AB 7 – VERERBUNG+METHODEN ÜBERSCHREIBEN

Der Atommüll im Endlager rottet vor sich hin… keiner traut sich mehr hinein! Wenigstens sollte eine Aufstellung über die Anzahl der dort lagernden Fässer und der beschädigten Fässer, aus denen die abgebrannten Brennstäbe schon herausgefallen sind, gemacht werden. Eine neue Aufgabe für unseren Rescue-Robot...

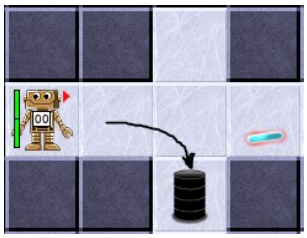
Nicht jedes Mal muss man alles neu programmieren ...

ZIEL: *Vererbung einsetzen, um Fähigkeiten eines Roboters in ein neues Modell zu übernehmen.*

Aufgaben:

1. Verändere die Klasse ***AB6_AB7*** so, dass sie nicht mehr eine direkte Unterklasse von Roboter, sondern eine Unterklasse des ***AB5***-Roboters ist. Sie erbt auf diese Weise alle Attribute und Methoden der Klasse ***AB5***.
   Compiliere die Klasse neu. Wie verändert sich das Klassendiagramm?
   Überprüfe, dass der ***AB6_AB7***-Roboter jetzt die Fähigkeiten des ***AB5***-Roboters hat. Du findest sie unter „inherited from AB5 >“. (inherited = engl. geerbt).

2. Erkläre, warum dein Roboter weiterhin auch die Fähigkeiten eines normalen Roboters hat.



3. **Kreuzungen zählen:** Implementiere im ***AB6_AB7*** eine Methode ***zaehleKreuzungen()***, die einen Gang ans Ende zur Wand läuft und zählt an wie vielen Kreuzungen der Roboter vorbei kam. Benutze dazu einen der bei ***AB5*** programmierten Methoden. Speichere den Wert in einem geeigneten ***Attribut***. Stelle auch eine ***get-Methode*** für dieses Attribut zur Verfügung. Verschiebe zum Testen mit der Maus das Fass vor dem Roboter oben links, damit er einen Gang mit Kreuzungen vor sich hat.

4. **Überschreiben von Methoden + Schlüsselwort super**: Ändere den Namen der Methode ***einsVorMitZaehlen()*** in ***einsVor()***. Rufe innerhalb dieser Methode die „alte“ und gleichnamige Methode der Superklasse/Oberklasse mittels **super.einsVor()** auf, damit sich dein Roboter weiterhin bewegen kann.
   Hinweis: Damit du die Klasse kompilieren kannst, muss du natürlich überall die Namensänderung vornehmen, wo du ***einsVorMitZaehlen()*** verwendet hast.

   Überprüfe die Auswirkung des Überschreibens der Methode einsVor():
   - Rufe ***einsVor()*** beim ***AB6_AB7*** auf. Beobachte die Auswirkung im Inspect-Fenster.
   - Rufe die von Roboter vererbte Methode ***einsVor()*** auf. Beobachte die Auswirkung im Inspect-Fenster.
   - Rufe die von AB5 geerbte Methode ***laufeBisWand()*** auf. Beobachte die Auswirkung im Inspect-Fenster.
   Was fällt dir auf?

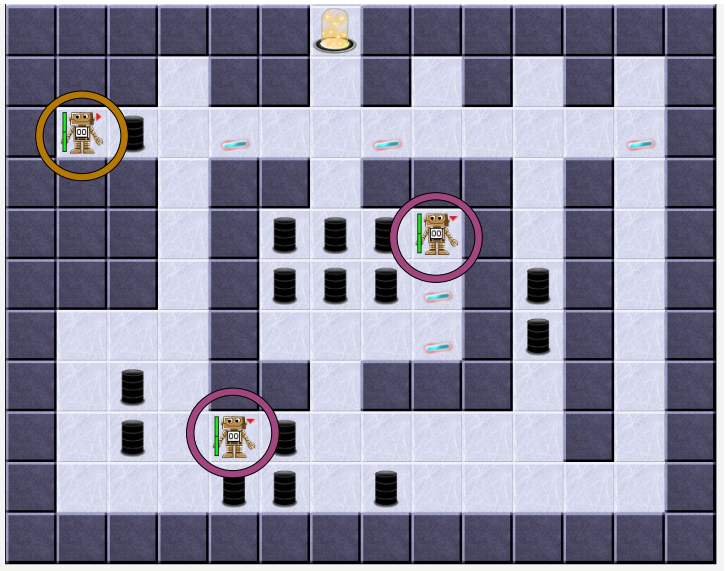
5. **Brennstäbe zählen:** Ändere die überflüssig gewordene Methode ***bisWandMitZaehlen()***, sodass sie jetzt in einem neuen Attribut ***brennstaebe*** zählt, wie viele Brennstäbe auf dem Weg zur Wand liegen. Verwende dazu die Methode ***istAufGegenstand(...)***. Implementiere ebenfalls auch die dazugehörige get-Methode.
   Achtung: Ein direkt vor der Wand liegender Brennstab muss auch erkannt werden.

6. **Standardmethoden überschreiben:** Überschreibe auf die gleiche Weise die Methode ***dreheLinks()*** und ***dreheRechts()*** von Roboter analog zu ***einsVor()***, sodass jetzt immer die Drehungen gezählt werden. Überprüfe, ob der Pledge-Algorithmus (***einsatz6()***) immer noch funktioniert.

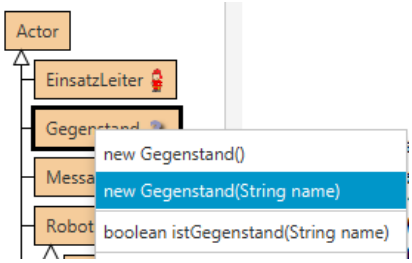
7. **Fass links:** Implementiere eine neue Methode ***istFassLinks()***, der erkennt, ob links ein Fass (egal, ob ein normales oder ein Atommüllfass) ist. Verwende dazu die Methode ***istFassVorne()*** (siehe Klasse ***AB5***). Der Roboter soll am Ende der Methode wieder in dieselbe Richtung wie vorher schauen.

8. **Überschreiben weiterer Methoden:** Überschreibe die Methoden ***istWandVorne()*** und ***istWandLinks()*** so, dass sie **true** zurückgeben, falls wirklich eine Wand vorne (super.istWandVorne()) bzw. links (super.istWandLinks()) ist **oder** falls ein Fass vorne bzw. links ist. Verwende dazu die Methoden istFassVorne() und istFassLinks().

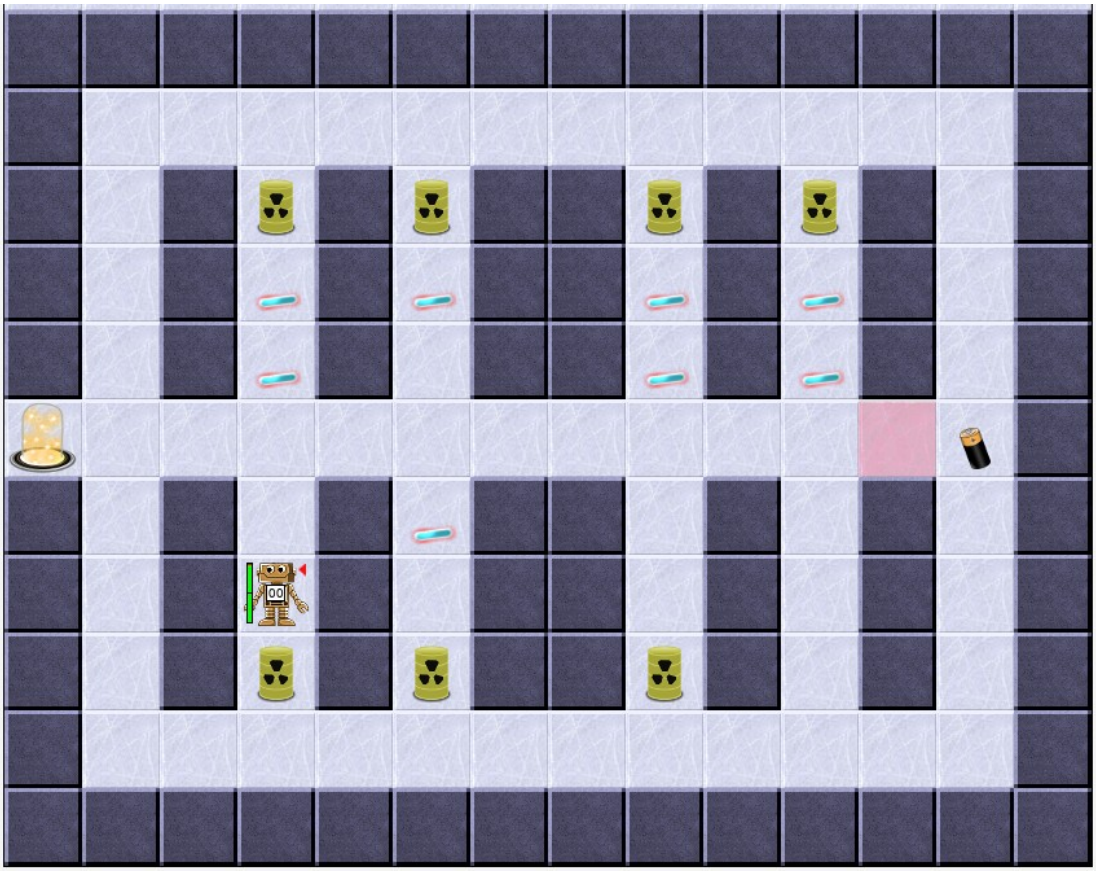
9. Teste die Auswirkungen dieser Änderungen auf den Pledge-Algorithmus, indem du direkt die Methode ***einsatz6()*** der Roboter ***AB6_AB7*** aufrufst (nicht das Level einsatz6() auf einem Bodenfeld).
   Der Roboter **unten links** als auch **oben rechts** sollten, den Weg zum Portal finden, während des Roboter oben links das Fass vor sich nicht mehr verschieben wird und deshalb den Weg zum Portal nicht findet.



Du kannst auch einige Fässer zusätzlich in der Welt platzieren. Erzeuge dazu mit einem Rechtsklick auf new Gegenstand(String name) einen neuen Gegenstand und gib als Name "Atommuell" oder "Fass" ein.



Einsatz 7:



In einem alten Bergwerk wurde vor etlichen Jahren ein Atommüllendlager eingerichtet. Leider rosten die Fässer dort vor sich hin, so dass einige Brennstäbe in den Gängen herumliegen. Für Menschen ist es zu gefährlich, diese zu zählen, da sie stark strahlen. Außerdem ist das Bergwerk stark einsturzgefährdet. Jeden Moment kann es einen Steinschlag geben. Unser Roboter soll das übernehmen.

Auftrag: Zähle die Brennstäbe in den Gängen, hol dir den Akku und kehre zum Portal zurück.

Tipps:
- Der Anzahl der Gänge sowie deren Abstände voneinander als auch die Anzahl der Brennstäbe sind zufällig. Starte hierzu den Einsatz mehrmals.
- Nutze die Methoden, die du bereits für den Roboter ***AB6_AB7*** programmiert hast.
- Auf dem Rundweg außen herum liegen keine Brennstäbe.
- Nutze den Pledge-Algorithmus, um aus dem eingestürzten Bergwerk zu entkommen.

- In seltenen Fällen kann es sein, dass die Energie nicht ausreicht, um das Bergwerk zu verlassen. In diesem Fall musst du den Einsatz erneut ausführen.

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