Sheridan

PROG 10082

Object Oriented Programming 1

Notes from Chapter 1.3, 1.5, 1.6

Sheridan

Programs

Computer *programs*, known as *software*, are instructions to the computer.

You tell a computer what to do through programs. Without programs, a computer is an empty machine. Computers do not understand human languages, so you need to use computer languages to communicate with them.

Programs are written using programming languages.

Sheridan

What is Programming?

- What does a programmer do?
  - Write programs!
- What is a program?
  - Software, applications, etc.
  - Sets of instructions that tell the computer to do something.
- What is a programming language?
  - A specific instruction set you can use to code programs.

Sheridan

Programming Languages

- Machine Language
  - The native language understood by a particular machine
  - Platform specific
  - Binary!
- Assembly Language
  - More symbolic
  - Easier for programmers to code with
  - Not understood by the machine!

Sheridan

Programming Languages

- Assembler
  - Programmer would write assembly code
  - Computer needs to understand it to execute it
  - Assembler program translated the assembly code into machine language

Sheridan

Programming Languages

- High-Level Languages
  - Assembly language is machine-specific
  - Eventually high-level languages were developed
    - These were easier to learn
    - They had more English-like words
- Hundreds of languages exist
  - Each was designed for a particular purpose

Sheridan

Programming Languages

- Name some!

Sheridan

Writing Programs

- A programmer picks a programming language
- The programmer writes **code** in that language
  - This is called **source code**

```
1 public class  MyFirstProgram
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("Hello World!");
6     }
7 }
8
```

Source Code

Sheridan

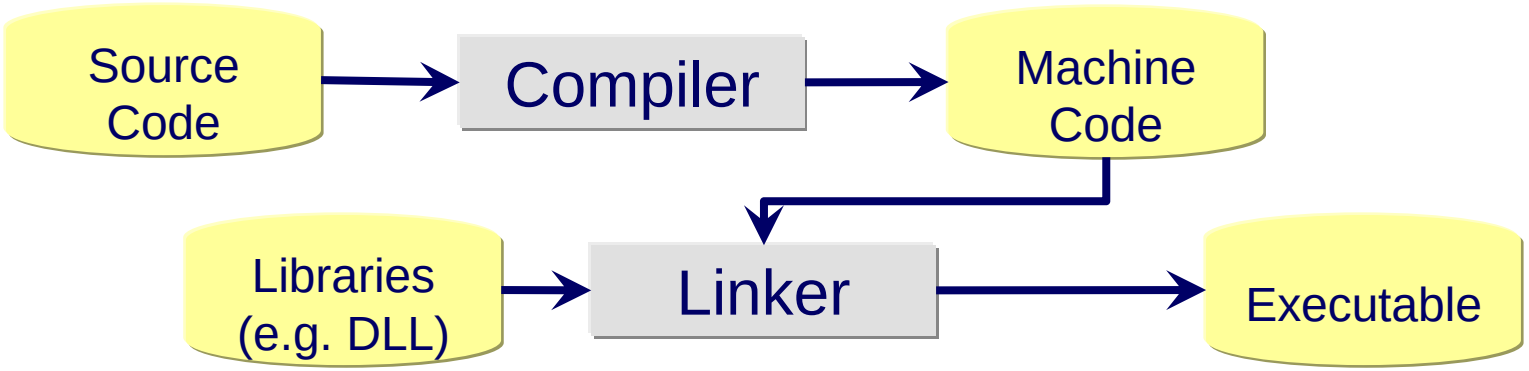
Writing Programs

- The computer can't understand the source code
  - It was written in a high-level language
  - The computer only understands machine language
- A **compiler** program translates the source code into machine code

Sheridan

Writing Programs

- With larger programs, existing libraries are added
  - E.g. .DLL files that contain standard Windows dialog boxes
- A **linker** combines library code with machine code to create .EXE



Writing Programs

- Source code is plain text
- A compiler works on a specific machine
- The compiled code can only run on that specific machine
  - E.g. you can write the source code on a Mac, but if you compile it on a Windows machine, it will only run on that machine.

Summary

- A computer understands Machine Language.
- High-level languages like Java are easier to learn and code with.
- A programmer writes source code.
- Source code is compiled into machine language.
- The compiled machine language can only run on the machine on which it was compiled.

Sheridan

Java

- Java was developed by **James Gosling** at Sun Microsystems
  - Bought by Oracle Jan 2010
  - http://www.oracle.com/technetwork/java/index.html
- He was trying to program some small remote control devices
  - *7 (“Star Seven”)
- Existing languages were too slow and took up too much space
- He developed Java as an alternative

Sheridan

Java

- Java's features made it suitable for the *7 project:
  - Portable or Platform Independent
    - compiled code can run on any platform
  - Secure and Reliable
  - Small and Compact
- The internet was starting to become popular
  - These features made Java suitable for Internet-based and networked apps

Sheridan

Java, Web, and Beyond

- Java can be used to develop standalone applications.
- Java can be used to develop applications running from a browser.
- Java can also be used to develop applications for hand-held devices.
- Java can be used to develop applications for Web servers.

Sheridan

Writing Java Programs

- Recall the process of coding and compiling a program:

How does this work in the Java language?

Sheridan

Writing Java Programs

- Write the program source code in an editor and save it to a file
  - Source file has a .JAVA extension
- Source code must be compiled with a compiler
  - **Syntax**:
    - The rules and grammar of the language
  - If there are syntax errors:
    - Compile is not successful!

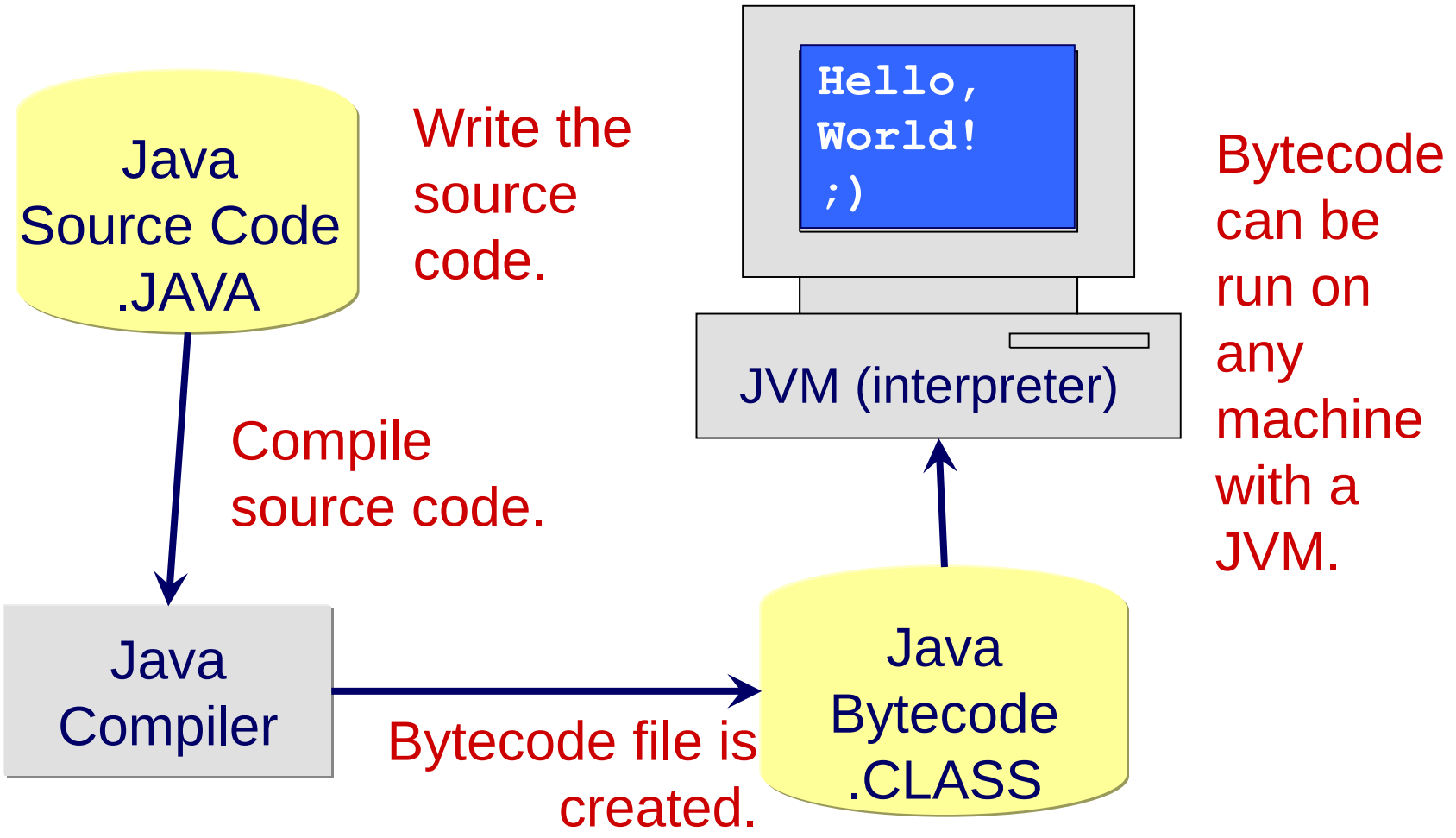
Sheridan

Writing Java Programs

- If the compile is successful
  - The result is a **bytecode** file
    - Bytecode file has a .CLASS extension
    - Not quite machine language
- Bytecode can be run on any platform
- JVM – Java Virtual Machine
  - Interprets bytecode line by line
  - Executes each line as it's interpreted

Sheridan

Writing Java Programs



Sheridan

Java API

- API = Application Program Interface
  - Contains pre-defined classes and interfaces
  - These are building blocks used to write programs in Java
  - The API is always growing and being improved
    - You should always have the latest version
    - Download it from Java web site

Sheridan

Java API

- There are three** sets of Java APIs:
- Java SE – Standard Edition
  - Used for regular desktop or internet applications
- Java EE – Enterprise Edition
  - Used for larger applications using servlets and JSP (Java Server Pages)
- Java ME – Micro Edition
  - Used for machines that have limited resources such as cell phones and PDAs
  - can be used to develop applications for mobile devices such as cell phones.

Java API

- **There is also a Java FX
  - Used for developing interactive web apps
  - Java's answer to Flash, Silverlight, etc.
  - Now bundled with Java SE as of version 7
- OpenJDK
  - http://openjdk.java.net/
  - Open source implementation of Java SE

Java SE

- Standard Edition consists of:
  - SDK/JDK – Software/Java Development Kit
    - Some versions say “SDK”, some use “JDK”
    - They mean the same thing!
  - JRE – Java Runtime Environment
    - A machine must have this in order to run Java programs.

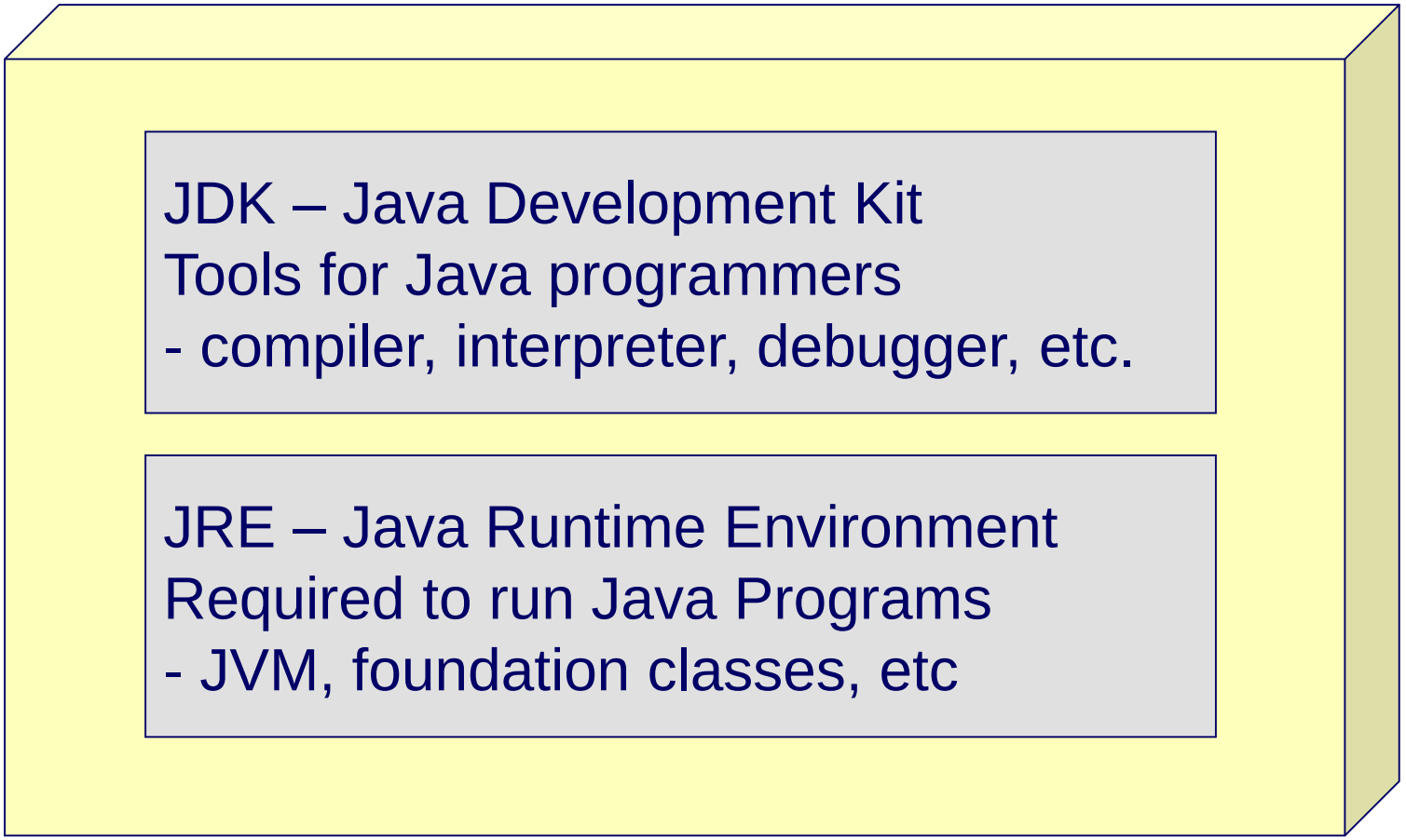
Java SE

- JDK – Java Development Kit
  - The current version of Java calls it the JDK
  - The tools and utilities used by programmers to make Java programs
  - Compiler, interpreter, debugger, and other programs you'll use in other Java courses

Java SE

- JRE – Java Runtime Environment
  - Any machine running Java programs must have the JRE
  - Contains the JVM
  - Contains core foundation classes and other supporting files that are the building blocks of Java programs

Java SE


JDK – Java Development Kit
Tools for Java programmers
- compiler, interpreter, debugger, etc.
JRE – Java Runtime Environment
Required to run Java Programs
- JVM, foundation classes, etc

Java SE Versions

- 1995 – Java 1 introduced
  - JDK 1.0 and the 1.1 series
- 1998 – Java 2 Standard Edition
  - Used to be referred to as J2SE
  - JDK 1.2
  - This is the "second generation" of Java that we are still using today
- 2000 – J2SE 1.3
- 2002 – J2SE 1.4

Sheridan

Java SE Versions

- 2004 – J2SE 5.0
  - This is actually J2SE 1.5
  - They decided to drop the "1." and just call it version 5.0
- Dec. 2006 – Java SE 6.0
  - J2SE 1.6 or J2SE 6.0
  - They also decided to change the "J2" to just "Java"

Sheridan

Java SE Versions

- July 2011 – Java SE 7.0
  - J2SE 1.7 or J2SE 7.0
  - Added some neat new language features

Sheridan

Java in this Course

- We'll be installing and using Java SE 7 (Java Standard Edition 7.0)
- The JDK includes the compiler and interpreter
  - And some other tools you'll use in other Java courses
- We will use Dr.Java as our editor/IDE
  - We'll get this from the web site

Sheridan

Popular Java IDEs

- NetBeans
- Eclipse

Summary

- Java is platform-independent.
- When you write Java source code, it is stored in a .JAVA file.
- You compile the .JAVA file with the Java compiler.
- If there are any syntax errors, you must fix them and recompile.
- When your compile is successful, a bytecode file is created.

Summary

- A bytecode file has a .CLASS extension.
- This file can then interpreted and executed on any machine that has a JVM.
- A JVM is part of the JRE, which comes with the Java SE.
- The Java SE also contains the JDK, which you use to compile and interpret Java programs.

Sheridan

Summary

- We will write our source code in Java using the EditPlus editor
- We will compile and run our programs using the existing tools in the JDK

Sheridan

A Simple Java Program

Listing 1.1

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

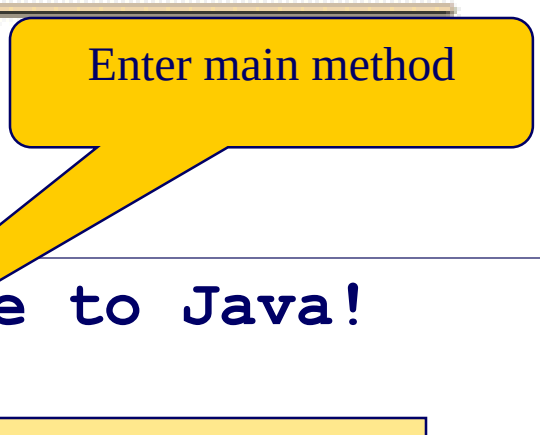
Welcome

Run

Note: Clicking the green button displays the source code with interactive animation. You can also run the code in a browser. Internet connection is needed for this button.

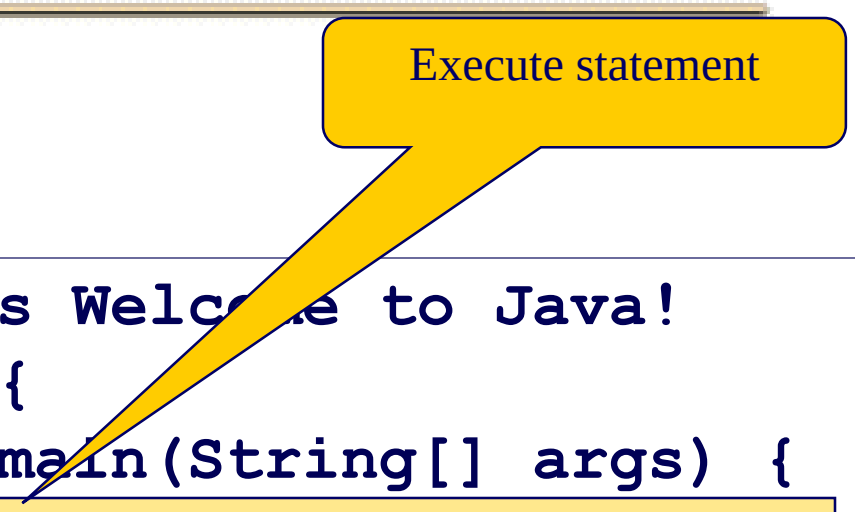
Note: Clicking the blue button runs the code from Windows. If you cannot run the buttons, see www.cs.armstrong.edu/liang/javaslidenote.doc.

Trace a Program Execution




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

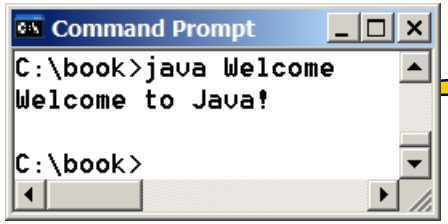
Trace a Program Execution




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

Trace a Program Execution

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



print a message to the console

Sheridan

Anatomy of a Java Program

- Class name
- Main method
- Statements
- Statement terminator
- Reserved words
- Comments
- Blocks

Sheridan

Class Name

Every Java program must have at least one class. Each class has a name. By convention, class names start with an uppercase letter. In this example, the class name is Welcome.

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

Sheridan

Simple Java Program

public class Welcome {

- This is a class Header – it means that the class is public and is called **Welcome**
  - The word **public** is a keyword – it means that the class/program can be used by other classes/programs

Sheridan

Main Method

Line 2 defines the main method. In order to run a class, the class must contain a method named main. The program is executed from the main method.

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

Sheridan

Statement

A statement represents an action or a sequence of actions. The statement System.out.println("Welcome to Java!") in the program in Listing 1.1 is a statement to display the greeting "Welcome to Java!".

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

Sheridan

Statement Terminator

Every statement in Java ends with a semicolon (;).

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

Sheridan

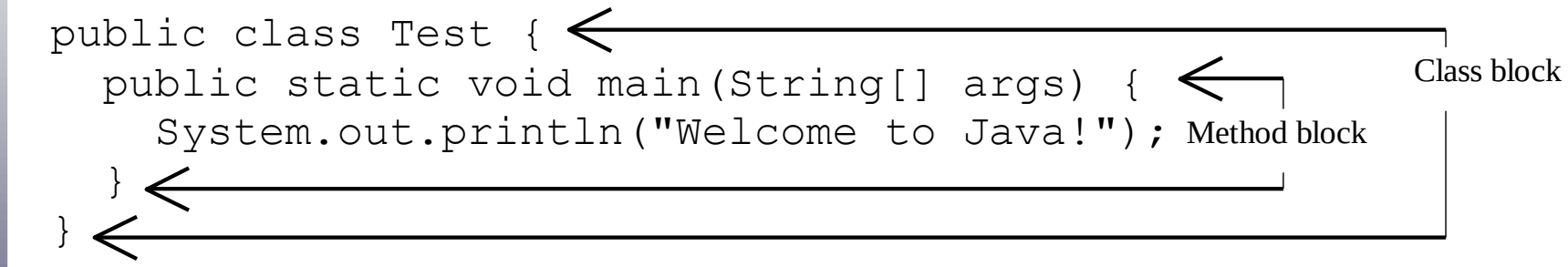
Reserved words

Reserved words or keywords are words that have a specific meaning to the compiler and cannot be used for other purposes in the program. For example, when the compiler sees the word class, it understands that the word after class is the name for the class.

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

Blocks

A pair of braces in a program forms a block that groups components of a program.



Sheridan

Special Symbols

Character	Name	Description
{}	Opening and closing braces	Denotes a block to enclose statements.
()	Opening and closing parentheses	Used with methods.
[]	Opening and closing brackets	Denotes an array.
//	Double slashes	Precedes a comment line.
" "	Opening and closing quotation marks	Enclosing a string (i.e., sequence of characters).
;	Semicolon	Marks the end of a statement.

Sheridan

{ ... }

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

Sheridan

(...)

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

Sheridan

;

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

Sheridan

// ...

// display a message to the user

This is a **comment** - these are notes or remarks that the programmer leaves for themselves or other programmers.

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

Sheridan

" "
...

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

This is an executing statement. All executing statements end in a line-terminator (a semi-colon, in Java).
This statement displays the **string literal Welcome to Java!** on the monitor.

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

Sheridan

Simple Java Program

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

- System.out refers to the default output device, which on most systems is the monitor.
- System is the name of a class that is part of the JRE (it's one of the "building blocks" of Java.
- The System.out object is executing or invoking a method called println().
- println() (pronounced "print line") is a method that sends its **argument** to the System.out object.
- In other words, the println() method sends the string literal "Welcome to Java!" to the default output device (monitor).

Sheridan

Programming Style and Documentation

- Appropriate Comments
- Naming Conventions
- Proper Indentation and Spacing Lines
- Block Styles

Sheridan

Appropriate Comments

Include a summary at the beginning of the program to explain what the program does, its key features, its supporting data structures, and any unique techniques it uses.

Include your name, class section, instructor, date, and a brief description at the beginning of the program.

Sheridan

Naming Conventions

- Choose meaningful and descriptive names.
- Class names:
  - Capitalize the first letter of each word in the name. For example, the class name `ComputeExpression`.

Sheridan

Proper Indentation and Spacing

- Indentation
  - Indent two spaces.
- Spacing
  - Use blank line to separate segments of the code.

Sheridan

Block Styles

Use end-of-line style for braces.

Next-line style

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

End-of-line style

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

Sheridan

Programming Errors

- Syntax Errors
  - Detected by the compiler
- Runtime Errors
  - Causes the program to abort
- Logic Errors
  - Produces incorrect result

Sheridan

Syntax Errors

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

ShowSyntaxErrors Run

Sheridan

Runtime Errors

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

ShowRuntimeErrors Run

Logic Errors

```
public class ShowLogicErrors {
  public static void main(String[] args) {
    System.out.println("Celsius 35 is Fahrenheit degree ");
    System.out.println((9 / 5) * 35 + 32);
  }
}
```

ShowLogicErrors Run

Sheridan

Displaying Text in a Message Dialog Box

you can use the showMessageDialog method in the JOptionPane class.

JOptionPane is one of the many predefined classes in the Java system, which can be reused rather than "reinventing the wheel."

WelcomeInMessageDialogBox Run

The showMessageDialog Method

```
JOptionPane.showMessageDialog(null,
  "Welcome to Java!",
  "Display Message",
  JOptionPane.INFORMATION_MESSAGE);
```

Sheridan

Two Ways to Invoke the Method

There are several ways to use the showMessageDialog method. For the time being, all you need to know are two ways to invoke it.

One is to use a statement as shown in the example:

```
JOptionPane.showMessageDialog(null, x,
  y, JOptionPane.INFORMATION_MESSAGE);
```

where x is a string for the text to be displayed, and y is a string for the title of the message dialog box.

The other is to use a statement like this:

```
JOptionPane.showMessageDialog(null, x);
```

where x is a string for the text to be displayed.

Sheridan

Implicit Import and Explicit Import

```
java.util.* ; // Implicit import

java.util.JOptionPane; // Explicit Import
```

No performance difference

For Next Class

- Finish reading Chapter 1
  - Fill in your Notes for today's lesson
- Do your SLATE profile and upload photo
- Do the Week 1 Intro
- Look over the online notes for the next session