

MODULE 1B – JAVA PROGRAMMING

ITSE 1003

Introduction to Programming Languages

Goals

- Install Java 7
- Write first Java program
- Run first Java program
- Play!

Installing Java 7

- We require the Java Development Kit (JDK)
 - Need the tools as well as the Java Runtime Environment (JRE)
- Java 7 JDK should be installed on lab machines
- Java 7 JRE may already be installed on your laptop
- Verification
 - Open a Terminal/Shell/Command Prompt
 - Type **java -version**
 - You should see something like **java version "1.7.0_45"**
 - Type **javac -version**
 - You should see something like **javac 1.7.0_45**

Installing Java 7

- Instructor has the JDKs for Windows and Mac
- For instructions on installing JDK on Windows, see <http://docs.oracle.com/javase/7/docs/webnotes/install/windows/jdk-installation-windows.html>
- For instructions on installing JDK on OS X, see <http://docs.oracle.com/javase/7/docs/webnotes/install/mac/mac-jdk.html>

Typical Java development cycle

- Edit
 - Programmer writes program (and stores program on disk)
- Compile
 - Compiler creates bytecodes from program
- Load
 - Class loader stores bytecodes in memory
- Verify
 - Verifier ensures bytecodes do not violate security requirements
- Execute
 - Interpreter translates bytecodes into machine language
 - JITC (Just in time compilers)

First Java program

```
// first Java program
public class HelloWorld {

    // Java application begins with main method
    public static void main(String[] args) {

        // output the hello message
        System.out.println("Hello World!");

    }

}
```

Go Live

- Edit the program
- Compile the program
- Run (load, verify, execute) the program
- Force some errors

Java template

- The first program is a template for other Java programs
 - There are “universal aspects”
- The basis form will be identical; the details will change

Universal Java program aspects

```
// first Java program  
public class HelloWorld {
```

// Java

The **name** of the 'program' or **application**

- Really the name of a Java **class** (class keyword)
- Must be the same as the file name for applications

public

```
    // output the hello message  
    System.out.println("Hello World!");
```

```
}
```

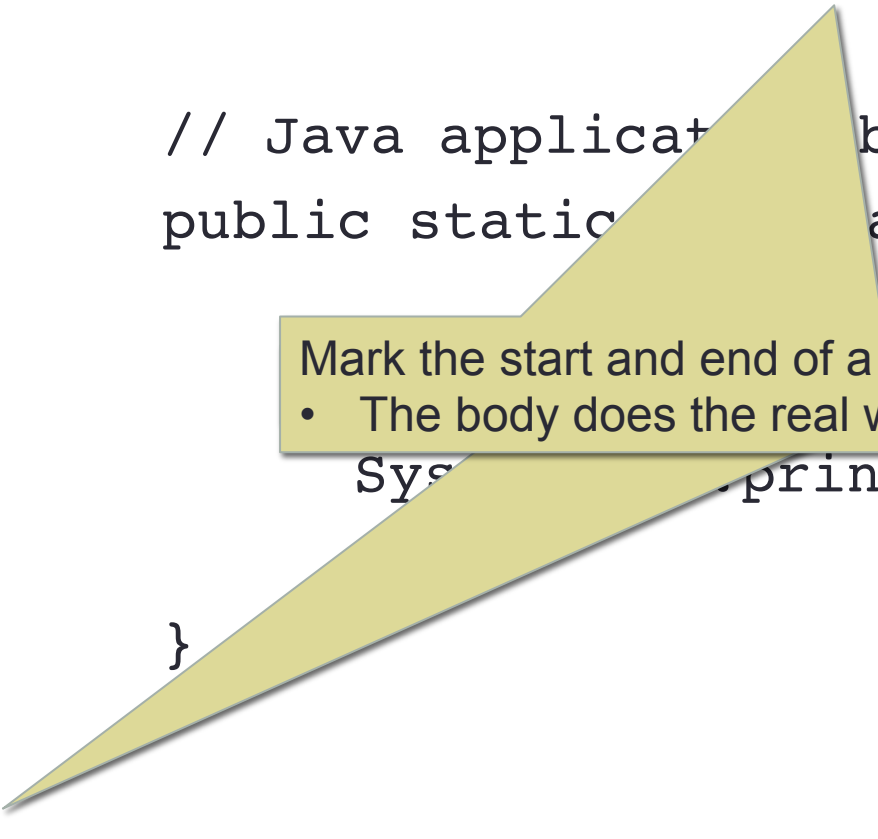
```
}
```

Universal Java program aspects

```
// first Java program
public class HelloWorld {

    // Java application begins with main method
    public static void main(String[] args) {

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



Mark the start and end of a **block** of code (the **body** of the class)

- The body does the real work of the class

Universal Java program aspects

```
// first Java program
```

```
public class
```

Name of a **method** in the class

- Java applications must have a **main** method

```
// Java application begins with main method
```

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

```
    // output the hello message
```

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

```
}
```

Arguments for the method (unused)

```
}
```

Universal Java program aspects

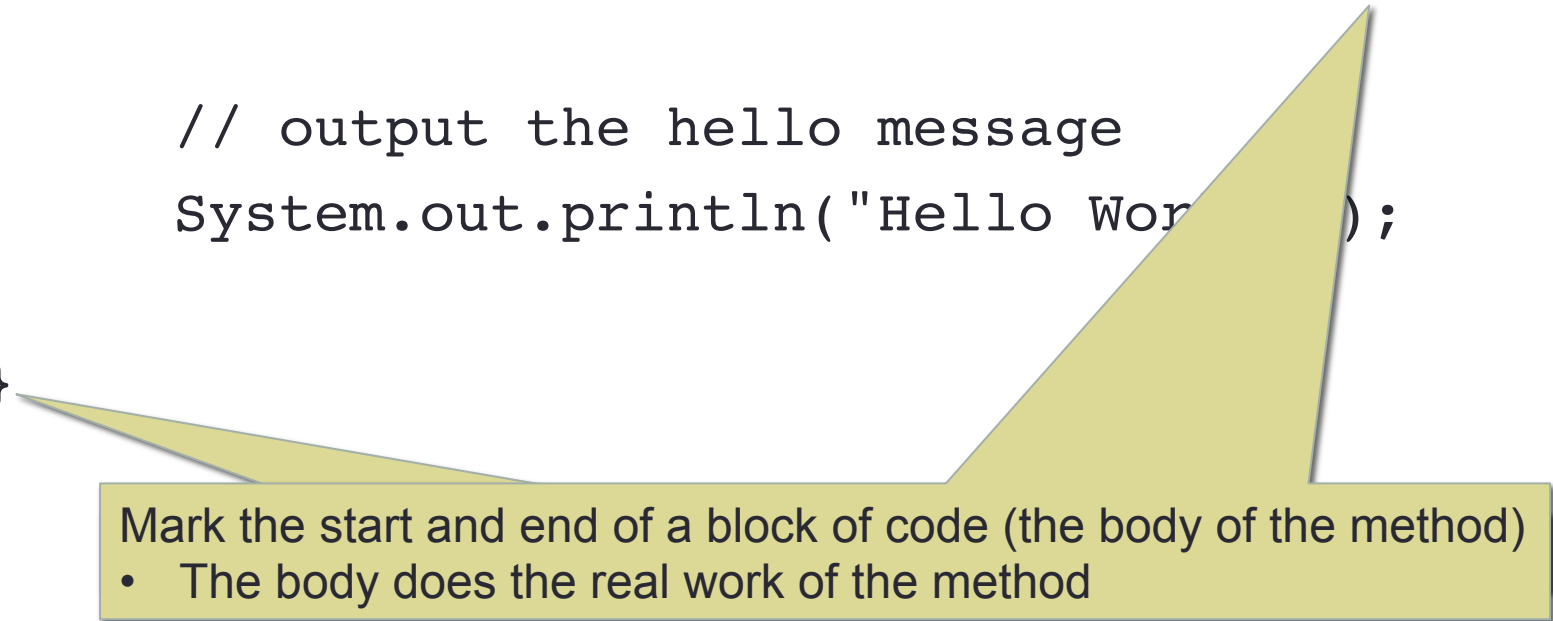
```
// first Java program
public class HelloWorld {

    // Java application begins with main method
    public static void main(String[] args) {

        // output the hello message
        System.out.println("Hello World");

    }

}
```



Mark the start and end of a block of code (the body of the method)

- The body does the real work of the method

Universal Java program aspects

```
// first Java program
public class HelloWorld {
```

```
// J
publ
```

A statement

- Does work in a method
- In this case, the body (block) consists of a single statement

```
// output the hello message
System.out.println("Hello World!");
```

```
}
```

Every statement must **terminate** with a semicolon (;)

```
}
```

Universal Java program aspects

```
// first Java program
```

System.out Is the standard output to “console”

- Outputs to terminal, shell, command window from which program is executed

application begins with main method

```
public static void main
```

println method displays a line of **text** in the ‘console’

```
// output the Hello message
```

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

```
}
```

Argument for System.out.println

- The text to be displayed
- Technically a Java **String**

```
}
```

Universal Java program aspects

```
// first Java program
public class HelloWorld {

    // Java application begins with main method
    public static void main(String[] args) {

        // output the hello message
        System.out.println("Hello World!");
    }
}
```

A comment

- Begins with //; can start anywhere
- Helps describe what a program, class, method or statement is doing
- Extremely important for code maintenance

Activity: Play!

- Write your own first program
- Feel free to experiment with what you print
 - Multiple lines
 - Special characters, foreign language characters
 - Make boxes, diamonds, etc. by printing *, blanks, tabs (\t)
 - ...
- Some potentially helpful elaborations
 - You can use + to make chunks of text a single chunk
 - “Hello “ + “World” equals “Hello World”
 - You can use System.out.print to print with no return
 - Allows you to use multiple statement to create a single line on console