AB 9 – METHODEN MIT PARAMETERN

Ein Platz für ein neues Endlager ist gefunden! In einem Bergwerk soll dies geschaffen werden. Sprengarbeiten müssen vorgenommen und Brennstäbe eingelagert werden. Eine Aufgabe wie gemacht für unsere Roboter… Ein Einsatzleiter wurde bestimmt, die anstehenden heiklen Arbeiten zu koordinieren.

Methoden mit Parametern

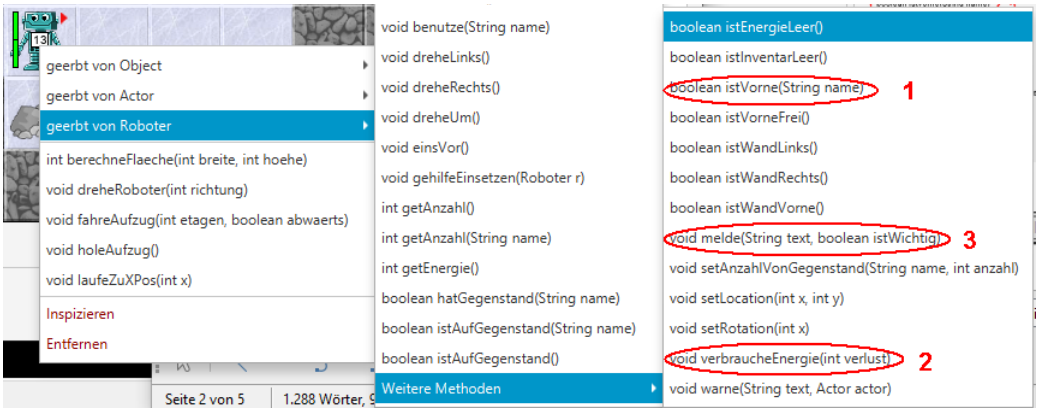
ZIEL: *Du kannst Probleme mithilfe von flexibel einsetzbaren Methoden lösen. Diese Methoden enthalten Parameter, mit denen sich das Verhalten der Roboter flexibel steuern lässt.*

Aufgaben:

1. Markiere im unten stehenden Bild alle Methoden, die beim Aufruf eine Eingabe verlangen.
   Welche Methode hat die längste Parameterliste?
   Woran erkennst du, ob eine Methode einen Wert zurückgibt?
   Welche zwei Methoden haben sowohl Parameter als auch einen Rückgabewert?

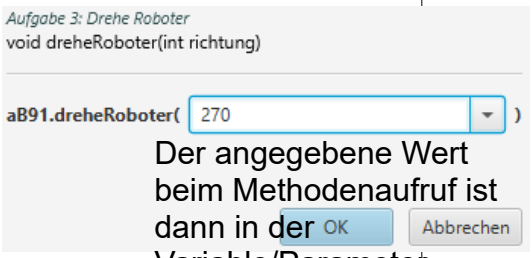
```
void ablegen(String name)
int berechneFlaeche(int breite, int hoehe)
float berechneVolumen(float breite, float hoehe, float tiefe)
String gibAntwortsatz()
int gibAnzahlBrennstaebe()
float gibRadioaktivitaet()
void holeBombeUndKehreZurueck(int xPos, int yPos)
boolean istDunkel()
void legeAnzahlBrennstaebe(int anzahl)
void legeBrennstab(int x, int y, boolean aufKontaktplatte)
void legeBrennstaebe(int pos1X, int pos1Y, int pos2X, int pos2Y, int pos3X, int pos3Y)
void sprenge(int startX, int startY)
```

2. Alle Roboter erben ja schon einige Methoden mit Parameter aus der Klasse ***Roboter***.
   Teste an einem ***AB9*** die drei unten markierten Methoden.
   Beschreibe jeweils kurz, welche Aufgabe der Parameter der Methode erfüllt.



AB 9 – METHODEN MIT PARAMETERN

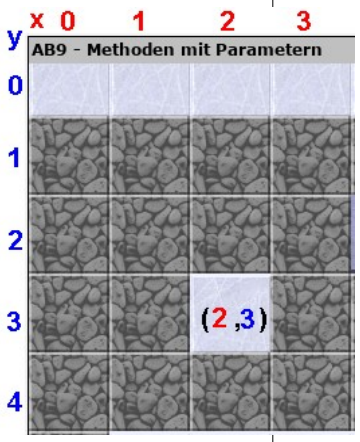
Im Bergwerk bringt ein Aufzug die Roboter in das richtige Stockwerk. Durch einen „Schalter“ kann der Aufzug in das oberste Stockwerk gerufen werden. Dafür gibt es schon die Methode ***holeAufzug()***.



Der angegebene Wert beim Methodenaufruf ist dann in der Variable/Parameter **richtung** gespeichert.

3. **Drehe Roboter:** Vervollständige die Methode ***dreheRoboter(int richtung)***, die den Roboter in die gewünschte Richtung dreht (0=Blick nach rechts, 90=Blick nach unten, 180=Blick nach links, 270=Blick nach oben). Die Zahlen stehen hierbei für den Winkel. Mit ***getRotation()*** kann man die aktuelle Richtung bzw. Winkel des Roboters erfragen.
   Tipp: Drehe den Roboter solange, bis ***richtung*** erreicht ist.

Wie du schon richtig erkannt hast, schauen wir nun nicht mehr auf die Roboterwelt von oben, sondern wir betrachten einen Querschnitt eines Bergwerks. Hier spielt – wie im wirklichen Leben – die **Schwerkraft** eine wichtige Rolle. Pass auf, dass deine Roboter **nicht in den Aufzugschacht fallen**.



4. **Laufe zu:** Vervollständige die Methode ***laufeZuXPos(int x)***, die den Roboter zu der angegebenen x-Koordinate laufen lässt. Die y-Position kann sich auch aufgrund der Schwerkraft verändern.

   Tipp: Durch Rechtsklick auf die Welt (blauer Hintergrund) kannst du die Methode ***zeigeKoordinaten()*** (ggf. unter more methods) aufrufen, somit erkennst du schnell, wie die Koordinaten eines Feldes lauten.

Der alte Grubenaufzug funktioniert etwas eigentümlich und muss noch programmiert werden. Bisher muss ein Roboter nach unten bzw. nach oben schauen und einsVor() gehen, damit der Aufzug ein Stockwerk nach unten bzw. nach oben fährt.

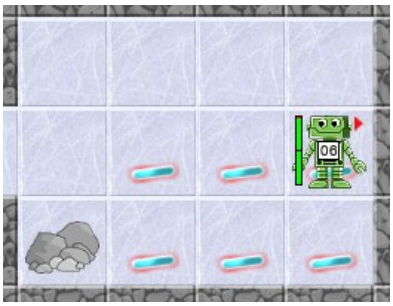
5. **Fahre Aufzug**: Vervollständige die Methode ***fahreAufzug(int stockwerke, boolean abwaerts)***, die einen Roboter, der auf einem Aufzug steht (istAufGegenstand(“Aufzug“)?), die angegebene Anzahl von Stockwerken (ein Stockwerk entspricht einem Schritt) abwärts oder aufwärts fahren lässt. Diese Methode soll funktionieren, egal in welche Richtung der Roboter am Anfang schaut. Steht der Roboter nicht auf einem Aufzug, soll nichts passieren.

6. **Fahre ins Stockwerk**: Implementiere die Methode ***fahreInsStockwerk(...)***., wobei als Parameter das gewünschte Stockwerk übergeben werden soll. Die Stockwerke werden dabei durch die y-Koordinate repräsentiert (2 ≤ y ≤ 10, siehe Koordinaten).

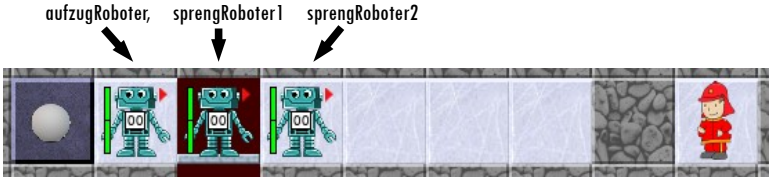
7. **Bombe sichern:** Implementiere die Methode ***holeBombeUndKehreZurueck(...)***, welche als Parameter die Koordinaten der Bombe übergeben bekommt. Der Roboter, der auf dem Aufzug steht soll zur angegeben Position laufen und dort die Bombe einsammeln. Danach soll er zum Aufzug zurückkehren.

8. **Sprengen:** Nun soll ein ***AB9***-Roboter einen Bereich im Bergwerk sprengen. Implementiere dazu die Methode ***sprenge(...)***, die einen Roboter, der auf dem Aufzug steht, zur **übergebenen** (x|y)-Position laufen lässt, mit benutze(“Bombe“) die Bombe zündet.

9. **Legen**: Der Roboter soll einen Raum mit einer Zweierreihe von Brennstäben (von denen er ausreichend dabei hat) belegen. Implementiere dazu eine Methode, die als Parameter die (korrekte) Breite des Raumes bekommt.
   Hinweis: Die Schwerkraft bewirkt, dass ein abgelegter Brennstab nach unten rutscht. Das heißt, legt man einen Brennstab auf einen Felsen, rutscht dieser runter, sofern kein Fels oder anderer Gegenstand da liegt. Beachte, dass der Roboter sich nicht mehr nach oben bewegen/springen kann.



Nun kommt der Einsatzleiter ins Spiel. Das ist das kleine Männchen oben rechts. Der Einsatzleiter kennt die vier Roboter. Der erste ganz links heißt ***aufzugRoboter***, der rechts daneben ***sprengRoboter1***, dann ***sprengRoboter2*** und unten steht der ***legeRoboter***.



Im Folgenden muss der Einsatzleiter jeweils Methoden bei einem der vier Roboter aufrufen. Im Quelltext schreibt man dazu beispielsweise folgenden Befehl:

```
sprengRoboter2.fahreAufzug(2,true);
```

Diese Befehle sind in der dir bekannte Punktnotation (Reihenfolge: **Wer** macht **was**?) Drückt man die Tastenkombination STRG+Leertaste nach der Eingabe des Punktes, so bekommt man eine Liste aller Methoden, die man bei diesem Objekt (hier: Roboter) aufrufen kann.

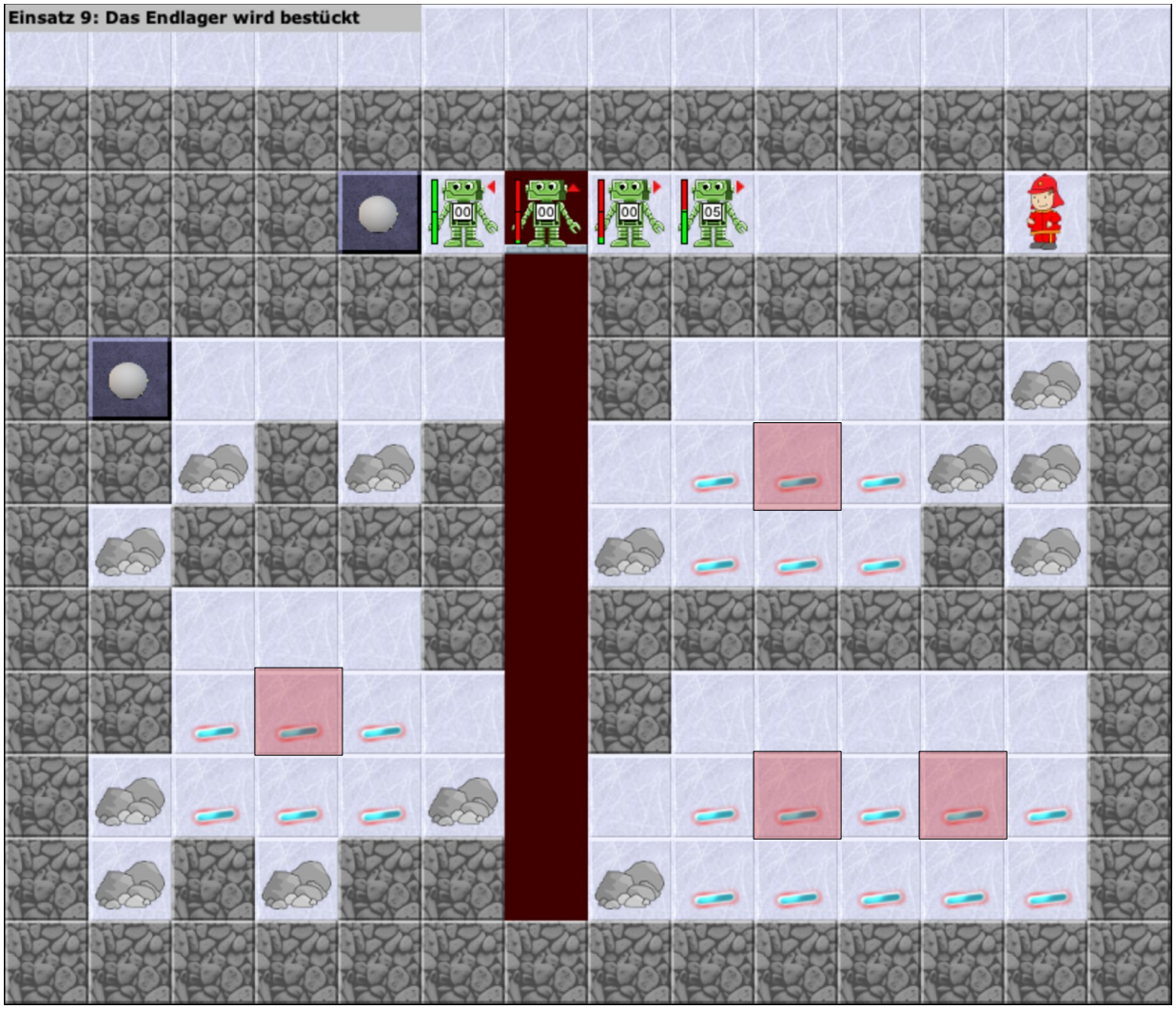
10. **Sprengkommando**: Öffne die Klasse ***EinsatzLeiter*** und vervollständige die Methode ***holeBombeUndSprenge (int xBombe, int yBombe, int xPos, int yPos)***. Dabei sind (xBombe|yBombe) die Koordinaten, wo sich eine Bombe befindet und (xPos|yPos) die Koordinaten der Sprengposition. Der Einsatzleiter soll dazu dem ***aufzugRoboter*** und dem ***sprengRoboter1*** die passenden Befehle geben!

Einsatz 9: Nun übernehmen Sie die Verantwortung – Einsatzleiter!

Im Folgenden soll dein ***Einsatzleiter*** den Einsatz im Endlager koordinieren. Die ***AB9***-Roboter haben alles gelernt, was sie für diesen Einsatz benötigen:

Sie können in einem Stollen Sprengarbeiten vornehmen und einen freigewordenen Bereich mit Brennstäben belegen. Aber Achtung, die Ressourcen sind knapp – sowohl die Anzahl der Sprengsätze, als auch die Energie der Roboter. Die Arbeit unter Tage ist nicht nur für Menschen anstrengend…

Auftrag: Deine Aufgabe ist es, den *Einsatzleiter* so zu programmieren, dass sie den Robotern klare Anweisungen gibt, um die drei verschütteten Stollen zu erweitern. Der *legeRoboter* soll dann seine Brennstäbe wie unten abgebildet in den frei gewordenen Hohlräumen ablegen. Hoffentlich geht ihm dabei nicht die Puste aus. Alle Roboter müssen sich am Ende wieder im oberen Eingangsstollen befinden.



Tipp: Im Bild sind die Sprengpositionen mit ▭ markiert.

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