# Java

## Kontrollstatements and OOP

Tobias Hanf, Manik Khurana

25. Oktober 2021

Java-Course

# Übersicht

1. Wiederholung
    - Rechnen
    - Text mit Strings
2. Input
3. Control Statements
    - Ite
    - for
    - while
4. Methoden

# Wiederholung

# Blocks

```
1          public class Hello {
2              // gibt "Hello, world!" in der Konsole aus
3              public static void main(String[] args) {
4                  System.out.println("Hello World!");
5              }
6              }
7
```

Alles zwischen { und } ist Inhalt eines *Blockes*.
Blöcke können geschachtelt sein.

# Benennung von Variablen

- Ein Variablenname kann mit einem Buchstaben oder einen _ Underscore beginnen.
  Normalerweise beginnen sie mit einem kleinen Buchstaben.
- Für zusammengesetzte Namen verwendet man die CamleCase-Notation.
- Es sollten immer aussagekraeftige Namen verwendet werden.

```
1     public class Calc {
2         public static void main(String[] args) {
3             int a = 0; // nicht sehr aussagekraeftig
4             float myFloat = 5.3f; // auch nicht sehr
  aussagekraeftig
5             int count = 7; // ein besserer Name
6
7             int rotationCount = 7; // so sollte es sein
8         }
9     }
10
```

# Primitive Datentypen

Java unterstützt einige primitive Datentypen:

| | |
|---|---|
| boolean | ein Wahrheitswert (entweder **true** oder **false**) |
| int | a 32 bit integer (ganze Zahl) |
| long | a 64 bit integer |
| float | a 32 bit floating point number (Gleitkommazahl) |
| double | a 64 bit floating point number |
| char | an unicode character |
| void | leerer Typ (für Später) |

# Rechnen mit *int* i

```
1   public class Calc {
2       public static void main(String[] args) {
3           int a;
4           a = 7;
5           System.out.println(a);
6           a = 8;
7           System.out.println(a);
8           a = a + 2;
9           System.out.println(a);
10      }
11  }
12
```

# Rechnen mit *int* ii

```
1          public class Calc {
2              public static void main(String[] args) {
3                  int a; // deklariert die Variable a
4                  a = 7; // weis a den Wert 7 zu
5                  System.out.println(a); // prints: 7
6                  a = 8;
7                  System.out.println(a); // prints: 8
8                  a = a + 2;
9                  System.out.println(a); // prints: 10
10             }
11         }
12
```

Nach der ersten Zuweisung ist eine Variable vollständig initialisiert.

# Rechnen mit *int* iii

```
1   public class Calc {
2       public static void main(String[] args) {
3           int a = -9;
4           int b;
5           b = a;
6           System.out.println(a);
7           System.out.println(b);
8           a++;
9           System.out.println(a);
10      }
11  }
12
```

# Rechnen mit *int* iv

```
1   public class Calc {
2       public static void main(String[] args) {
3           int a = -9; // Deklaration mit Zuweisung
4           int b; // declaration of b
5           b = a; // assignment of b
6           System.out.println(a); // prints: -9
7           System.out.println(b); // prints: -9
8           a++; // increments a
9           System.out.println(a); // prints: -8
10      }
11  }
12
```

# Rechnen mit *int* v

Einige grundlegende Rechenoperationen:

| | |
|---|---|
| Addition | `a + b;` |
| Subtraction | `a - b;` |
| Multiplication | `a * b;` |
| Division | `a / b;` |
| Modulo | `a % b;` |
| Increment | `a++;` |
| Decrement | `a--;` |

# Rechnen mit *float* i

```
1          public class Calc {
2              public static void main(String[] args) {
3                  float a = 9;
4                  float b = 7.5f;
5                  System.out.println(a); // prints: 9.0
6                  System.out.println(b); // prints: 7.5
7                  System.out.println(a + b); // prints: 16.5
8              }
9          }
10
```

# Rechnen mit *float* ii

```
1        public class Calc {
2            public static void main(String[] args) {
3                float a = 0.1f;
4                float b = 0.2f;
5
6                System.out.println(((a + b) == 0.3));
7            }
8        }
9
```

# Rechnen mit *float* iii

```
1         public class Calc {
2             public static void main(String[] args) {
3                 float a = 0.1f;
4                 float b = 0.2f;
5
6                 System.out.println(((a + b) == 0.3)); // false
7                 System.out.println((a + b));
8             }
9         }
10
```

Float besitzt nur eine begrenzte Genauigkeit.

*Dies kann zu unvorhersehbaren Ergebnissen führen!*

# Rechnnen mit *int* und *float*

```
1         public class Calc {
2             public static void main(String[] args) {
3                 float a = 9.3f;
4                 int b = 3;
5                 System.out.println(a + b); // prints: 12.3
6                 float c = a + b;
7                 System.out.println(c); // prints: 12.3
8             }
9         }
10
```

Java konvertiert automatisch von **int** zu **float**, wenn dies nötig ist.
Aber nicht andersherum.

# Strings

Ein String ist ein primitiver Datentyp. Es ist ein Objekt
Objekte werden in der nächsten Stunde besprochen.

```
1         public class Calc {
2             public static void main(String[] args) {
3                 String hello = "Hello World!";
4                 System.out.println(hello); // print: Hello World
    !
5             }
6         }
7
```

Ein String ist nichts anderes als eine Zeichenfolge. Strings beginnen u

# Concatenation

```
1         public class Calc {
2             public static void main(String[] args) {
3                 String hello = "Hello";
4                 String world = " World!";
5                 String sentence = hello + world;
6                 System.out.println(sentence);
7                 System.out.println(hello + " World!");
8             }
9         }
10
```

Man Strings mit den + Operator aneinander hängen. Dadurch sehen beide Zeilen gleich aus.

Man kann ein String einer Variable zuweisen.Um Strings miteiar

# Strings und Zahlen

```
1        public class Calc {
2            public static void main(String[] args) {
3                int factorA = 3;
4                int factorB = 7;
5                int product = factorA * factorB;
6                String answer =
7                factorA + " * " + factorB + " = " + product;
8                System.out.println(answer); // prints: 3 * 7 =
    21
9            }
10       }
11
```

Wenn ein primitiver Datentyp mit einen String verbunden wird, wird dieser in eine *String* umgewandelt.

# Zusammenfassung

Datentypen

- int, long
- float, double
- String

Hello World Beispiel

# Input

# Eingabe vom *Benutzer* i

Um eine Eingabe von einen Nutzer zu lesen benötigen wir ein Objekt vom Typ Scanner. Dafür müssen wir das Paket java.util.Scanner importieren.

```
1         import java.util.Scanner;
2         public class Input
3         {
4             public static void main(String[] args)
5             {
6                 //do something
7             }
8         }
9
```

Pakete sind Code, welcher von anderen Leute geschrieben wurde. Dadruc

# Eingabe vom *Benutzer* ii

Um den Scanner zu benutzen, müssen wir zuerst ein Objekt von der Klasse Scanner erzeugen (??? wird in der nächsten Stunde besprochen).

```
1   import java.util.Scanner;
2   public class Input
3   {
4       public static void main(String[] args)
5       {
6       Scanner sc = new Scanner(System.in);
7       System.out.print("Please input a number: ");
8       int a = sc.nextInt();
9       System.out.println("Input number = "+a);
10
11      }
12  }
13
```

sc.nextInt() sagt den Scanner er soll auf die nächste Eingabe warten, diese

# Eingabe vom *Benutzer* iii

Um den Scanner zu benutzen, müssen wir zuerst ein Objekt von der Klasse Scanner erzeugen. (Beispiel-Code, bitte nicht in realen Programmen verwenden)

```
1   import java.util.Scanner;
2   public class Input
3   {
4       public static void main(String[] args)
5       {
6       Scanner sc = new Scanner(System.in);
7       System.out.print("Please input a number: ");
8       int a = sc.nextInt();
9       System.out.println("Input number = "+a);
10      System.out.print("Please input a decimal number: ");
11      Double b = sc.nextDouble();
12      System.out.println("Input number = "+b);
13      }
14  }
```

# Eingabe vom *Benutzer* iv

```
1   import java.util.Scanner;
2   public class Input
3   {
4       public static void main(String[] args)
5       {
6       Scanner sc = new Scanner(System.in);
7       System.out.print("Please input a number: ");
8       int a = sc.nextInt();
9       System.out.println("Input number = "+a);
10      System.out.print("Please input a decimal number: ");
11      Double b = sc.nextDouble();
12      System.out.println("Input number = "+b);
13          System.out.print("Please input a String: ");
14          String c = sc.nextLine();
15          System.out.println("Input String = "+c);
16      }
17  }
```

# Control Statements

# Control Statements

- if, else, else if
- for
- while

# If Then Else

```
if(condition) {
    // do something if condition is true
} else if(another condition){
    // do if "else if" condition is true
} else {
    // otherwise do this
}
```

Wenn die erste Aussage wahr ist, tue das was in den ersten Block steh

# If Then Else example

```
1  public class IteExample {
2
3      public static void main(String[] args) {
4          int myNumber = 5;
5
6          if(myNumber == 3) {
7              System.out.println("Strange number");
8          } else if(myNumber == 2) {
9              System.out.println("Unreachable code");
10         } else {
11             System.out.println("Will be printed");
12         }
13     }
14
15 }
```

# Vergleichsoperatoren

Wie man Dinge vergleicht:

- == Equal
- != Not Equal
- > Greater Than (Größer als)
- >= Greater or Equal than (Größer gleich)

3 == 3 -> wahr, 3 == 5 -> falsch3 != 3

*Note*: Man kann mehrere Vergleiche mit && (AND) or || (OR) verknüpfen.

(3 == 3) || ( 3 == 5) -> wahr, wenn mindestens eine Aussage/Vergleich wa

# for

Eine Schleife hat einen Schleifenkörper (Block). Dieser Körper kann mehrma

```
for(initial value, condition, change) {
    // do code while condition is true
}
```

initial value: ein Befehl der zuerst ausgeführt wird, meist etwas wie int i = 0

# for example

```
1  public class ForExample {
2
3      public static void main(String[] args) {
4          for(int i = 0; i <= 10; i++) {
5              System.out.print("na ");
6          }
7          System.out.println("BATMAN!");
8      }
9
10 }
```

1. Durchlauf i = 0 -> i <= 10 (ist wahr) -> Ausgabe: na -> Schleifenkörper End

# while

```
while(condition) {
    // do code while condition is true
}
```

# while example

```
1  public class WhileExample {
2
3      public static void main(String[] args) {
4          int a = 0;
5          while(a <= 10) {
6              System.out.println(a);
7              a++; // Otherwise you would get an endless loop
8          }
9      }
10
11 }
```

# Methoden

# Was ist eine Methode?

Was ist eine Methode?

- ein Stück Code zum Wiederverwenden

# Was ist eine Methode?

Was ist eine Methode?

- ein Stück Code zum Wiederverwenden
- kann Daten von Außen bekommen

# Was ist eine Methode?

Was ist eine Methode?

- ein Stück Code zum Wiederverwenden
- kann Daten von Außen bekommen
- kann Daten nach Außen zurückgeben

# Was ist eine Methode?

Was ist eine Methode?

- ein Stück Code zum Wiederverwenden
- kann Daten von Außen bekommen
- kann Daten nach Außen zurückgeben

Andere Namen

- function
- procedure
- subroutine

# Warum sollten man Methoden verwenden?

Warum sollten man Methoden verwenden?

- Programmierer sind faul $\rightarrow$ Mehr mit weniger erreichen

# Warum sollten man Methoden verwenden?

Warum sollten man Methoden verwenden?

- Programmierer sind faul $\rightarrow$ Mehr mit weniger erreichen
- bessere Struktur und weniger Änderungen

# Warum sollten man Methoden verwenden?

Warum sollten man Methoden verwenden?

- Programmierer sind faul $\rightarrow$ Mehr mit weniger erreichen
- bessere Struktur und weniger Änderungen
- reduziert Fehler

# Warum sollten man Methoden verwenden?

Warum sollten man Methoden verwenden?

- Programmierer sind faul $\rightarrow$ Mehr mit weniger erreichen
- bessere Struktur und weniger Änderungen
- reduziert Fehler
- wichtig für OOP

# Einführung in die *Methoden* i

Die einfachste Methode

```
1     static void helloMethod(){
2         System.out.println("Hello, method!");
3     }
4
```

# Einführung in die *Methoden* ii

## Eine Methode aufrufen

```
1    class Hello{
2        public static main(String[] args){
3            helloMethod();
4        }
5        static void helloMethod(){
6            System.out.println("Hello, method!");
7        }
8    }
9
```

# Einführung in die *Methoden* iii

Daten an eine Methode übergeben (Parameter)

```
1        static void printHello(String input){
2            System.out.println("Hello, " + input + "!");
3        }
4
```

# Einführung in die *Methoden* iv

Daten aus einer Methode zurückgeben (Return)

```
1        static String getHello(String input){
2            String hello = "Hello, " + input + "!";
3            return hello;
4            // return "Hello, " + input + "!";
5        }
6
```

# Einführung in die *Methoden* v

DEMO

# Übung

Gebe alle geraden Zahlen zischen 1 und 100/n aus. (n - User input)

# Übung

Gebe alle geraden Zahlen zischen 1 und 100/n aus. (n - User input)

- benutze while und if

# Übung

Gebe alle geraden Zahlen zischen 1 und 100/n aus. (n - User input)

- benutze while und if
- benutze for (und nicht if)

# Übung

Gebe alle geraden Zahlen zischen 1 und 100/n aus. (n - User input)

- benutze while und if
- benutze for (und nicht if)
- benutze eine Methode zur Überprüfung


GOOD LUCK AND
HAVE FUN!!!!