Contents **Topic 01 - Java Fundamentals**

I. Introducing JAVA

- The White Paper for Java was announced in May 1996
James Gosling , Henry McGilton - Sun engineers

https://www.oracle.com/java/moved-by-java/

- Java is designed to achieve:
  - Simple — Java is partially modeled on C++, but simplified and improved.
  - Object oriented — Java was designed from the start to be object-oriented.
  - Distributed — Java is designed to make distributed computing easy with networking capability. Writing network programs is like sending and receiving data to and from a file.
  - Multithreaded — Multithread programming is smoothly integrated.
  - Dynamic — Designed to adapt to an evolving environment. Libraries can freely add new methods and instance variables without effecting clients. Straightforward to find out runtime type information.
  - Architecture neutral, Portable — With a Java Virtual Machine (JVM), one program can run on any platform without being recompiled.
  - High performance — High performance of interpreted bytecodes, efficient translation of bytecodes to machine code.
  - Robust — Java compiler, modified program constructs, runtime exception-handling
  - Secure — Security mechanisms to protect against harm caused by stray programs.

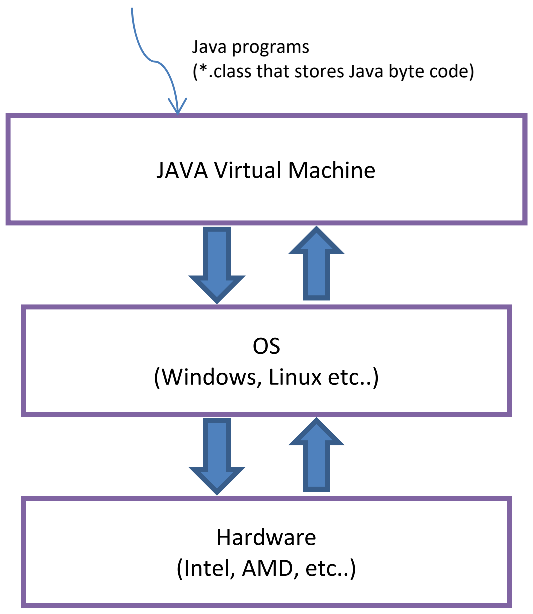
- The Java platform is available as different packages:

https://www.oracle.com/java/technologies/language-environment.html

  - JRE (Java Runtime Environment) – For consumers to run Java programs.
  - JDK (Java Development Kit) – For programmers to write Java programs.
    Includes JRE plus tools for developing, debugging, and monitoring Java applications.

- Once installed, the Java Virtual Machine (Java VM) is launched in the computer.

- During runtime, the Java VM interprets Java byte code and translates into OS calls.



- Java Versions:

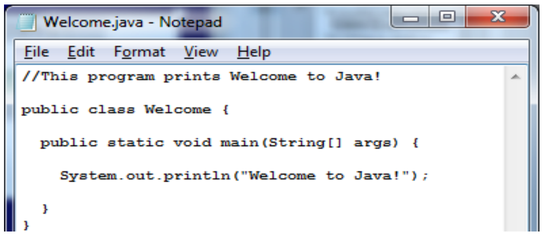
Version 1.0 (1995)	Version 1.5 (2004) a. k. a. Java 5
Version 1.1 (1996)	Version 1.6 (2006) a. k. a. Java 6
Version 1.2 (1998)	Version 1.7 (2011) a. k. a. Java 7
Version 1.3 (2000)	..
Version 1.4 (2002)	Version ?? https://www.oracle.com/java/ https://www.oracle.com/technetwork/java/java-se-support-roadmap.html

- Editions for different development purposes:
  - Java Standard Edition (J2SE)
    J2SE can be used to develop client-side standalone applications or applets.
  - Java Enterprise Edition (J2EE)
    Server-side applications such as Java servlets, Java ServerPages, and Java ServerFaces.
  - Java Micro Edition (J2ME)
    Applications for mobile devices such as cell phones.

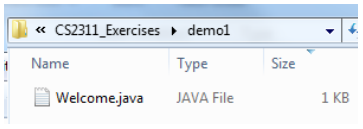
II. Compiling and Launching from Command-Line, IDE, A Simple JAVA Program

With JDK installed, you can compile and run Java programs in this way:

1. Create the source file: `Welcome.java`

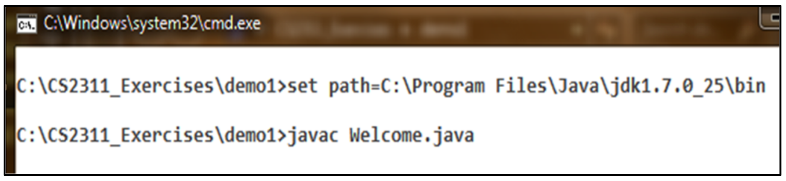


```
//This program prints Welcome to Java!

public class Welcome {

  public static void main(String[] args) {

    System.out.println("Welcome to Java!");

  }
}
```

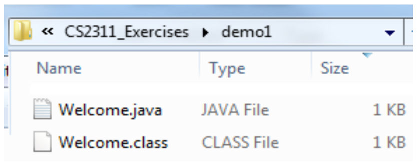


Name	Type	Size
Welcome.java	JAVA File	1 KB

2. At the command prompt, set path to JDK and then compile to give `Welcome.class`

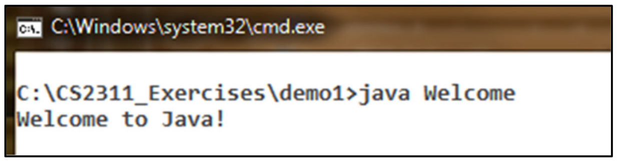


```
C:\CS2311_Exercises\demo1>set path=C:\Program Files\Java\jdk1.7.0_25\bin

C:\CS2311_Exercises\demo1>javac Welcome.java
```



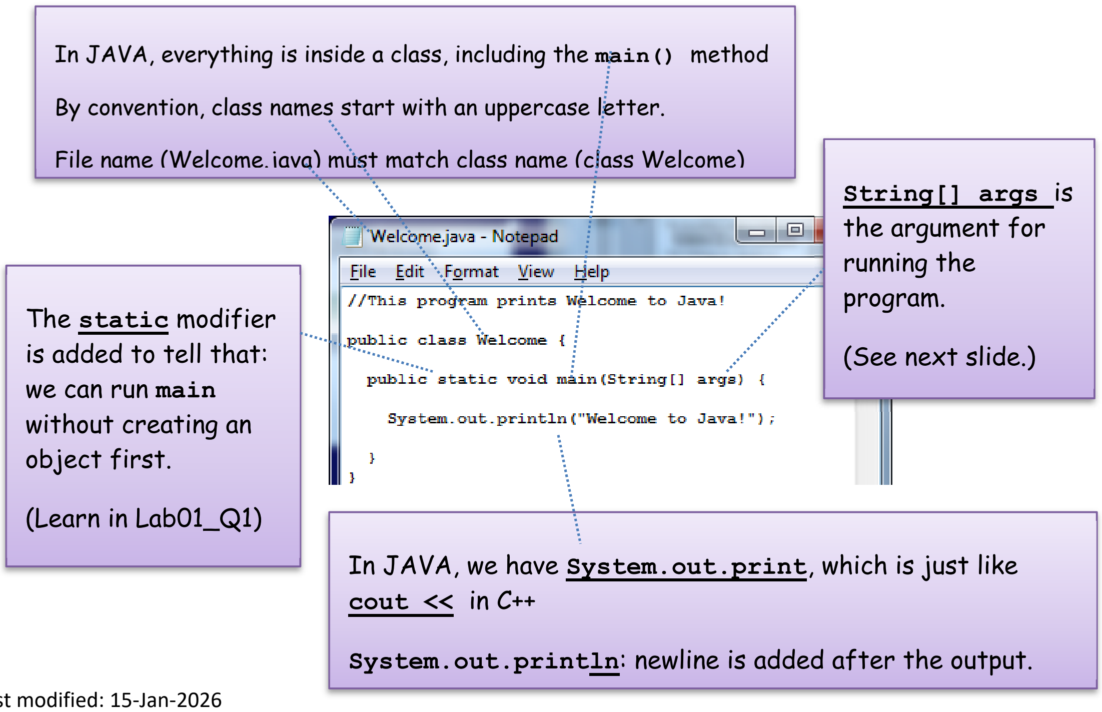
Name	Type	Size
Welcome.java	JAVA File	1 KB
Welcome.class	CLASS File	1 KB

3. Run it:



```
C:\CS2311_Exercises\demo1>java Welcome
Welcome to Java!
```

Explanation of the program:



In JAVA, everything is inside a class, including the `main()` method

By convention, class names start with an uppercase letter.

File name (Welcome.java) must match class name (class Welcome)

`String[] args` is the argument for running the program.

(See next slide.)

The `static` modifier is added to tell that: we can run `main` without creating an object first.

(Learn in Lab01_Q1)

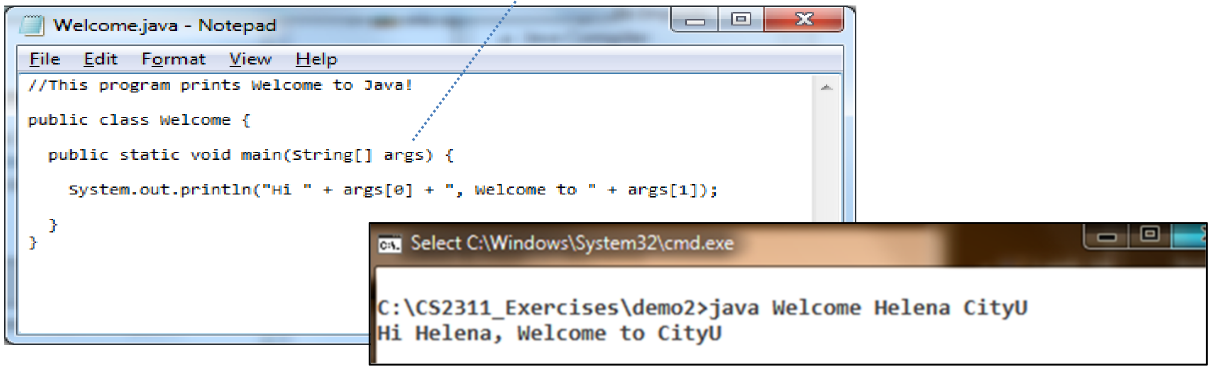
```
//This program prints Welcome to Java!

public class Welcome {

  public static void main(String[] args) {

    System.out.println("Welcome to Java!");

  }
}
```

In JAVA, we have `System.out.print`, which is just like `cout <<` in C++

`System.out.println`: newline is added after the output.

Arguments can be supplied to `main()` as an array of strings:

Example:



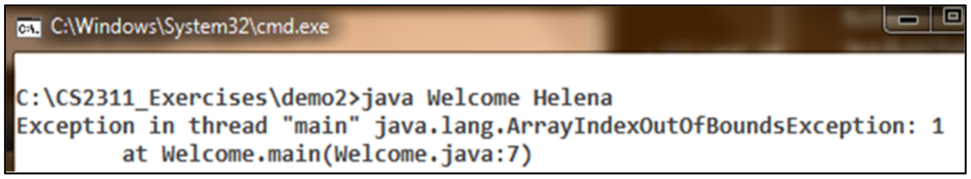
Welcome.java - Notepad
File Edit Format View Help
Select C:\Windows\System32\cmd.exe

```java
//This program prints Welcome to Java!

public class Welcome {

  public static void main(String[] args) {

    System.out.println("Hi " + args[0] + ", Welcome to " + args[1]);

  }
}
```

```
C:\CS2311_Exercises\demo2>java Welcome Helena CityU
Hi Helena, Welcome to CityU
```

Run-time exception:

The program code expects 2 arguments. But the only one is given.



```
C:\CS2311_Exercises\demo2>java Welcome Helena
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 1
        at Welcome.main(Welcome.java:7)
```

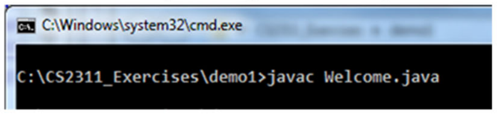
- Integrated Development Environments (IDE):

- VS Code
- IntelliJ IDEA
- Eclipse
- repl.it

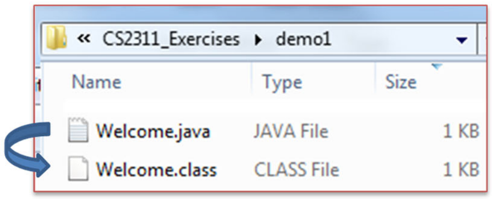
III. How does JAVA work

Compiling and Running Programs

- Source files (.java) are compiled into .class files by the javac compiler.

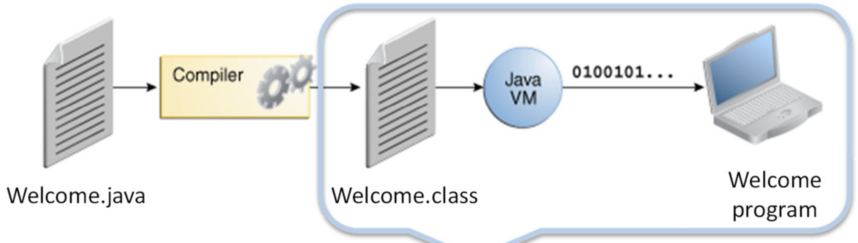


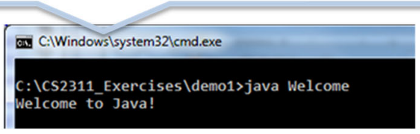
```
C:\CS2311_Exercises\demo1>javac Welcome.java
```



Name	Type	Size
Welcome.java	JAVA File	1 KB
Welcome.class	CLASS File	1 KB

- A .class file does not contain code that is native to the computer;
  It contains **bytecodes** — in machine language of Java Virtual Machine (Java VM).
- The *JRE* runs .class with an instance of the Java Virtual Machine.



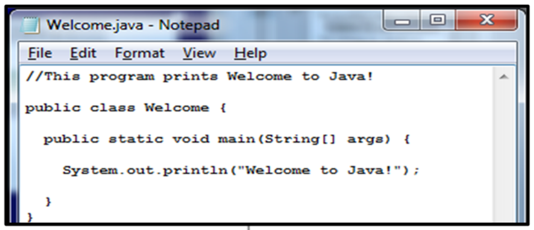


```
C:\CS2311_Exercises\demo1>java Welcome
Welcome to Java!
```

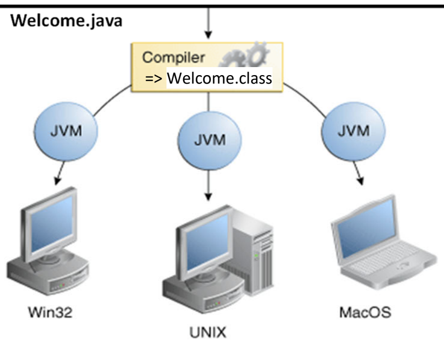
How are JAVA Programs “Architecture neutral”, “Portable” ?

<u>The role of Java VM</u>

- Java VM is available on many different operating systems.
- Once you install JRE or JDK, Java VM is ready in your computer.
- The **same .class file** is capable of running on Microsoft Windows, the Solaris™, Linux, or Mac OS.



```
//This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```



IV. Review - Programming Style, Documentation, Syntax error / Runtime error / Logic error

Programming Style and Documentation

- Appropriate Comments
- Naming Conventions
  - Choose meaningful and descriptive names.
- Proper Indentation and Spacing Lines
  - Tabs, tidy spacing
  - Use blank line to separate segments of the code.
- Block Styles should be applied:

```
public class Day {
  private int year;
  private int month;
  private int day;
  public Day(int y, int m, int d) {
    this.year = y;
    this.month = m;
    this.day = d;
  }
  public String toString() {
    return day + "-" + month + "-" + year;
  }
}
```

Poor! Hard to read!
Please add line breaks before methods

Next-line style

(OK)

```
public class Test
{
  public static void main(String[] args)
  {
    System.out.println("Block Styles");
  }
}
```

End-of-line style

(OK)

```
public class Test {
  public static void main(String[] args) {
    System.out.println("Block Styles");
  }
}
```

Good examples:

```
public class Day {
  private int year;
  private int month;
  private int day;

  public Day(int y, int m, int d) {
    this.year = y;
    this.month = m;
    this.day = d;
  }

  public String toString() {
    return day + "-" + month + "-" + year;
  }
}
```

(OK)

(OK)

Bad examples:

```
public class Day {
  private int year;
  private int month;
  private int day;

  public Day(int y, int m, int d) {
    year = y;
    month = m;
    day = d;
    }

  public String toString(){
    return day + "-" + month + "-" + year; }
}
```

✗ wrong!

✗ wrong!

- Other than the Block Styles:

  For CS2312 teaching, occasionally you will see a complete method in one single line. That is, it violates the Block Styles.

  Reason: It concerns spacing when it is projected on the screen for presentation in class.

  Q: Can I do so in my programs for CS2312?
  A: Well, OK. But may not be welcome outside CS2312.

```
public class Day {
  private int year;
  private int month;
  private int day;
  public Day(int y, int m, int d) {
    year = y; month = m; day = d;
  }
  public String toString()      {
    return day + "-" + month + "-" + year;
  }
  public int getDay()    {return day;}
  public int getMonth()  {return month;}
  public int getYear()   {return year;}
  // …
}
```

Three types of programming errors

- Syntax Errors
  - Detected by the compiler

```
public class ShowSyntaxErrors {
  public static main(String[] args) {
    System.out.println("Welcome to Java);
  }
}
```

- Runtime Errors
  - Causes the program to abort

```
public class ShowRuntimeErrors {
  public static void main(String[] args) {
    System.out.println(1 / 0);
  }
}
```

- Logic Errors
  - Produces incorrect result

```
public class ShowLogicErrors {
  public static void main(String[] args) {
     System.out.print("Five plus six is ");
     System.out.println("5"+"6");
  }
}
```

Debugging

(1) A video on Canvas => CS2312 => https://www.cs.cityu.edu.hk/~helena/cs231220... :
debugger in VS Code (Tracing Lec01 Q12 Fib and Lab01 Q02 Day)

(2) https://code.visualstudio.com/docs/java/java-debugging