Inhalt

Variablen

- Speichern von Werten und Ergebnissen.
- Bestandteile einer Variable
  - Datentyp
  - Name (werden klein geschrieben)
  - Wert (muss mit Datentyp übereinstimmen)
  - Semikolon `;` beendet Ausdruck

Variablen

Wert 34 in Variable a speichern

```
int a = 34;
```

- Datentyp: `int`
- Name: `a`
- Wert: `34`
- Semikolon `;` beendet Ausdruck

Ausgabe auf Konsole

```
System.out.println(a);
```

Variablen

Wert 34 in Variable a speichern

```
int a = 34;
```

Überschreiben der Variable a

```
int a = 10; // Funktioniert nicht!
```

```
a = 10;
```

- Datentyp wird nur beim **Deklarieren** angegeben

Variablen

Rechnen mit int Variablen

```
int a = 34;
int b = 6;
```

```
int c1 = a + 1; // ?
```

Variablen

Rechnen mit int Variablen

```
int a = 34;
int b = 6;
```

```
int c1 = a + 1; // 35
```

Variablen

Rechnen mit int Variablen

```
int a = 34;
int b = 6;
```

```
int c1 = a + 1; // 35
int c2 = a + b; // ?
```

Variablen

Rechnen mit int Variablen

```
int a = 34;
int b = 6;
```

```
int c1 = a + 1; // 35
int c2 = a + b; // 40
```

Variablen

Rechnen mit int Variablen

```
int a = 34;
int b = 6;
```

```
int c1 = a + 1; // 35
int c2 = a + b; // 40
int a = a - 4; // ?
```

Variablen

Rechnen mit int Variablen

```
int a = 34;
int b = 6;
```

```
int c1 = a + 1; // 35
int c2 = a + b; // 40
int a = a - 4; // FEHLER, da a bereits deklariert
a = a - 4; // 30
```

Variablen anlegen

- Variablen vom gleichen Typ können in einer Zeile deklariert werden.

```
int a, b, c; // Deklaration

a = 1; // Wertzuweisung
b = 2;
c = 3;
```

Primitive Datentypen

Datentyp	Java-Syntax	Wertebereich	Länge
Boolischer Wert	`bool`	0,1	1 Byte
Byte	`byte`	-128...127	1 Byte
Zeichen	`char`	'a', 'b', ..	1 Byte
Short	`short`	±32768	2 Byte
Integer	`int`	±2147483648	4 Byte
Long	`long`	groß	8 Byte
Gleitkommazahl	`float`	groß	4 Byte
Gleitkommazahl	`double`	groß	8 Byte

Primitive Datentypen

```
bool bt = true;
bool bf = false;

short s = 100;

byte b = 5;

char c = 'a';

int i = -1999;

long l = 99999;

float f = 12.41f; // Achtung bei Float muss f angehängt werden

double f = 33.9987;
```

Maximale/Minimale Werte

```java
// BYTE
int min = Byte.MIN_VALUE; // - 128
int max = Byte.MAX_VALUE ; //+ 127

// INT
int min = Integer.MIN_VALUE; // - 2147483648
int max = Integer.MAX_VALUE ; //+ 2147483647
```

Komplexer Datentyp String

- Aneinanderkettung von `char`
- `H` , `a` , `l` , `l` , `o` => `Hallo`
- **char** in `''` , **String** in `""`
- char nur ein Zeichen, Strings beliebige Zeichen

```
char x1 = 'H';
char x2 = 'a';
char x3 = 'l';
char x4 = 'l';
char x5 = 'o';
System.out.println(x1+x2+x3+y4+x5);
```

```
String hallo = "Hallo";
System.out.println(hallo);
```

Konstanten

- Schlüsselwort `final`
- Können nicht überschrieben werden
- GROSSSCHREIBUNG (Trennung mit _)

```
final int MY_INT_CONST = 12;
final String MY_STRING_CONST = 12;
```

Globale/Lokale Variablen/Konstanten

- **Lokale Variablen** sind nur im Funktionsblock `{...}` sichtbar
- **Globale Variablen** (bei uns `static`) sind in der ganzen Datei sichtbar

```
class Main{

  static int globalVariable = 1;

  public static void main(String[] args){
    int localVariable = 33;

    globalVariable = 2;
    localVariable = 55;
  }

  // localVariable nicht mehr vorhanden
}
```

Arithmetik

Rechenoperatoren

- Addieren
  - `int x = 2 + 4;`
- Subtrahieren
  - `int x = 5 - 4;`
- Dividieren
  - `int x = 4/2;` ACHTUNG! Ergebnis kann Gleitkommazahl sein.
- Multiplizieren
  - `int x = 5 * 4;`
- Restwertberechnung (**Modulo**)
  - `int x = 5 % 4;` // Ergebnis ist 1

Arithmetik

Restwertberechnung

```
int x = 10;
int y = 3;

int xDurchY = x/y;  // 3
//ergibt 3,33 in int => 0,33 wird abgeschnitten

int rest = x % y; / 1
// Rest von 10/3  => 10-(3*3)  = 1
```

Modulo

- 1 % 3 = 1 // Teilbar: 0, Rest: 1
- 2 % 3 = 2 // Teilbar: 0, Rest: 2
- 3 % 3 = 0 // Teilbar: 1, Rest: 0
- 4 % 3 = 1 // Teilbar: 1, Rest: 1
- 5 % 3 = 2 // Teilbar: 1, Rest: 2
- 6 % 3 = 0 // Teilbar: 2, Rest: 0
- 7 % 3 = 1 // Teilbar: 2, Rest: 1
- 8 % 3 = 2 // Teilbar: 2, Rest: 2
- 9 % 3 = 0 // Teilbar: 3, Rest: 0

Übungen mod

- `10 / 2`
- `10 % 2`
- `10 % 3`
- `5 % 3`
- `19 % 10`
- `10 % 10`
- `100 / 1`
- `100 % 1`

Variable erhöhen um 1

`int i = 1;`

- `i++;` // gleich wie `i = i + 1;`
- `++a` erhöht die Variable und verwendet erhöhte
- `a++` verwendet Variable und erhöht sie anschließend

```
int a = 1
System.out.println(a++); // Output: 1
// Hier ist a == 2
System.out.println(++a); // Output: 3
```

Variable verringern um 1

`int i = 1;`

- `i--;` // gleich wie `i = i - 1;`
- `--a` verringert die Variable und verwendet erhöhte
- `a--` verwendet Variable und verringert sie anschließend

```
int a = 3
System.out.println(a--); // Output: 3
// Hier ist a == 2
System.out.println(--a); // Output: 1
```

Übung Variable

Ausgaben nennen:

- Integer Variable `a` mit Wert 10 anlegen;
- `System.out.println(--a);`
- `System.out.println(a);`
- `System.out.println(a++);`
- `System.out.println(a);`

Typumwandlung

- Ändern des Datentyps, um kompatibel zu sein
- **IMPLIZIT**
  - ein "kleinerer" Datentyp kann in größeren gespeichert werden
  - `byte a = 17;`
  - `int b = a;`
- **EXPLIZIT**
  - "größerer" Datentyp in "kleineren" stecken.
  - Werte können verloren gehen
  - `int c = 100;`
  - `byte d = (byte) c;` ⇐ **TYPECAST**
  -

Typumwandlung

```
int c = 127;
byte d = (byte) c; // funktioniert, Ergebnis ist richtig

int e = 200;
byte f = (byte) e; // funktioniert, Ergebnis ist -56
```

- **Byte Overflow**
  - 1 Byte = 2^8 Bit = 256 Bit (-127 bis +128 inklusive 0) = **256 mögliche Werte**
  - 200-256 = **-56**

Typumwandlung von double in int

```
double a = 123.35;
int b = (int) a; // 0.35 wird abgeschnitten => verloren
```

- von double zu int

```
int a = 123;
double b = a; // funktioniert
```

Typumwandlung von Chars

- Recap: char=1 Byte und byte=1 Byte

```
char a = 'a';
byte aAsByte = (byte) a;
System.out.println(aAsByte); // 97
```

ASCII TABLE

Letter	ASCII Code	Binary	Letter	ASCII Code	Binary
a	097	01100001	A	065	01000001
b	098	01100010	B	066	01000010
c	099	01100011	C	067	01000011

Typumwandlung von Chars

- Recap: char=1 Byte und byte=1 Byte

```
byte aAsByte = (byte) a;
System.out.println(aAsByte); // 97

// hier Umwandlung in byte erforderlich, da es zu Overflow führen könnte
aAsByte = (byte)(aAsByte+1); // 98

char aAsChar = (char) aAsByte;
System.out.println(aAsChar); // b
```