Java

Controll Statements and Methods

Hannes Ueck, Jakob Krude

19. November 2020

Java-Kurs

Overview

1. Recalling last session
2. Methods
   - The Java main method
   - Methods with Arguments
   - Return Value
3. Controll Statements
   - Ite
   - for
   - while
   - break
   - continue

Recalling last session

Conclusion

Variables

Datatypes

- int, long
- float, double
- String

Methods

What is a Method

A method is a block of code which only runs when it is called.

```
1 public class Main {
2
3     //Declaration of the method printHello()
4     static void printHello() {
5         System.out.println("Hello");
6     }
7 }
8
```

printHello() is the name of the method

Just ignore *void* and *static* for now.

The Java main method

This is a small program printing *Hello World!* to the console:

```
1     public class Hello {
2         public static void main(String[] args) {
3             //Everything in here is executed
4             System.out.println("Hello World!");
5         }
6     }
7
```

In Java the method "public static void main()" is the entry into the program.
This method is automatically called when you execute the code.

Calling a Method

```
1  public class Main {
2
3      public static void main(String[] args) {
4
5          printHello(); // method call
6
7      }
8
9      //Declaration of the method printHello()
10     static void printHello() {
11         System.out.println("Hello")
12     }
13 }
14
```

You can call a method by appending () to the name of the method.

Methods with Arguments

```
1  public class Calc {
2
3      static void add(int summand1, int summand2) {
4          System.out.println(summand1 + summand2);
5      }
6
7      public static void main(String[] args) {
8          int summandA = 1;
9          int summandB = 2;
10         System.out.print("1 + 2 = ");
11         add(summandA, summandB);
12         // prints: 3
13     }
14
15 }
16
```

Methods with Return Value

A method without a return value is indicated by **void**:

```
1     static void add(int summand1, int summand2) {
2         System.out.println(summand1 + summand2);
3     }
4
```

A method with an **int** as return value:

```
1     static int add(int summand1, int summand2) {
2         return summand1 + summand2;
3     }
4
```

Calling Methods with a return value

```
1   public class Calc {
2
3       static int add(int summand1, int summand2) {
4           return summand1 + summand2;
5       }
6
7       public static void main(String[] args) {
8           int sum = add(3, 8);
9           System.out.print("3 + 8 = " + sum);
10          // prints: 3 + 8 = 11
11      }
12
13  }
14
```

Controll Statements

Controll 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
}
```

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 }
```

Conditions?

How to compare things:

- == Equal
- != Not Equal
- > Greater Than
- >= Greater or Equal than

Note: You can concatenate multiple conditions with && (AND) or || (OR)

for

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

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 }
```

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 }
```

break

The *break* statement can be used to jump out of a loop.

```
\\Prints the numbers from 0 to 3.
for (int i = 0; i < 10; i++) {
    if (i == 4) {
        break;
    }
    System.out.println(i);
}

```

continue

The *continue* statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

```
\\Prints only the number 4.
for (int i = 0; i < 10; i++) {
    if (i == 4) {
        continue;
    }
    System.out.println(i);
}

```