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.

Wertzuweisungen

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, … .

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

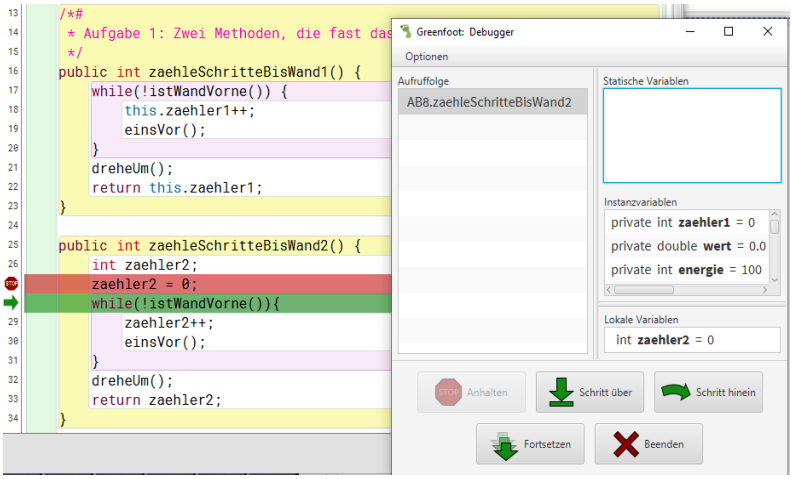
Deklaration	Innerhalb der Methode:
	public class AB8 {
	…
	//Methoden
	public int berechneXY() {
	int zaehler2;
	…
	}
	…
	}
	→ *ohne private!*
		Innerhalb der Klasse, vor allen Methoden:
		public class AB8 {
		private int zaehler1;
		…
		//Methoden
		…
		}
		
		→ *mit private!*

Initialisierung	Direkt nach der Deklaration:
	public int berechneXY() {
	int zaehler2;
	zaehler2 = 0;
	…
	}
		Im Konstruktor der Klasse:
		
		public AB8() {
		this.zaehler1 = 0;
		}

Verwendung	Ohne this. zaehler2++;	Zur Unterscheidung kann/sollte this. verwendet werden this.zaehler1++;

Aufgaben:

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 danach die Methoden erneut auf. Erkläre den Unterschied.*
2. ***Debugging:*** *Lasse dir mit INSPECT die Attribute anzeigen, um den Ablauf der Methoden zu verfolgen. 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. Setze dazu einen Breakpoint, indem du auf die Zeilennummer klickst, an der das Programm unterbrochen werden soll (hier Zeile 27). Wenn du dann die Methode aufrufst, hält das Programm am Stoppschild an und kann mit Step Schritt für Schritt ausgeführt werden. Dabei werden Attribute (= Instance variables) und lokale Variablen angezeigt. Du siehst, dass die lokale Variable erst entsteht, wenn sie das erste Mal benutzt wird.*



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

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

- Wertzuweisungen:
  An beliebigen Stellen im Programm kann der Variable ein Wert zugewiesen werden:
  `anzSchritte = 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.

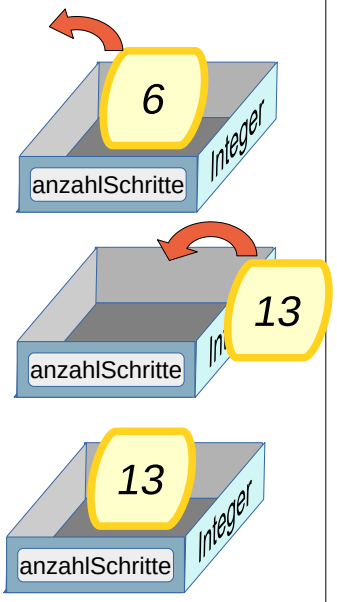
Grundsätzlich gilt: Rechts vor Links! Zuerst wird die Rechnung/Methode auf der rechten Seite ausgeführt und dann der Variable links zugewiesen.

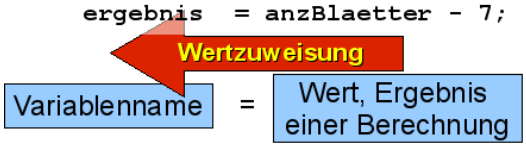
Ablauf der Wertzuweisung:

`anzSchritte = 13;`

1. Bisherigen Wert entfernen

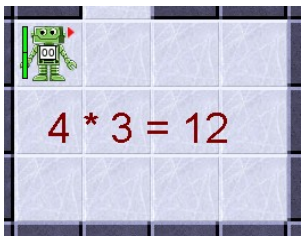
2. Neuen Wert in die Variable stecken



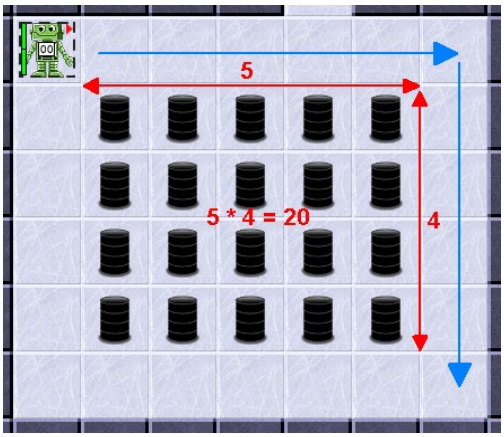


Aufgaben:

5. ***Summe:*** *Rufe die Methode gibPositionssumme() an einem Roboter auf. Bewege ihn (per einsVor- und drehen-Befehle) an eine andere Stelle. Was meldet er jetzt als Antwort auf diesen Fragebefehl? Wie musst du ihn bewegen, damit das Ergebnis gleich bleibt? Begründe dies mit dem Quelltext. Wo findet welche Art von Wertzuweisung statt?*
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 …



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.*
8. ***Zähle Fässer:*** *Implementiere eine Methode, die zurück gibt, aus wie vielen Fässern ein rechteckiges Areal von Fässern besteht. Es ist sinnvoll, zunächst einen Sensor istFassRechts() zu implementieren. Damit kann man eine Methode gibLaengeFassreihe() implementieren, die man dann zum Bestimmen der Anzahl der Fässer benutzt.*



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

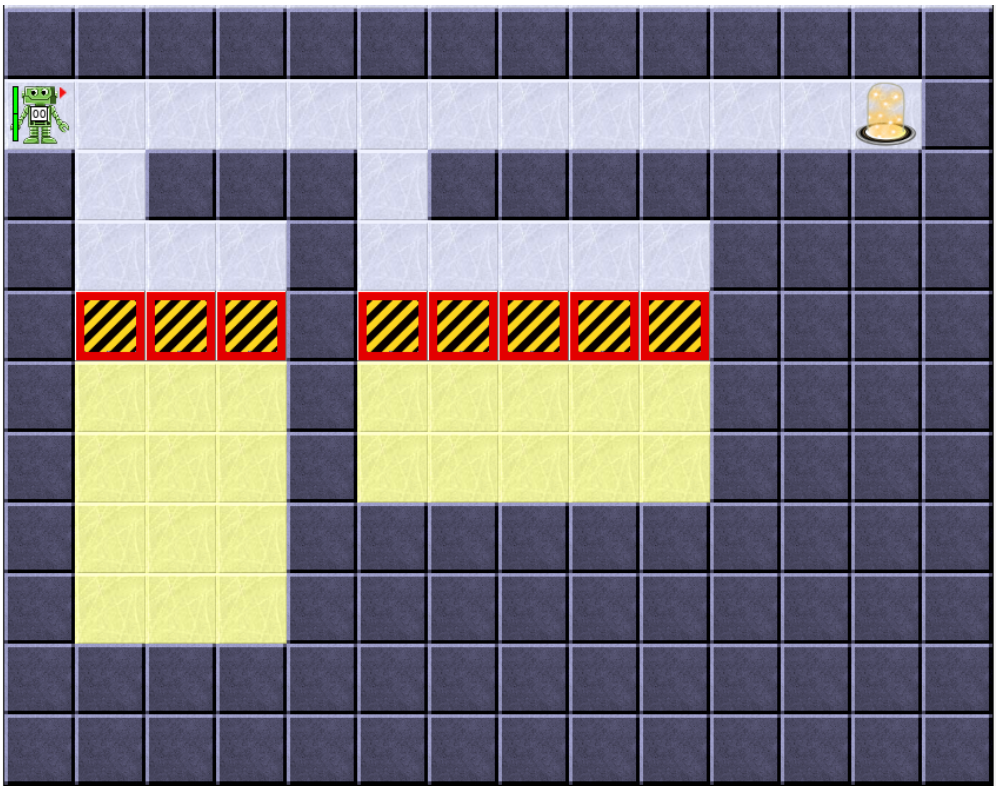
```
int summe;
summe = 10;
summe = summe – 2;
summe = summe * 4;
summe = summe + 6;
```

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. Er soll dann 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).
11. ***Finanzamt:*** *Der Wert aller gesammelter Gegenstände (aus vorheriger Aufgabe) muss besteuert werden, d.h. für jeden gefundenen Gegenstand, der verkauft wird, muss die Mehrwertsteuer hinzugerechnet und ans Finanzamt abgeführt 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.



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