Java

- Object oriented programming language
  - Syntax is similar to C/C++
  - A program is a class with a static **`main()`** method.

- Programs compiled to bytecode:
  - Bytecode runs on a Java Virtual Machine (JVM)
  - Same compiled code can run on any platform with a JVM without needing to be recompiled
  - JVM uses various techniques (e.g. JIT compilation) for improving performance by compiling all or some of the bytecode to native code before it is executed

Java

- Features garbage collection
  - Do not need to explicitly free dynamically allocated memory
- Substantial language and library support for:
  - Multithreaded programming
  - Network programming
  - Pattern matching & string manipulation

Simple Java program

```
/** This is a simple hello world program
 *  @author Rodney Summerscales
 */

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

Simple interactive Java program

```
import java.util.*;

public class HelloWorld {
  /** Compute the square of a number
   *  @param x The number to square
   *  @return The square of the number
  */
  public static int sq(int x) {
    return x * x;
  }
  /** Prompt the user for a number and display its square */
  public static void main(String[] args){
    Scanner in = new Scanner(System.in);
    System.out.println(”Enter a number: ");
    int n = in.nextInt();
    System.out.println(n + ” squared is "
                       + sq(n));
    in.close();
  }
}
```

Running Java programs

- To compile a java program at the command line type:

  `javac  HelloWorld.java`

  This generates the bytecode file **`HelloWorld.class`**

- To execute the main method in the class **`HelloWorld`** type:

  `java  -cp` *PATH* **`HelloWorld`**

  Where *PATH* specifies the path of **`HelloWorld.class`**

Running Java Programs

- If your system does not have a java compiler (**`javac`**) or JVM (**`java`**), you can obtain them by downloading and installing a java development kit (JDK) from

  http://java.sun.com/javase/index.jsp

  This site also contains documentation on the java API

Eclipse

- Popular IDE for developing java programs
  - Has industry support
  - Allows you to develop, run and debug java programs within a nice GUI environment
- Installed on lab computers in Stuart Building.
- Recommended (but not required) for this course
- Runs on Windows, OS X, Linux
- Free download
  - http://www.eclipse.org/downloads/
  - Depending on your system you may need to download a JRE (http://www.java.com)

Eclipse

- If using Eclipse, you need to create a java project before you begin to develop a new java program

- To create a new project:
  - From the main menu select:

    File -> New -> Java Project...
  - Enter the project name (e.g. HelloWorld)
  - Click Finish

Eclipse

- To add a new java file to your project:
  - From the main menu select:

    File -> New -> File...
  - Select the src directory in your project (e.g. HelloWorld)
  - Then give the file a name (e.g. HelloWorld.java)
  - Click Finish

- To run your program:
  - From the main menu select:

    Run -> Run...
  - Or, click the button on the toolbar with a white triangle inside a green circle

Working with multiple files

- It is relatively easy to compile and run java programs that use classes defined in different files and packages if you use an IDE such as Eclipse or NetBeans.
- If just using **`javac`** and **`java`** it can be more tricky.
- If all the java source files are in the same directory and they are all in the default package
  - compile each file with **`javac`**
    - **`javac -cp .`** <*SourceFile*>.java
  - Pass the name of the class with the main method you want to execute to the JVM
    - **`java -cp .`** <*NameOfClassWithMainMethod*>

Working with multiple files

- If some classes are in a separate package (e.g. **`mypackage`**)
  - Each file containing classes in **`mypackage`**, should be in the same subdirectory (i.e. **`mypackage/`**)
  - From the main source directory, compile all source files in the package subdirectory with

    `javac -cp . -d . mypackage/`<*SourceFileName*>**`.java`**

  - Compile all source file in default package with

    `javac -cp . -d .` <*SourceFileName*>**`.java`**

  - Run the program

    `java -cp .` <*NameOfClassWithMainMethod*>