Auftrag AB8: Wertzuweisungen

Immer mehr Atommüll. Wie viel passt eigentlich noch in unser Endlager? Einige Räume im Endlager sind noch frei. Dort sollen Fässer eingelagert werden. Vorher muss aber bestimmt werden, wie viel Platz dort noch ist.

Ziel: Lokale Variable als Zwischenspeicher für einen Wert kennen und einsetzen können. Lokale Variable von Objektvariablen unterscheiden können. Wertzuweisungen nutzen können.

Lokale Variable als Kurzzeitgedächtnis

Manchmal muss sich der Roboter bestimmte Werte nur vorübergehend merken, um eine Aufgabe zu erfüllen. Wenn er z.B. die Schrauben einer lückenlosen Schraubenspur zählen soll, so muss er sich nur kurzfristig merken, wie viele Schrauben er schon gezählt hat und kann das wieder vergessen, sobald er die Antwort gegeben hat. Bisher hatten wir Attribute der Objekte als Gedächtnis. Diese sind aber nur für Werte gedacht, die der Roboter sich dauerhaft merken soll: z.B. wie viele Schritte er insgesamt schon gelaufen ist, welche x-Position er im Moment hat, u.ä.

Man könnte auch das „Kurzzeitgedächtnis“ mit Attributen realisieren, würde dann aber sehr viele Attribute bekommen, die immer nur kurzfristig zum Einsatz kämen. Das fördert nicht gerade die Lesbarkeit.

Daher verwenden wir sogenannte lokale Variablen, deren Geltungsbereich nur eine Methode (oder auch nur eine Schleife) ist. Man verwendet sie fast genauso wie Attribute.

	Lokale Variable	**Attribut**
Deklaration	Innerhalb der Methode:	Innerhalb der Klasse, vor allen Methoden:

	`public class AB8 {`
	`  ...`
	`  //Methoden`
	`  public int berechneXY()`
	`{`
	`  int zaehler2;`
	`  ...`
	`  }`
	`}`
	`  ...`
	`}`
	→ **ohne** private!
		`public class AB8 {`
		`  private int zaehler1;`
		`  ...`
		`  //Methoden`
		`  ..`
		`}`
		→ **mit** private!

Initialisierung	Direkt nach der Deklaration:	Im Konstruktor der Klasse:

	`public class AB8 {`
	`  //Methoden`
	`  public int berechneXY() {`
	`  int zaehler2;`
	`  zaehler2 = 0;`
	`  ...`
	`  }`
	`  ...`
	`}`
		`// Konstruktor: Methode mit dem Namen der Klasse`
		`// ohne Rückgabtyp`
		`  public AB8() {`
		`  this.zaehler1 = 0;`
		`  }`

	Lokale Variable	**Attribut**
Verwendung	Ohne `this.`	Zur Unterscheidung sollte `this.` verwendet werden

	`zaehler2++;`
	`...`
	`return zaehler2;`
		`this.zaehler1++;`
		`...`
		`return this.zaehler1;`

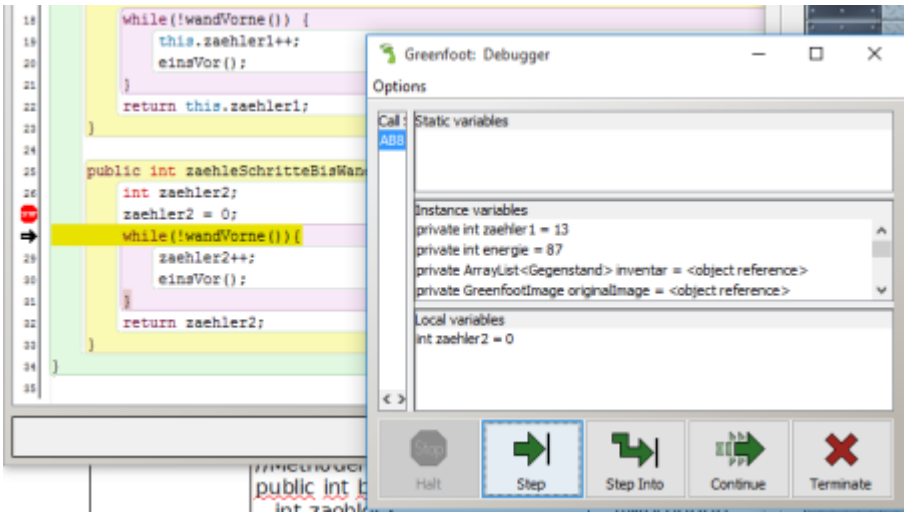
Aufgabe 1: Finde den Unterschied

Rufe bei den beiden AB8-Robotern links oben die Methoden `zaehleSchritteBisWand1` bzw. `zaehleSchritteBisWand2` auf. Sie sollten das gleiche Ergebnis liefern.

Rufe die Methoden erneut auf, um die Roboter auch auf dem Rückweg die Schritte zählen zu lassen. Erkläre den Unterschied.

Aufgabe 2: Debugging

Öffne im Kontextmenü der Roboter den Punkt `Inspect`, um die Attribute und Ihre Werte anzeigen zu lassen. Starte dann jeweils die Methoden `zaehleSchritteBisWand1` bzw. `zaehleSchritteBisWand2` und beobachte wie sich die Werte der Attribute verändern.



- Warum sieht man hier nur den `zaehler1`?
- Wo sieht man die *lokalen Variablen*?

Dafür benötigt man den *Debugger*, mit dem man auch bei lokalen Variablen den Verlauf kontrollieren kann.

1. Setze dazu einen *Breakpoint* im Quelltexteditor von AB8, indem du auf die Zeilennummer klickst, an der das Programm unterbrochen werden soll (hier Zeile 27).
2. Rufe dann die Methode `zaehleSchritteBisWand2` auf. Jetzt hält das Programm am Stoppschild an und kann mit `Step/Schritt über` Schritt für Schritt ausgeführt werden. Dabei werden die Objektattribute (= Instance variables) und lokale Variablen angezeigt. Du siehst, dass die lokale Variable erst entsteht, wenn sie das erste Mal benutzt wird. Beobachte auch, wie im Quelltext stets die Zeile markiert ist, an der das Programm gerade ist.

Aufgabe 3

Entscheide, ob die folgenden Werte besser als lokale Variable oder als Attribut gespeichert werden sollten:

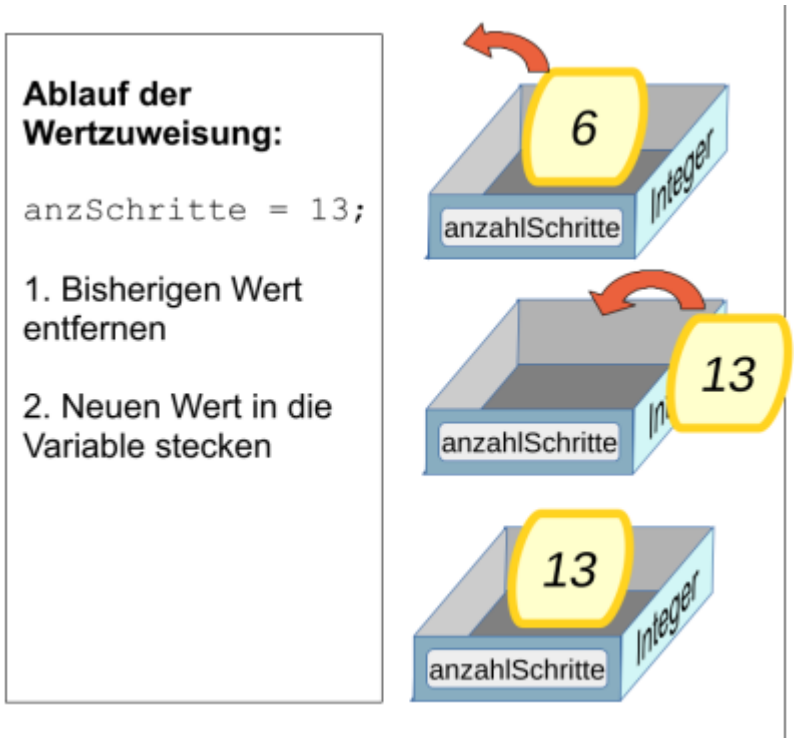
- die Farbe des Roboters
- die Anzahl der insgesamt gelöschten Feuer
- die Anzahl der mit dem aktuell benutzten Feuerlöscher gelöschten Feuer

Aufgabe 4

Entscheide, ob man bei dem Pledge-Algorithmus die Anzahl der Drehungen auch als lokale Variable hätte speichern können.

Wertzuweisungen

Bisher haben wir Variablenwerte nur initialisiert (z.B. `anzahl = 0`) oder um 1 erhöht/erniedrigt (`anzahl++` bzw. `anzahl--`). Da geht noch mehr:



- An beliebigen Stellen im Programm kann der Variable ein Wert zugewiesen werden:

```
anzBlaetter = 13;
```

- Wertzuweisung um Ergebnisse von Methoden zu speichern:

```
anzSchritte = zaehleSchritteBisWand1();
```

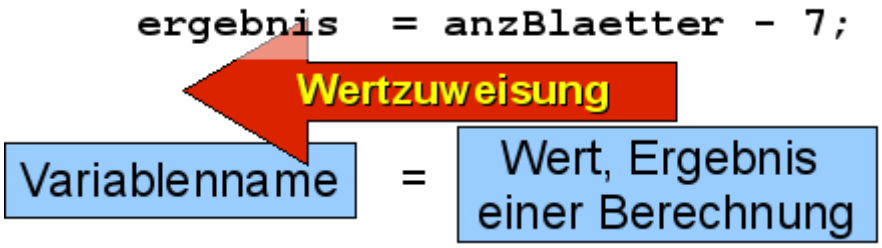
Dabei wird die Methode aufgerufen und das Ergebnis in der Variable `anzSchritte` gespeichert.

- Wertzuweisung mit Rechnungen:

```
flaeche = breite * hoehe;
```

  Dabei wird zunächst die Rechnung auf der rechten Seite ausgeführt und dann das Ergebnis in der Variablen gespeichert.

Grundsätzlich gilt: **Rechts vor Links!** Zuerst wird die Rechnung/Methode auf der rechten Seite ausgeführt und dann der Variable links zugewiesen.



Aufgabe 5: Summe

Rufe die Methode `gibPositionssumme()` an einem Roboter auf. Ziehe ihn an eine andere Stelle. Was meldet er jetzt als Antwort auf diesen Fragebefehl? Wie musst du ihn ziehen, damit das Ergebnis gleich bleibt? Begründe dies mit dem Quelltext. Wo findet welche Art von Wertzuweisung statt?

Aufgabe 6: Abstände

Implementiere die Methode `gibLaenge()` mit `int`-Rückgabewert: Deklariere drei lokale Variablen `x1`, `x2`, `x_diff`. Weise `x1` den Wert des Aufrufs von `getX()` zu. Lasse den Roboter bis zur nächsten Wand laufen. Weise nun `x2` den Wert des Aufrufs von `getX()` zu. Berechne die Differenz von `x2` und `x1` und speichere sie in `x_diff`. Gib diese Differenz als Ergebnis deiner Methode zurück.

Teste diese Methode. Welches Ergebnis liefert sie, wenn du den Roboter an eine Wand nach rechts laufen lässt. Was passiert bei Robotern die nach links/oben/unten laufen?

Verbessere deine Methode so, dass sie immer den Abstand zur Wand zurück gibt. Dazu wirst du auch die y-Koordinaten speichern müssen.

Anmerkung: Ok, einfacher als Schritte zählen ist das nicht...

Aufgabe 7: Fläche

Implementiere eine Methode, die die Fläche eines Raumes (Breite * Höhe) bestimmt und zurück gibt. Du kannst davon ausgehen, dass der Roboter in einer Ecke des Raumes steht. Du kannst dir aussuchen in welcher.

Aufgabe 8: Zähle Fässer

Implementiere eine Methode, die zurück gibt, aus wie vielen Fässern ein rechteckiges Areal von Fässern besteht. Sinnvoll ist es dazu zunächst einen Sensor istRechtsFass() zu implementieren. Damit kann man eine Methode gibLaengeFassreihe() implementieren, die man dann zum Bestimmen der Anzahl der Fässer benutzt.

Aufgabe 9

Gib an, welchen Wert die Variable summe hat, wenn folgende fünf Zeilen ausgeführt wurden:

```
int summe;
summe = 10;
summe = summe – 2;
summe = summe * 4;
summe = summe + 6;
```

Aufgabe 10: Sammler

Implementiere eine Methode, die den Roboter bis zur nächsten Wand laufen lässt und dabei alles aufhebt, was auf dem Boden liegt. Dabei soll der Wert der gesammelten Gegenstände ermittelt werden. Jede Schraube ist dabei 20 Cent Wert, jeder Akku 2,40 Euro, jeder Schlüssel 5 Euro, jeder Feuerlöscher 40 Euro.

Der Gesamtwert der gesammelten Gegenstände soll mittels einer get-Methode jederzeit abgefragt werden können.

Tipp: Überlege dir, ob du eine lokale Variable oder ein Attribut für den Wert verwenden möchtest. Kommazahlen kann man nicht in einer integer-Variable speichern. Dafür gibt es den Typ double. Kommazahlen werden in Java mit einem Punkt geschrieben (z.B. 2.40).

Aufgabe 11: Finanzamt

Auch Roboterfirmen müssen Steuern bezahlen. Für jeden gefundenen Gegenstand, der verkauft wird, muss Mehrwertsteuer bezahlt werden. Daher soll der Roboter mit `getMehrwertsteuer()` den Steuerbetrag der gefundenen Gegenstände berechnen können. Implementiere eine Methode, die die Steuer berechnet und zurück gibt.

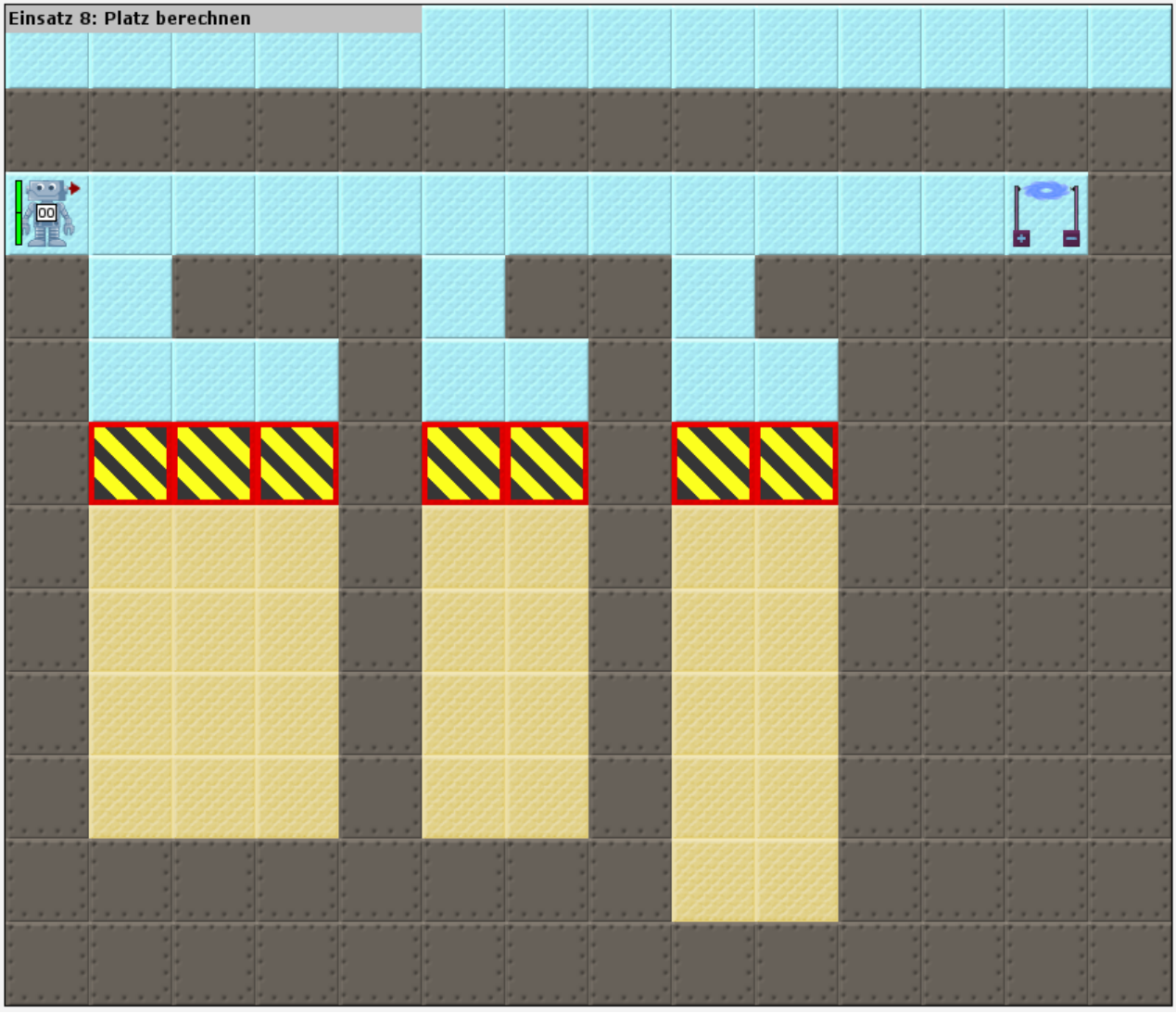
Einsatz 8

> **Bestimme die Lagerkapazität des Endlagers**
>
> Einige Räume im Endlager sind noch frei. Dort sollen Fässer eingelagert werden. Vorher muss

aber bestimmt werden, wie viel Platz dort noch ist.

- Die Räume gehen alle nach rechts von einem Gang ab.
- Man kommt immer am linken oberen Eck in den Raum.
- Die ersten zwei Zeilen müssen frei bleiben, damit man mit dem Gabelstapler rangieren kann. Daher können die Fässer nur auf den gelben Feldern platziert werden.

Bestimme die Anzahl der gelben Felder und gehe zum Portal. Der Leveltest (also die Methode einsatz8()) muss als Ergebnis die Anzahl der Felder zurückgeben.



<<< Zurück zu Level 7 **Level 8** Weiter zu Level 9 >>>
