

Java Introduction

Patrick Nagel

University of Applied Sciences Bonn-Rhein-Sieg

March 29, 2018



**Hochschule
Bonn-Rhein-Sieg**
University of Applied Sciences

- 1 Java - Yet Another Programming Language
- 2 Java Basics
 - Properties
 - How does Java work?
- 3 Syntax and Further Characteristics
- 4 Basic Elements
 - Repetition
 - Examples in Java
 - Demo via Terminal
 - Challenge
- 5 Debugging
- 6 Object Oriented Principles
 - Inheritance
 - Abstract Classes
 - Interfaces



Java - Yet Another Programming Language

Courses in which Java will play a role:

- Advanced Software Technology.
- Planning and Scheduling.
- Multiagent Systems.

Advantages of Java in these subjects:

- Extensive libraries.
- Elaborated concepts.

Further reasons to learn Java:

- Used in Android development.
- Average salary of a Java developer in Germany: \$53,763
(source: [stack overflow developer survey results 2017](#)).

Java Basics - Properties

Java is a high-level programming language developed by *Sun Microsystems* and released in 1995.

Java is:

- Object Oriented.
- Platform Independent.
- Multithreaded.
- Interpreted.
- Dynamic.



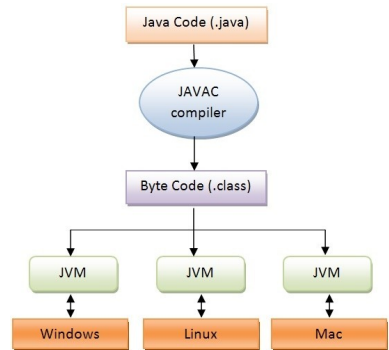
**Hochschule
Bonn-Rhein-Sieg**
University of Applied Sciences

Java Basics - How does Java work?

- High-level language \neq machine code.
- Compilation and interpretation needed!

The steps in Java are:

- 1 Write code in .java file.
- 2 Compile into Java byte code.
- 3 Execute byte code with help of Java Virtual Machine (JVM).



Source: www.quora.com

Syntax and Further Characteristics

- Java is *case sensitive* - difference between **cool** and **Cool**.
 - Java follows the *CamelCase* convention.
 - Java is a strict data typing language: Every variable must be declared with a data type.
-
- Class names start with a capital letter.
 - Method and variable names start with a small letter.
 - Names used for classes, methods and variables are referred to as *identifiers*. For that certain rules exist:
 - All identifiers start with a letter, a currency symbol or an underscore.
 - After the first character identifiers can have any combination of characters.
 - A key word cannot be used as an identifier.
 - Reserved key words may not be used as identifier names. A list of all keywords can be found [here](#).

Basic Elements - Repetition

Java programs consist of objects, that communicate via invoking each other's methods.

- **Comments:** Allow the programmer to make notes inside the code.
- **Variables:** The values assigned to the variables define the state of an object or a class.
- **Objects:** Have states and behaviors.
- **Class:** Blueprint/template of an object. A class describes the behavior and states that the object supports.
- **Methods (functions):** Collections of statements that are grouped together to perform an operation. Methods represent the behavior. A class can contain many methods.
- **Decisions:** Structures that have one or more conditions to be evaluated or tested by the program.
- **Loops:** Statements that allow to execute a block of code several number of times.
- **Input/Output:** Self explaining.
- ...



Basic Elements - Examples in Java (1/3)

```
// This is a single-line comment.

/*
 * This is a multi-line comment.
 */

// The following is class declaration.
public class BasicElements {

    // The following is a variable of type String.
    public String name;
    // The following is a variable of type string array.
    public String[] favoriteColors;

    /*
     * The following is the constructor of the class. With the help of a constructor
     * objects of the type of the class are instantiated (Later more about that).
     */
    public BasicElements(String name) {
        this.name = name;
        this.favoriteColors = new String[] { "red", "blue", "green" };
    }
}
```


Basic Elements - Examples in Java (2/3)

```
// The following is a method.
public void printFavoriteColors() {
    int numberOfFavoriteColors = this.favoriteColors.length;

    // The following is a for loop.
    for (int i = 0; i < numberOfFavoriteColors; i++) {
        System.out.println(this.favoriteColors[i]);
    }
}
```

Basic Elements - Examples in Java (3/3)

```
public static void main(String[] args) {  
  
    // The following is a condition.  
    if (args.length > 0) {  
        for (String inputArgument : args) {  
            System.out.println(inputArgument);  
        }  
    }  
  
    // The following is using a class to perform input in Java.  
    Scanner      sc = new Scanner(System.in);  
    String name = sc.nextLine();  
    sc.close();  
    System.out.println("Hello, " + name + " nice to meet you!");  
  
    // The following is an object of the class BasicElements.  
    BasicElements firstInstance = new BasicElements(name);  
    firstInstance.printFavoriteColors();  
    System.out.println("Saved in the object's variable: "  
                        + firstInstance.name);  
}
```

Basic Elements - Demo via Terminal

- 1 Download the file *BasicElements.java* from the repository.

foundation_cours > content > programming > java_introduction > code

- 2 Compile the file with the help of the terminal and the command

javac BasicElements.java

- 3 Execute the created byte code with
java BasicElements <argument>



Basic Elements - Challenge

Create a class representing a student. The student is suppose to have the following properties:

- *First name* and *last name*.
- *Age*.
- *Favorite subject*.

Further write a method that prints all the aforementioned information in a sentence.

The student is created on runtime and the necessary information are passed via console.

Good luck! You have 20mins.




Debugging

Types of errors:

- **Syntactical Errors:** e.g. missing semicolon.
- **Logical Errors:** e.g. the method *withdrawMoney()* adds money to the bank account rather than subtracting it.

Debugging is the process of finding and resolving defects or problems within a computer program.

Eclipse:

- Shows *syntax errors* right away: 
- Provides for *logical errors* a debug tool.



**Hochschule
Bonn-Rhein-Sieg**
University of Applied Sciences

Object Oriented Principles -Inheritance

Inheritance can be defined as the process where one class acquires the properties (methods and fields) of another. With the help of inheritance information are managed in a hierarchical manner.

Example:

```
public class Parent {  
    public Parent() {  
    }  
}
```

```
public class Child extends Parent {  
    public Child() {  
        super();  
    }  
}
```

What does that mean? Among other:

- Child inherits all properties and skills.
- Redundancies can be avoided.



Object Oriented Principles - Abstract Classes

Abstract classes represent classes that cannot be instantiated. However other classes can inherit from abstract classes and then be instantiated with the properties of the abstract class. It is used as a process of hiding the implementation from the user.

Facts about abstract classes:

- Contain the word *abstract* in their declaration.
- May or may not contain *abstract methods*(=methods whose body still needs to be defined by the heirs).
- If a class has at least one abstract method it needs to be declared *abstract*.



Object Oriented Principles - Interfaces

An interface in Java is a collection of constants and abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface.

Facts about interfaces:

- A class can implement more than one interface.
- Similar to abstract classes, an interface cannot be instantiated.
- Interfaces do not contain any constructors.
- Fields in interfaces can only be declared both *static* and *final*.
- An interface can extend multiple interfaces.



Object Oriented Principles - Challenge

Create five classes. Four classes that represent mathematical shapes, namely:

- Square.
- Rectangle.
- Parallelogram.
- Kite.

Include methods to compute the area and circumference of each of the shapes.

One class that contains the main method and outputs the parameters, areas and circumferences of each object.

Good luck! You have 30mins.

