Auftrag AB 10 – Gezählte Wiederholungen

Eine Diamantminenfirma hat vom großen Erfolg des ReaktorRobots gehört und überlegt, ob sie nicht auch die gefährlichen Arbeiten in den Diamantenminen durch Roboter ausführen lassen soll, da es immer wieder zu Stolleneinbrüchen kommt oder gefährliche Sprengungen notwendig sind. Eine letzte Aufgabe für unsere Rescue-Robots... zum Glück kommen diese gerade aus dem Trainingslager. Schau mal an, was sie dort gelernt haben.

Ziel: Zählschleifen kennen und (in mindestens einer Art) in Java notieren können.

Die Methode `rundeDrehen()` in der Klasse AB10 wurde umständlich notiert:

```
public void rundeDrehen() {
     laufeBisWand();
     dreheLinks();
     laufeBisWand();
     dreheLinks();
     laufeBisWand();
     dreheLinks();
     laufeBisWand();
     dreheLinks();
 }
```

Der Roboter umrundet gegen den Uhrzeigersinn ein von Wänden eingeschlossenes rechteckiges Feld. Dabei macht er vier Mal das Gleiche. Eine Befehlssequenz ist also eine mehrere male auszuführen.

Daher gibt es in praktisch allen Programmiersprachen die Möglichkeit solche Wiederholungen zu notieren.

Wir müssen dem Roboter mitteilen, was er wiederholen soll (`laufeBisWand()` und `dreheLinks()`) und wie oft. Dabei muss er selbst mitzählen, wie oft er es schon gemacht hat.

Er muss also solange die Befehle erneut ausführen wie die Rundenzahl kleiner ist als 4:

```
public void rundeDrehen() {
  int anzGemacht;
  anzGemacht = 0;
  while (anzGemacht < 4) {
    laufeBisWand();
    dreheLinks();
    anzGemacht++;
  }
}
```

Die Wiederholungsrunden muss er zählen. Dafür benötigt er innerhalb der Methode `rundeDrehen()` eine Variable. Sie wird als ganzzahlige Variable mit dem Namen `anzGemacht` für die Wiederholung bereitgestellt: `int anzGemacht;` Die Anweisung `anzGemacht = 0;` legt den Anfangswert der Zählvariable auf Null fest. Bisher hat der Roboter auch noch keine Runde gedreht.

Der Wert dieses Wiederholungszählers muss bei jedem Schleifendurchgang um 1 erhöht werden.

Daher findest du am Ende der zu wiederholenden Anweisungen den Erhöhe-Befehl für die Zählvariable

```
anzGemacht++
```

Die Ausführungsbedingung `(anzGemacht<4)` sorgt dafür, dass die Schleife genau viermal durchlaufen wird: das erste Mal mit dem Wert 0 für `anzGemacht` , das zweite Mal mit dem Wert 1, das dritte Mal mit dem Wert 2 sowie das vierte und letzte Mal mit dem Wert 3.

Eine 13-fache Wiederholung kannst du daher so schreiben:

```
int i=0;
while (i < 13) {
   // Anweisungen, die wiederholt werden
   i++; // Mitzählen!!
}
```

Statt dem Namen `i` könntest du auch einen Namen wie `wdhZaehler` benutzen oder sonst einen Namen. Programmierer verwenden gern kurz und knapp `i` als Zählvariable. Längere Namen blähen die Ausdrücke auf. Daher werden wir meist auch `i` oder `j` als Namen für Zählvariablen verwenden.

Wenn von Anfang an feststeht, wie oft etwas wiederholt werden muss, dann kann man Zählschleifen benutzen. Es gibt eine andere Formulierung für Zählschleifen, die wir aber nicht benutzen müssen, denn jede Wiederholung lässt sich wie gesehen mit while formulieren. Diese spezielle Formulierung mit dem Schlüsselwort `for` lautet:

```
for (int i=0; i<13; i++) {
  // Anweisungen, die wiederholt werden
  // if ( i == 2) { ...}
}
```

Der Wert der Zählvariable `i` steht im Schleifenkörper zur Verfügung.

Gezählte Wiederholungen in Java	
while-Schleife	for-Schleife

`public void vierVor() {`
`  int i=0;`
`  while (i < 4) {`
`  einsVor();`
`  i++;`
`  }`
`}`
	`public void vierVor() {`
	`  for (int i=0; i<4; i++){`
	`  einsVor();`
	`  }`
	`}`

Aufgaben:

Aufgabe 1: Patrouille

Teste mit dem Roboter unten links die Methode `rundeDrehen()` und betrachte anschließend den Quelltext. Wie oft läuft er die Strecke ab? Ändere die Methode so, dass er 10x hin und herläuft, d.h. 10x nach rechts und 10x wieder nach links.

Welche Zahl darf `anzGemacht` nicht überschreiten, damit der Roboter genau 800 Runden dreht?

Aufgabe 2

Warum wird die Variable `anzGemacht` aus der Methode `rundeDrehen()` nicht im Objektinspektor angezeigt?

Aufgabe 3

Verändere in der Methode `rundeDrehen()` die Zählschleife so, dass du anstatt einer `while`-Schleife eine `for`-Schleife verwendest. Oben stehende Tabelle kann dir als Hilfe dienen. Der Roboter soll nach Methodenaufruf genau 5x hin und herlaufen.

Aufgabe 4: Zu Befehl

Vervollständige die Methode `dreheAnzahlRunden(int anz)` so, dass der Roboter `anz`-Runden dreht, je nach übergebenem Parameter für `anz`. So lässt der Aufruf `dreheAnzahlRunden(7);` den AB10 sieben Mal hin und herlaufen.

Blöderweise kann ihm der Strom ausgehen... zum Glück hat er jedoch drei Akkus dabei. Ergänze die Methode dreheAnzahlRunden(int anz) so, dass er seine Akkus sinnvoll einsetzt. (Tipp: mit if(getEnergie()<60) kann er überprüfen, ob sein Ladezustand ausreicht.)

Aufgabe 5: Drehwurm

Was macht ein Roboter bei einem Wiederholeschritt wie rechts gezeigt?

```
int i;
i = 5;
while (i < 17) {
  dreheUm();
  einsVor();
  dreheLinks();
  i++;  // Mitzählen!!
}
```

Wie oft wird hier wiederholt? Schreibe das einfacher als *for*-Schleife in der Methode `drehWurm()` und lasse den AB10-Roboter diese in der Methode `mirWirdsGanzSchwindlig(int x)` x-Mal ausführen.

(Hinweis: möchtest du mal einen Methodenablauf unterbrechen, kannst du in Greenfoot auf die Reset-Taste drücken)

Aufgabe 6: Hartes Training

Bringe den AB10-Robotern bei, folgende Befehle auszuführen:

- gehe3Schritte();
- geheSchritte(int anz);
- lege7Brennstaebe();
- legeBrennstaebe(int anz);

Den letzten Befehl soll der Roboter nur ausführen, wenn sein Vorrat an Brennstäben dazu groß genug ist und wenn er nicht schon auf einem Gegenstand steht (!istAufGegenstand()). Er soll dabei so viele Brennstäbe ablegen, wie anz angibt, auch wenn der Vorrat noch viel größer ist. Nutze in den Methoden jeweils eine Zählschleife.

Aufgabe 7

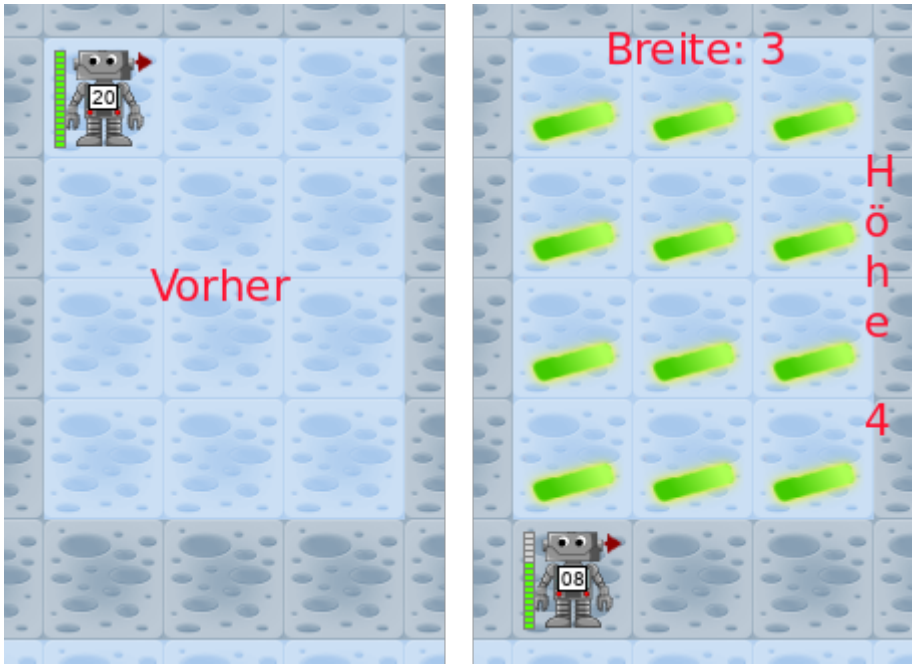
Der letzte Einsatz naht: Nun müssen die Roboter im Trainingslager noch lernen, wie man Brennstäbe präzise deponiert. Da darf beim Einsatz nichts schief gehen. Aber keine Angst wir sind ja noch im Training.

Es sollen die Methoden

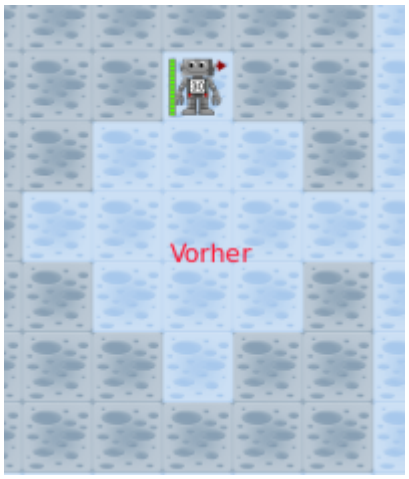
- `legeRechteck(int breite, int hoehe)` und
- `legeRaute(int hoehe)` implementiert werden.

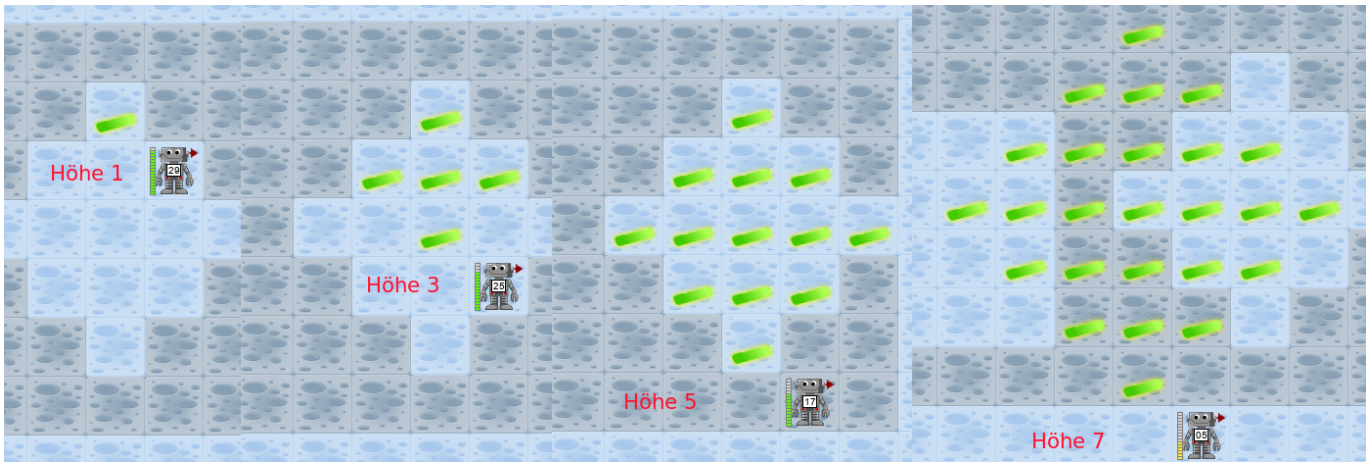
Der Einsatzleiter gibt zum Beispiel die Befehle `legeRechteck(3, 4)` oder `legeRaute(5)`.

Das Ergebnis von `legeRechteck(3, 4)` soll wie wie folgt aussehen:



Das Ergebnis von `legeRaute(5)` soll wie unten abgebildet aussehen:





Anmerkungen: Der Roboter beginnt beim Rechteck immer in der linken oberen Ecke bzw. bei der Raute oben mittig. Die Raute darf als Parameter für die Höhe nur ungerade Zahlen bekommen!

Der Roboter soll nur Brennstäbe legen, wenn er welche hat. Benutze die Methoden aus dem harten Training von Aufgabe 6.

Einsatz 10

Hilfe! Vier Stollen sind bei Sprengarbeiten eingestürzt und haben ein riesiges Loch hinterlassen. Die Situation ist prekär. Überall Staub – man kann kaum etwas sehen. Ein Trupp Rescue-Robots hat schon alle verteilten Brennstäbe gesichert. Nun müssen diese präzise deponiert werden. Die Einsatzleitung hat uns folgenden Plan überlassen:

Ihr Roboter hat 51 Brennstäbe für die Endlagerung und ein Akku für seine Stromversorgung erhalten. Sie müssen unbedingt genau nach Plan arbeiten. Beeilen Sie sich …

<<< Zurück zu Level 9 | **Level 10**
