CT874 Programming I

Introduction to the Java Platform & Java Applications

Contact Details

- Dr. Séamus Hill
- seamus.hill@nuigalway.ie

Objectives

- This module introduces the theory of object oriented programming (OOP) as well as its practice with the Java™ programming language
- The objectives of this course is to provide an understanding of OOP and to develop the fundamental programming skills required to create applications using Java™

Resources

- Resources
  - Head First Java 2nd ed., O'Reilly, by Kathy Sierra & Bert Bates
  - An Introduction to Object Oriented Programming with Java™ 5th Edition, by Thomas C. Wu
  - *Think Java*, by Allen B. Downey
    - http://greenteapress.com/thinkapjava/index.html
  - *Introduction to Java*, 8th Edition, by David J. Eck
    - http://math.hws.edu/javanotes/
- Oracle Java Tutorials
  - http://docs.oracle.com/javase/tutorial/

Course Notes

- Course Website
  - https://nuigalway.blackboard.com
  - Virtual Classroom
  - Lecture Slides, sample programs, videos & assignemnts
  - Announcements
- Lectures
  - Monday 15:00-16:50, Online/IT125 (TBC COVID)
- Lab Sessions
  - 2 hours per week, Online/Lab (TBC COVID)
- Assessment:
  - 20% Continuous Assessment (Lab Assignments)
  - 80% Written Exam

Learning Objectives

After this lecture you should be able to

- Describe Java's main characteristics
- Describe the java platform and how hardware independence is achieved
- Write simple Java programs
- Describe the process of creating and running Java programs

– ***Reading*** and ***studying*** recommended readings is essential to improve your understanding of the above

Technology - Compiler

- *Machine language* is made up of simple instructions that can be executed by a computer's CPU

- Programs written in *high level programming languages* need to be converted to machine language by a program known as a *complier*
  - *grossPay = basePay + overTimePay*
  - The complier translates the program to an executable machine language program which can be run the particular machine

- To run the program on a different machine architecture you need to re-translate the code using a different compiler

Technology - Interpreter

- An *interpreter* translates a high level programming program instruction by instruction
- An interpreter acts much like a CPU, with a form of fetch and execute cycle
- The interpreter runs in a repetition like fashion reading the program instruction by instruction performing the machine language commands to carry out each instruction
- Interpreters execute high level language programs but also allow the use of a machine language program for one computer on a different type of computer

Java Programming Language

- Java is a high-level programming language developed by Sun Microsystems (now merged into Oracle Corporation)
- Syntax based on C/C++
- Object Oriented
- Write once run anywhere (WORA) no need to recompile code to run on a different architecture
- Versions
  - Standard Edition (SE) – developing applications & applets
  - Micro Edition (ME) – development for consumer devices
  - Enterprise Edition (EE)- Adds additional technologies to SE in order to develop enterprise sever applications

Java Virtual Machine (JVM)

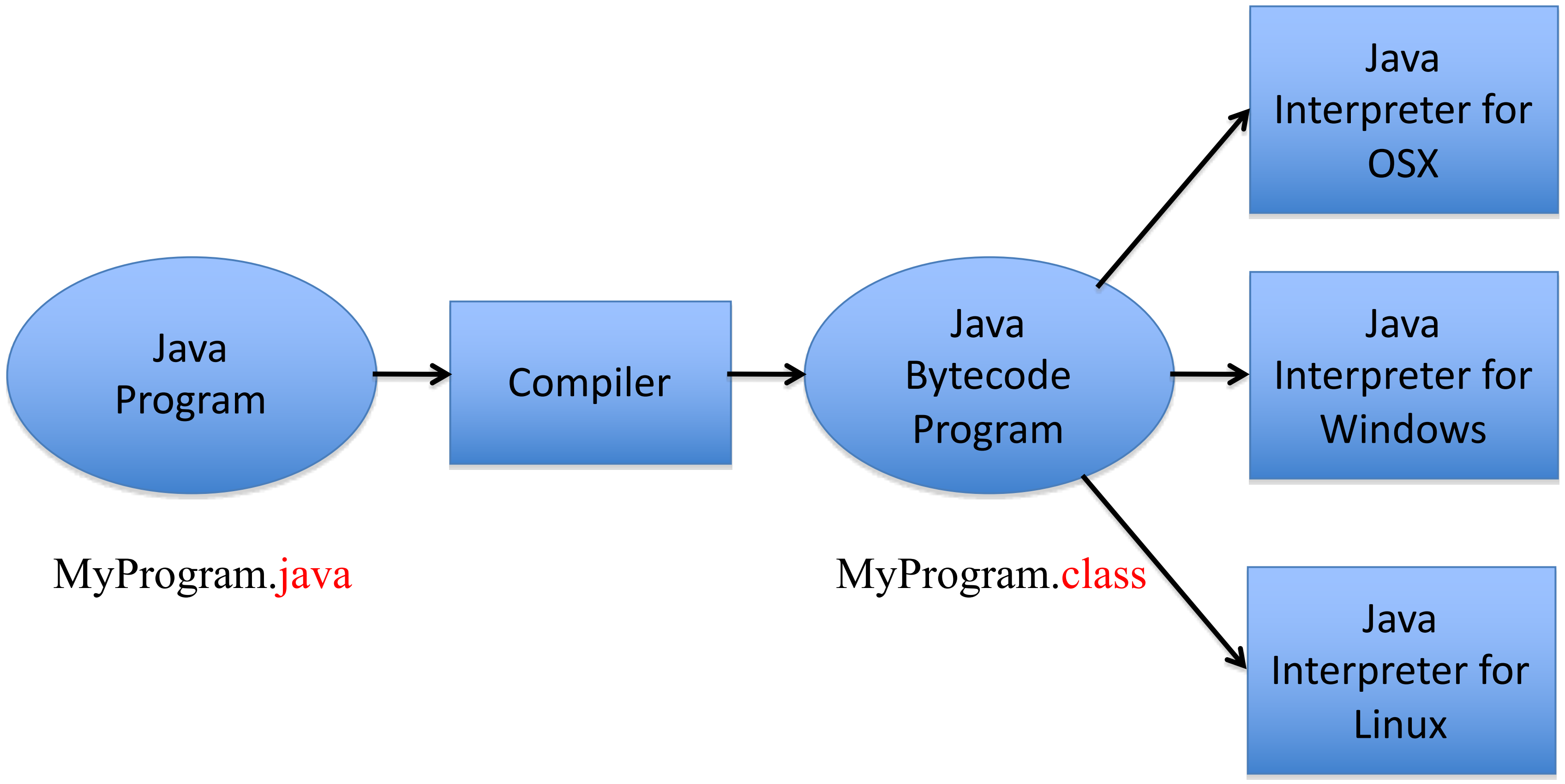
- Java programs are compiled into machine language (Java bytecode) for a virtual machine – the *Java Virtual Machine* (JVM)

- Any machine with an interpreter for Java bytecode can run a Java program

  – Each computer needs a JVM suitable for its architecture - *the* interpreter implements *the JVM*

  – With Java the same compiled program can be run on different computer architectures

Java Virtual Machine

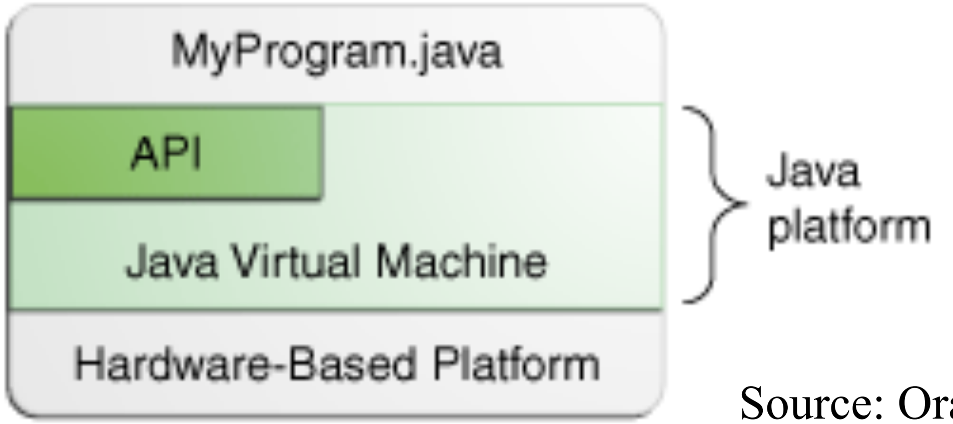


Java Platform

- Made up of two components
  - JVM
  - Java Application programming interface (API)
    - code libraries which provide pre-written functionality for programmers.
    - defines how a programmer can access the functionality contained within a code library
    - grouped into libraries of related classes and interfaces; these libraries are known as *packages.*
- Specifications
  - http://docs.oracle.com/javase/15/docs/api

Java Platform

- JVM & API insulate the program from the underlying computer hardware



Source: Oracle Java Tutorials

- API offers a large selection of useful classes which you can use in your own applications.

HelloWorld Program

```
/*
 * Text Output to Console/standard output
 */

class HelloWorld {   // begin class
    /* main method begins execution of application */
    public static void main(String[] args) {

        System.out.println("Hello World"); //display string

    } // end method

} // end class
```

Hello World Program

- Comments are ignored by the compiler

```
/*
 * Text Output to Console/standard output
 */

/* main method begins execution of application */

// display string
```

Hello World Program

- Class Definitions
  - keyword class begins the class definition for a class named *HelloWorld*, and the code for each class appears between the opening and closing curly braces
  - A class name is an identifier - a series of characters
  - Java is case sensitive

```
class HelloWorld {
        // display code here
}
```

Hello World Program

Main method

- main must be defined as shown; otherwise, the JVM will not execute the application

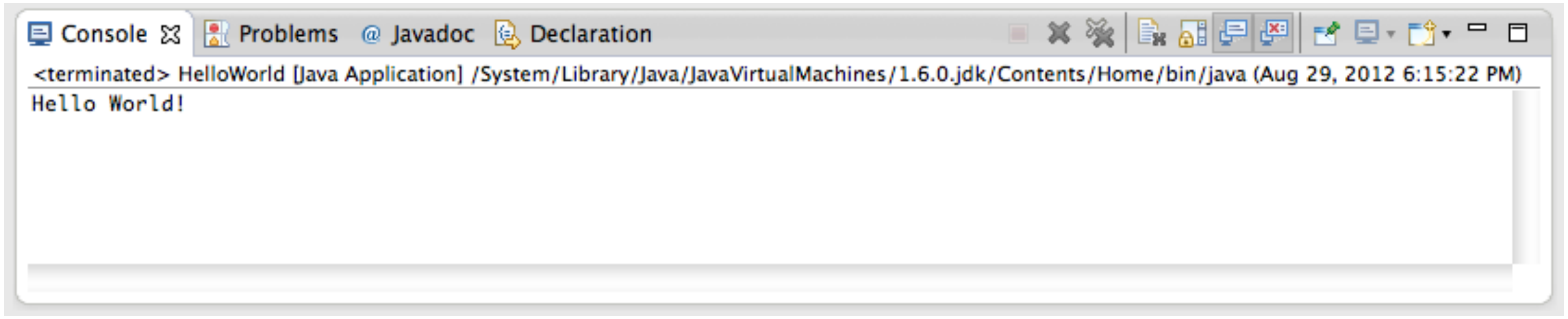
```
public static void main(String[] args) {
        …
}
```

- methods perform tasks and can return information when they complete their tasks
- keyword void indicates that this method will not return any information.
- the entry point for your application and will subsequently invoke all the other methods used in your program

Hello World Program

- The System class from the core library is used to print the "Hello World!" message to standard output (System.out object)
- A string is sometimes called a character string or a string literal.
- White-space characters in strings are not ignored by the compiler.

```
System.out.println("Hello World!");
```



```
<terminated> HelloWorld [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Aug 29, 2012 6:15:22 PM)
Hello World!
```

Java IDEs

- IDE: Integrated Development Environment
  - Editor, compiler, debugger, etc.
  - Graphical user interface
  - Aid productivity

- Some examples:
  - Eclipse: www.eclipse.org
  - IntelliJ www.jetbrains.com/idea/
  - NetBeans: www.netbeans.org/
  - Jbuilder: www.embarcadero.com/products/jbuilder
  - BlueJ: www.bluej.org

Eclipse IDE

- Eclipse can be used for this module:
  - Free, open-source
  - Backed by an industry consortium including IBM, Borland, RedHat and others
  - Widely used in industry
  - Wide range of available plug-ins and productivity tools for advanced users
  - Downloads and documentation on www.eclipse.org
- Installation is very simple:
  - Select appropriate version: **Eclipse IDE for Java Developers**
  - See installation note for Windows & Mac on Blackboard

Optional Exercise

- Optional exercises for this week are as follows;
  - Using Eclipse, write a short Java application which displays a string