University of York
Europe Campus
CITY College

Writing a Java program

CCS1110
Programming
Principles and Algorithms

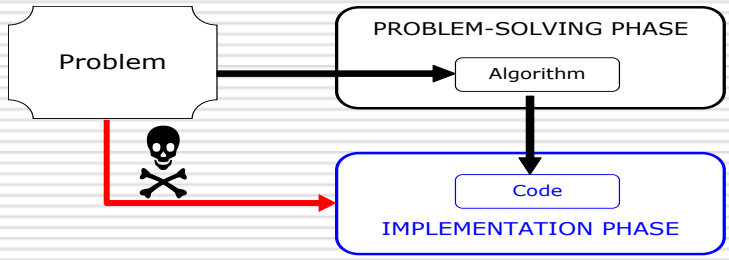
Dr Ioanna Stamatopoulou

What are computers and programs

- **Computers** are general purpose devices that **execute programs**
- **Programs** consist of instructions to the computer to perform a task in order to **solve a specific problem**
- Programs are written by **programmers** in specialised languages called **programming languages**
- Programs are referred to as **software,** whereas computers are referred to as **hardware**

So Programming is Problem-Solving!

- An algorithm a sequence of steps that solves a problem
- We cannot start coding/implementing if we have not figured out the solution first



Code Writing

- A program is written as **text** in a text editor and is saved in a file
- We call this the **source code**
- All files containing Java source code have the extension **.java**

Defining a Class

```
public class HelloWorld
{

} // end of class
```

- All Java source code must be enclosed within a **class**
- Start by declaring a **`public class`** and giving it a name (NO SPACES)
  - Class names start with Capital letters
  - Capitalise again when word changes
- Use **{** and **}** to denote the beginning and end of the class, respectively

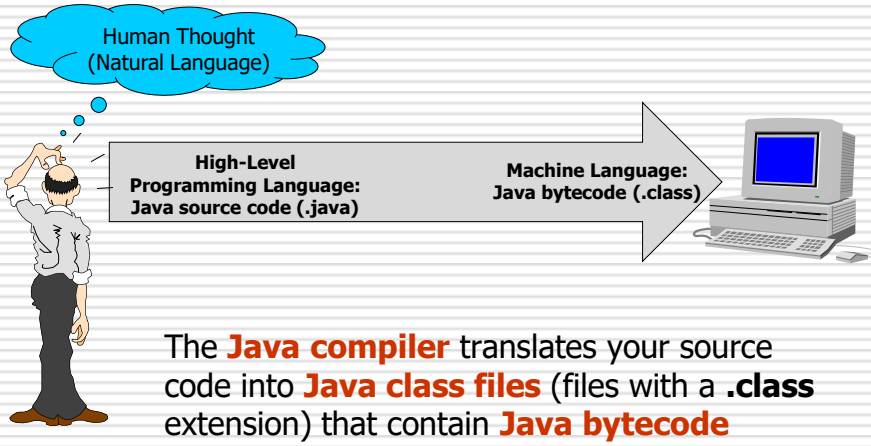
Saving your Class

- Save your file in a .java file
- The **name of the file should be exactly the same as the name of the class**
  - Otherwise you get a compiler error
- E.g. The code for the class HelloWorld should be saved in a file named: "HelloWorld.java"

Let's compile the class

- Open the command prompt:
  Windows Start > Run > cmd
- **C**hange the **D**irectory to where your .java file is:
  `cd M:\prog\lab01`
- Call the **Java c**ompiler for your file:
  `javac HelloWorld.java`
- Do you get errors?

What does the compiler do?



The **Java compiler** translates your source code into **Java class files** (files with a **.class** extension) that contain **Java bytecode**

Let's run this

- Call:
  `java HelloWorld`
- **Oops!** What's wrong?

Writing a Program

```
public class HelloWorld
{
  public static void main(String[] args)
  {

  } //end of main
} //end of class
```

- A **class is not a program**
  - It is not executable
- You need a **`main`** method **inside the class** to have a program
  - All program commands must be enclosed within the main, i.e. within the **{** and **}** of the main method
- Let's compile the program
- Let's run the program

Output

Output to the user

- The **`System.out.println()`** method is the simplest way to print something on the screen
- It prints the **`String`** we provide as an argument (enclosed in **`""`** within the parentheses) to the standard output (screen)

```
public class HelloWorld {
  public static void main(String[] args) {
    /*
       The method println prints "Hello World" to the standard
       output (screen)
    */
    System.out.println("Hello World!");
  } // end of main()
} // end of class HelloWorld
```

Output on the Screen:

```
Hello World!
```

Strings and Escape sequences

- A String is a sequence of characters (text)
- In Java, strings are enclosed within double quotes:
  `"some text…"`
- The **\n inside a String** are an **escape sequence**:
  - It has a special meaning and does not get printed per se
  - It create a **n**ew line within the String
- Try this:

Output on the Screen:

```
Hello World!
Hello students!
```

```
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello World!\nHello students!");
  } // end of main()
} // end of class HelloWorld
```


Output with a Graphical User Interface (GUI)

```
import javax.swing.JOptionPane;
/**
   My First Java program with a GUI output
   @author Ioanna
*/

public class HelloWorld
{
  public static void main (String[] args)
  {
    JOptionPane.showMessageDialog(null, "Welcome to CITY College");
  }
}
```

The method **`showMessageDialog()`** of the **`JOptionPane`** class expects two arguments (**`null`** and a String) and creates a window with the String printed on it.

General Principles of Java Programming

Java is **case sensitive!**

Case Sensitivity

- All names in Java are case sensitive (variable names, class names, method names, etc.)

```
MyFirstJavaClass
myfirstjavaclass
myFirstJavaClass
my_first_java_class
```

All these names are completely different for the JAVA compiler

Comments

Good programming practice

- They are addressed to humans reading our code
- They are ignored by the compiler
- There are two types of comments:
  - Single-line (inline) comments:

```
// Filename: HelloWorld.java
// A simple program, which prints a line of text
…
}     // class ends
```

  - Multi-line (block) comments:

```
/*
   Filename: HelloWorld.java
   A simple program that prints a line of text
*/
```


Code Formatting Rules

- Spaces and new lines are ignored by the compiler (unless inside a String) so the two following programs are identical:

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

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

- **BUT…**

Code Formatting Rules (cont'd)

- By convention all programmer strictly follow some formatting rules:
  - Keep curly brackets in a line of their own
  - Move **one tab to the right after** an opening curly bracket **{**
  - One command per line
  - **A closing curly bracket must appear exactly below its corresponding opening curly bracket**

All commands end with a semicolon ;

ATTENTION!

NEVER put a semicolon:
• before or after **{**
• after a **}**

Importing other classes

- Java provides a lot of useful classes, organised in class collections that are called packages

```
import javax.swing.JOptionPane;
```

- The above command imports the class JOptionPane, which may be found in the javax.swing package
- The showMessageDialog() method we used earlier is defined inside the JOptionPane class, and we may not use it unless we import the class

Importing other classes (cont'd)

- Why did we not import the System class in order to use the println() method?
  - All classes of the java.lang package are by default imported

Check list

- Can I create a class?
- Can I create an (empty) program inside my class (main method)?
- Can I compile my code?
- Do I know what the compiler does?
- Can I run my code?
- Do I know what a String is?
- Can I output a message to the user?
- Can I properly format my code?