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. Im Gegensatz zu den Methoden, die du bisher implementiert hast, geben diese eine Antwort zurück.

Aufgaben:

1. **a)** Teste die Methode ***istFassVorne()*** an allen Robotern ***AB5***. Wie beantwortet ein Roboter diese Anfrage? Welche Antwortmöglichkeiten gibt es?
   b) Ein Roboter erkennt ein Atommüllfass (noch) nicht als Fass.
   Teste dies an dem Roboter vor dem gelben Fass.

2. Analysiere nun den Quelltext zur Methode ***istFassVorne()***: Wo tauchen die Antwortmöglichkeiten im Quelltext auf? Welcher Befehl sorgt dafür, dass eine Antwort zurückgegeben wird?

3. Füge in die Methode ***istFassVorne()*** die Anweisung **dreheUm();** ein:
   a) Vor der if-Anweisung.
   b) Jeweils vor dem return.
   c) Nach der if-else-Anweisung / vor der letzten Klammer.
   Welche Auswirkungen hat dies jeweils? **Entferne** die dreheUm() Anweisung wieder.

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.

5. **Out of Power:** Implementiere eine Methode ***istEnergieSchwach()***. Diese soll **true** zurückgeben, falls der Roboter höchstens **20** Energiepunkte hat (benutze ***getEnergie()***). Ansonsten gibt sie **false** zurück.
   Teste deine Methode an den drei Robotern im Raum unten links (nur der unterste hat einen schwachen Ladezustand).

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.

Fässer kann man schieben. Aber nur, wenn vor dem Fass Platz ist. Daher ist es hilfreich zu wissen, ob vor dem Fass frei ist oder nicht. Probiere die Methode ***einsVor()***, wenn der Roboter vor einem Fass steht.

7. **Look Ahead:** Implementiere eine Methode ***istVorFassFrei()***. Diese soll **false** zurückgeben, falls vor dem Fass eine Wand ist. Dabei können die folgenden Fälle auftreten:

Der Roboter muss **rechts** um das Fass herumlaufen – sofern möglich – und nachschauen, ob vor dem Fass frei ist. Falls 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 soll er am Ende wieder an seinen Ausgangspunkt stehen.

8. **Aufräumen:** Implementiere eine Methode ***schiebeFassBisWand()***, die ein Fass bis zur nächsten Wand schiebt. Teste deine Methode an dem Roboter unten rechts.
Hinweis: Da man ***istVorFassFrei()*** als Bedingung für eine while-Schleife verwenden kann, ist der Quelltext dieser Methode nur 3 Zeilen lang. Der Roboter rennt dann aber ganz schön wild durch die Gegend.

Für die nächsten Aufgaben solltest du dir nochmal die Verknüpfung von Bedingungen mit **&&** (UND), **||** (ODER) und **!** (NICHT) an.

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 zweite weitere Methode ***geheBisKreuzung()*** und teste sie am Roboter oben links.

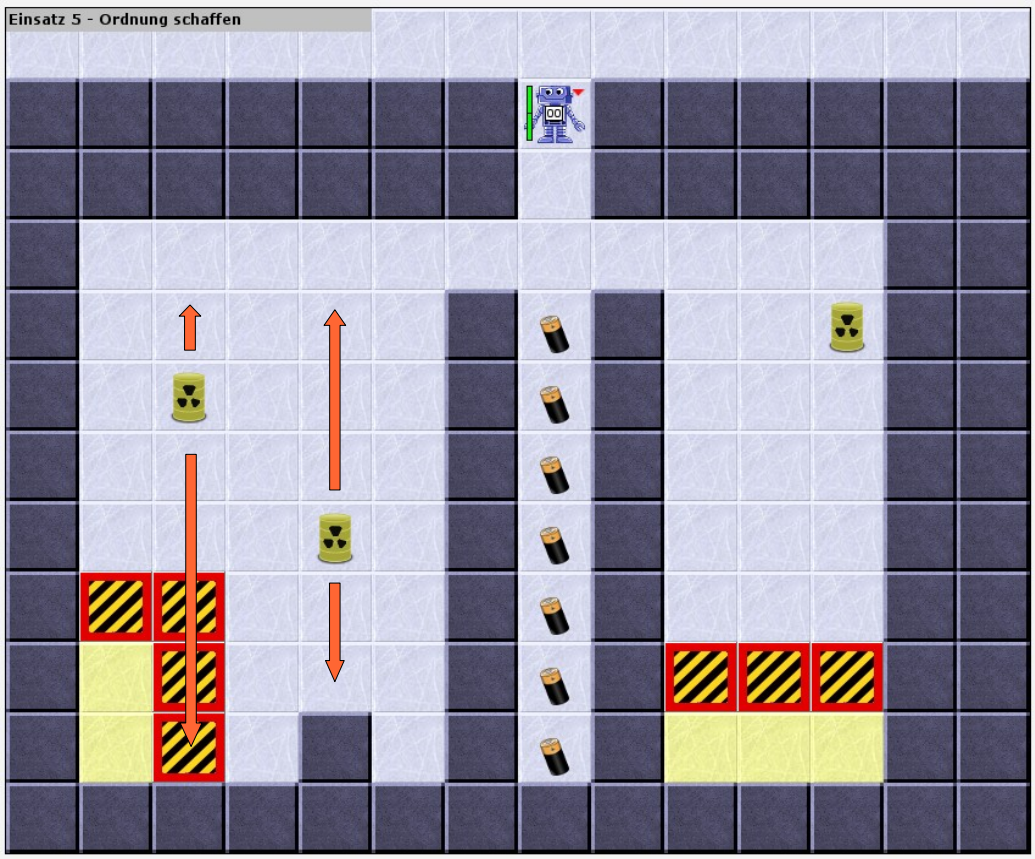
10. **Upgrade 1:** Erweitere die Methode ***istFassVorne()*** so, dass nicht nur die Übungsfässer erkannt werden, sondern auch Atommüllfässer (→ siehe Methode istVorne(String gegenstand). Teste deine Methode an den beiden Robotern im rechten Raum.

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 funktionieren.
Hinweis: istVorneFrei() prüft nur, ob eine Wand vorhanden ist.

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 genau zwei Fässer. Ich weiß nicht genau, wie viele Fässer im anderen Raum stehen. Die Fässer können an den mit den Pfeilen gekennzeichneten 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.

Tipp 3: rechter Raum: das zu schiebende Fass ist immer ganz an der Wand.

Tipp 4: linker Raum: nutze die bereits implementierte Methode aus Aufgabe 12 geschickt, um die Fässer auf die gelben Felder zu schieben.

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