Auftrag AB3

In einem anderen Kraftwerk ist ein Feuer ausgebrochen. Deine ersten Erfolge haben sich rumgesprochen. Daher wird deine RoboRescue-Firma angesprochen, ob sie auch in diesem Fall helfen kann. Natürlich bist du sofort bereit... aber wie löscht man ein Feuer?

Die Roboter machen vieles immer wieder...

Ziel

Wiederholungen in Handlungen erkennen, als SOLANGE-Schleife (while) formulieren und in Programmiersprache umsetzen können. Methoden mit Parametern benutzen können.

Aufgaben

Aufgabe 1: Löschen

Der Roboter AB3 ganz oben links soll den Feuerlöscher, der vor ihm steht aufnehmen, damit vor *zum* Feuer (nicht auf das Feuer!) gehen und es durch Benutzung des Feuerlöschers löschen. Führe dies zunächst von Hand durch (die Methoden `einsVor()`, ... findest du nun bei Rechtsklick unter "geerbt von Roboter").

Hinweis: Es gibt Methoden, die benötigen eine Zusatzinformation (einen sogenannten Parameter) um korrekt arbeiten zu können. Die Methode `benutze(String name)` ist so eine. Es wird also der Name des Gegenstandes erwartet, der benutzt werden soll. Dieser ist ein *String* (=Text). Strings müssen in Anführungszeichen gesetzt werden. Um den Feuerlöscher zu benutzen, muss man also `benutze("Feuerloescher");` schreiben.

Vervollständige dann die Methode `loesche()`, die dies alles automatisch machen soll. Teste deine

Methode.

Aufgabe 2: Mehr Feuerwehrmänner...

Die weiteren Roboter darunter sollen ihre ersten Feuer in der Reihe auch mit dem Befehl `loesche()` löschen können. Diese Feuer sind aber in einer anderen Entfernung platziert. Welche Zeilen deiner Methode `loesche()` Programms müssen überarbeitet werden? Worauf muss der Roboter reagieren können?

Aufgabe 3: Vor bis Feuer!

Gib jedem Roboter den Auftrag: `vorBisFeuer()` und beobachte, was sie tun.

Lies danach im Quelltext bei der Methode `vorBisFeuer()` die Anweisungen und versuche sie zu verstehen.

Hinweis: das Ausrufezeichen bedeutet "nicht".

Rufe diese Methode sinnvoll in der loesche()-Methode auf, indem du den Quelltext durch den Befehl `vorBisFeuer();` ergänzt. Teste die nun neue `loesche()`-Methode mit den vier Löschrobotern.

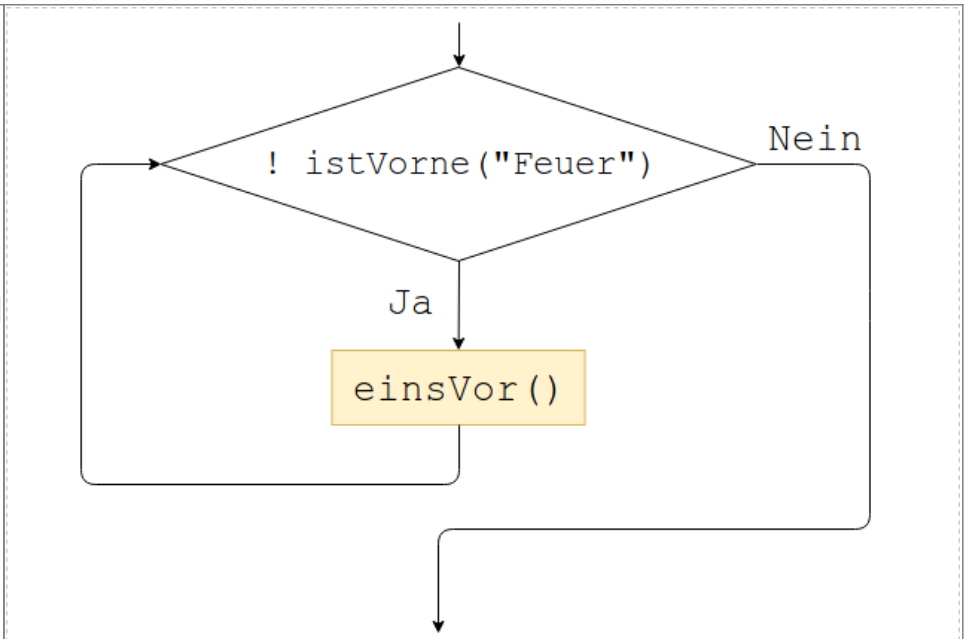
Aufgabe 4: Ins Feuer!

- Ergänze die Methode `vorBisFeuer` so, dass die Roboter danach noch einen Schritt ins Feuer hinein machen. Was passiert dann?
- Was passiert, wenn du innerhalb der lila Farbhinterlegung (innerhalb der `{}`-Klammern) noch ein zweites `einsVor();` einfügst?

Hintergrundinfos: Die while-Schleife

Der Auftrag `vorBisFeuer()` muss korrekt ausgeführt werden, egal wie weit das Feuer entfernt ist. Daher muss der Roboter prüfen, wie lange er vorwärts gehen muss. Auf die Anfrage: `istVorne("Feuer")` liefert er die Antwort `true` bzw. `false` ("wahr" oder "falsch"). Damit können wir diese bedingte Wiederholung so formulieren:

Normierte Sprache	Programmiersprache	Flussdiagramm

SOLANGE (vorne kein Feuer ist)
  WDH: ein Schritt vor
  ENDE WDH
	while (!istVorne(''Feuer''))
	  {
	  einsVor();
	  }
		

Genau so steht es auch im Quelltext. Im runden Klammerpaar hinter dem Schlüsselwort `while (…)` steht die Bedingung, welche die Wiederholungen steuert. Solange diese Ausführungsbedingung gilt, werden alle Anweisungen im Schleifenblock zwischen { und } wiederholt.

Oder anders ausgedrückt: Die Ausführungsbedingung wird überprüft. Ist sie wahr, so wird jede Anweisung in der Blockklammer zwischen { und } ausgeführt. Danach wird wieder überprüft, ob die Ausführungsbedingung immer noch wahr ist. Die Anweisungen innerhalb der Blockklammer werden wieder ausgeführt. Und so weiter und so fort. Erst wenn die Ausführungsbedingung falsch ist, wird die Schleife beendet und die Befehle hinter der schließenden Klammer } ausgeführt. **Innerhalb des wiederholten Vorgangs muss sich die Ausführungsbedingung verändern**, damit die Wiederholungen schließlich aufhören und nicht endlos laufen.

Aufgabe 5: Feuersbrunst löschen

Die Feuer des 3. und 4. Roboters bestehen aus mehreren Flammen hintereinander. Um diese zu löschen muss man immer wieder löschen, dann einen Schritt gehen, dann wieder löschen usw. Das wiederholt man so lange, wie nach dem Schritt noch ein Feuer vor einem ist.

Implementiere eine Methode `loescheReihe()`, die diese Aufgabe löst. Verwende dazu eine while-Schleife. Rufe diese Methode anschließend in deiner `loesche`-Methode auf.

Aufgabe 6: Reichweite testen

Mit einem Feuerlöscher kommt man nicht weit. Die sind recht schnell leer. Schreibe eine Methode `testeReichweite()`, bei der der Roboter unten links den ersten Feuerlöscher einsammelt, zur Feuerspur darunter geht und dort so viel Feuer wie möglich löscht.

Verwende eine while-Schleife um zu überprüfen, ob der Feuerlöscher noch nicht leer ist. **Hinweis:** Die Ausführungsbedingung `getAnzahl("Feuerloescher")>=1` testet, ob der Roboter noch

mindestens einen Löscher hat. Wenn er leer ist, wirft er ihn automatisch weg.

Aufgabe 7: Feuerlöscher einsammeln

Für ein großes Feuer braucht man mehrere Feuerlöscher. Vor dem Roboter unten links ist ein Gang, mit vielen Feuerlöschern in den Nischen (drücke ggf. unten auf den Reset-Knopf). Er soll nach vorne bis zur Wand laufen und dabei die Löscher einsammeln. Das soll funktionieren, egal auf welchem roten Feld er startet.

- Implementiere eine Methode `sammleLoescher()`, die zunächst den Roboter bis zur Wand laufen lässt, ohne die Löscher einzusammeln. (Die Methode `istVorneFrei()` testet, ob vor dem Roboter keine Wand ist).
- Ergänze die Methode dann so, dass er die Löscher einsammelt.
- Teste verschiedene Startpositionen (irgendwo auf einem roten Feld).

(Tipp: über den Geschwindigkeitsregler unten rechts kannst du die Ausführungsgeschwindigkeit anpassen!)

Aufgabe 8: Homerun

Lasse den Roboter rechts unten in die Sackgasse (Wand vorne, links und rechts) laufen. Verwende dazu mehrere Aufrufe von einer Methode `laufeBisWand()`, die du auch implementieren musst.

Implementiere `laufeBisSackgasse()`, indem du zunächst `laufeBisWand()` aufrufst und dann eine `while`-Schleife verwendest, um zu testen, ob du noch mal bis zur Wand laufen musst oder schon in der Sackgasse angekommen bist. Dies erkennst du, indem du testet, ob links keine Wand ist (`!istWandLinks()` - das ! steht für "nicht")

Einsatz 3: Feuerlöschen im Kernkraftwerk

"Oh mein Gott, im Kernkraftwerk ist ein Feuer ausgebrochen. Sie müssen unbedingt helfen! Wenn das Feuer nicht schnell gelöscht wird, gibt es eine Katastrophe. Unsere Arbeiter können wir nicht mehr rein schicken. Die Gefahr ist einfach zu groß!

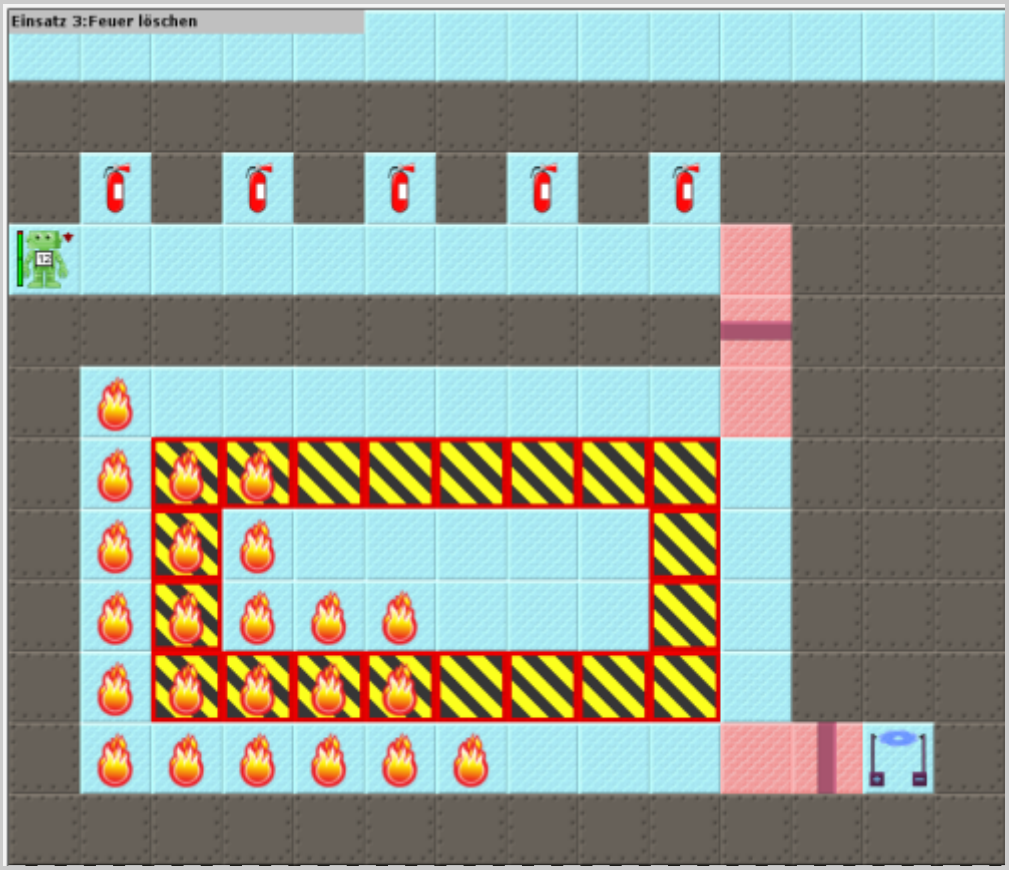
Wenn man reinkommt, gibt es gleich links eine Reihe von Nischen, in der die Feuerlöscher hängen. Sammeln Sie am besten alle ein. Wir wissen nicht, wie groß das Feuer schon ist.

Gehen Sie dann in die große Halle. Löschen Sie dort das Feuer. Es ist in der linken unteren Ecke ausgebrochen. Wie weit es sich inzwischen ausgebreitet hat, müssen Sie

selbst herausfinden [1)].

Rechts unten befindet sich ein Schutzraum (mit Portal), in das sich der Roboter am Ende begeben muss. Schaffen Ihre Roboter das?

Ach so, der Weg ist übrigens weit – nutzen Sie die Akkus, die der Roboter für die Mission bei sich hat!"



Tipps:

- Versuche zunächst, nur die oberste Reihe Flammen zu löschen und danach wieder nach rechts

zur Wand zu laufen. Drehe den Roboter wieder so, dass er nach unten blickt (Ausgangsstellung).

- Welche der benutzen Befehle müssen wiederholt werden, damit der Roboter die nächste Zeile

löscht? Wiederhole diese mit einer `while`-Schleife, solange das Portal noch nicht erreicht ist (warum erreicht man es automatisch?)

<<< Zurück zu Level 2 | **Level 3** | Weiter zu Level 4 >>>

02_ablauf_von_schleifen.odp 1.8 MiB 29.09.2021 17:41

02_ablauf_von_schleifen.pdf	268.5 KiB	29.09.2021 17:41
ab3_01.png	143.8 KiB	04.12.2019 11:55
ab3_02.png	12.9 KiB	04.12.2019 11:55
ab3_03.png	11.9 KiB	04.12.2019 11:55
ab3_04.png	160.9 KiB	08.10.2023 17:07
ff.png	12.5 KiB	04.12.2019 11:55

---

1)

Hinweis: in jeder Zeile ist direkt vor der Wand mindestens eine Flamme und es gibt keine Lücken im Feuer