A brief history of Java

The history of Java begins in 1991, when a group of engineers from the **Sun** company under the leadership of **James Gosling** and **Patrick Naughton** started to work on the project that got a code name **Green**. The team aimed at developing a possibly "`light`" programming language that would be independent of the hardware platform. Initially, the language was to be applied in cable TV tuners.

The engineers derived from Unix circles, therefore they based their new language on **C++** and *Smalltalk*. Thanks to this, they created an object-oriented, and not a procedural language.

At the initial stage of the project, the language was called *Oak– Object Application Kernel*. Reportedly, the name was chosen by James Gosling because of the view of the oak tree outside the window of his office. It turned out that a language with this name already existed, and a decision was made to change the name to `Java` (which is a coffee bush variety), which turned out to be a hit.

The first version of Java was launched in early 1996. In the next version (1.1) many features were added and improved, but the language still had many restrictions.

Then from the version to the version new functionalities were introduced to Java, yet retaining its backward compatibility.

yet retaining its backward compatibility.

Since 2010, Java has been developed by **Oracle**.

Language assumptions

In May 1996, James Gosling and Henry McGilton published "A White Paper" where they described, among other things, the assumptions and goals of the language. The basic assumptions were as follows:

- **Simplicity**Unlike other programming languages (e.g. *C++*), Java has been deprived of many rarely used or difficult and erroneous structures and functions. Also, it is relatively easy to learn this language.
- **Object-orientation**Java is an object-oriented language where data, or objects, play the most important role. They have *identity*, *status* and *behavior*.
- **Reliability**A Java compiler can detect many errors that in other languages would not occur until the application was launched.
- **Networking**Java provides a broad library of functionalities that enable communication through many popular network protocols.
- **Architecture Neutral and Portable**A compiled Java code can be run on many processors, regardless of their architecture. Thanks to (in simple terms) *the Java Virtual Machine*, it is translated into a machine code on the fly. In addition, it is platform-independent, which means you can focus on programming without worrying about the specific mechanisms of a particular operating system.
- **High Performance**Despite the need to compile and then interpret a byte code, programs written in Java can be very efficient. Compilers feature optimizing mechanisms, e.g. they can optimize the most frequently executed code snippets in terms of speed or eliminate the number of function calls (*inlining*).
- **Multithreading Capability**Concurrent and parallel programming is relatively difficult. Java provides an accessible abstraction that covers complex constructions, yet allows you to write a code to be run on various processors (cores).

Basic concepts

While learning Java, you will encounter many names and abbreviations. Below you fill find the ones that you should know from the very beginning of your adventure with the language.

- **Java SE** – Java Standard Edition, a platform to be used on standard computers and in simple server applications.
- **Java EE** – Java Enterprise Edition, a platform designed for complex and expanded server applications.
- **Java ME** – Java Micro Edition, a platform for applications on mobile phones, mobile devices and small devices.

- **IDE** – **I**ntegraed **D**evelopment **E**nvironment, integrated development environment; it is used to develop, test, build and run programs.
- **JVM** – **J**ava **V**irtual **M**achine, executes a Java byte code by interpreting/compiling into machine code.
- **Source code** – A record of a computer program using a specific programming language that describes the operations to be performed by the computer on collected or received data. The source code is the result of a programmer's work and makes it possible to express the structure and operation of a computer program in a human-readable form. In Java, the source code is included in text files with a `.java` extension.
- **Compiler** – A program that converts **a source code** into a code written in other language. In addition, a compiler is to find lexical and semantic errors and optimize the code. In Java, a compiler processes the code stored in files with a `.java` extension and places the result of the operation in files with a `.class` extension.
- **Bytecode** – The outcome of compiling a program written in Java. This code is understandable for the Java runtime environment (JVM).

Working Environment

To execute exercises and assignments, you will need the following:

- **JDK** (*Java Development Kit*) – tools necessary for programming in Java
- **IDE** (*Integrated Development Environment*) – the environment in which writing a code in the Java language is possible

> ***Note:*** If for any reasons you do not want to or cannot install the above applications or during the installation unexpected problems occurred, you can use one of the tools available online, such as OnlineGDB. It is sufficient for the performance of exercises and assignments from this Handbook.

In order to install **Java** and **IntelliJ IDEA IDE** please go to this section and choose appropriate guide depending on your operating system (Windows, Linux or MacOS). Please note that installation guide contains information how to install **all** necessary software that will be used during the course. For now, you don't need anything else than Java and Intellij IDEA.

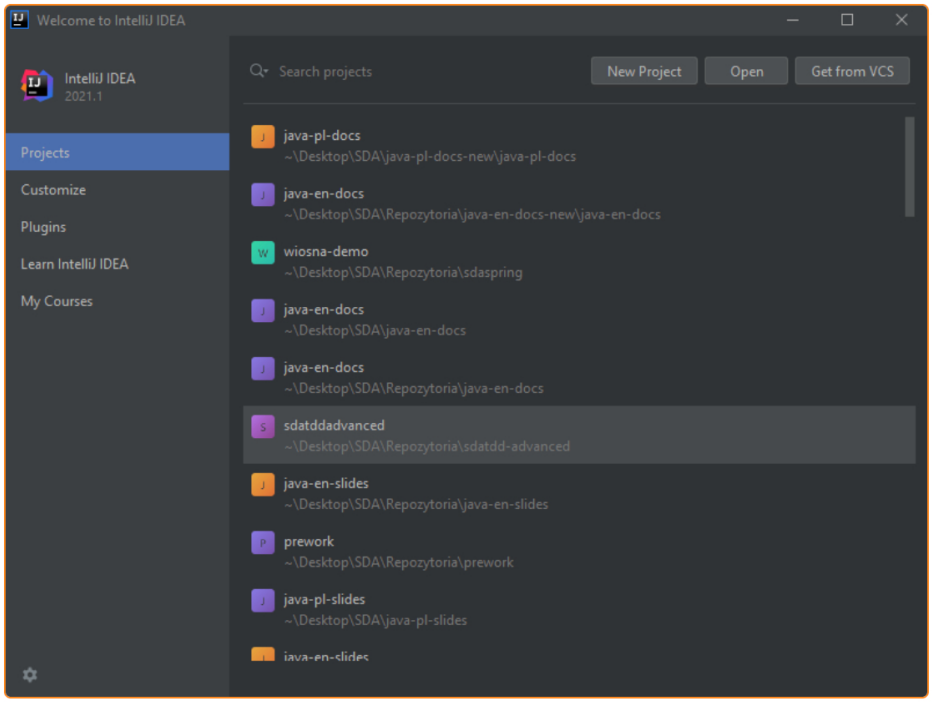
Creating a project

When you have installed Java and IntelliJ IDEA, it is time to create a project in which you will locate Java programs. After you run IntelliJ IDEA, the Welcome to

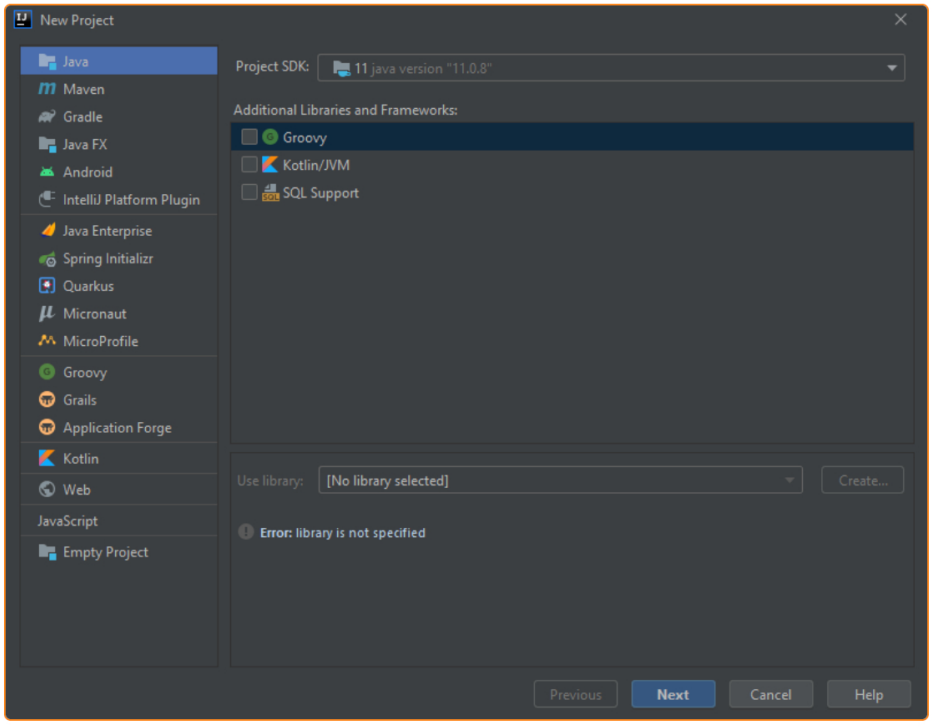
- [ ] Test 1st attempt | after the block: HTML,CSS,JS
- [ ] Test 2nd attempt | after the block: HTML,CSS,JS
- [ ] Frontend Technologies coursebook
- [ ] Frontend technologies slides
- [ ] Frontend Technologies tasks
- [ ] Test 1st attempt | after the block: FRONTEND TECHNOLOGIES (ANGULAR)
- [ ] Test 2nd attempt | after Frontend technologies
- [ ] Spring coursebook
- [ ] Spring slides
- [ ] Spring tasks
- [ ] Test 1st attempt | after the block: spring
- [ ] Test 2nd attempt | after the block: spring
- [ ] Mockito
- [ ] PowerMock
- [ ] Testing exceptions
- [ ] Parametrized tests
- [ ] Final project coursebook
- [ ] Final project slides
- [ ] Class assignments

IntelliJ IDEA window will appear with several options to choose from.

- select **New Project**

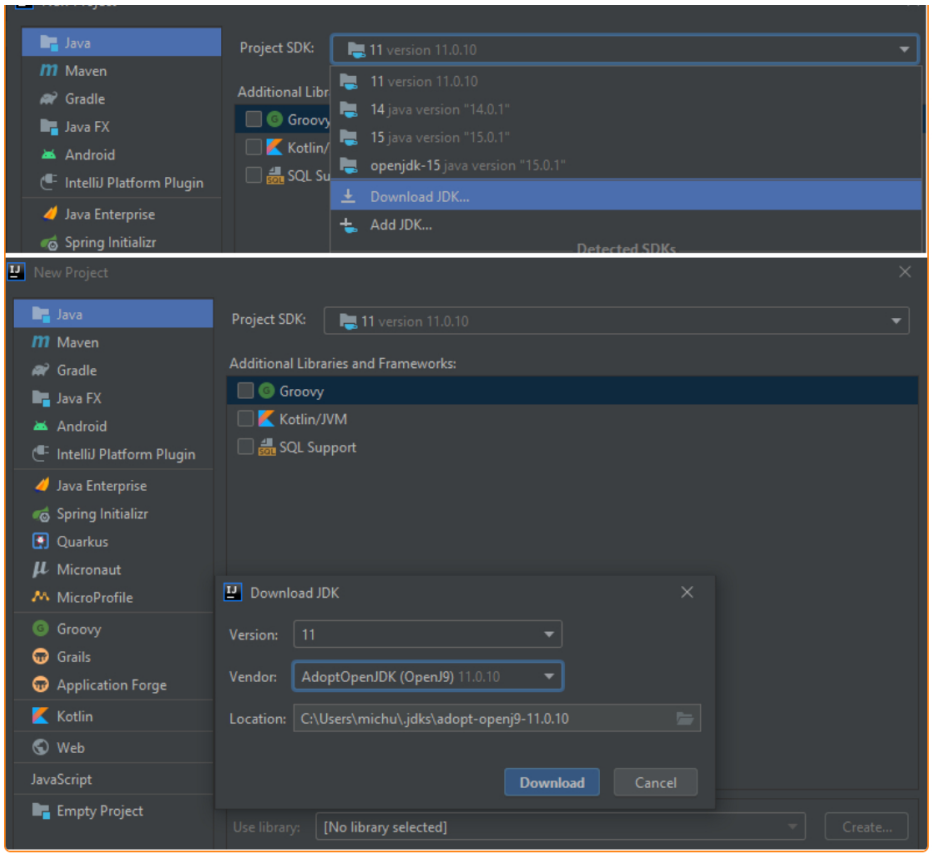


- on the next screen choose Java project of left side of the window. In the project SDK choose installed Java version and click **Next**.

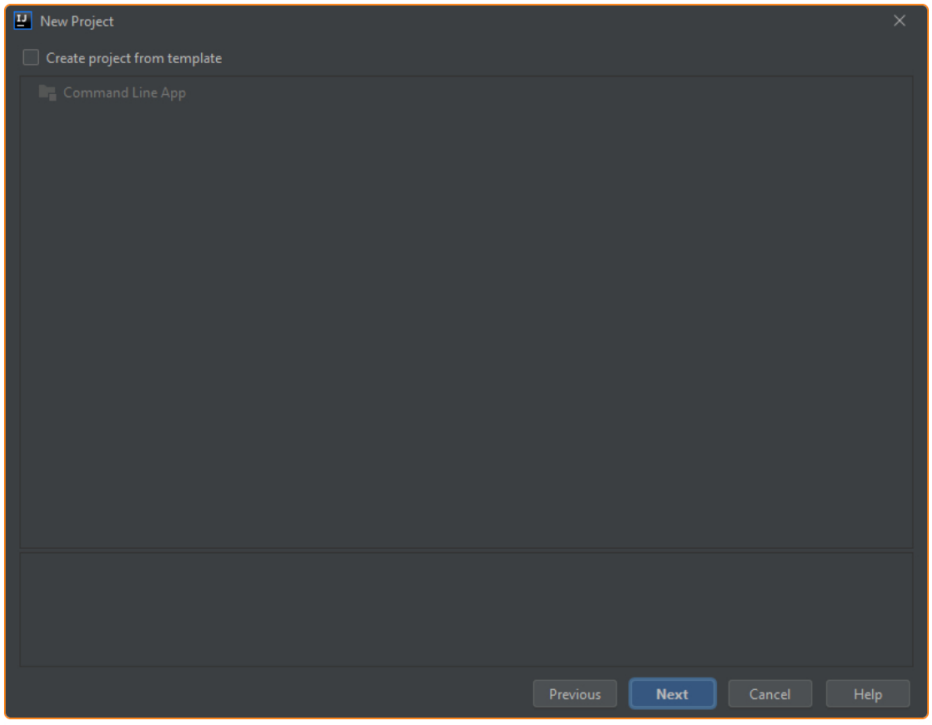


- in case no Java (i.e. no SDK) is available, click the arrow on the SDK select list and choose *Download SDK...*

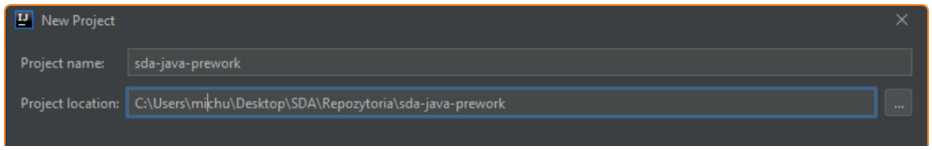
Choose Java version to install (11 or 17 in case it is available) and choose any vendor from the list (e.g. *AdoptOpenJDK*) and click *Download*.

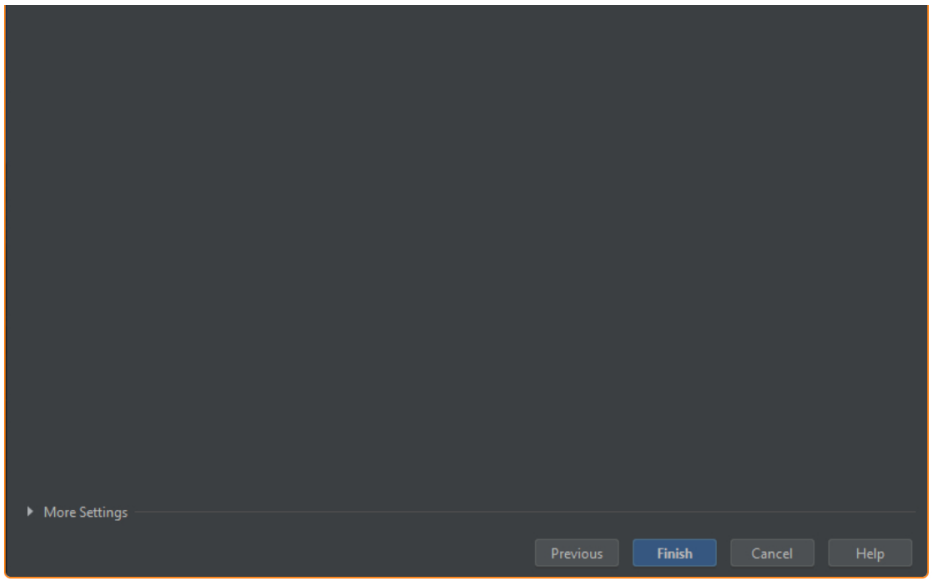


- on the next screen click *Next*

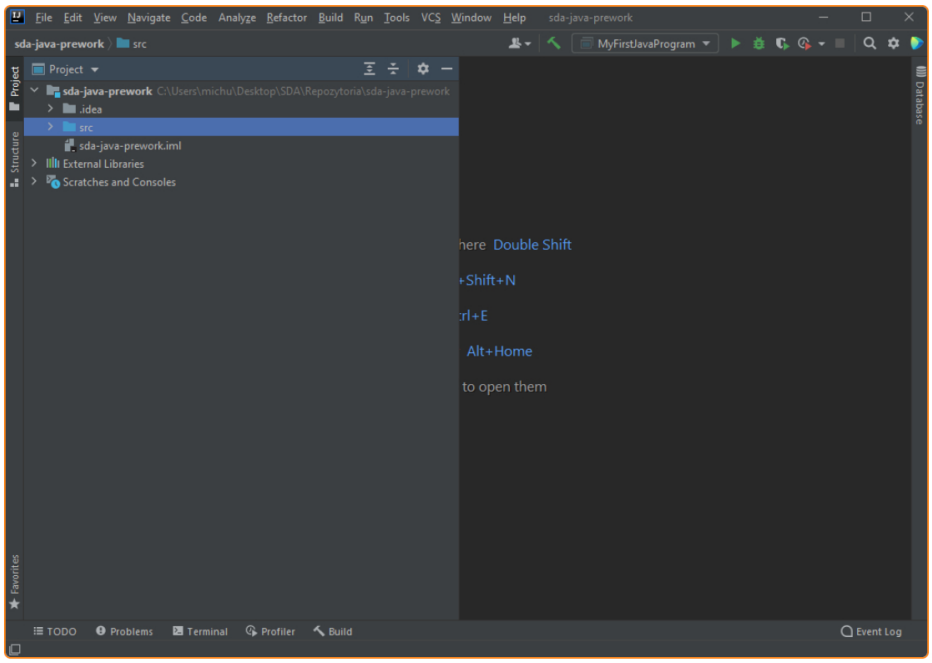


- give a name to the project (e.g. *sda-java-prework*) and choose directory where project will be stored. Click *Finish*. An empty project should be created.





- You should be ready to create your first program!



Complete Lesson