Learning Outcomes

1. Understand the fundamentals of Java programming
2. Classify the Java programming language based on principal characteristics of programming languages.
3. Describe the role of the JRE and JDK in Java programming
4. Explain the role of bytecode in Java programs and their execution
5. Explore the Dr. Java IDE
6. Use the java compiler and the java virtual machine to compile and run Java programs

Why Java?

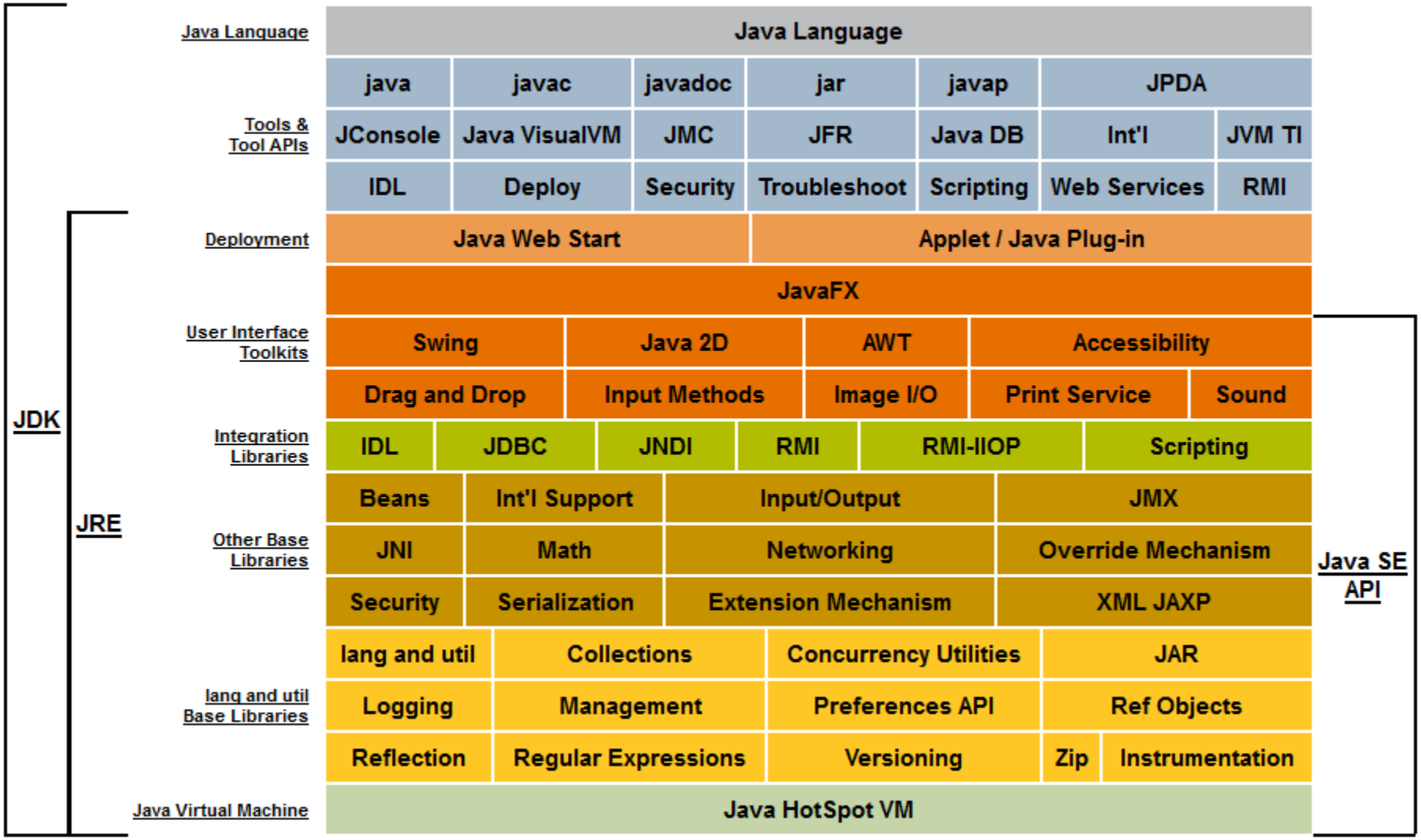
- Why is Java so successful?
  - Java is platform independent, Java programs run on many different operating systems
  - Develop once run everywhere…
  - It's one way to achieve platform independence (not the only way)
- Java is a complete platform, not just a programming language
  - Rich user interface (GUI)
  - Database connectivity
  - Web development
  - Security (?)
  - Middleware platform
- Java is a true object-oriented language

Platform Independence

- How is platform independence achieved?
- Java Virtual Machine
  - Hides the operating system and the underlying machine
  - Programs are compiled to run on the virtual machine which in turn is implemented for each particular platform (Windows, Linux, Android, OSX etc)
- Other ways to achieve platform independence
  - C / C++ write once compile everywhere
  - Platform independent frameworks: Qt

Java programs are
platform independent

Complete Platform



http://www.oracle.com/technetwork/java/javase/tech/index.html

True Object-Oriented Language

- Java is a true object-oriented language
- You can't write a java program without creating a class and/or object(s)
- In other languages like C++, Visual Basic, Python, PHP the object-oriented parts were added later
  - You can write totally non-object-oriented programs

Interpreted Languages

- Interpreted languages are programming languages which are not compiled into machine code
  - They are compiled to an intermediate form (think of it as higher-level assembly language or machine code)
  - An interpreter program is used to interpret (execute) the files in this intermediate form
- Advantages of interpreted languages
  - Platform independence: unlike native code the intermediate form can be platform independent
  - Language independence: programs written in different languages can be compiled into the same intermediate form and thus able to cooperate
  - Dynamic typing: depending on the runtime context code can be interpreted in different ways
  - Runtime checking: security, validation

Java is an interpreted language

Java Concepts

- Java source code is compiled by the Java Compiler (javac.exe) into an intermediate language
  - Java source files have the extension .java
- Java Bytecode is the intermediate language Java is compiled into
  - Java bytecode is saved in *class files* with the extension .class
  - A class file is produced by the Java compiler for each source file compiled
- Java Virtual Machine (JVM) interprets Java instructions one at a time by translating each one into native machine instructions
- Just-In-Time Compiler (JIT) converts java bytecode into native machine code while the program is running → runs much faster
  - JIT is part of JVM, works behind the scenes

JRE vs. JDK

- Java Runtime Environment (JRE) = JVM + class libraries
  - It is needed to run java programs
  - Contains java.exe, the Java virtual machine that is used to run any Java program
- Java Development Kit (JDK) = JRE + tools to build Java programs like the compiler and others
  - Contains javac.exe, the Java Compiler that translates Java source code into bytecode
  - Contains other tools like documentation compiler, archiving, deployment tools, security tools etc
- JDK vs. JRE in detail: http://www.java.com/en/download/faq/techinfo.xml

JVM and Java Compiler on Your Computer

- Locate the Java Virtual Machine executable, java.exe on your machine
  - It is present only if the Java Runtime Environment (JRE) or the Java Development Kit (JDK) are installed
  - C:\Program Files\Java\jre<version>\bin\java.exe
  - C:\Program Files\Java\jdk<version>\bin\java.exe
- Locate the Java Compiler executable, javac.exe on your machine
  - It is present only if the Java Development Kit (JDK) is installed
  - Not part of the JRE
  - C:\Program Files\Java\jdk<version>\bin\javac.exe

The Java Compiler
translates
Java source code
into
bytecode
not machine language

Bytecode is NOT executable by the computer directly.
You need another program to interpret the bytecode:
the JVM (java.exe)

In Java,
the bytecode is the
machine code
of the
virtual machine

Java Source vs. Java Bytecode

Source Code

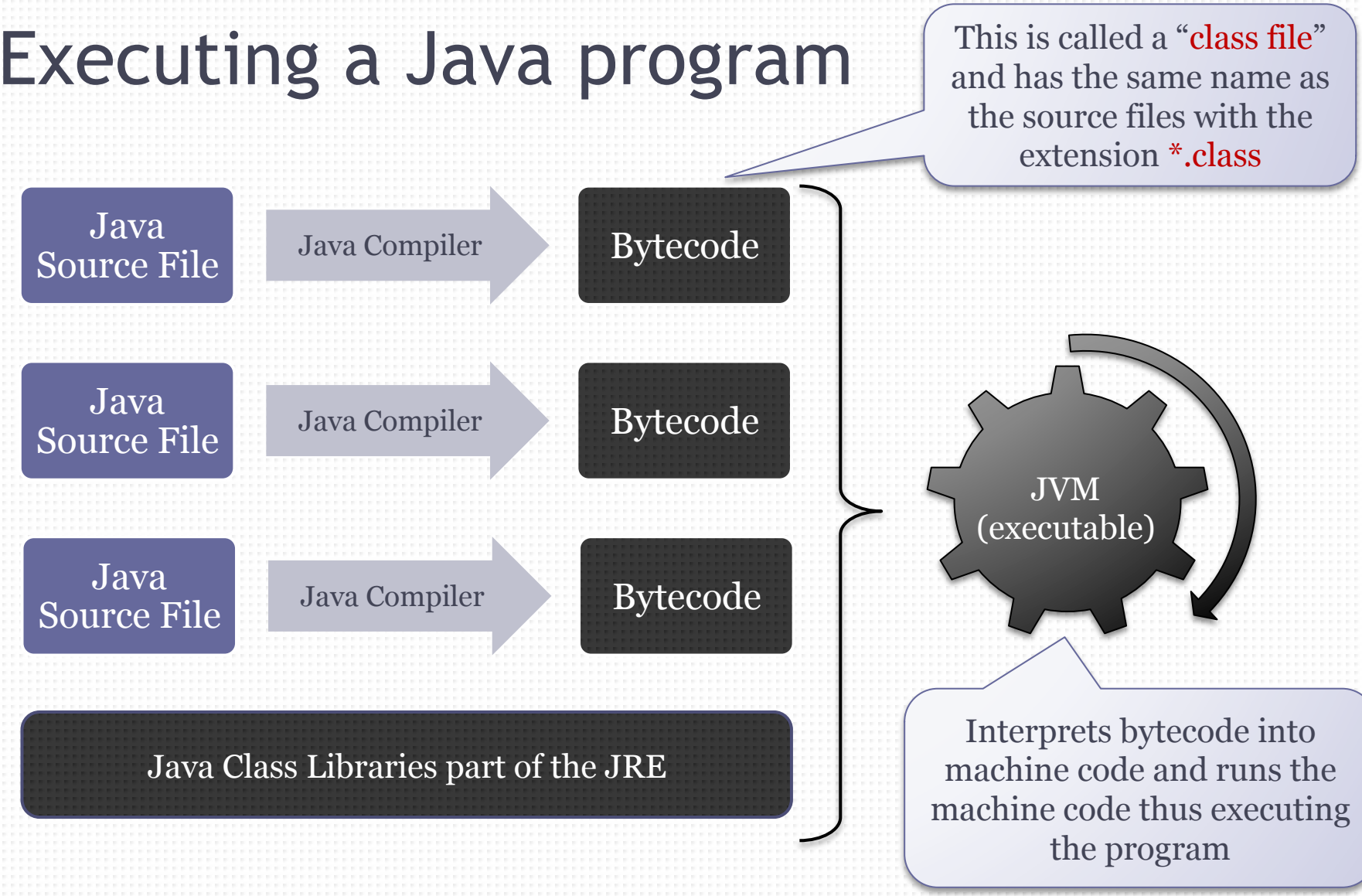
```
public static void main(String[] args)
{
    System.out.println("Hello First Program");
}
```

Bytecode *(before conversion to binary numbers)*

```
0 getstatic #2 <java/lang/System.out>
3 ldc #3 <Hello First Program>
5 invokevirtual #4 <java/io/PrintStream.println>
8 return
```

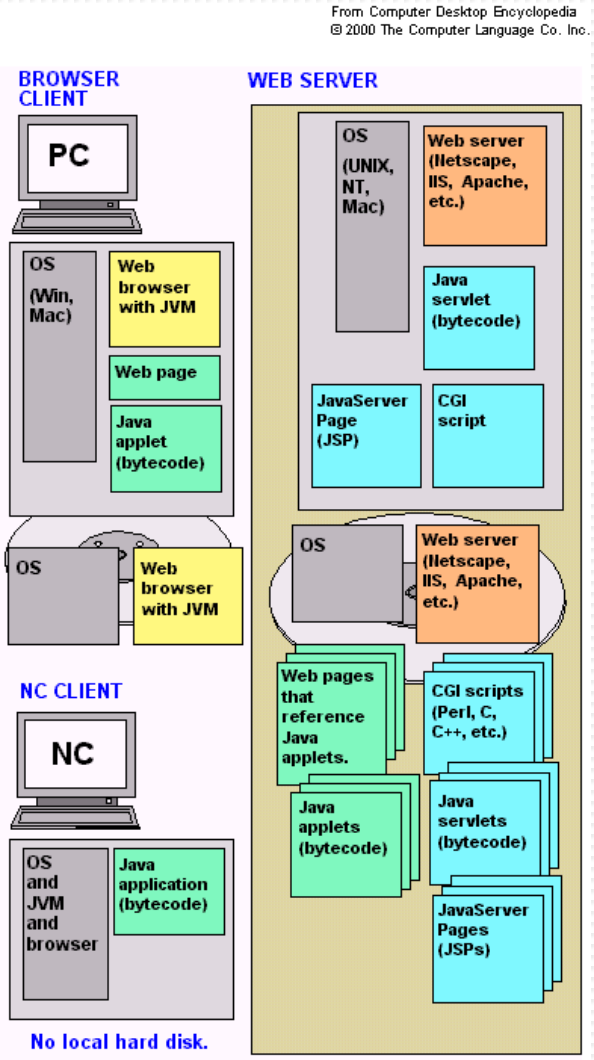
The Just in Time Compiler (JIT)
translates bytecode
into
machine code
(1s and 0s)

Executing a Java program



Java Program Types

- Java application – program that runs standalone on a user's machine
- Java applet – program that runs in a web browser
- Java servlet – program that runs in a web server



Let's Review

- Java's Main Characteristics
  - ...
  - ...
  - ...
- Java bytecode is ...
- Bytecode is executable directly by the CPU. True or False?
- JVM is ...
- JIT is ...
- JRE is ...
- JDK is ...

Programming Language: Statements

- Programming language statements are commands given to the computer to do something
  - Think of them as "sentences" in a natural language like English
  - Only statements can cause the computer to do something just like only full sentences have meaning
  - Statements have a statement terminator, a symbol to mark the end of the statement just like "." marks the end of natural language sentence
- A program is a (very long) sequence of statements
  - Millions of statements for large programs, e.g. Windows 7 OS
  - To make sense and be able to control and maintain large amounts of anything one must organize them
- Program statements are organized into statement blocks

Statement Blocks

- Are used to organize statements, like countries, provinces, cities, and postal codes are used to organize addresses
- Statement blocks have a start marker that marks the start of the block (e.g. “begin” or “{“)
- Statement blocks have an end marker that marks the end of the block (e.g. “end” or “}”)
- The statements that are part of the block must be placed between the start and end markers
- A block of statements can contain other blocks
- A block of statements can be empty

Indenting

- Statements in a block are indented to show they are part of the block
  - This makes your program easier to read and understand
  - You'll make fewer mistakes if you indent properly!
- The compiler does not care if you indent, but other programmers (and your boss) do
- In this course you should indent each block by 4 spaces
  - If there's another statement block inside, indent by 4 more spaces… etc.
- I will deduct marks for incorrect/inconsistent indenting

Example: Statements and Blocks

```
_____ <end>
_____ <end>
_____ <end>
_____ <end>
```

```
<block-begin>
<block-end>
```

```
_____ <end>
```

```
<block-begin>
        _____ <end>
        _____ <end>
        _____ <end>
        _____ <end>
<block-end>
```

```
<block-begin>
        _____ <end>
        _____ <end>
        <block-begin>
                _____ <end>
                _____ <end>
        <block-end>
        _____ <end>
<block-end>
```

Must remember to indent by 4 spaces

Preview of our first program: “Hello World”

```
public class HelloWorld {

    public static void main(String[] args) {
        // A Java program starts with the first
        // statement in main
        System.out.println("Hello PROG10082 class");
    }

}
```

Java Programming Absolute Basics

- The statement terminator in Java is the semicolon **;**
- A Java statement block is placed inside curly brackets { … }
  - "{" starts a statement block (start marker)
  - "}" ends a statement block (end marker)
- Java source files are text files saved with the extension *.java
- The name and location of the source files depends on what is in the source file
  - The name must match the name of the class in the file
  - The location (directory path) must match the full package name
  - Wrong name or location will generate compile or runtime errors
- Java programming language is case sensitive: name, Name, NAME and nAmE are four different things
  - File names are not case sensitive on Windows, but I really encourage you to keep them consistent with the Java code

Comments

- Important to make your program understandable, ignored by compiler
- Single line comments
  - Start with //
  - End at the end of the line. It doesn't matter what is in the comment, the rest of the text until the end of the line is considered a comment
  - Examples:

```
// This is a comment on one line

//Some developers prefer these type of comments
//even when they span multiple lines
```

- Block comments can span multiple lines
  - Start with /* and end with */
  - Anything in between is considered a comment. Anything goes except the end of comment marker, */
  - Example:

```
/* This is a multi-line
   comment. */
```

Exercise 1: Writing Java Programs using the command line

- Create the Hello World program shown on the next slide
- Create a source file using Notepad named Hello1.java
  - You can copy and paste from the next slide into Notepad
  - Save it on your desktop
  - Make sure the file name is exactly as written above (starts with an upper-case letter)
- Start the Windows command prompt
- Type `cd Desktop`
- To compile the program, type `javac Hello1.java`
  - Notice that the compiler creates the file Hello1.class
  - What does this file contain?
- To run the program, type `java Hello1`

Example: “Hello World”

```
public class Hello1 {
    public static void main(String[] args) {
        // A Java program starts with the first
        // statement in main
        System.out.println("Hello PROG10082 class");
    }
}
```

Integrated Development Environments

- An Integrated Development Environment (IDE) is a graphical computer program that helps us do many different programming tasks
- It's called "integrated" because it integrates multiple types of tools necessary when programming
  - More like a toolbox than a tool
- What tools are integrated in an IDE?
  - A text editor for creating source files and writing source code
  - A compiler (may be part of the IDE or may be installed separately)
  - An easy-to-use interface for running the compiler and running the program
  - A debugger used for interactive execution of the code to find bugs
  - A help system that provides access to information that helps us program
  - Advanced coding tools like automatic code-completion, refactoring etc.
  - Advanced code generation tools for building graphical user interfaces, database code
  - Tools to create diagrams that help us think about or document a program

Exercise 2: Writing Java Programs using an IDE

- Install the Dr. Java IDE from http://drjava.org/download.shtml
  - Click on "Download Windows App via HTTP"
- Using Dr. Java, create the Hello World program shown in the next slide (you can copy and paste from the slide into Dr. Java)
  - Create a source file using Dr. Java named Hello2.java
  - Save it in a folder named sheridan
- Click the "Compile" button (near the top)
  - Check that a .class file was created for this program
- Click the "Run" button and look in the bottom window for the
- What's different about this program compared to the previous one?
  - It has a new line at the top, "package sheridan;"
  - This means it *must* be saved in a folder called sheridan

Example: “Hello World” (v2)

```
package sheridan;

public class Hello2 {
    public static void main(String[] args) {
        // A Java program starts with the first
        // statement in main
        System.out.println("Hello PROG10082 class");
    }
}
```

Exercise 3: Writing Java Programs using an IDE

- If you just did Exercise 2 press the Close button in Dr. Java to close the Hello2.java file
- Using Dr. Java, create the Hello World program shown in the next slide
  - Create a source file named Hello3.java
  - Save it in a directory (folder) named sheridan (same as Ex. 2)
- Click the "Compile" button
- Click the "Run" button
- What is different about this program compared to the previous one?
  - It uses a String variable (we'll learn all about strings later)

Example: “Hello World” (v3)

```
package sheridan;

public class Hello3 {
    public static void main(String[] args) {
        // A Java program starts with the first
        // statement in main
        String message = "Hello From Program 3";
        System.out.println(message);
    }
}
```

Recommended Homework Exercises

- Using Dr. Java, type in or copy/paste the example program from the slides "2. Programming Languages", then compile and run it
  - How is this program different from the exercises in the preceding slides?
- From your textbook, Chapter 1, implement the following exercises using Dr. Java
  - 1.1 (Displaying three messages)
  - 1.3 (Displaying a pattern)
- For more fun create a program that displays ASCII art. For examples of ASCII art see this link
  - http://www.chris.com/ascii/
- If you do recommended exercises please show me in class or send me an email to let me know how it went

References

- Introduction to Java Programming
  - Chapter 1: Introduction to Computers, Programs and Java
- JRE vs. JDK
  - http://www.java.com/en/download/faq/techinfo.xml
- How to check the type of Operating System / Processor architecture on your Windows machine:
  - http://support.microsoft.com/kb/827218