Kapitel 3: Variablen

Im letzten Kapitel haben wir gelernt, bestimmte Ereignisse zu wiederholen solange eine Bedingung erfüllt ist. Nun möchten wir aber folgendes tun:  
*Kara soll eine Spur von fünf Kleeblättern legen.*

Dies wäre natürlich recht einfach, wenn wir einfach putLeaf() und move() fünfmal aufrufen, jedoch wenig elegant. Schöner wäre es, wenn Kara mitzählt, wie viele Blätter er bereits abgelegt hat. Damit Kara zählen kann, braucht er ein „Gedächtnis“, d.h. konkret einen Speicher. Die Speicherplätze beim Programmieren sind durch **Variablen** ansprechbar.

**Zählen mit Kara:**

int i;

i = 0;

while (i < 5) {

putLeaf();

move();

i = i + 1;

}

**Erklärungen:**

1. Mit **int i;** wird Speicherplatz für eine Variable mit dem Namen **i** und dem Typ **int**, d.h. ‚integer‘, reserviert. Man sagt: Die Variable **i** wird ***deklariert***. In Java gibt es unterschiedliche Typen, die verwendet werden können (siehe nachfolgende Tabellen).
2. Durch **i = 0;** wird der Variable **i** der Wert **0** zugewiesen. Da es die erste Wertzuweisung für die Variable ist, sagt man auch: Die Variable **i** wird ***initialisiert***.
3. Deklaration und Initialisierung werden meistens zusammengefasst wie folgt:  
   **int i = 0;**
4. Für die Bedingung **i < 5** wird der Vergleichsoperator **<** benutzt (weitere Vergleichsoperatoren siehe nachfolgende Tabellen).
5. Bei der Zuweisung **i = i + 1;** muss man zuerst den rechten Teil anschauen. Es bedeutet: „Nimm den aktuellen Wert von **i**, addiere **1** dazu und speichere den neuen Wert wieder unter dem Namen **i** ab.“

**Weitere Hinweise zu Variablen:**

1. Es ist möglich, eine Variable mit einem endgültigen, unveränderlichen Wert zu versehen, d.h. sie zu einer Konstanten zu machen:  
   **final int ANZAHL = 5;**  
   Dann könnte man im obigen Programmbeispiel **while (i < ANZAHL)** schreiben.  
   Konstanten schreibt man komplett mit Grossbuchstaben.
2. Variablen schreibt man wie Methoden beginnend mit einem Kleinbuchstaben!

# Elementare Datentypen in Java

Diese Datentypen heissen „elementar“, da es die Grunddatentypen in Java sind. (Später werden wir lernen, wie man Variablentypen für ganze Objekte erstellen kann).

**Ganze Zahlen und Zeichen**

|  |  |  |  |
| --- | --- | --- | --- |
| **Typ** | **von** | **bis einschliesslich** | **Speicherbedarf** |
| **byte** | -128 | 127 | 8 bit |
| **short** | -32‘768 | 32‘767 | 16 bit |
| **int** | -2‘147‘483‘648 | 2‘147‘483‘647 | 32 bit |
| **long** | -9‘223‘372‘036‘854‘775‘808 | 9‘223‘372‘036‘854‘775‘807 | 64 bit |
| **char** | 0 | 65‘635 | 16 bit |

**Fliesskomma-Zahlen**

|  |  |  |  |
| --- | --- | --- | --- |
| **Typ** | **von** | **bis einschliesslich** | **Speicherbedarf** |
| **float** | -3.4\*1038 | 3.4\*1038 | 32 bit |
| **double** | -1.7\*10308 | 1.7\*10308 | 64 bit |

**Wahrheitswerte**

|  |  |  |
| --- | --- | --- |
| **Typ** | **Wertebereich** | **Speicherbedarf** |
| **boolean** | **true** oder **false** | 1 bit |

# Vergleichsoperatoren

Folgende Operatoren können in Java für Vergleiche verwendet werden. Das Ergebnis ist jeweils ein boolean (entweder true oder false).

|  |  |  |
| --- | --- | --- |
| **Operator** | **Bedeutung** | **Beispiel** |
| **<** | Kleiner als | k < 12 |
| **<=** | Kleiner als oder gleich | k <= 23 |
| **>** | Grösser als | k > 67 |
| **>=** | Grösser als oder gleich | k >= 45 |
| **==** | Gleich | k == 2 |
| **!=** | Ungleich | k != 32 |

**Achtung: Der Vergleich auf „Gleichheit“ hat immer zwei Gleichheitszeichen ‚==‘. Mit einem einzelnen Gleichheitszeichen ‚=‘ geschieht eine Zuweisung!**

# Arithmetische Operatoren

Zur Berechnung können folgende arithmetische Operatoren verwendet werden:

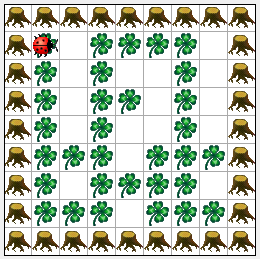
|  |  |  |
| --- | --- | --- |
| **Operator** | **Bedeutung** | **Beispiel** |
| **+** | Addition | h = wert + 34 |
| **-** | Subtraktion | z = 3.4 – t |
| **\*** | Multiplikation | value = h \* 3.56 |
| **/** | Division | d = m / v |
| **%** | Modulo (liefert den Rest der Division) | count = w % 2 |

Aufgabe 21: Kara zählt Kleeblätter[[1]](#endnote-1)

Kara soll waagrecht von links nach rechts gehen bis zum Baum und dabei die Kleeblätter zählen.

*Hinweise*: Am Schluss können Sie das Resultat mit dem folgenden Befehl auf die ‚Konsole‘ schreiben:  
  
System.out.println(„Das Resultat ist: “ + **count**);   
  
Alles, was in der Klammer des Befehls steht wird ausgegeben. Text muss man in Anführungszeichen schreiben und kann mit einem Pluszeichen zusammengesetzt werden.   
**count** ist der Name der Variable (kann natürlich auch anders benannt werden).

*Verfügbarkeit von Variablen: Variablen sind immer nur innerhalb des Blockes (zwischen den geschweiften Klammern) gültig, in welchem sie deklariert wurden. Man kann sie auch ausserhalb der Methoden deklarieren, dann gelten sie für die ganze Klasse.*

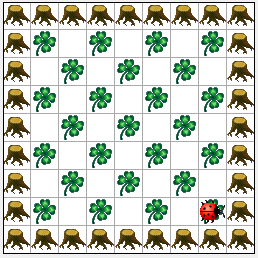
Aufgabe 22: Kara in a box I

Eine quadratische Fläche ist von Bäumen umrandet. Innerhalb der Fläche ist ein Muster aus Kleeblättern gelegt, das von Kara invertiert (= umgedreht) werden soll. Kara startet links oben in der Ecke mit Blick nach rechts.

*Hilfe:*

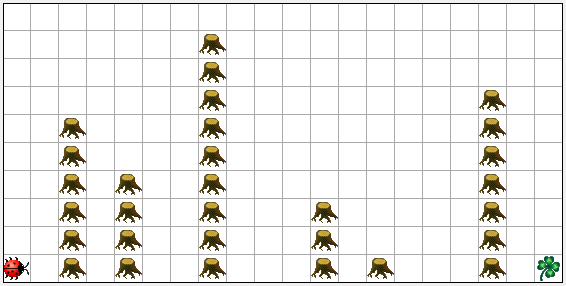
*Bei dieser Aufgabe ist es hilfreich, mit booleschen Variablen zu arbeiten, z.B.:*

boolean goingRight = false; // Deklaration und Initialisierung  
goingRight = !goingRight; // Aus true wird false und umgekehrt  
if (goingRight) // Boolesche Variable als Bedingung



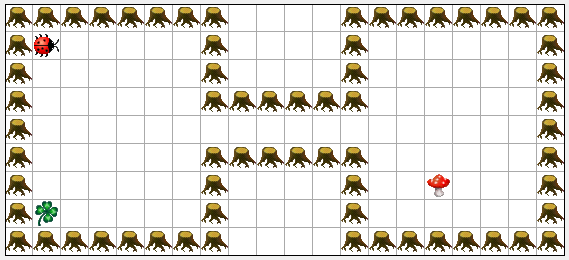
Zusatzaufgabe 23 (für Schnelle): Kara in a box II

Eine quadratische Fläche ist von Bäumen umrandet. Innerhalb der Fläche soll ein schachbrettartiges Muster aus Kleeblättern von Kara gelegt werden. Kara startet links oben in der Ecke mit Blick nach rechts.

Zusatzaufgabe 24 (für Schnelle): Die längste Baumreihe

In der Kara Welt hat es verschiedene Baumreihen. Kara soll nun die Länge (in Anzahl Bäumen) der längsten Baumreihe ermitteln und auf die Konsole ausgeben. Zwischen d en Baumreihen ist immer mindestens ein Feld Platz. Auf dem letzten Feld liegt ein Kleeblatt.

Zusatzaufgabe 25 (für sehr Schnelle): Pilz durch Tunnel schieben



Die Welt von Kara hat zwei Boxen, welche durch einen Tunnel verbunden sind. In der Box links befindet sich Kara und ein Kleeblatt. In der Box rechts befindet sich ein Pilz. Kara soll nun auf die andere Seite gelangen, den Pilz finden und ihn auf die andere Seite schieben. Auf der anderen Seite angekommen soll der Pilz schliesslich auf das Kleeblatt geschoben werden.

Sie können davon ausgehen, dass Kara immer oben links in der Ecke startet und das Kleeblatt sich unten links befindet. Der Pilz kann jedoch an einer beliebigen Stelle rechts vom Tunnel stehen.

Hinweis: Diese Aufgabe kann auch in Zusammenarbeit zu zweit gelöst werden. Dabei können die Teilprobleme untereinander aufgeteilt werden:

* Tunneleingang finden
* Pilz finden
* Pilz vor den Tunneleingang stossen
* Pilz auf das Kleeblatt stossen

# Vertiefung zu Variablen

Wir hatten bereits einen ersten Kontakt zu Variablen. Nun noch ein paar zusätzliche Bemerkungen zu den verschiedenen Typen:

## Elementare Datentypen

**x**

byte

**x**

short

**x**

int

**x**

long

**x**

boolean

**boolean x = true;**

**int x = 12;**

## Referenztypen

**Kara k = new Kara();**

**k.move();**

**k**

Kara

Kara-Objekt



🡪 Der Wert in **k** ist eine Referenz auf   
das Kara-Objekt. Mit dem Punktoperator (**k.**)   
kann **k** wie eine Fernbedienung auf das   
Kara-Objekt benutzt werden!

1. ***Hinweis zu Kara:***

   * Ideen und Konzepte von Kara wurden entwickelt von Jürg Nievergelt, Werner Hartmann, Raimond Reichert et al., <http://www.swisseduc.ch/informatik/karatojava/>, abgerufen Februar 2011.
   * Einige Kara-Übungen basieren auf Unterlagen von Horst Gierhardt, <http://www.swisseduc.ch/informatik/karatojava/javakara/material/>, abgerufen Februar 2011.

   [↑](#endnote-ref-1)