp 2:** Baue in die Methode `istVorFassFrei()` eine Energiekontrolle ein, da sonst die Energie ausgehen kann. Teste dazu am Anfang der Methode mit `istEnergieSchwach()` ob der Roboter mit `benutze("Akku")` aufgeladen werden muss.

---

<<< Zurück zu Level 4 **Level 5** Weiter zu Level 6 >>>

---

| | | |
|---|---|---|
| 03_methoden_unterklassen.odp | 1.8 MiB | 29.09.2021 20:47 |
| 03_methoden_unterklassen.pdf | 449.1 KiB | 29.09.2021 20:47 |
| ab501.png | 16.3 KiB | 04.12.2019 13:00 |
| ab502.png | 19.4 KiB | 04.12.2019 13:00 |
| ab503.png | 170.0 KiB | 10.10.2023 10:09 |

---

Alle Arbeitsaufträge in diesem Namensraum basieren auf den Materialien von `Schaller/Zechnall` zur Informatikfortbildung Baden-Württemberg 2016 und stehen unter einer CC-BY-SA-NC Lizenz.

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

Permanent link:
**https://info-bw.de/faecher:informatik:mittelstufe:robot:arbeitsauftraege:ab5:start**

Last update: **19.10.2023 10:15**