TND002 Object-oriented programming

Lecture 1: Introduction

Mark Eric Dieckmann,
MIT division,
ITN, Linköping University.

January 11, 2023

Outline of the lecture

- Course information.
- Course organisation and assessment.
- How do I get started? (What software do I need?)
- Primitive variables and Strings in Java.
- We write "Hello world" to the console.

Course content

- Primitive and reference variables.
- Inbuilt and own classes.
- Console and disk IO.
- Static and dynamic arrays.
- Inheritance.
- Abstract classes, interfaces and methods.
- Graphical user interfaces.

Course organization

12 Lectures: Lectures are primarily theoretical and supported by some coding examples (2h).

10 Lessons: We will implement code together (2h).

6 Labs: You will implement code. A lab assistant will provide some support (4h).

Revision: You can present completed labs via Zoom. Additional sessions can be scheduled if needed.

In the labs, priority is given to the implementation. We can arrange extra times when you can present your completed lab.

Course literature and assessment

Literature:

- The course does not follow a coursebook and I don't recommend buying one.
- Course material is placed on LISAM.

Examination

- Labs (3 hp).
- Exam with computer support (3 hp).
- The course grade equals the grade in the exam.

Object-orientation

- Object orientation pushes further the concept of subprograms (list of tasks).
- Object-oriented programming bundles related information, which is similar to collecting documents in folders.
- We will use Java as an object-oriented programming language.
- Java is an important language for writing internet-based applications.
- Java provides a standard (*Java Standard Edition* (*SE*)) that is understood by all computers.

The Java virtual machine (JVM)

- We use an *Integrated Development Environment* (IDE) to implement and run *Java* programs.
- Source code is stored in files ending with the suffix *.java*.
- "Compiling" with the IDE creates *byte code*, which is a standardized and portable binary format.
- Files with byte code end with the suffix *.class*.
- Java is an interpreted language: .class files are not executable (C++ compilers create executables).
- Byte code can be run on any computer with the Java Virtual Machine (JVM) installed (in Java SE).

Getting the software

You download the Eclipse IDE 2022-12 from https://www.eclipse.org/downloads/

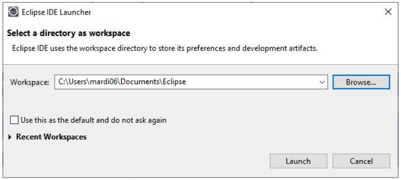
You also need the Java Standard Edition (It allows you to compile and run Java code)

You download it from: https://jdk.java.net/java-se-ri/19

Version 19 is the latest official one. I run Version 17.

You can copy the directory into the Eclipse directory in order to avoid setting search paths in Eclipse.

Launching Eclipse 2022-12


Eclipse IDE Launcher

Select a directory as workspace

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

Workspace: C:\Users\mardi06\Documents\Eclipse

The first window asks you for the directory, which will be used by the Eclipse IDE.

Create the directory before you launch the IDE.

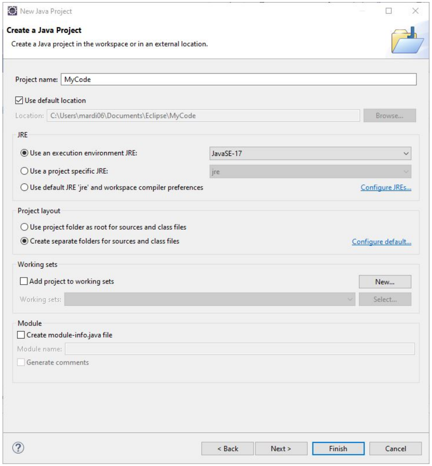
If you use the PCs in the computer labs, then select a location where the code is stored permanently.

Once you click on launch, your IDE will appear.

Launching Eclipse 2022-12

Close the Welcome window and click on "New" on the top left.

Select "Project" and "Java Project" and click on "Next".



Create a Java Project

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

Project name: MyCode

[✓] Use default location

Location: C:\Users\mardi06\Documents\Eclipse\MyCode

JRE

- [● Use an execution environment JRE:] JavaSE-17
- ○ Use a project specific JRE: jre
- ○ Use default JRE 'jre' and workspace compiler preferences

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:

Module

- ☐ Create module-info.java file
- Module name:
- ☐ Generate comments

This window pops up.

Pick a "Project" name that starts with an uppercase letter. I use "MyCode".

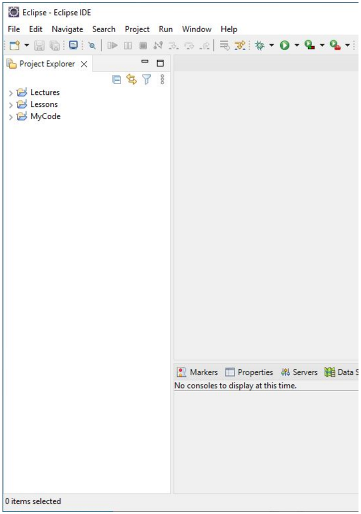
Do not "Create a module-info.java file".

Do not "Open the Java perspective".

Click on Finish.

Launching Eclipse 2022-12

The project directory is visible on the left side.


Eclipse - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Project Explorer
Lectures
Lessons
MyCode
Markers Properties Servers Data S

No consoles to display at this time.

0 items selected

Click on the icon to the left of the directory "MyCode".

Right-click on "src" on the project directory.

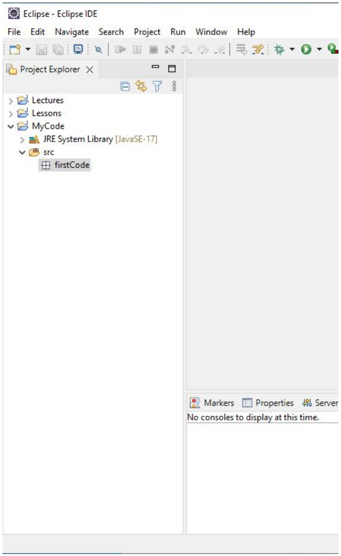
Select "New" and "Package".

Package names should start with a lowercase letter.

Do not create a package-info.java file.

Create the package (for example "firstProgram").

Launching Eclipse 2022-12


Eclipse - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Project Explorer
Lectures
Lessons
MyCode
JRE System Library [JavaSE-17]
src
firstCode
Markers Properties Server
No consoles to display at this time.

You will have one "executable" that accesses other "files".

⇒ Your code is distributed over several files.

Packages collect files that belong to the same program framework.

You will create one package for every lab, lesson, or for your own program frameworks.

Launching Eclipse 2022-12

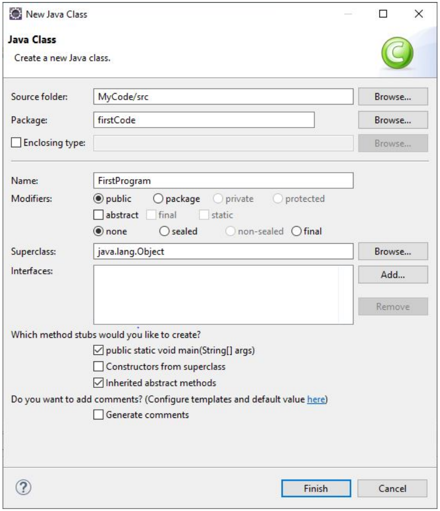
Right-click on the package name and select "New" followed by "Class".

Class names must start with an uppercase letter.

Separate words should start with an uppercase letter.

I chose "FirstProgram".

Select "public static void main(String[] arg)".



Source folder: MyCode/src
Package: firstCode
☐ Enclosing type:

Name: FirstProgram
Modifiers: [✓ public] package private protected
☐ abstract ☐ final ☐ static
[✓ none] sealed non-sealed final
Superclass: java.lang.Object
Interfaces:

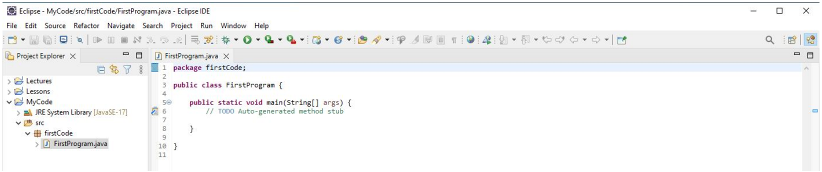
Which method stubs would you like to create?
☑ public static void main(String[] args)
☐ Constructors from superclass
☑ Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
☐ Generate comments

Launching Eclipse 2022-12

The package names in the package explorer and in the source code must be equal.

The class name (behind "public class") in the code must equal the name of the file in the package.

If you must change it, right-click the file. Choose "Refactor" followed by "Rename".


Eclipse - MyCode/src/firstCode/FirstProgram.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Lectures
Lessons
MyCode
JRE System Library [JavaSE-17]
src
firstCode
FirstProgram.java
FirstProgram.java

```
package firstCode;

public class FirstProgram {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}
```

The symbol // starts a comment.

The green button with the white triangle starts the program.

Our first program

public means that the code is visible and accessible from outside the class and package.

```
package firstCode;
public class FirstProgram {
public static void main(String[] args) {} }
```

All code of the class goes in between the outer braces { }.

main(..) is a method and *void* means that it does not return any value. It is called first when you run the code.

args is a string array through which you can pass values into the code when it starts. We will not use that in TND002.

Our first program

We can write "Hello world" to the console with the code:

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

The method "println(arg)" always interprets "arg" as a string. How about other variables?

The same "primitive variables" exist in C++ and Java.

The size of "primitive variables" is known at compile time.

Examples: integer variables "int" get 4 bytes, double precision variables "double" 8 bytes and booleans "boolean" get 1 bit.

Writing out primitive variables

In what follows, we drop the line starting with "package".

```
public class FirstProgram {
public static void main(String[] args) {
System.out.println(1); // interprets 1 as a string.
System.out.println(1+1); // interprets 1+1= 2 as a string.
System.out.println("Hello world\n" + 1 + 1); // writes "Hello
world" and "11" on two lines (\n is new line).
System.out.println(1 + 1 + " = result"); // writes "2 = result".
System.out.println(true); // writes the boolean "true" }}
```

The method "println(arg)" always interprets "arg" as a string.

The effect of "+" depends on the preceding argument.

Summary

We presented

- course information.
- conditions to pass and to get a better grade.
- how to get and use the required software.
- Primitive variables and Strings in Java.
- how to write "Hello world" to the console.