Foundations of Computer Science II
Group B (AY 2019-2020)
Laboratory 1

Nicola Fiorentino

n.fiorentino@dimes.unical.it

CUBO 44z

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica

Università della Calabria, Italy

25th Mar, 2020

Index

1. Introduction
2. Portability
3. Java Virtual Machine (JVM)
4. Java Platform, Standard Edition (Java SE)
5. Java SE Development Kit (JDK)
6. Your first Java program
7. TextPad
8. Eclipse
9. IO.java

Introduction

- Java is a high-level programming language developed by James Gosling at Sun Microsystems and first released in 1996. (The company has been acquired by Oracle Corporation in 2007.)
- Java is a general-purpose, concurrent, class-based, object-oriented language.
- The latest version is Java 14, released on March 17, 2020.
- Different Java Platforms are available:
  - Java Platform, Standard Edition (Java SE);
  - Java Platform, Micro Edition (Java ME);
  - Java Platform, Enterprise Edition (Java EE).
- We will focus on Java SE 14.

Portability

- Mission: pursue the **WORA** principle.

  "Write once, run anywhere."

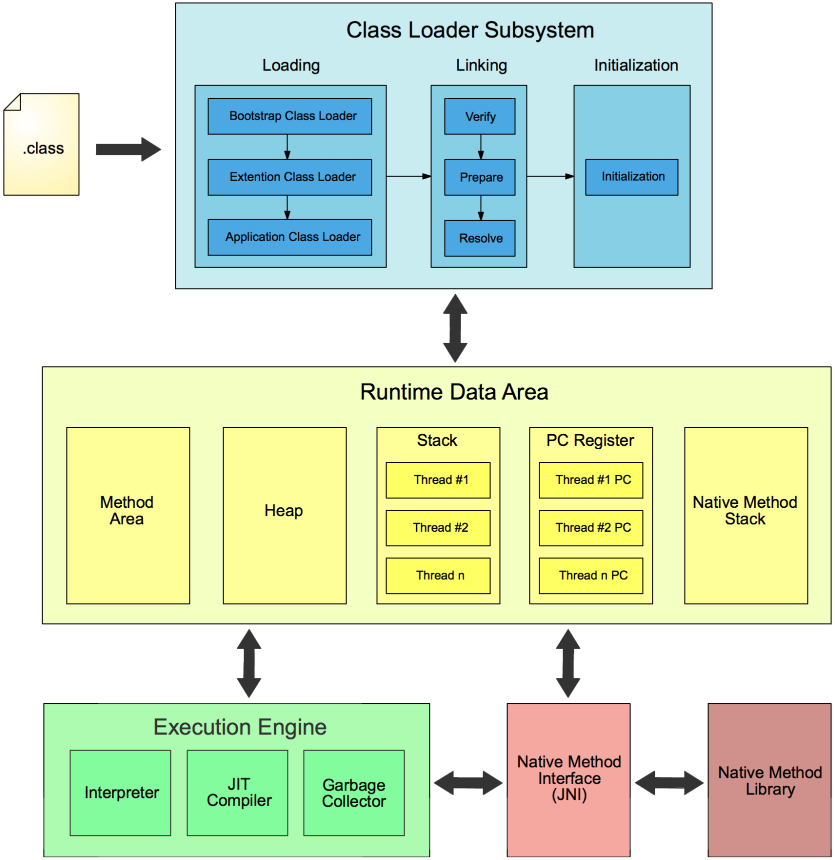
- A Java program is **cross-platform**: the same program may run on Linux, macOS, Solaris, Windows, etc.
- **Portability** across operating systems is ensured by a two-phase process, made of **compilation** and **interpretation** of the source code.
  - First, the source code is compiled (by the **compiler**). A(n executable) **bytecode program** is generated.
  - Then, the **interpreter** translates the bytecode program into native machine code and run it on the fly.
- Who does what?

Java Virtual Machine (JVM)

- As said, compilation transforms the source code into bytecodes. It is performed by the **Java compiler**.
- Practically, bytecodes are stored in a file (or a set of files) in a hardware- and operating system-independent binary format, called the `class` file format.
- The bytecodes are run by a virtual machine, called the **Java Virtual Machine (JVM)**.
- The JVM ensures portability: it is the component of the Java technology responsible for hardware- and operating system-independence.
- There is a specific version of JVM for each platform supported by the Java development team.

JVM Architecture

- The JVM is not just an interpreter.



Java Platform, Standard Edition (Java SE)

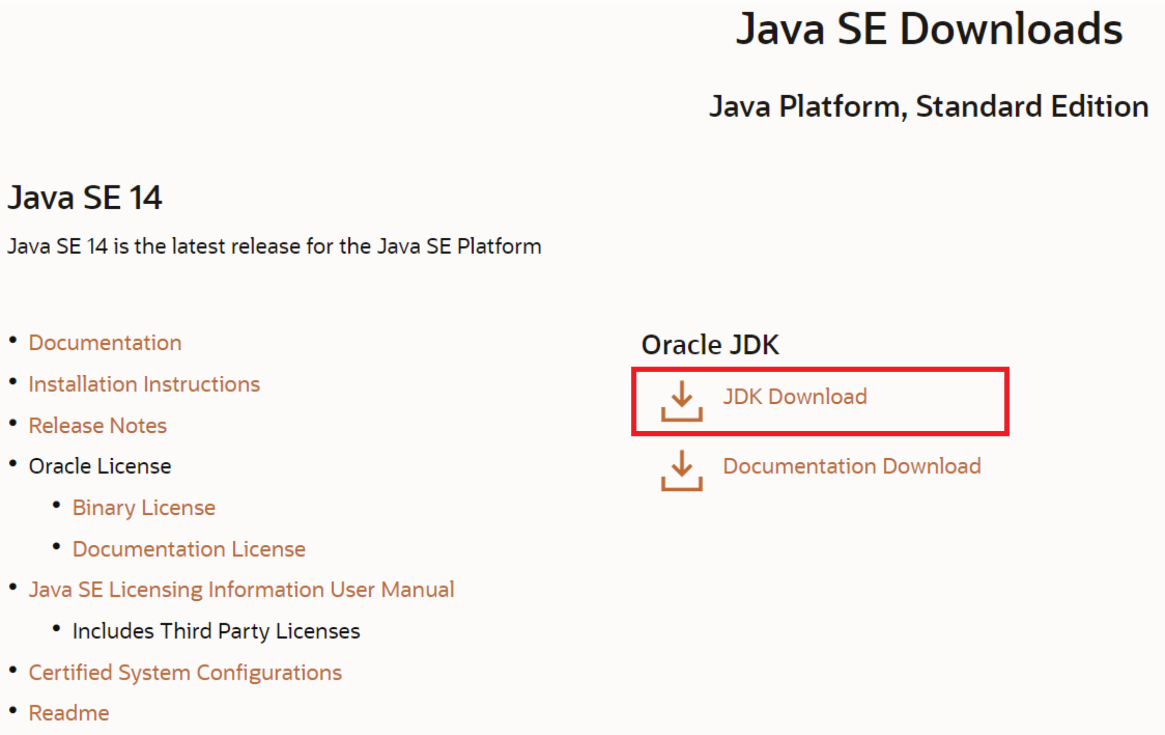
- It includes:
  - The Java Language Specification.
    - A document specifying what a well-formed Java program is, i.e. a document defining the programming language itself.
  - The Java Virtual Machine Specification.
    - A document specifying an abstract computing machine able to execute bytecode instructions. The document does not describe any particular implementation of the Java Virtual Machine.
  - The Java Class Library (JCL).
    - It provides the programmer with a set of helpful features.
- The **Java SE Development Kit (JDK)** is an implementation of Java SE.

Java SE Development Kit (JDK)

- It is the software you need to develop Java programs.
- JDK includes (at least):
  - the **Java Runtime Environment (JRE)**. It includes:
    - the Java Virtual Machine;
    - Java APIs (they implement the Java Class Library).
  - a set of tools:
    - **javac** (the Java compiler);
    - **java** (the bytecode interpreter);
    - **javadoc** (the documentation generator);
    - **jshell** (an interactive tool for learning the Java programming language. It is included in JDK 9 or higher).
- The Java Development Kit is freely downloadable from the Oracle website.

JDK - Download

- Visit the website at `https://www.oracle.com/java/technologies/javase-downloads.html`
  and download the latest release (jdk-14).



Java SE Downloads

Java Platform, Standard Edition

Java SE 14

Java SE 14 is the latest release for the Java SE Platform

- Documentation
- Installation Instructions
- Release Notes
- Oracle License
  - Binary License
  - Documentation License
- Java SE Licensing Information User Manual
  - Includes Third Party Licenses
- Certified System Configurations
- Readme

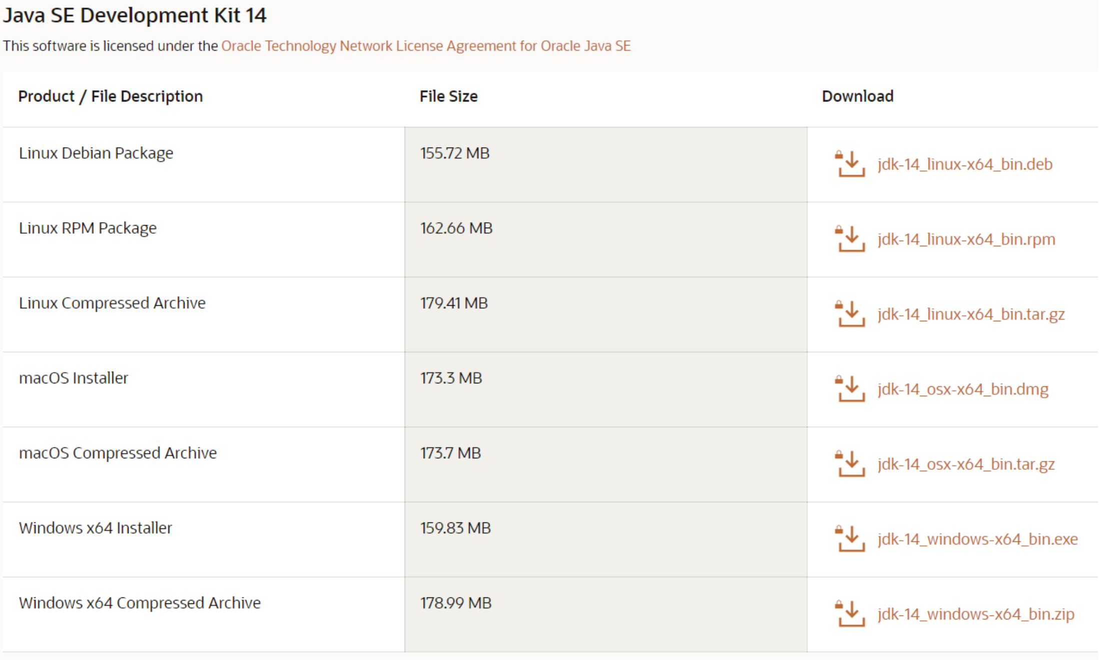
Oracle JDK

JDK Download

Documentation Download

JDK - Download

- Select the JDK version according to your platform and start the download.



Java SE Development Kit 14

This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE

Product / File Description	File Size	Download
Linux Debian Package	155.72 MB	jdk-14_linux-x64_bin.deb
Linux RPM Package	162.66 MB	jdk-14_linux-x64_bin.rpm
Linux Compressed Archive	179.41 MB	jdk-14_linux-x64_bin.tar.gz
macOS Installer	173.3 MB	jdk-14_osx-x64_bin.dmg
macOS Compressed Archive	173.7 MB	jdk-14_osx-x64_bin.tar.gz
Windows x64 Installer	159.83 MB	jdk-14_windows-x64_bin.exe
Windows x64 Compressed Archive	178.99 MB	jdk-14_windows-x64_bin.zip

- Once the download has completed, install the software.

JDK - Install

- Under Windows, launch the setup program. You will be asked where to install the JDK (default location: `C:\Program Files\Java\jdk-14`).
- On the Mac, run the installer. It installs the software into `/Library/Java/JavaVirtualMachines/jdk-14.jdk/Contents/Home/`.
- On Linux, uncompress the `.tar.gz` file to a location of your choice, such as your home directory.

JDK - Configuration

- If you want to develop and run Java applications under Windows or on Linux, one last step is required: setting up the **PATH** environment variable.
- The **PATH** environment variable tells the operating system where to find **javac** and **java** programs (namely, the compiler and the interpreter). It must store the location of the Java directory containing them (the **bin** directory nested inside the JDK root directory). Once set it up, you can compile and run Java programs from any directory.
- On Linux, add a line such as the following to the end of yuor **~/.bashrc** or **~/.bash_profile** file:

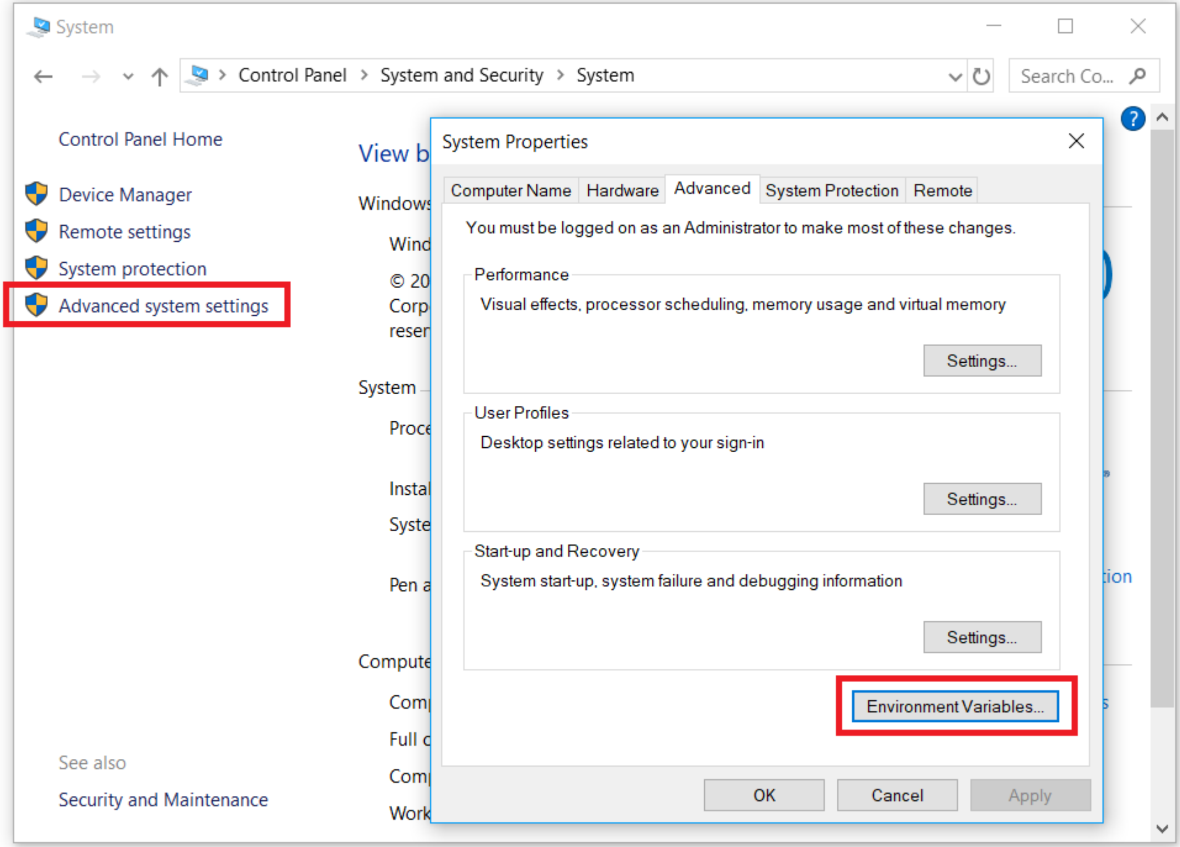
  ```
  export PATH=/home/usr/jdk-14/bin:$PATH
  ```

  (We are assuming **/home/usr/jdk-14** is the path locating the JDK root directory.)
- Under Windows, start the Control Panel, select System and Security, select System, then select Advanced System Settings. In the System Properties dialog, click the Advanced tab, then click the Environment Variables button.
  Scroll through the System Variables list until you find a variable named **PATH**. Click the Edit button.
  Add **C:\Program Files\Java\jdk-14\bin** to the beginning of the path, using a semicolon to separate the new entry from the (possible) old stuff:

  ```
  C:\Program Files\Java\jdk-14\bin;<old stuff>
  ```

  Save your settings.
  (We are assuming **C:\Program Files\Java\jdk-14** is the path locating the JDK root directory.)

JDK - Configuration (Windows 10 snapshots)



System

Control Panel > System and Security > System

Search Co...

Control Panel Home

Device Manager

Remote settings

System protection

Advanced system settings

See also

Security and Maintenance

System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage and virtual memory

Settings...

User Profiles

Desktop settings related to your sign-in

Settings...

Start-up and Recovery

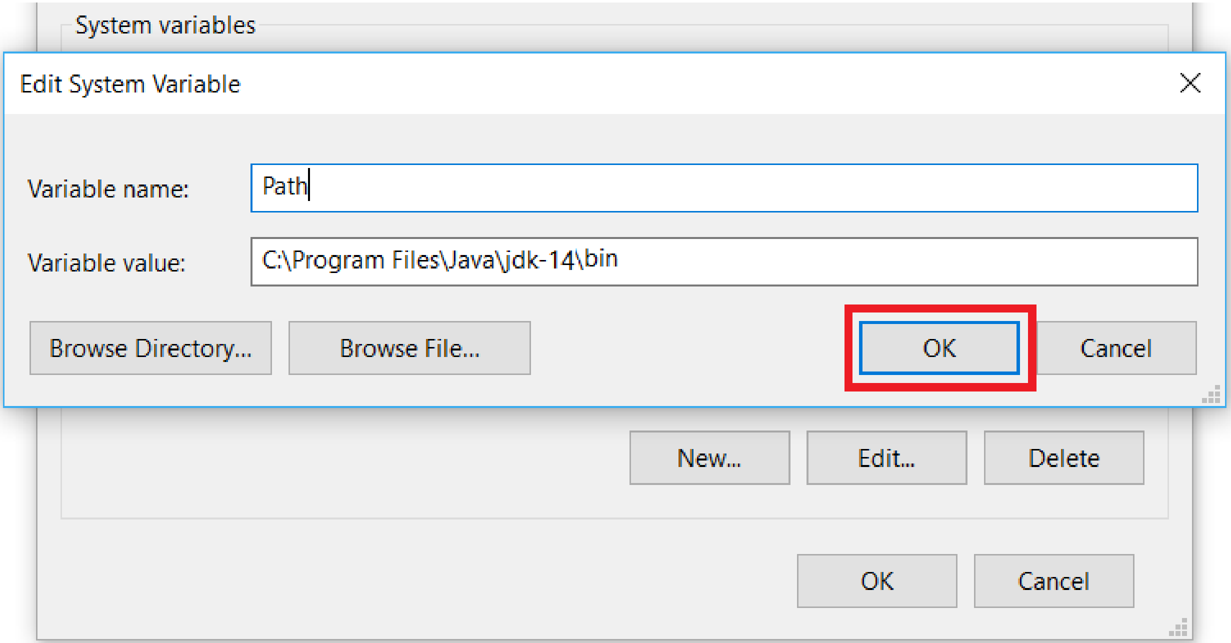
System start-up, system failure and debugging information

Settings...

Environment Variables...

OK | Cancel | Apply

JDK - Configuration (Windows 10 snapshots)



Variable name: Path

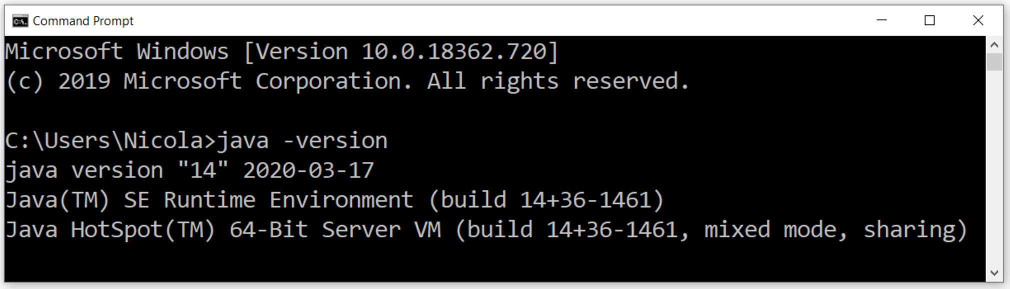
Variable value: C:\Program Files\Java\jdk-14\bin

JDK - Configuration

- Check that the installation and configuration have been successful. Start a terminal window, type the command

  `java -version`

  and press the Enter key. You should get a display such as this one:

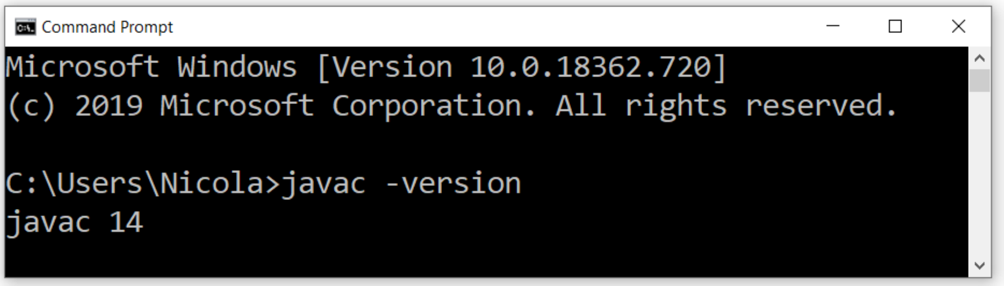
  

  ```
  Microsoft Windows [Version 10.0.18362.720]
  (c) 2019 Microsoft Corporation. All rights reserved.

  C:\Users\Nicola>java -version
  java version "14" 2020-03-17
  Java(TM) SE Runtime Environment (build 14+36-1461)
  Java HotSpot(TM) 64-Bit Server VM (build 14+36-1461, mixed mode, sharing)
  ```

  Analogously, type the command

  `javac -version`

  and press the Enter key. You should get a display like this:

  ```
  Microsoft Windows [Version 10.0.18362.720]
  (c) 2019 Microsoft Corporation. All rights reserved.

  C:\Users\Nicola>javac -version
  javac 14
  ```

Your first Java program

- Open the Text Editor you prefer (e.g. *Notepad* under Windows) and write your first Java program.

```java
public class FirstProgram {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

- Save the file as `FirstProgram.java` in a directory of your choice (e.g. `C:\myJavaPrograms`).

Compile and run your program

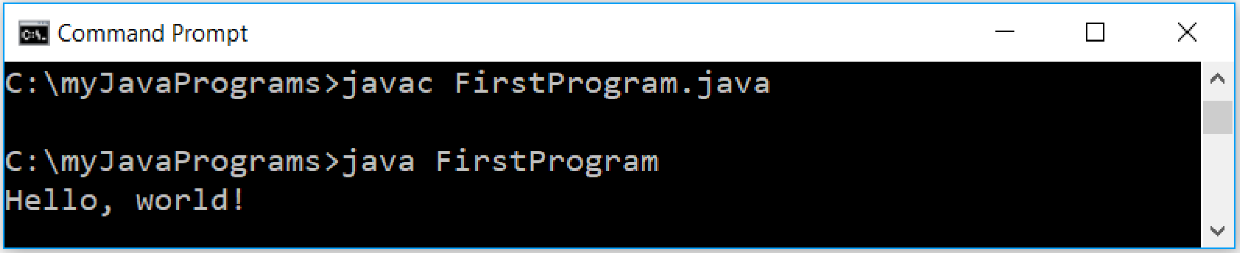
- Start a terminal window and browse to the folder containing your `FirstProgram.java` file (`C:\myJavaPrograms` in our case).
- Type the command

  ```
  javac FirstProgram.java
  ```

  and press the Enter key. The Java compiler will compile the program and will generate a file named `FirstProgram.class` (the bytecodes).
- To run the program, type the command

  ```
  java FirstProgram
  ```

  and press the Enter key. You should get a display such as this one:



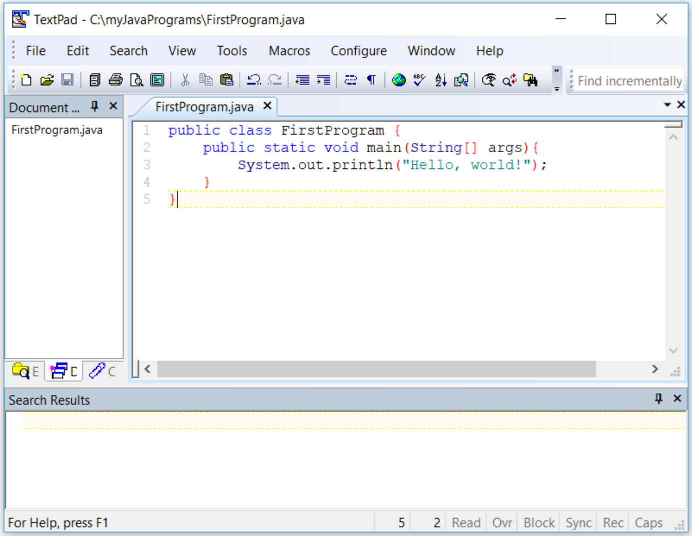
```
C:\myJavaPrograms>javac FirstProgram.java

C:\myJavaPrograms>java FirstProgram
Hello, world!
```

TextPad

- It is an advanced text editor for Windows platforms developed by Helios Software Solutions.
- Features:
  - syntax highlighting;
  - program compilation from within the editor;
  - program execution from within the editor.
- Visit the website at `https://www.textpad.com/download/index.html` and download the latest release (TextPad 8).

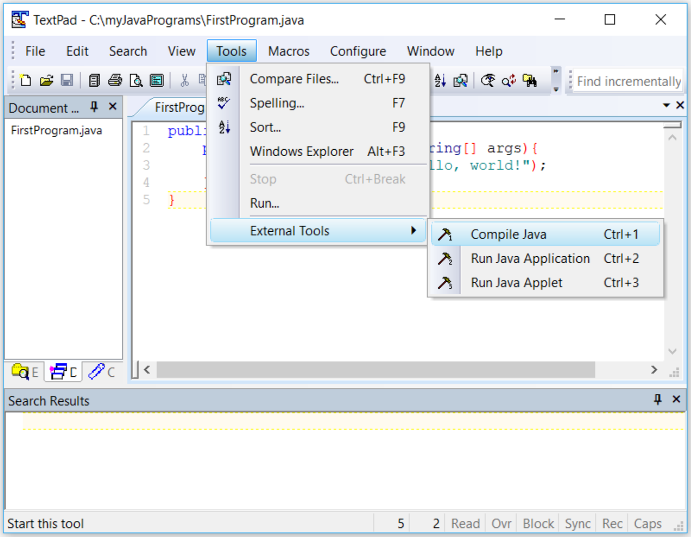
TextPad

- Open your `.java` file within TextPad.
- Go to `Tools -> External Tools -> Compile Java` (or press `Ctrl+1`) to compile the file.
- Go to `Tools -> External Tools -> Run Java Application` (or press `Ctrl+2`) to run the program.



TextPad - C:\myJavaPrograms\FirstProgram.java

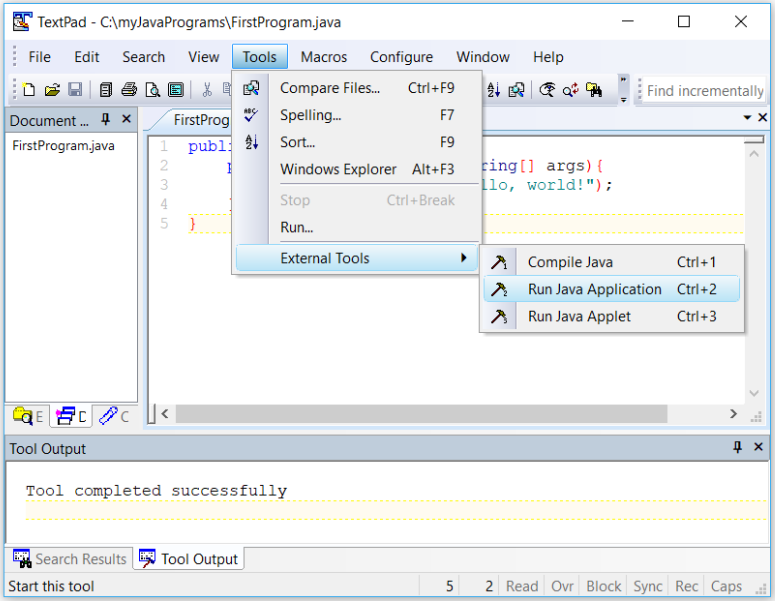
```
public class FirstProgram {
    public static void main(String[] args){
        System.out.println("Hello, world!");
    }
}
```



TextPad - C:\myJavaPrograms\FirstProgram.java

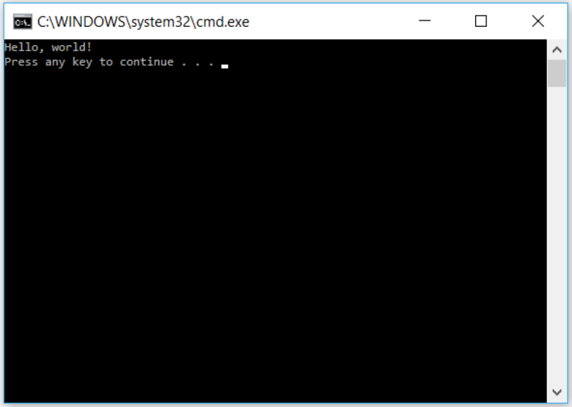
TextPad

- Open your `.java` file within TextPad.
- Go to `Tools -> External Tools -> Compile Java` (or press `Ctrl+1`) to compile the file.
- Go to `Tools -> External Tools -> Run Java Application` (or press `Ctrl+2`) to run the program.



```
publ                    ring[] args){
                        llo, world!");
}
```

```
Tool completed successfully
```



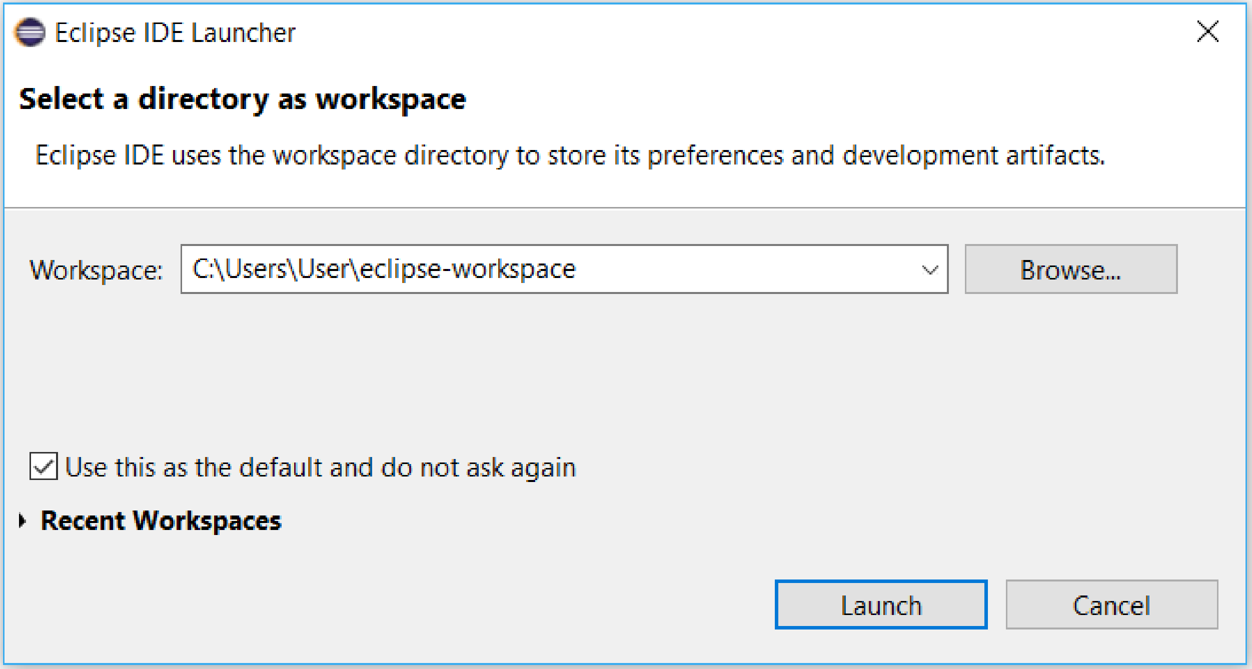
```
Hello, world!
Press any key to continue . . .
```

Eclipse

- It is an **Integrated Development Environment (IDE)** available for Windows, Linux and macOS.
- Features (non-exhaustive list):
  - syntax highlighting;
  - automatic compilation;
  - program execution from within the editor;
  - debugging;
  - autocomplete option.
- Visit the website at `https://www.eclipse.org/downloads/packages` and download **Eclipse IDE for Java Developers**.

Eclipse

- At the first start, set the workspace.



Eclipse IDE Launcher

Select a directory as workspace

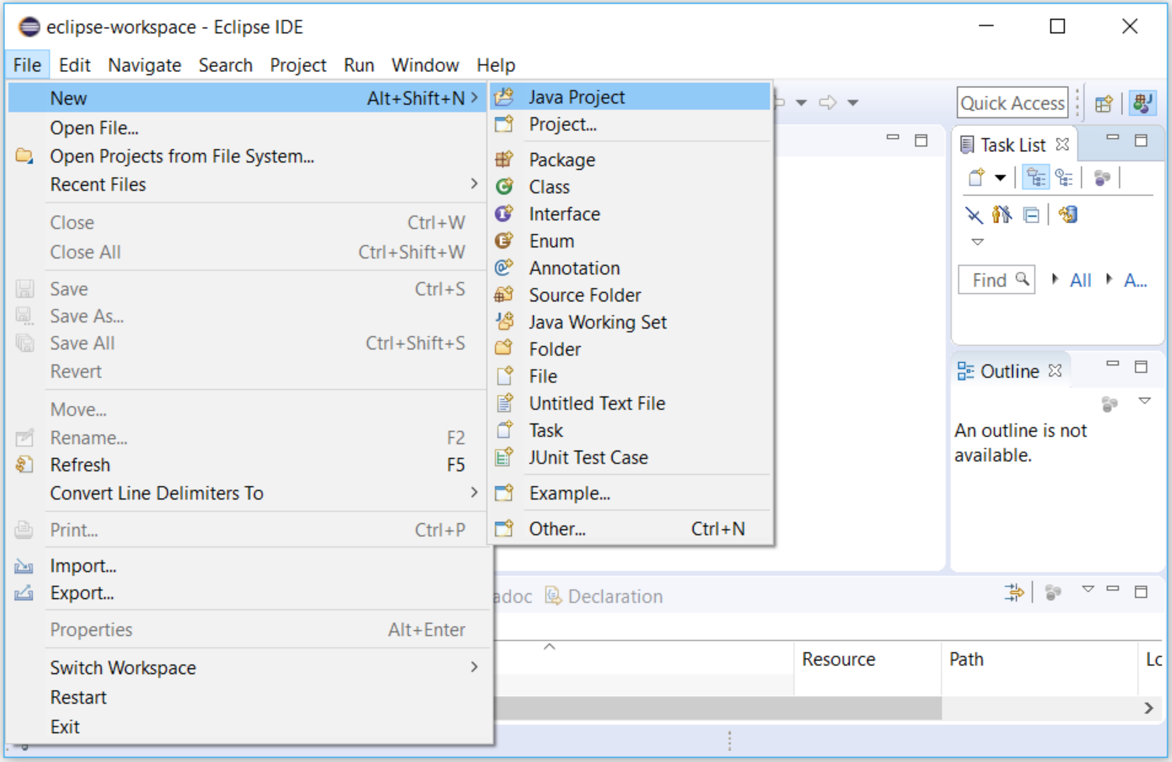
Eclipse IDE uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Users\User\eclipse-workspace

☑ Use this as the default and do not ask again

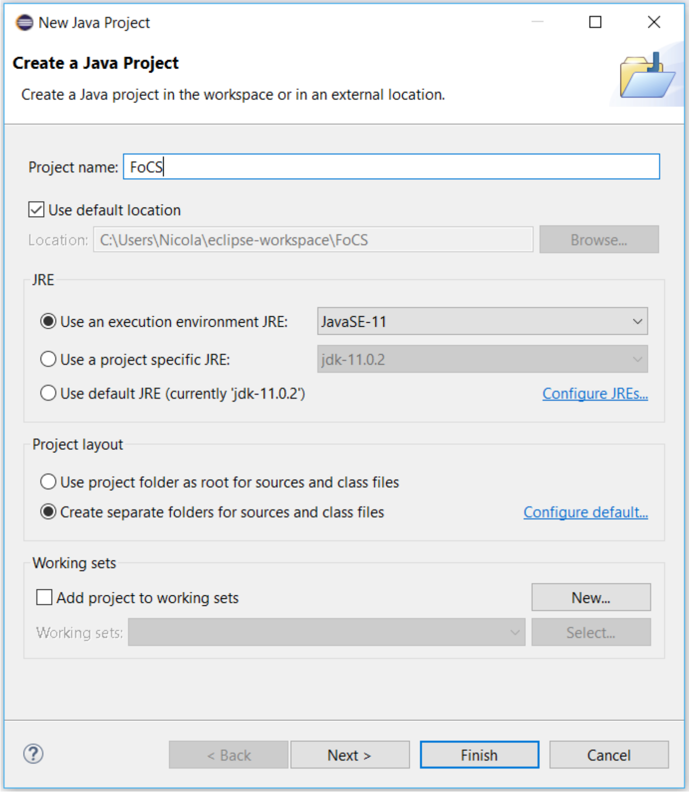
Eclipse

- Create a new Java Project. Go to `File -> New -> Java Project` (or press `Alt+Shift+N` and select `Java Project`).



Eclipse

- Give a name to your project (e.g. `FoCS`).



New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: FoCS

☑ Use default location

Location: C:\Users\Nicola\eclipse-workspace\FoCS

JRE

● Use an execution environment JRE: JavaSE-11

○ Use a project specific JRE: jdk-11.0.2

○ Use default JRE (currently 'jdk-11.0.2')

Project layout

○ Use project folder as root for sources and class files

● Create separate folders for sources and class files

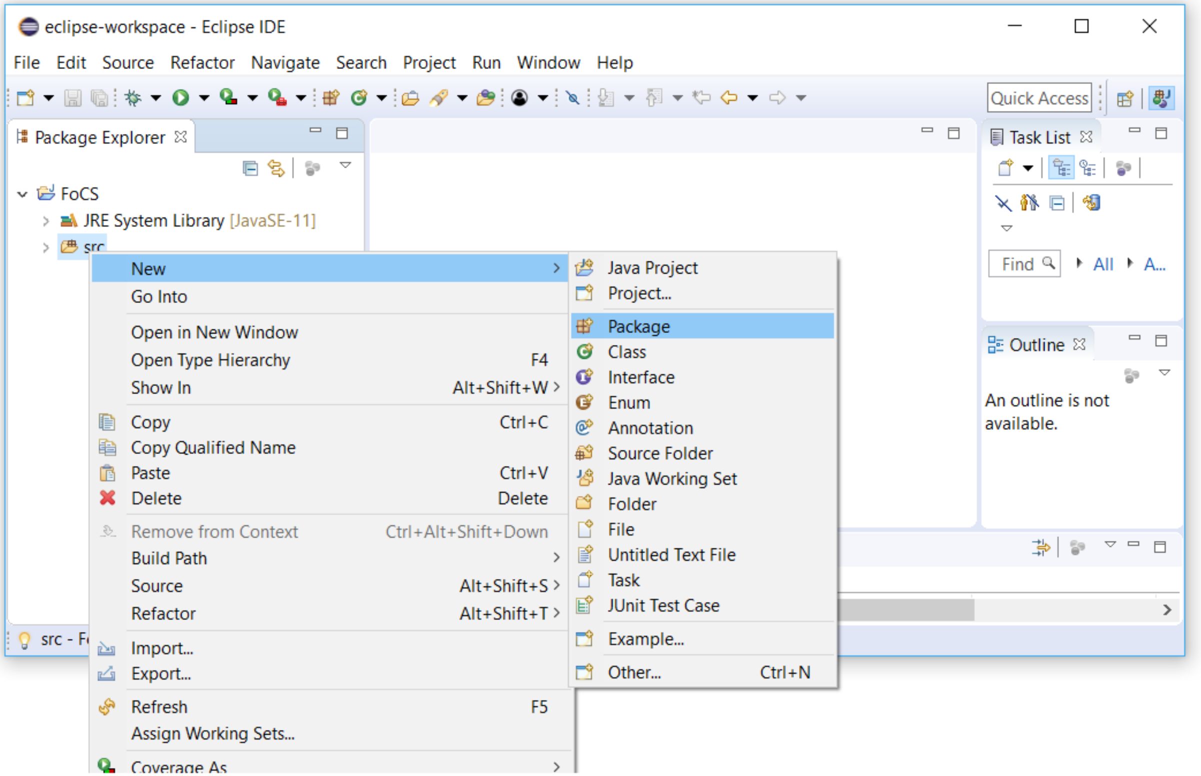
Working sets

□ Add project to working sets

Working sets:

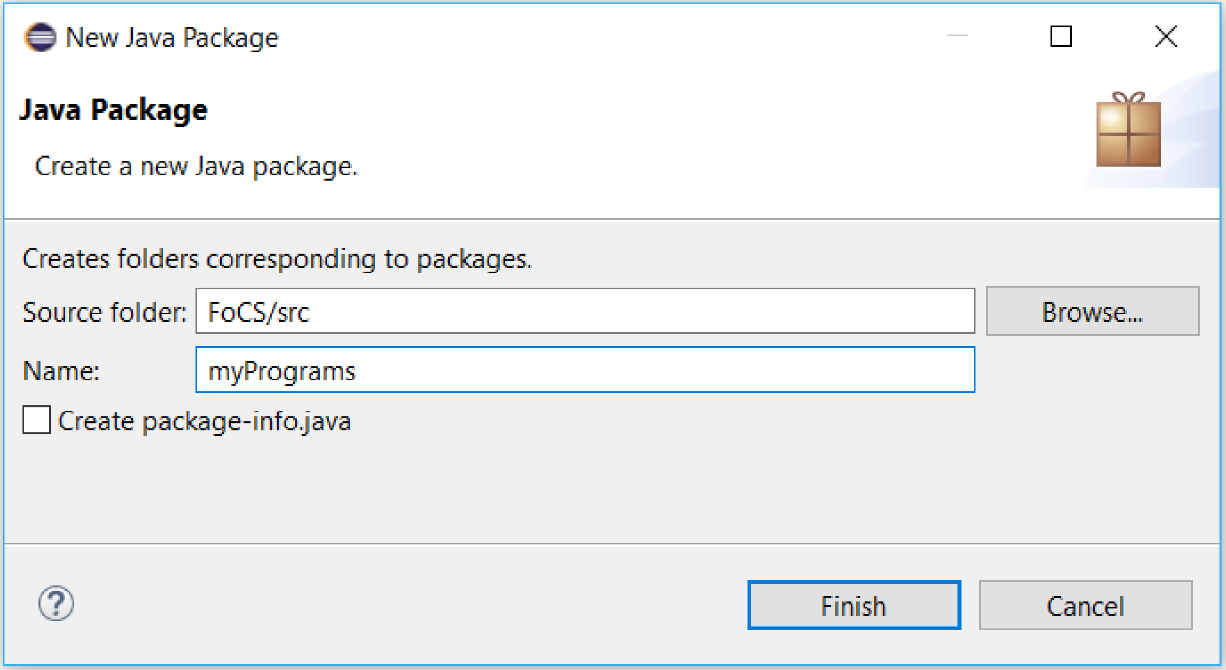
Eclipse

- Create a new package. Right-click on the `src` folder inside your project and go to `New -> Package`.



Eclipse

- Give a name to the package (e.g. `myPrograms`) and click on Finish.



New Java Package

Java Package

Create a new Java package.

Creates folders corresponding to packages.

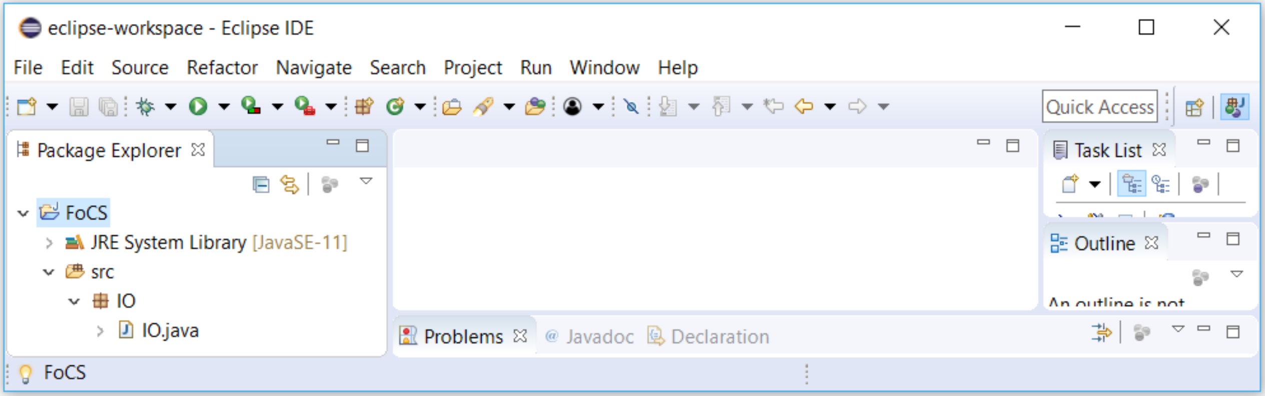
Source folder: FoCS/src

Name: myPrograms

□ Create package-info.java

Eclipse

- Download the `IO` package from the website of the course.
- Import the `IO` package inside your project (just *cut and paste* the package inside the `src` folder of your project).


eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
FoCS
JRE System Library [JavaSE-11]
src
IO
IO.java
Task List
Outline
An outline is not
Problems
Javadoc
Declaration
FoCS

Eclipse

- Create a new class. Right-click on the `myPrograms` package and go to `New -> Class`.
- A dialog will open. Give a name to the class (e.g. `FirstProgram`) and click on Finish. A new `.java` file (called `FirstProgram.java`) will be generated inside the package.

```java
package myPrograms;
public class FirstProgram {

}
```

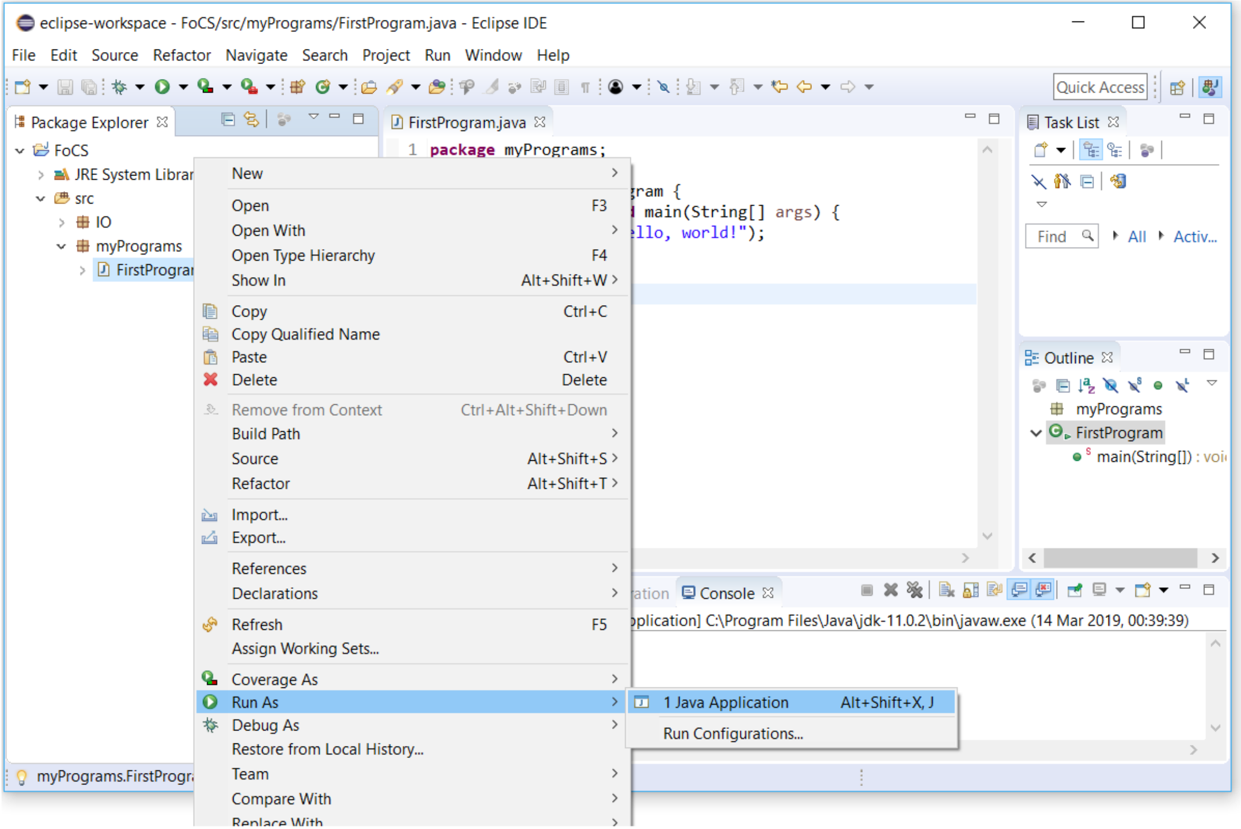
- Edit the class by writing your code.

```java
package myPrograms;
import IO.*;
public class FirstProgram {
    public static void main(String[] args) {
        IO.println("Hello, world!");
    }
}
```

- Save changes and run it.

Eclipse

- In the Package Explorer on the left, right-click on the `FirstProgram.java` file and go to `Run as -> Java Application` (or press `Alt+Shift+X,J`).



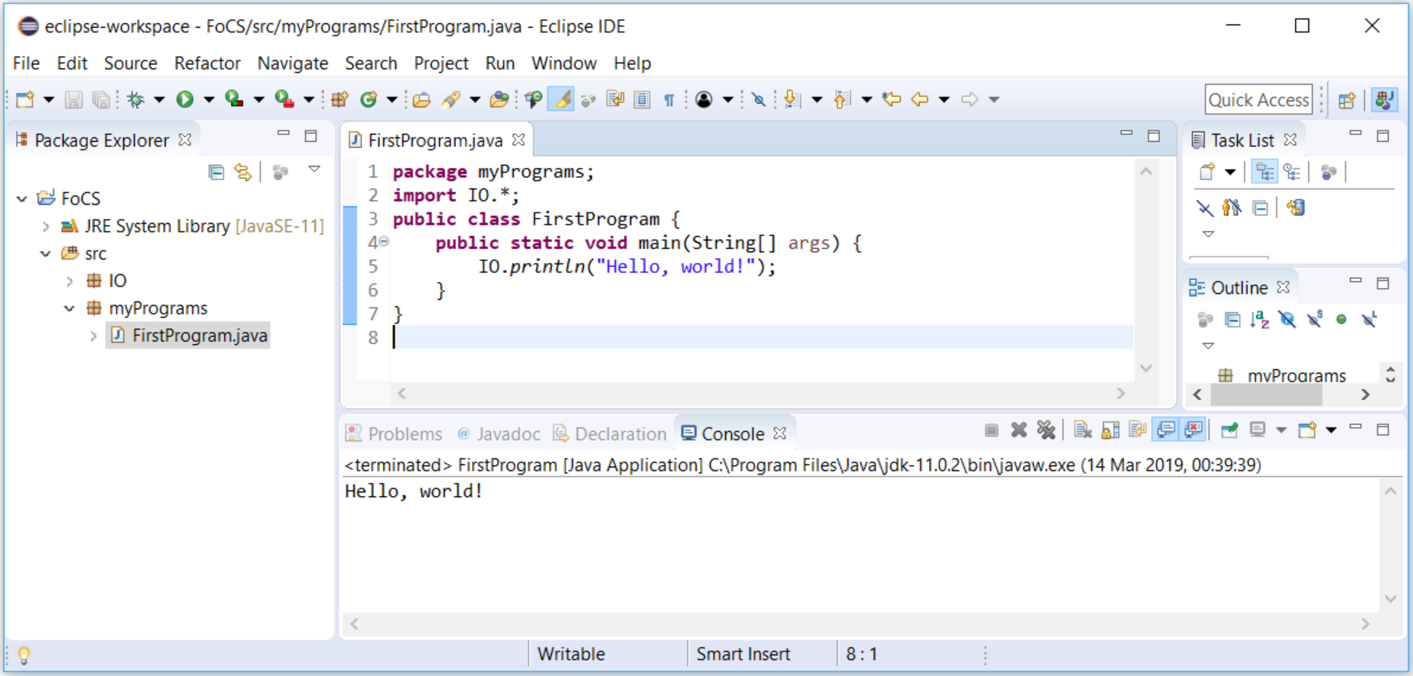
eclipse-workspace - FoCS/src/myPrograms/FirstProgram.java - Eclipse IDE

```
package myPrograms;

ram {
d main(String[] args) {
llo, world!");
```

```
pplication] C:\Program Files\Java\jdk-11.0.2\bin\javaw.exe (14 Mar 2019, 00:39:39)
```

Eclipse

- You will see your prints in the console below.



```java
package myPrograms;
import IO.*;
public class FirstProgram {
    public static void main(String[] args) {
        IO.println("Hello, world!");
    }
}
```

```
<terminated> FirstProgram [Java Application] C:\Program Files\Java\jdk-11.0.2\bin\javaw.exe (14 Mar 2019, 00:39:39)
Hello, world!
```

IO.java

- You will be provided with the `IO` package. It contains the `IO.java` file, hosting the `IO` class definition.
- It is a utility class designed to support you in IO operations.
- Output methods:
  - `IO.print(<expression>);`
  - `IO.println(<expression>);`
  - `IO.println().`
- Input methods:
  - `IO.readString(<prompt>);`
  - `IO.readInt(<prompt>);`
  - `IO.readLong(<prompt>);`
  - `IO.readShort(<prompt>);`
  - `IO.readByte(<prompt>);`
  - `IO.readFloat();`
  - `IO.readDouble().`