Java-Applikationen (Java-Programme)

Eine erste Applikation:

```
1  // Quelltext  HalloWelt.java
2  // Programm gibt den Text Hallo Welt aus
3
4  public class HalloWelt
5  {
6    public static void main(String[] args)
7    {
8      System.out.println("Hallo Welt !!");
9    }                                    // End of main
10 }                                      // End of class
```

Die Zeilennummern gehören nicht zum Quelltext !

In den Zeilen 1 und 2 stehen Kommentare, die zur besseren Lesbarkeit des Quelltextes dienen. Sie werden vom Compiler ignoriert.

In Java gibt es mehrere Arten von Kommentaren:

- **Die Zeichenfolge // kann irgendwo in einer Zeile stehen und kennzeichnet den folgenden Text bis zum Zeilenende als Kommentar.**
- **Mehrzeilige Kommentare werden durch die Zeichen /* .... */ begrenzt.**

Die Klasse, die main enthält muss öffentlich sein

Jedes Java-Programm ist eine Klasse

Programmname, d.h. Name der öffentlichen Klasse

```
public class HalloWelt
{
```

Beginn der Klassendefinition

Definition einer Methode mit dem Namen main

Parameterliste (arguments) für die Methode main

```
public static void main(String[ ] args)
```

Die Methode ist öffentlich, statisch und gibt nichts zurück. (void : leer)

Die variable args ist ein Array (Feld) dessen Komponenten Stringvariablen sind, in denen die Parameter gespeichert sind. args[0];args[1];..... siehe auch args

```
{
```

Beginn der Methodendefinition

```
System.out.println("Hallo Welt !!");
```

Dies ist eine Kommandozeile, die mit ; abgeschlossen wird.

Aufruf der Methode println im Objekt System.out

Kommentar bis Zeilenende

```
}
```

Ende der Methodendefinition main

```
// End of main
```

```
}
```

Ende der Klassendefinition d.h. Programmende

```
// End of class
```

In der 4. Zeile wird die Klasse HalloWelt deklariert.

Klassennamen sollten großgeschrieben und ggf. durch Großbuchstaben strukturiert werden.

Der Klassenkörper wird durch geschweifte Klammern in Zeile 5 und 10 als Block zusammengefasst.

Einziges Mitglied der Klasse ist im Beispiel die Methode namens main, die in Zeile 6 deklariert wird.

Die Methode main ist

- **public d.h. sie ist öffentlich in dem Sinn, dass von außen auf sie zugegriffen werden kann.**
- **static d.h. auf diese Methode kann zugegriffen werden, ohne dass schon eine Instanz der betreffenden Klasse erzeugt wurde. Wenn ein Java-Programm gestartet wird, können noch keine Instanzen von Klassen vorliegen, daher muss die Methode main immer static sein!**

Der Methodenkörper wird ebenfalls durch geschweifte Klammern in Zeile 7 und 9 als Block zusammengefasst.

Üblicherweise beginnen Methodennamen mit einem Kleinbuchstaben.

main() enthält nur eine Anweisung, die den Text "Hallo Welt !!" ausgibt. Dies wird durch Aufruf der Methode println() erreicht.

Am Ende einer jeden Anweisung steht in Java ein Semikolon.

Damit der Quelltext verständlich wird:

- **Pro Zeile nur eine Anweisung**
- **Einrückungen von Klassenkörper gegen Klassenkopf, von Methodenkörper gegen Methodenkopf um eine feste Anzahl von Spalten (z.B. 2).**
- **Die den Block schließende Klammer steht in einer eigenen Zeile in der Spalte des ersten Zeichens der Struktur.**

  (Die öffnende Klammer steht meistens unmittelbar hinter dem Klassenkopf, kann aber auch unmittelbar unter dem ersten Zeichen des Klassenkopfes stehen.)

```
public class HalloWelt {

    public static void main(String[] args) {
      System.out.println("Hallo Welt !!");
    }
}
```

oder:

```
public class HalloWelt
{
      public static void main(String[] args)
      {
         System.out.println("Hallo Welt !!");
      }
}
```

Namen (Bezeichner)

Java verwendet den Unicode-Zeichensatz der 65536 verschiedene Zeichen aller gängigen Weltsprachen enthält.

Vorsicht: Trotzdem sollten in Java-Programmen Umlaute und ß nicht verwendet werden! Die DOS-Kommandozeile kennt diese Zeichen nicht, da sie einen 7 Bit-Zeichensatz verwendet.

Regeln zur Bildung von Namen:

- **Das erste Zeichen darf keine Ziffer sein, Buchstaben, '$' und '_' sind erlaubt**
- **weitere Zeichen dürfen auch Ziffern sein, jedoch keine Leerzeichen.**
- **Übereinstimmung mit *reservierten Worten* (class, int, char, public,...) sind verboten.**
- **Gross/Kleinschreibung wird unterschieden: eine Rose ist keine rose und keine ROSE und keine RoSe.**

Zur besseren Lesbarkeit sollen lange Namen durch Großbuchstaben strukturiert werden:
z.B. RechnungsDatum ; SchuelerWohnort

Klassennamen sollten mit Großbuchstaben, Variablen- und Methodennamen mit Kleinbuchstaben beginnen.

Struktur einer Java-Applikation

Applikationen enthalten immer die *Deklaration einer Klasse* (1. Zeile in folgendem Beispiel), in der zumindest die *Methode* main() (3. Zeile) deklariert ist. Die Ausführung einer Applikation ist gleichbedeutend mit der Ausführung der Methode main().

Java-Applikationen sind also selbst Klassen !

```
class KlassenName
{
   public static void main (String[] args)
    {
      // hier stehen die Anweisungen
    }
}
```

Anweisungen und Ausdrücke

Eine *Anweisung* (Statement) ist ein einzelner Befehl einer Programmiersprache, der dafür sorgt, dass etwas passiert.

Beispiele für einfache Java-Anweisungen :

```
int zahl = 30 ;
import java.io.* ;
System.out.println("Hallo Welt !!") ;
a=0.15 ;
```

Ein Ausdruck (engl. expression) ist eine Anweisung, die als Ergebnis einen Wert produziert.

Dieser Wert kann

- **zur späteren Verwendung im Programm gespeichert werden,**

  Bsp.: z=x+y; //Der Wert des Ausdrcks x+y wird als Wert der //Variablen z übergeben

- **direkt in einer anderen Anweisung verwendet werden**

  Bsp.: System.out.println(x+y)

  // Der Wert von x+y wird ausgegeben

- **oder überhaupt nicht beachtet werden.**

  x+y; // wenig sinnvoll !

Der Wert, den eine Anweisung erzeugt, wird Rückgabewert genannt.

Manche Ausdrücke erzeugen numerische Rückgabe-werte, andere erzeugen einen booleschen Wert (true oder false) oder aber kompliziertere Datenstrukturen, sogenannte Objekte.

Variable

In der Mathematik stehen Variablen als Platzhalter für einen Wert. In Programmen sind Variablen dagegen **Speicher für Werte**.

Entsprechend bedeutet die Anweisung `x = 7;`

dass im Speicher x der Wert 7 abgelegt wird. Links von = muss somit eine Variable, rechts kann eine *Konstante (s. unten)*, eine *Variable* oder ein *Ausdruck* stehen.
Fehlerhaft wäre 7 = x !

Um eine Variable zu erstellen, muss Sie ***deklariert*** werden d.h.

- sie erhält einen Namen
- und eine Typzuweisung, damit der Compiler weiß, wie viel Speicherplatz er für diese Variable reservieren muss.

Bsp.: `int Zahl;` `/* Die Variable hat den Namen Zahl und ist vom Typ int (Integer Zahl) für die 2 Byte=16 Bit Speicherplatz reserviert werden müssen. */`

Sie können einer Variablen auch bei der Erzeugung direkt einen Wert zuweisen (***Initialisierung***).

Bsp.:
```
int plz = 54516;
String wohnort = "Wittlich";
boolean versetzt = true;
int alter = 18,  gewicht_in_kg= 70, groesse = 161;
char geschlecht='w'
```

Der Gültigkeitsbereich einer Variablen erstreckt sich über den Block, in dem sie deklariert wurde.

Außerhalb dieses Blocks kann die Variable nicht verwendet und auch nicht verändert werden.

Wenn mehrere Variablen desselben Typs deklariert werden sollen, können diese in einer einzigen Zeile aufgeführt werden. Dazu trennt man die einzelnen Variablennamen mit Kommas.

Die folgende Anweisung erzeugt drei Double-Variablen mit den Namen zahl1, zahl2 und zahl3:

```
Double zahl1 , zahl2 , zahl3 ;
```

Verwenden Sie zur besseren Lesbarkeit des Quelltextes „sprechende“ Variablennamen,

z.B. schuelerName statt s
primzahl statt p

Primitive Datentypen

JAVA kennt 8 verschiedene einfache Datentypen:

Typ	Wertebereich	Bit
boolean	true, false	1
char	Unicode-Zeichen con \u0000..... \uFFFF	16
byte	-128.....127	8
short	- 32768..... 32767	16
int	- 2147483648..... 2147483647	32
long	- 92 233 772 036 854 775 808 ..... 92 233 772 036 854 775 807	64
float	$-3{,}40282347 \cdot 10^{38}$ ..... $3{,}40282347 \cdot 10^{38}$	32
	$-1{,}40239846 \cdot 10^{-45}$ ..... $1{,}40239846 \cdot 10^{-45}$	
double	$-1{,}7976931348623157 \cdot 10^{308}$ .... $1{,}7976931348623157 \cdot 10^{308}$	64
	$-4{,}9406564584124654 \cdot 10^{-324}$ .... $4{,}9406564584124654 \cdot 10^{-324}$	

Vorsicht: Bereichsüberschreitungen führen zu Fehlern !

Beispiel:

```
Byte  x ;       // Deklariert eine Variable x vom Typ Byte
X=127+1;        // berechnet die Summe 127+1=128
                // und weist das Ergebnis der Variablen x zu
System.out.println(x);  // gibt den Wert  –1  aus
```

Erklärung: Zum Abspeichern einer Variablen vom Typ byte werden 8 Bit verwendet. Das führende Bit (MSB most significant bit) ist das Vorzeichenbit. Steht dort eine 1, so ist die Zahl negativ, bei 0 ist sie positiv.
Die größte positive Zahl, die sich mit 8 Bit abspeichern lässt ,ist also 127

0	1	1	1	1	1	1	1

Addiert man zu dieser Zahl 1 so erhält man

1	0	0	0	0	0	0	0

Die Zahl ist negativ und hat den Dezimalwert -1

8-Bit-Darstellung positver und negativer ganzer Zahlen:

0	0 0 0 0 0 0 0 0
1	0 0 0 0 0 0 0 1
2	0 0 0 0 0 0 1 0
3	0 0 0 0 0 0 1 1
	. . . .
127	0 1 1 1 1 1 1 1
-128	1 1 1 1 1 1 1 1
-127	1 1 1 1 1 1 1 0
-126	1 1 1 1 1 1 0 1
	. . . .
-1	1 0 0 0 0 0 0 0

char

Der Datentyp char wird für einzelne Zeichen, wie z.B. Buchstaben, Ziffern, Interpunktionszeichen und andere Symbole verwendet, da Java den Unicode-Zeichensatz unterstützt.

```
char z;
z='a';      //weist der Variablen z das Zeichen a zu.
```

Nichtsichtbare Zeichen und andere Steuerzeichen müssen durch sogenannte *Escape-Sequenzen* dargestellt werden, die mit \ eingeleitet werden und aus insgesamt zwei Zeichen bestehen. Ein Beispiel ist '\n', das Zeichen für einen Zeilenvorschub (Line Feed).

Escape-Sequenz	Bedeutung
\n	LF (Neue Zeile)
\t	TAB (Tabulator)
\b	BS (Rückschritt (Backspace))
\ r	CR (Wagenrücklauf (Carriage return)
\f	FF (Seitenvorschub (Formfeed))
\\	Inverser Schrägstrich (Backslash)
\'	Einfache Anführungszeichen
\"	Doppelte Anführungszeichen
\d	Oktal
\xd	Hexadezimal
\ud	Unicode-Zeichen

boolean

Boolesche Variablen speichern entweder den Wert true der den Wert false.

Typnamen beginnen mit Kleinbuchstaben und müssen auch in dieser Form in Programmen verwendet werden.

Vorsicht :Es gibt Klassen in Java, die denselben Namen wie einige dieser Datentypen besitzen, allerdings mit anderer Groß-/Kleinschreibung - z.B. Boolean und Char. Diese haben eine andere Funktionalität in einem Java-Programm, sodass diese nicht im Austausch füreinander verwendet werden können.