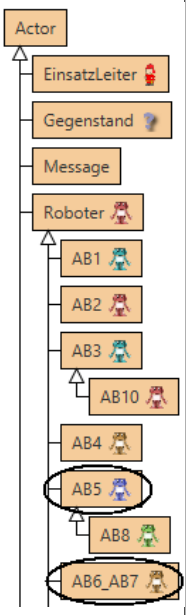
AB 7 – Vererbung

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..*

Im AB5 hast du viele neue Sensoren für den Roboter programmiert, in AB6 hat er gelernt, Schritte und Drehungen zu zählen. Aber dafür sind alle Fähigkeiten des AB5 wieder verloren gegangen. Natürlich könnte man per Copy-Paste die Methode in die neue Klasse übernehmen. Das geht aber auch eleganter.



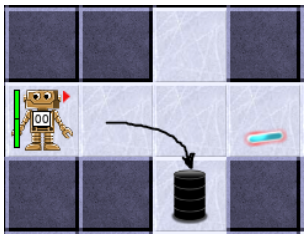
Du hast ja schon gemerkt, dass deine Roboter alles können, was ganz allgemein ein Roboter kann, ohne dass du jemals dafür eine einzige Methode hast kopieren müssen. Das liegt daran, dass deine ABx-Roboter Unterklassen der Klasse Roboter sind. Sie erben alle Eigenschaften und Fähigkeiten der Oberklasse, können sie dennoch erweitern oder verändern.

Dies erreicht man durch folgende Anweisung am Anfang der Klasse:

public class AB6 extends Roboter {...}

Aufgaben:

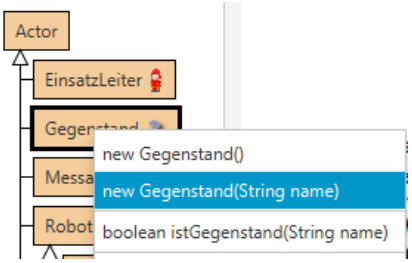
1. *Verändere die Klasse AB6_AB7 so, dass sie nicht mehr eine direkte Unterklasse von Roboter, sondern eine Unterklasse des AB5-Roboters ist. Sie erbt auf diese Weise alle seine Sensoren.*
   Compiliere die Klasse neu. Wie verändert sich das Klassendiagramm?
   Überprüfe, dass der AB6_AB7-Roboter jetzt 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 Sensoren. Speichere den Wert in einem geeigneten Attribut. Stelle auch eine get-Methode zur Verfügung. Verschiebe zum Testen mit der Maus das Fass vor dem Roboter oben links, damit er einen Gang mit Kreuzungen vor sich hat.*



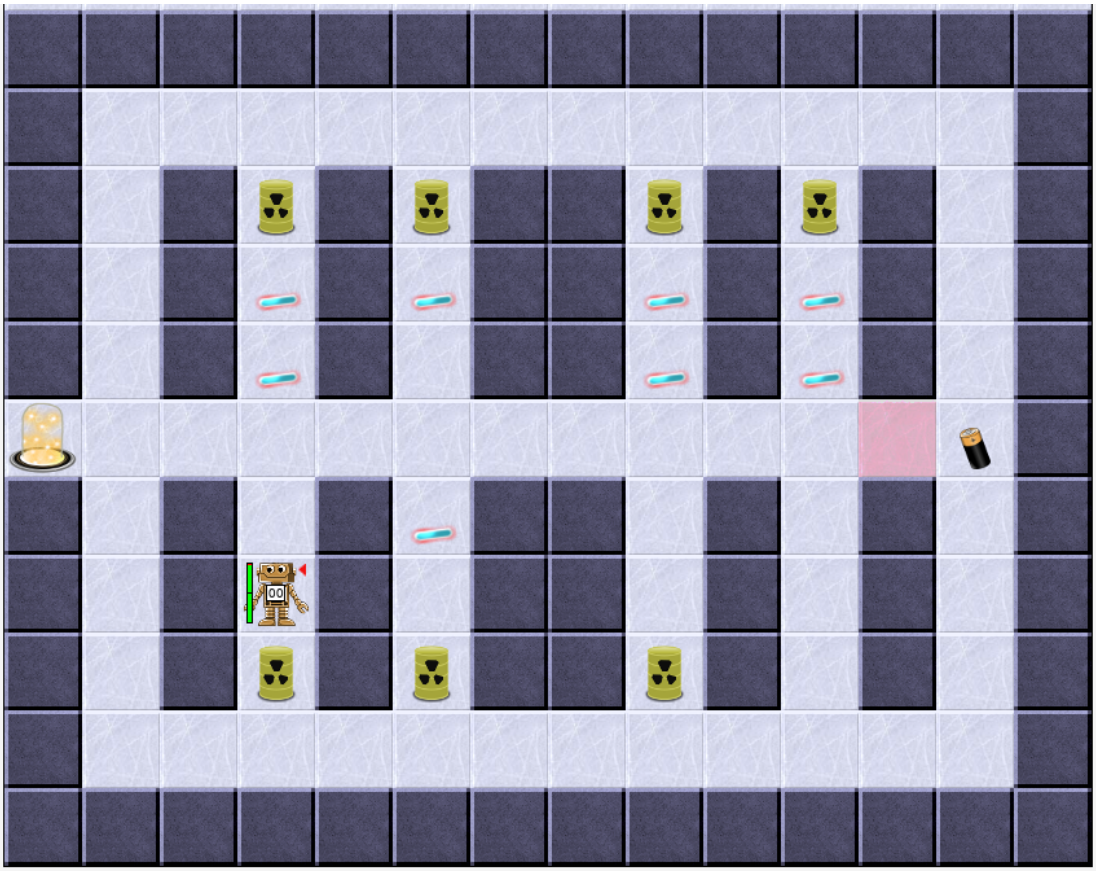
Vererbung kann aber noch mehr. Man kann schon existierende Methoden auch verbessern. Beim letzten Arbeitsblatt wurde eine neue Methode einsVorMitZaehlen eingeführt, die eine Verbesserung der Methode einsVor() des Standardroboters darstellt. Statt eine Methode mit einem neuen Namen zu erfinden, kann man auch die alte Methode verbessern. Ändere dazu den Namen auf einsVor(). Nun gibt es diese Methode zweimal. Einmal beim Roboter und einmal beim AB6_AB7-Roboter. Um die alte Methode aufzurufen, musst du beim AB6_AB7-Roboter super.einsVor(); schreiben (lies: rufe die Methode einsVor() bei der übergeordneten Klasse auf).

Aufgaben:

4. ***Super**: Ändere den Namen der Methode einsVorMitZaehlen in einsVor. Rufe innerhalb dieser Methode die gleichnamige Methode der Oberklasse mit super.einsVor() auf.*
   Anmerkung: Damit du die Klasse kompilieren kannst, muss du natürlich überall die Namensänderung vornehmen, wo du einsVorMitZaehlen() verwendet hast.
   Überprüfe die Auswirkungen dieser Änderung:
   - Rufe einsVor() beim AB6 auf. Beobachte die Auswirkung im Inspect-Fenster.
   - Rufe die von AB5 geerbte Methode laufeBisWand() auf. Beobachte im Inspect-Fenster.
5. ***Brennstäbe zählen:** Ändere die überflüssig gewordene Methode bisWandMitZaehlen(), so dass sie in einem neuen Attribut brennstaebe zählt, wie viele Brennstäbe auf dem Weg zur Wand liegen (wir verwenden auf dem Testgelände natürlich nur ungefährliche Attrappen, die aber täuschend echt aussehen). Verwende dazu istAufGegenstand("Brennstab"). Ergänze 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, so dass die Drehungen gezählt werden. Überprüfe, ob der Pledge-Algorithmus (einsatz6()) immer noch funktioniert.*
7. ***Fass links:** Implementiere einen neuen Sensor istFassLinks(), der erkennt, ob links ein Fass (egal, ob ein normales oder ein Atommüllfass) ist. Verwende dazu den Sensor istFassVorne() (Öffne zum Vergleich AB 5).*
8. ***Überschreiben:** 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 Sensoren istFassVorne() und istFassLinks().*
9. *Teste die Auswirkungen dieser Änderungen auf den Pledge-Algorithmus, indem du direkt am Roboter (nicht auf einem Bodenfeld) den einsatz6() aufrufst, nachdem du einige Fässer in der Welt zusätzlich platziert hast. (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, hebe sie auf und kehre zum Portal zurück.

Hinweis:
- Auf dem Rundweg außen herum liegen keine Brennstäbe.
- Mach dir vorher ein Bild von der Situation, indem du den Einsatz7 startest.
- Eventuell reicht die Energie nicht aus, 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*.