



# 326.041 (2015S) – Practical Software Technology

(Praktische Softwaretechnologie)

## Introduction

Alexander Baumgartner  
Alexander.Baumgartner@risc.jku.at

Research Institute for Symbolic Computation (RISC)  
Johannes Kepler University, Linz, Austria



- Language is English.
- Moodle: <https://moodle.risc.jku.at/course/view.php?id=115>.
- You have to **register yourself** as a course participant **in Moodle**.
- **Weekly exercises:** Submission via Moodle before the deadline!
- **Presentation** about a chosen topic **in English, 45 minutes**.
- No Exam.
- Slides, exercises, . . . will be available at the Moodle page.



- 200 points are available.
  - 100 points for the presentation.
  - 100 points for the exercises.
- You need at least 140 points.
  - $[140, 155) = 4$
  - $[155, 170) = 3$
  - $[170, 185) = 2$
  - $[185, 200] = 1$





- Presentation topics will be available on the Moodle page.
- Everybody must choose a topic: First come, first served!
- It is not acceptable if you read your presentation:
  - You can check your notes during your presentation.
  - You can read some terms on your slides.
- Please upload your slides as pdf file in Moodle after your presentation.



- **Java:** Currently one of the most popular programming languages.
- **Eclipse:** A modern integrated development environment (IDE).
- **Object Oriented Design:** Robustness, adaptability, reusability.
- **Algorithms:** Solving some tasks (searching, sorting, . . . ) in Java.
- **Design Patterns:** Solutions to “typical” software design problems.
- **JUnit:** A simple framework to write repeatable tests.
- **GIT/SVN:** Two widely used revision control systems.
- **UML:** A modeling language for software engineering.
- **Networking:** Client-server solutions and web services.



-  Grady Booch. *Object-Oriented Analysis and Design with Applications*. Addison Wesley Longman Publishing Co., Inc., 3rd edition, 2004.
-  Bruce Eckel. *Thinking in Java*. Prentice Hall Professional Technical Reference, 3rd edition, 2002.
-  Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns – Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman, 1 edition, 1995. 37. Reprint (2009).
-  Xiaoping Jia. *Object-oriented Software Development Using Java: Principles, Patterns, and Frameworks*. Addison-Wesley, 2000.
-  Robert Lafore. *Data Structures and Algorithms in Java*. Sams, 2nd edition, 2002.



- Specification: <http://docs.oracle.com/javase/specs/>
- Tutorial: <http://docs.oracle.com/javase/tutorial/>
- Javadoc: <http://docs.oracle.com/javase/7/docs/api/>



```
1 class HelloWorld {  
2     public static void main(String [] args) {  
3         System.out.println(" Hello World!");  
4     }  
5 }
```

The obligatory "Hello World!" program.

- **class** defines a "container" for:
  - Construction plan to create objects (instances) of a certain kind.
  - Member variables (fields) provide state information for a certain object.
  - (Member) methods define the behavior for a certain state and allow interaction with a certain object.
  - Static variables (e.g. constants) and static methods (e.g. functions).
- **HelloWorld** is the name of this class (container).
  - Class names start with upper case.
  - CamelCase is used to form meaningful names.





- Typically (not always) one .java file contains one class with the same name. In our case the file name is **HelloWorld.java**.
- **{ }** is used to group logical blocks of code.
- **public** blocks are visible to the entire world.
- **static** methods and variables are independent of any instances created for the class. They exist without/besides the objects.
- **void** is the “empty type”. Such a method has no return value.
- **main** is the name of the method.
  - Method names and variable names start with lower case.
  - camelCase is used to form meaningful names.

Method name **main** is used to declare an **entry point** for a program.

- **String[] args** is a single argument of the method main.
  - String is a class which represents a sequence of unicode characters.
  - String[] designates an array of Strings.



- args is the name of the argument.
- **System** is a class which contains static variables and methods to access the runtime environment: I/O, etc.
- **.** (the dot operator) is used to access class members of certain classes or instance members of a certain objects.
- **out** is a static variable of System and denotes an object representing the standard output stream. It is of type java.io.PrintStream.
- **println** is a method of the class PrintStream and it is invoked on the object out.
- **"Hello World!"** is a shortcut to instantiate an object of type String which is initialized by the given character sequence.
- The application of println to the argument "Hello World!", has the behavior of writing "Hello World!\n" into the Stream.
- **;** (the semicolon) denotes the end of a statement.



```
1 import javax.swing.JOptionPane;
2
3 class HelloPerson {
4     public static void main(String [] args) {
5         String name = JOptionPane.showInputDialog("Name:");
6         JOptionPane.showMessageDialog(null, "Hello " + name);
7     }
8 }
```

A simple dialog example.



- Java provides a powerful **Standard Class Library**.
- Fundamental classes (java.lang) do not need an **import**.  
Non-fundamental classes (outside current package) need **import**.
- **String name** declares a local variable, called name, of type String.
- The method showInputDialog has **return type String**.
- **=** assigns the return value to the local variable.
  
- **null** is a keyword, which stands for “no object”.
- **,** (the comma) separates arguments.
- **+** is syntactic sugar for String concatenation.

+ is also used for addition of primitive types... comes later.



- Check java version: **java -version**  
java version "1.7.0\_65"  
OpenJDK Runtime Environment (IcedTea 2.5.3) (7u71-2.5.3-2)  
OpenJDK 64-Bit Server VM (build 24.65-b04, mixed mode)
- Compile source code: **javac FILENAME.java**  
Creates **FILENAME.class** which contains **byte-code**.
- Run the program: **java FILENAME**  
Only files which contain a **static void main** method are executable!



- Register at Moodle and enroll yourself for the course.
- Install a Java Development Kit(JDK) version 6 or later.
  - Gentoo: emerge jdk
  - Debian: apt-get install openjdk-7-jdk
- java -version should print a number  $\geq 1.6$ .
- Write a “Hello World” program with a Text Editor.
- Compile and execute the program from command line.
- See the guidance for this exercise on the Moodle page.
- Upload your “Hello World” program in Moodle.

Recommendation: Install/Download the API-Source Code. **You can learn a lot from it!** (e.g. apt-get install openjdk-7-source)

API = Application Programming Interface