# Object-Oriented Programming

## Life-cycle of Programs and the Java Virtual Machine

Dr. Seán Russell
`sean.russell@ucd.ie`

School of Computer Science,
University College Dublin

September 4, 2018

# Table of Contents

1. Structure of a Computer
2. Life-Cycle of a Program
   - Problems with Machine Code
3. The Java Virtual Machine
4. Differences between C and Java
5. First Java Program
6. Integrated Development Environment
   - Basics of using Eclipse

## Learning outcomes

After this lecture and the related practical students should...

- understand the structure of computers, and the life cycle of C programs
- understand how Java code is compiled and executed
- understand some of the features of the Java programming language
- be able to compile and execute a basic Java program

## Main Components of a Computer

- Central Processing Unit (CPU)
- Main Memory (Short term)
- Hard Drives (Long term memory)

## Central Processing Unit (CPU)

- Very low level instructions called **machine code**
- Difficult to understand
- Represented in hexadecimal (binary is too long)
- Can be shown as an assembly language

## Machine Code and Assembly Language

Example of machine code:

```
78A9808D1503A92D8D14035860EE20D04C31EA
```

Example of assembly language:

```
SEI
LDA #$80
STA $0315
LDA #$2D
STA $0314
CLI
RTS
INC $D020
JMP $EA31
```

## High Level Languages

- Makes programming easier
- Easier for us to understand
- Code must be changed to something the CPU understands
- This is done by the compiler

## Main Memory

- Memory has all of the data that the computer is using
- Programs being executed and their data
- Data stored as groups of bits associated with an address

## Hard Drives

- This is where we store all of the data when it is not being used
- When code is not being executed, it is just a file
- It becomes a program when it is loaded into main memory and is executed

## Life-Cycle of a Program

- The stages of a C program
    1. As a text file
    2. As an executable file
    3. As an executing program

## Program as a Text File

- A C file is just a text file with a special name
- This file is saved on a hard drive of the computer, e.g. `E:\program.c`
- The CPU does not understand the text

## Program as an Executable file

- The compiler takes this C file and produces an exe file containing the machine code instructions
- The compiler is just a program that we must execute
- Creates new file (`E:\program.exe`) we can execute

## Program when it is Executing

- When we execute the program, the machine code is loaded into memory
- The CPU executes the instructions, one at a time
- Any variables stored in the program must also be created in memory when in scope

## Problems with Machine Code

- Computer systems are different
- Operating systems provide different libraries

## Different Computer Hardware

- Different types of CPUs cannot execute the same program
- Different CPU types can have different instructions
- Programs must be compiled again for the different CPU types

## Different Operating Systems

- Many operations (like printing) use parts of the operating system called **libraries**
- A program compiled on macOS uses macOS libraries
- Can not execute on Windows

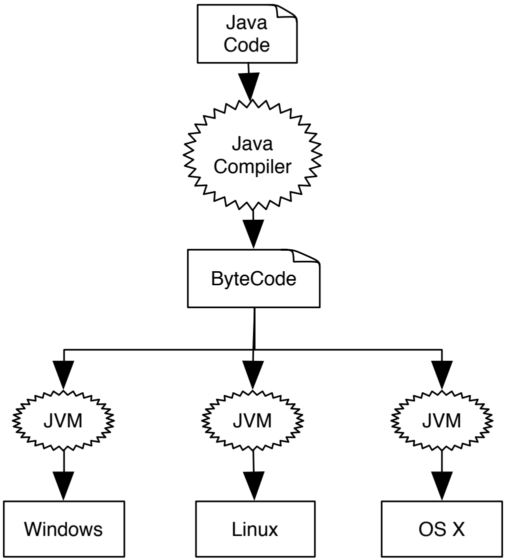
## The Java Virtual Machine

- Java makes it easier to develop code for **any** computer
- This uses the Java Virtual Machine (JVM)
- The JVM has a set of instructions and libraries
- For all CPUs and operating system, a JVM was created which converts the instructions for that CPU

## Bytecode

- A Java program is compiled to **bytecode**
- Bytecode is the program represented for the JVM
- Java bytecode can be executed by **any** JVM

# Java Virtual Machines



## Compiling Java Programs

- The compiler that we use in Java is called `javac`
- This converts our Java code into bytecode in one or more class files (`Something.class`)
- E.g. `E:>javac P.java`

## Executing Java Programs

- CPU doesn't understand Bytecode, must use JVM
- The application that we use to execute a Java program is called `java`
- We need to tell it the name of the class that contains the program
- Assuming that the compiler created a file named `P.class`, we use the name `P`
- E.g. `E:>java P`

## Differences between C and Java

- Java is an **object-oriented** programming language
- Java does not use pointers
- Java manages memory
- Java programs can recover when a problem happens during execution

## Pointers

- Java uses references
- Every object in Java is a reference

### Dereferencing

Java will automatically convert a reference to an object (called dereferencing) whenever we use it. This means that we never need to use * or & or anything similar.

## Memory

Java manages memory automatically for us.

- When we create an object the JVM allocates the memory (never need `malloc`)
- When we are no longer using an object the JVM frees the memory (never need `free`)

### Garbage collector

The part of the JVM that frees memory is called the garbage collector. Memory can only be freed when we don't have a reference to it, so we still have to think a little bit about memory!

## Errors

Java provides functionality to help us deal with errors that happen while our program is running (called runtime errors)

- Runtime errors in Java are called Exceptions
- It is possible to recover from a lot of different errors
- Sometimes we must include this code

## First C program

```c
int main(){

   printf("hello world!");

}
```

## First Java program

```java
public class Hello {

  public static void main(String[] args){

    System.out.println("hello world!");

  }
}
```

## First Java program

### Explanation of line 1

The first line performs one function, it declares a class

**public**

This specifies where in the program the class can be accessed from

**class**

This tells Java that this line will start the details of a class

**Hello**

This word tells Java what the name of the class will be

The bracket at the end is the beginning of the class

## First Java program

### Explanation of line 3

This line declares a method called **main**

- The same as `int main()` function in C
- This method will always be the same
- The parameter of the method is an array of Strings
- The bracket is the beginning of the method

## First Java program

### Explanation of line 5

This line prints out the message `hello world!`

- The program uses a method named println that prints a single parameter to the screen
- `System.out` tells Java where it can find the `println` method
- Just link in C most statement are finished with a semicolon (;)

## Components of a Java program

- Every program must have at least one class
- Within these classes there must be at least one main method
- Main methods are used to start executing all Java programs

### File name

The name of a class must match the name of File it is saved in. If we have a class named `Hello`, then it must be saved in the file Hello.java

## Integrated Development Environment

An Integrated Development Environment (IDE) is an application with a lot of tools that are useful for developing software. IDEs usually include:

- Text editor
- Compiler
- Syntax highlighting
- Built in program execution
- Code suggestions
- Debugging

There are a number of different IDEs, but we will be using the eclipse IDE - `http://www.eclipse.org`

# Using Eclipse

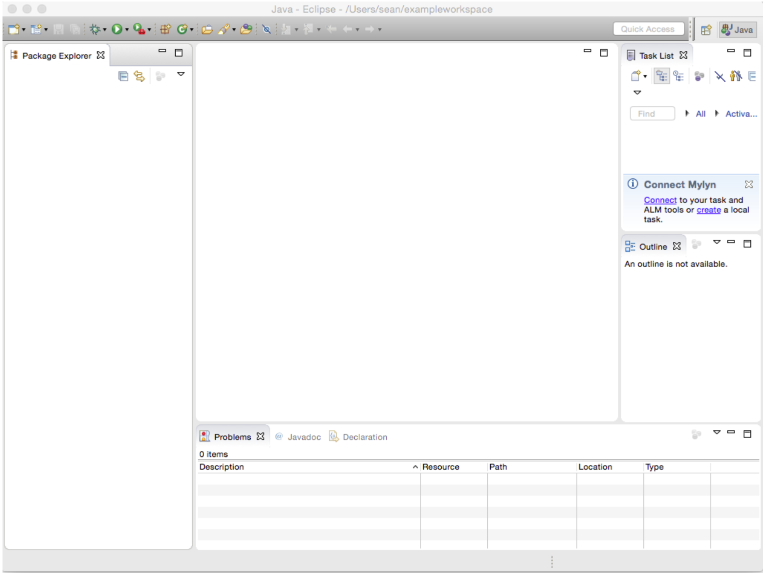
## Workspace

Eclipse is a very large and complicate program. This section will introduce some basics about how it works.

### Workspace

To use eclipse you must first select a workspace. This is just a folder on your computer where eclipse will save all of your code and programs. Select a location that is easy to get to because you will usually have to submit your work.

## Eclipse after you select your workspace



Connect Mylyn

Connect to your task and ALM tools or create a local task.

An outline is not available.

0 items

| Description | Resource | Path | Location | Type |
|---|---|---|---|---|

## Using Eclipse

### Projects

Eclipse groups programs together into projects. You cannot program using eclipse without a project.

- To create a new project click the new icon (📁▾) located at the top left.
- Then select Java project from the drop down menu
- In the Project name box type a name for your project and then click Finish
  - Use easy names such as Assignment 1 etc.
  - This will create a new folder in the workspace with that name
  - All of your code will be in this folder inside another folder named src
  - You cannot have two projects with the same name

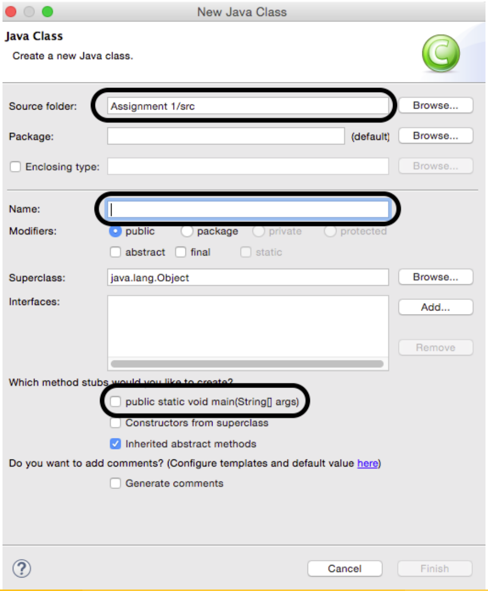
## Using Eclipse

### Creating Java files

Once you have a project created we can now start creating Java files.

- To create a Java file (the type we are creating is called a class file) click on the new icon again
- Then select class from the drop down menu
- In the window that appears we have to input some information
    1. The location we want to put the file (this should be the folder named src in our project) in the Source folder box
    2. The name of the class in the Name box
    3. We can get eclipse to automatically add a main method if we select the tick box named `public static void main(String[] args)`
    4. Finally click Finish and the Java file will be created!

# Eclipse new class wizard



New Java Class

**Java Class**

Create a new Java class.

Source folder: Assignment 1/src

Package: (default)

☐ Enclosing type:

Name:

Modifiers: [✓ public] package private protected

☐ abstract ☐ final ☐ static

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

## Using Eclipse

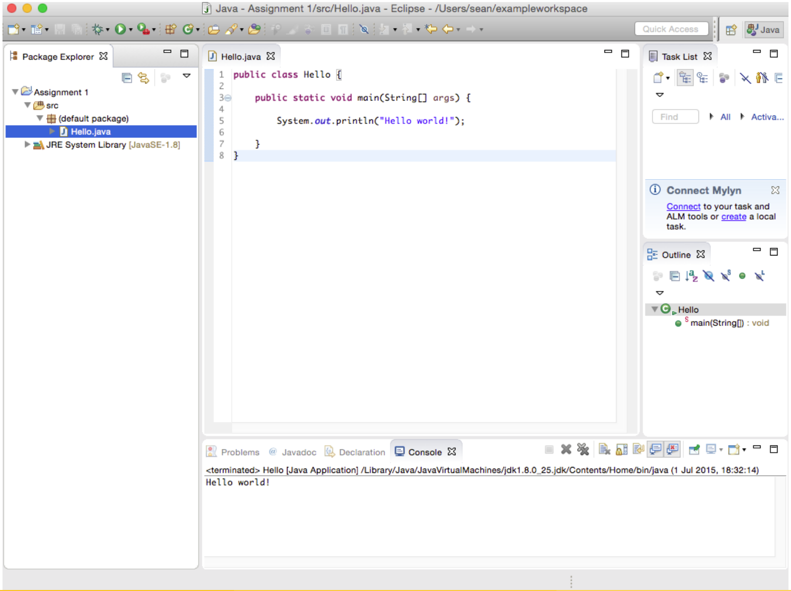
### Executing Java programs in eclipse

There are different ways to execute out code in eclipse. Here are a few steps to complete the task.

1. In the main window of eclipse right-click on the file you want to execute
2. in the drop down menu go down to the Run As menu and select Java Application

The program will be executed and the output will appear in a tab named console

# Eclipse after executing



```java
public class Hello {

    public static void main(String[] args) {

        System.out.println("Hello world!");

    }
}
```

Connect to your task and ALM tools or create a local task.

```
<terminated> Hello [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Contents/Home/bin/java (1 Jul 2015, 18:32:14)
Hello world!
```