



Einführung in die Programmierung

- JavaScript & Java -

Claudia Scheffler

19. September 2013



- 1 Eigenschaften von Java
- 2 Java Plattform
- 3 Hello World!
- 4 Grundlagen
 - Kommentare
 - Datentypen und Variablen
 - Typumwandlung
 - Operatoren
 - Kontrollstrukturen
 - Array
 - String



- 1 Eigenschaften von Java
- 2 Java Plattform
- 3 Hello World!
- 4 Grundlagen
 - Kommentare
 - Datentypen und Variablen
 - Typumwandlung
 - Operatoren
 - Kontrollstrukturen
 - Array
 - String



- 1 Eigenschaften von Java
- 2 Java Plattform
- 3 Hello World!
- 4 Grundlagen
 - Kommentare
 - Datentypen und Variablen
 - Typumwandlung
 - Operatoren
 - Kontrollstrukturen
 - Array
 - String



- 1 Eigenschaften von Java
- 2 Java Plattform
- 3 Hello World!
- 4 Grundlagen
 - Kommentare
 - Datentypen und Variablen
 - Typumwandlung
 - Operatoren
 - Kontrollstrukturen
 - Array
 - String



Fragen ?



Eigenschaften von Java



- einfach zu erlernende Programmiersprache im Gegensatz zu C/C++
- es ist vorgesehen, dass objektorientiert programmiert wird
→ ist dadurch gekennzeichnet, dass keine Methode außerhalb von Klassen programmiert werden kann
- bietet eine sehr hohe Sicherheit, da unbeabsichtigter Zugriff auf Arbeitsspeicherbereiche und dessen Manipulation durch den Interpreter verhindert wird
- **Vorteil:**
 - Plattformunabhängig → ein Java-Programm läuft unter jedem beliebigen Betriebssystem gleich



- Java Runtime Environment (JRE)
- diese Laufzeitumgebung beinhaltet die Java Virtual Maschine (JVM)

Java Virtual Maschine

Die virtuelle Maschine ist ein Programm, welches von Betriebssystem abhängig ist. Des weiteren interpretiert sie den erzeugten Java-Bytecode für das verwendete Betriebssystem.

- Java-Bytecode ist somit unabhängig vom Betriebssystem



Fragen ?



Hello World!



- aufgrund der zugrunde liegenden Objektorientierung immer in Klassen
- eine Klasse allein ist jedoch kein ausführbares Programm
→ damit ein ausführbares Programm entsteht, ist es notwendig eine Klasse mit der **main()** - Methode zu implementieren

Beispiel

```
public class HelloWorld {  
    public static void main(String[] args)  
    {  
        System.out.println(" Hello World!");  
    }  
}
```



Fragen ?



- ein Quellcode kann mit kommentierten Text versehen werden, der vom Compiler ignoriert wird
- **Vorteil:**
 - Programmcode wird nachvollziehbar und strukturiert
 - bestimmte Abläufe sind auch im Nachhinein noch verständlich
 - bei der Zusammenarbeit von mehreren Personen

Syntax für einzeiligen Kommentar

```
// Dies ist ein einzeiliger Java-Kommentar bis zum Zeilenende
```

Syntax für mehrzeiligen Kommentar

```
/* Dies ist ein Kommentarblock und  
kann sich über mehrere Zeilen erstrecken */
```



Fragen ?



Datentypen und Variablen

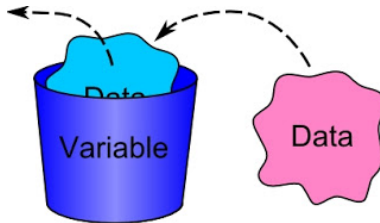
Was ist eine Variable?



Variable

Eine Variable ist ein "Behälter" zur Speicherung von Informationen.

- eine Variable kann man sich als ein Gefäß für veränderliche Daten vorstellen





Datentypen und Variablen

Namensgebung



- Variablennamen bestehen aus einer Folge von:
 - Buchstaben
 - Ziffern
 - Unterstrichen (_)
- Variablennamen beginnen entweder mit einem Buchstaben, Dollarzeichen (\$) oder Unterstrich (_)

Achtung!

Das erste Zeichen darf jedoch keine Ziffer sein!

Tipp!

Symbole wie %, *, @ usw. sind in Java meist für Operatoren reserviert und sollten deshalb in Variablennamen nicht verwendet werden.



Datentypen und Variablen

Namensgebung



- Variablennamen sind **Case Sensitiv**

Case Sensitiv

- Unter Case Sensitiv versteht man die Beachtung der Groß- und Kleinschreibung der Daten.
- **Beispiel:** *java, Java, jAvA*
- Schlüsselwörter und -begriffe dürfen nicht als Variablennamen verwendet werden
- Variablen haben grundsätzlich aussagefähige Namen, die aus mehreren zusammengesetzten Wörtern bestehen



Datentypen und Variablen

Namensgebung



- das erste Wort des Variablennamens ist klein geschrieben und alle anderen ohne Leerzeichen folgende Wörter beginnen mit großen Anfangsbuchstaben

Beispiel

```
int myAge;  
String myName;  
boolean isTired;
```



Datentypen und Variablen

Variablendeklaration



Variablendeklaration

Die Deklaration bezeichnet die Vereinbarung eines Namens für einen Speicherplatz des Rechners. Der Programmierer teilt also dem Rechner mit, dass er in seinem Programm eine Variable benötigt.

- **streng typisierte** Programmiersprache, d.h. das der Datentyp der Variable mit angegeben werden muss
- über den Datentyp wird festgelegt, welche Werte die Variable annehmen kann

Syntax

Datentyp Variablenname;



Datentypen und Variablen

Variablendeklaration



- Wertzuweisung einer Variable erfolgt über das Gleichheitszeichen =

Syntax

```
Datentyp Variablenname = Wert;
```

Beispiel

```
int variablenName = 12;  
char name = 'Hallo';
```



Datentypen und Variablen

Variableninitialisierung



Variableninitialisierung

Der Begriff Initialisierung bezeichnet die erste Zuweisung eines Wertes an eine Variable.

- erforderlich bei komplexen Datentypen, damit auf sie zugegriffen werden kann
- einfache Datentypen benötigen nicht zwingend eine Initialisierung → da sie standardmäßig durch den Compiler mit dem Wert 0 initialisiert werden



Fragen ?



Datentypen und Variablen

Datentyp



- in Java wird zwischen **primitiven** und **komplexen Datentypen** unterschieden
- Primitive Datentypen:
 - Boolean
 - char (Character)
 - byte
 - short
 - int (Integer)
 - long
 - float
 - double



Datentypen und Variablen

Datentyp-Boolean & Datentyp-Charakter



■ **boolean:**

- nimmt genau zwei Werte an:
 - true (wahr) und false (falsch)
- wird für bedingte Anweisungen sehr häufig verwendet

Beispiel

```
boolean fehlerAufgetreten = false;
```

■ **char:**

- Attribut von diesem Datentyp kann alle Unicode-Zeichen enthalten
- max. Länge umfasst 16 Bit
- der Werte Bereich reicht von 0 bis 65535

Beispiel

```
char zeichen = 'A';
```



Datentypen und Variablen

Datentyp-byte & Datentyp-short



■ byte:

- Wertebereich (8 Bit) reicht von $-128(-2^7)$ bis $+127((2^7) - 1)$
- wird häufig im Bereich der Datendatei- oder der grafischen Datenverarbeitung verwendet

■ short:

- Wertebereich ist doppelt so groß wie der des Datentyps byte
- in Dezimal ausgedrückt fängt der vorzeichenbehaftete bei $-32768 (-2^{15})$ an und endet bei $32767 ((2^{15}) - 1)$

Beispiel

```
byte byteVar = 0x000A;  
short shortVar = 077;
```



Datentypen und Variablen

Datentyp-int & Datentyp-long



■ int:

- hat eine Größe von 32 Bit
- Wertebereich von dezimal -2^{31} bis $2^{31} - 1$

Beispiel

```
int varZahl = 23;
```

■ long:

- doppelt so groß wie der Datentyp int
- Wertebereich liegt somit zwischen -2^{63} und $2^{63} - 1$

Beispiel

```
long longVar1 = 5556547842L;
```



Datentypen und Variablen

Datentyp-float & Datentyp-double



■ float:

- Datentyp für Fließkommazahlen
- hat eine Größe von 32 Bit
- der Wertebereich reicht von $-3,4 * 10^{38}$ bis $3,4 * 10^{38}$

Beispiel

```
float floatVar = 3.14E5F;
```

■ double:

- hat eine Größe von 64 Bit
- Wertebereich erstreckt sich von $-1,7 * 10^{308}$ bis $1,7 * 10^{308}$

Beispiel

```
double doubleVar = 3.14E5;
```



Datentypen und Variablen

Komplexe Datentypen



- jede Klasse, die über mehrere Attribute verfügt, definiert einen komplexen Datentyp
- setzen sich aus verschiedenen Elementen zusammen
- **Beispiel:** String
 - String-Datentyp repräsentiert Zeichenketten



Fragen ?



Typumwandlung

Type-Casting mit primitiven Datentypen



- Unterscheidung:
 - expliziten Typumwandlung
 - impliziten Typumwandlung
- **Implizite Typumwandlung:**
 - findet automatisch bei der Zuweisung statt
 - ist nur möglich, wenn ein niederwertiger Datentyp in einen höher wertigen Datentypen umgewandelt wird
 - **Beispiel:** vom int in den long.

Beispiel

```
int wert = 10;  
long wert2 = 30;  
wert2 = wert;
```



Typumwandlung

Type-Casting mit primitiven Datentypen



- man spricht von einem sogenannten **Typecast**
- **Explizite Typumwandlung:**
 - erfolgt durch den cast-Operator
 - bei dieser Variante wird von einem höher wertigen Datentyp in einen nieder wertigen Datentypen umgewandelt
 - in welchen Datentyp umgewandelt werden soll, muss beim **cast**-Operator angegeben werden

Beispiel

```
int wert = 10;  
float wert2 = 30.5f;  
wert = ( int ) wert2;;
```



Typumwandlung

Type-Casting mit primitiven Datentypen



Konvertierung in einen höher wertigen Datentyp

double

float

long

int

short

byte

char



Typumwandlung

Type-Casting mit primitiven Datentypen



Konvertierung in einen niederwertigen Datentyp

double

float

long

int

short

byte

char



Typumwandlung

Type-Casting mit komplexen Datentypen



- zwischen Klassen muss eine Vererbungsbeziehung bestehen
- Objekte gehören nicht nur ihrer eigenen Klasse, sondern automatisch auch der Oberklasse an → ohne **cast**-Operator
- jedes Objekt kann in den Datentyp Object umgewandelt werden
- umgekehrte Weg ist nicht immer problemlos möglich



Typumwandlung

Type-Casting mit komplexen Datentypen



Beispiel

```
String arr1= "123456";  
Object ref1= arr1;  
String []arr2 = (String[]) ref1;  
System.out.println("Zeichenkette arr1:" + arr1+ " wird zu "  
+ arr2);
```



Fragen ?



Operatoren

Arten von Operatoren



Operatoren

- Durch Operatoren können Variablen verändert oder mehrere Variablen miteinander verknüpft werden.
- Arten:
 - **Arithmetische Operatoren:** $+$, $-$, $*$, $/$, $\%$, $\neg$
 - **Inkrementierungs- & Dekrementierungsoperatoren:**
 $++$, $--$
 - **Boolesche Operatoren:** $\&\&$, $\|\|$, $\&$, $|$, $!$, $\wedge$
 - **Vergleichsoperatoren:** $==$, $!=$, $<$, $>$, $<=$, $>=$
 - **String-Operatoren:** $+$
 - **Sonstige Operatoren:** `new`, `instanceof`, `cast`



Fragen ?



Kontrollstrukturen

Eine Kontrollstruktur ist ein wesentliches Element zur Strukturierung von Computerprogrammen und regelt, in welcher Abfolge die im Quelltext aufgeführten Befehle abgearbeitet werden.

- Kontrollstrukturen in JavaScript:
 - Anweisungsblöcke
 - For-Schleife
 - While-Schleife / Do-while-Schleife
 - If-Anweisung
 - Switch-Case-Anweisung



- einfache Kontrollstruktur
- sie dienen zur Zusammenfassung von Anweisungen
- werden durch geschweifte Klammern dargestellt

Syntax

```
{ Anweisung1; Anweisung2; ...; AnweisungN; }
```

Syntax

```
{  
Anweisung1;  
Anweisung2;  
...;  
AnweisungN;  
}
```



- diese Art von Schleife führt eine Gruppe von Anweisung (Block) mit einer bestimmten Anzahl von Wiederholungen aus

Syntax

```
for (Initialisierung; Bedingung; Befehlsfolge)
{
    Anweisung;
}
```



■ Initialisierung:

- Anweisung in dem der Schleifenzähler definiert und initialisiert wird
- **Mit einfachen Worten:** Zählvariable, die die Anzahl der Wiederholungen zählt

■ Bedingung:

- die Schleife wird solange ausgeführt, bis diese Bedingung nicht mehr erfüllt ist.

■ Befehlsfolge:

- nach jedem Durchlauf der Anweisung wird diese Befehlsfolge ausgeführt



Beispiel

```
for(int i=0; i<5; i++)  
{  
    System.out.println("i ist " + i);  
}
```

Ergebnis:

```
i ist 0  
i ist 1  
i ist 2  
i ist 3  
i ist 4
```



Fragen ?



- die While-Schleife dient dazu, eine Abfolge von Anweisungen mehrfach auszuführen, solange eine Bedingung erfüllt ist
- die Überprüfung der Bedingung erfolgt vor der Abarbeitung des Anweisungsblockes

Syntax

```
while (Bedingung)
{
Anweisungsblock;
}
```



Beispiel

```
int i=5;
while(i>0)
{
    System.out.println("i ist "+i);
    i--;
}
```

Ergebnis:

```
i ist 5
i ist 4
i ist 3
i ist 2
i ist 1
```



- die Do-While-Schleife ist eine Variante der While-Schleife
- der Unterschied zur While-Schleife:
 - Anweisungsblock wird einmal ausgeführt und anschließend wird überprüft, ob die Bedingung erfüllt ist
 - danach wird die Schleife solange wiederholt, bis die Bedingung nicht mehr korrekt ist

Syntax

```
do  
{  
Anweisungsblock;  
} while (Bedingung);
```



Beispiel

```
int i = 0;  
do  
{  
    System.out.println("i ist " + i);  
    i += 10;  
}  
while(i <= 20);
```

Ergebnis:

```
i ist 0  
i ist 10  
i ist 20
```



Fragen ?



Kontrollstrukturen: Fallunterscheidung

If-Anweisung

- eine If-Anweisung wird unter bestimmten Anweisungen ausgeführt
- die Verzweigung legt fest, welcher von zwei oder mehreren Anweisungsblöcken, abhängig von einer oder mehreren Bedingungen ausgeführt wird

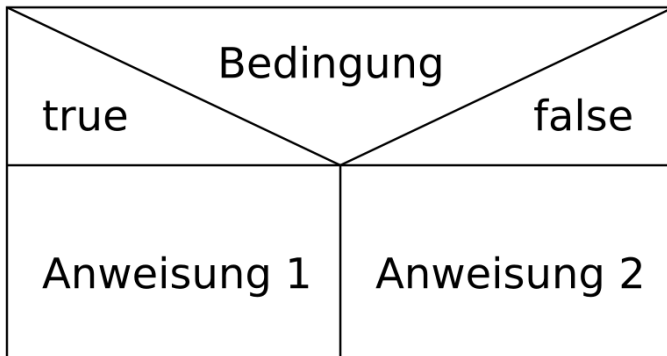
Syntax

```
if (Bedingung)
{
Anweisungsblock;
} else {
Anweisungsblock;
}
```



Kontrollstrukturen: Fallunterscheidung

If-Anweisung





Beispiel

```
int a=5;  
int b=6;  
if(a<b){  
  a=b;  
} else {  
  b=a;  
}
```



Kontrollstrukturen: Fallunterscheidung

Switch-Case-Anweisung

- Mithilfe dieser Anweisung ist es möglich mehr als nur zwei Unterscheidungen, wie bei der If-Anweisung, zu differenzieren

Syntax

```
switch (Ausdruck){  
  case Wert1:  
    Anweisungsblock;  
    break;  
  case Wert2:  
    Anweisungsblock;  
    break;  
  default:  
    Anweisungsblock;  
}
```



Kontrollstrukturen: Fallunterscheidung

Switch-Case-Anweisung

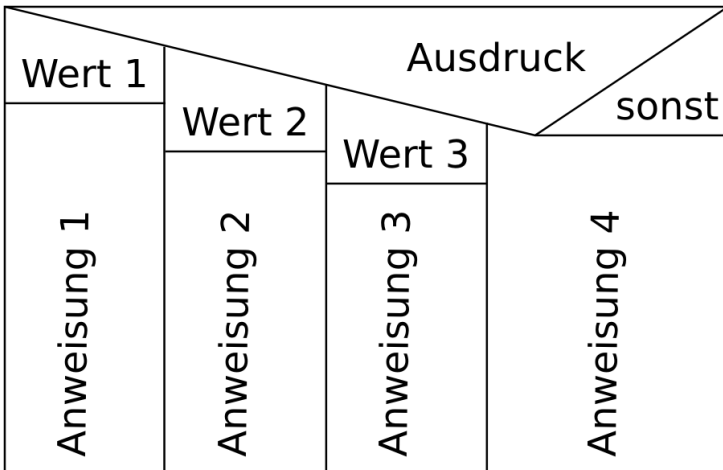
Achtung!

Das Schlüsselwort **break** darf am Ende der Switch-Case-Anweisung nicht fehlen, sonst wird die Überprüfung des Ausdrucks im passenden Fall nicht abgebrochen.



Kontrollstrukturen: Fallunterscheidung

Switch-Case-Anweisung





Beispiel

```
int x = 5;
switch(x) {
case 2: break;
case 3: //Mach irgendwas
break;
case 4:
case 5: System.out.println("Wert ist gleich 4 oder 5");
break;
default: // Mach irgendwas standardmäßiges
}
```



Fragen ?



Array

Was sind Arrays?



Array

Ein Array ist eine zusammengesetzte Datenstruktur, die mehrere Werte in getrennten und direkt adressierbaren Elementen zusammenfasst, um so thematisch zusammengehörende Variablen in einer Sammlung zu verbinden.

- zwei Arten von Arrays:
 - Eindimensionale Arrays
 - Mehrdimensionale Arrays



Array

Eindimensionale Arrays



■ einfache Liste

0	1	2	3	Index
Klaus	Mona	Hans	Lena	Daten

- werden mit einem Datentyp deklariert → aller Werte, die in diesem Array gespeichert werden, müssen vom selben Datentyp sein, mit dem das Array deklariert wurde
- erhält eine festgelegte unveränderbare Größe



Array

Deklaration eines Arrays



Syntax

```
Datentyp [ ] Arrayname = new Datentyp[Länge];  
Datentyp Arrayname [ ] = new Datentyp[Länge];
```

- charakteristisch für ein Array sind die eckigen Klammern
- **Position:**
 - zwischen dem Datentyp und dem Namen des Array
 - nach dem Namen des Arrays
- bei einem Array handelt es sich um einen komplexen Datentyp → aus diesem Grund benötigt man zur Erzeugung eines Arrays den **new**-Operator



Array

Initialisierung eines Elementes



- jedes Element des Arrays hat eine Nummer → den **Index**

Tipp!

Das erste Element hat immer den Index 0, das zweite den Index 1 usw.

Beispiel

```
nameDesArrays[Index] = Wert;
```



Array

Initialisierung eines Elementes



- mit der Initialisierungsliste ist es möglich, bereits bei der Erstellung des Arrays den Elementen in einem Schritt Werte zuzuweisen
- dann ist aber **kein** new-Operator erforderlich

Tipp!

Das erste Element hat immer den Index 0, das zweite den Index 1 usw.

Syntax

```
Datentyp [ ] Arrayname = {Element 0, Element 1, Element 2,  
Element n-1, Element n};
```



Array

Initialisierung eines Elementes



Beispiel

```
int [ ] arr1=new int[5];  
int [ ] arr2 = {1,2,3,4,5};
```



Array

Zugriff auf die Elemente



- um auf ein Element des Arrays zuzugreifen, ist es notwendig nach dem Array-Namen in eckigen Klammern den Index des Elementes anzugeben

Beispiel

```
String [ ] namenDesArrays= {"Gustaf", "Hans", "Klaus",  
"Maria", "Gustaf"};  
String name = namen[3];
```



Fragen ?



Array

Mehrdimensionale Arrays



- ineinander geschachtelte Arrays
 - Elemente der ersten Dimension sind Arrays, die selber wieder Arrays der zweiten Dimension enthalten usw.
 - jede Dimension wird durch ein Paar eckiger Klammer gekennzeichnet

Beispiel

```
double [ ] [ ] tabelle = new double [20] [10];
```



Array

Initialisierung und Zugriff



- bei der Initialisierung oder dem herauslesen eines Elementes müssen die kompletten Indizes angegeben werden

Beispiel

```
tabelle[11][3] = 2.74;  
double element = tabelle[11][3];
```

- eine Initialisierungsliste bei mehrdimensionalen Arrays ist ebenfalls möglich

Beispiel

```
int zahlenArr [ ] [ ] = { {4,7,2}, {8,5,3,1}, {6,9}, {3,6,4} };
```



Fragen ?



String



- der am meisten verwendete Datentyp
- String-Datentyp repräsentiert Zeichenketten
- Zeichenketten werden mit doppelten Ausführungszeichen umgeben
- Länge einer Zeichenkette ist variabel
- String-Objekt kann mit dem new-Operator erstellt werden
→ ist aber nicht erforderlich, da er bei der Initialisierung implizit aufgerufen wird

Beispiel

```
String zeichenkette1 = new String("Einfacher Text");  
String zeichenkette2 = "Einfacher Text";
```



String



- ein Array von einzelnen Zeichen (char)

Achtung

Aus diesem Grund ist zu beachten, das auch bei den Strings, das erste Zeichen immer den Index 0 hat. Und der Index des letzten Zeichen entspricht daher der Länge der Zeichenkette - 1.



Fragen ?

Claudia Scheffler
siclsche@stud.hszg.de