School of Science & Engineering
Department of CSE
Canadian University of Bangladesh

Lecture-2: A Simple Java Program

Prerequisite: CSE 1101
Semester: Summer 2024

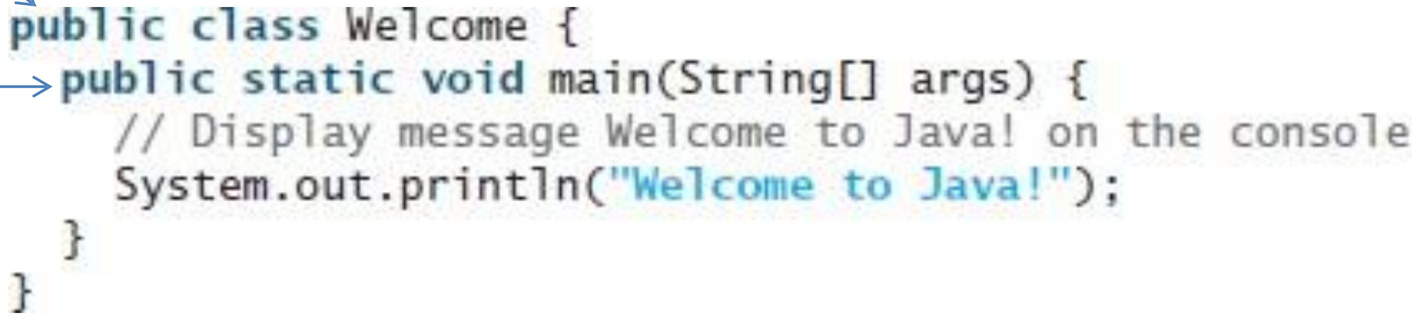
A Simple Java Program

- A Java program is executed from the *main method in the class*.
- We will begin with a simple Java program that displays the message **Welcome to Java!** on the *console*.
- The word *console* is an old computer term that refers to the text entry and display device of a computer.
  - Console input means to receive input from the keyboard.
  - Console output means to display output on the monitor

A Simple Java Program

Class definition

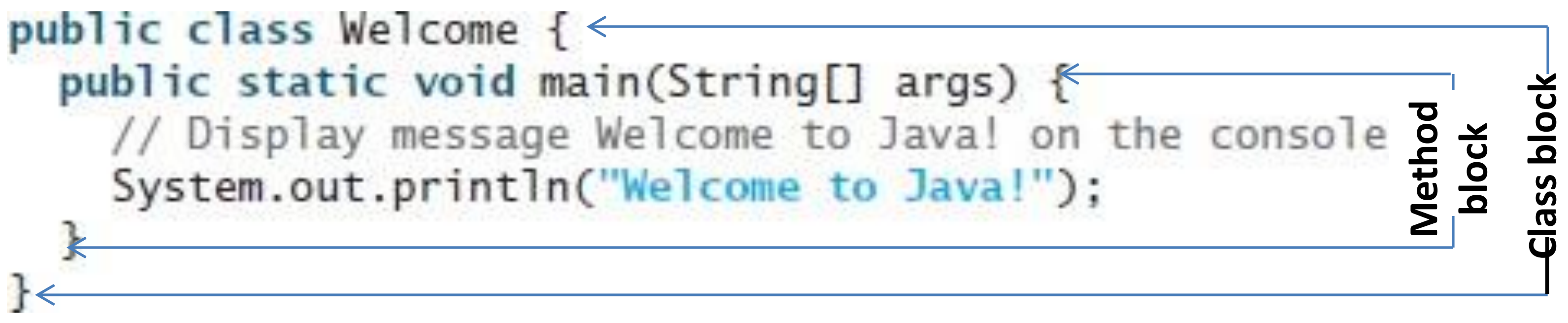
Main method definition



```
public class Welcome {
  public static void main(String[] args) {
    // Display message Welcome to Java! on the console
    System.out.println("Welcome to Java!");
  }
}
```

- Java source programs are <u>case sensitive</u>.
- Every Java program must have *at least* one *class*.
- Each class has a name. By convention, class names start with an uppercase letter.
- The *main* method is the entry point where the program begins execution.
- A class may contain several methods. A method is a construct that contains statements.

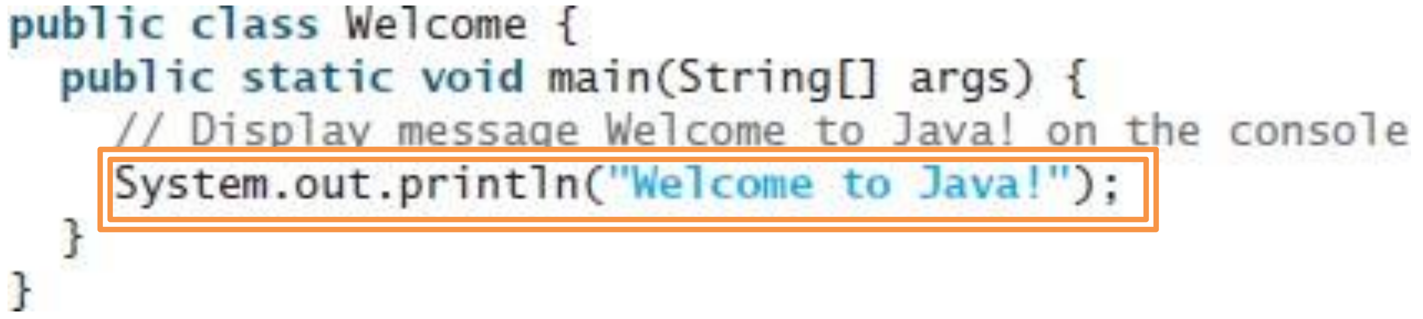
A Simple Java Program (Cont.)



```
public class Welcome {
  public static void main(String[] args) {
    // Display message Welcome to Java! on the console
    System.out.println("Welcome to Java!");
  }
}
```

- A pair of curly braces in a program forms a *block* that groups the program's component.
- Every class has a *class block* that groups the data and methods of the class.
- Every method has a *method block* that groups the statements in the method.

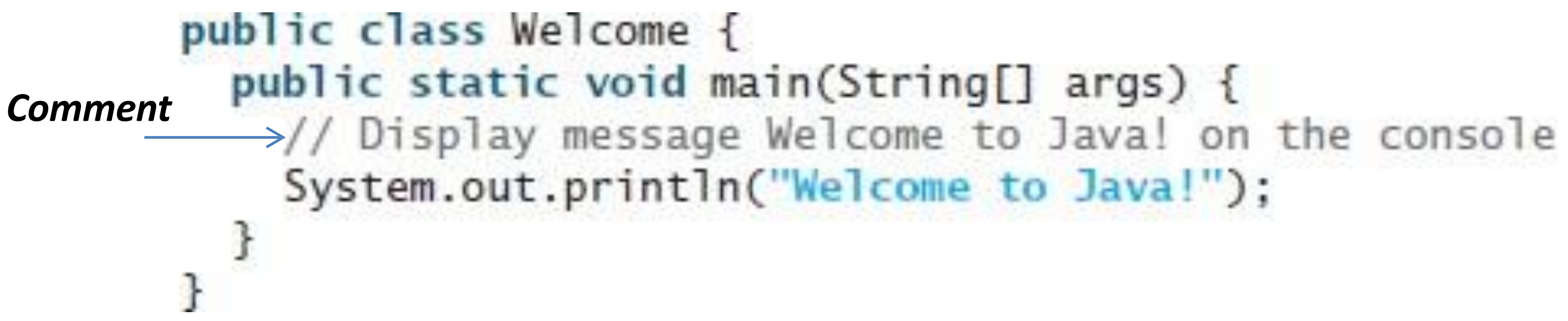
A Simple Java Program (Cont.)



```
public class Welcome {
  public static void main(String[] args) {
    // Display message Welcome to Java! on the console
    System.out.println("Welcome to Java!");
  }
}
```

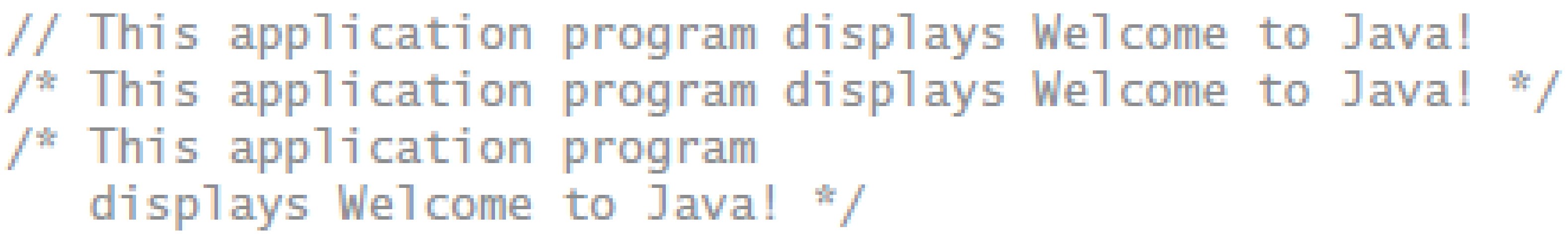
- The *System.out.println* statement displays the string *Welcome to Java* on the console.
- A *String* is a sequence of characters. Strings should be enclosed in double quotation marks.
- Every statement in Java ends with a semicolon (*;*).
- *public*, *class*, *static*, and *void* are reserved words : have a specific meaning to the compiler and cannot be used for other purposes. In the program.

A Simple Java Program (Cont.)



```
public class Welcome {
  public static void main(String[] args) {
    // Display message Welcome to Java! on the console
    System.out.println("Welcome to Java!");
  }
}
```

- Comments are ignored by the compiler.
- Two types of comments:
  - Line comments: preceded by two slashes (//).
  - Block (or paragraph) comments: enclosed between (/*) and (*/)



```
// This application program displays Welcome to Java!
/* This application program displays Welcome to Java! */
/* This application program
   displays Welcome to Java! */
```

Java Special Characters

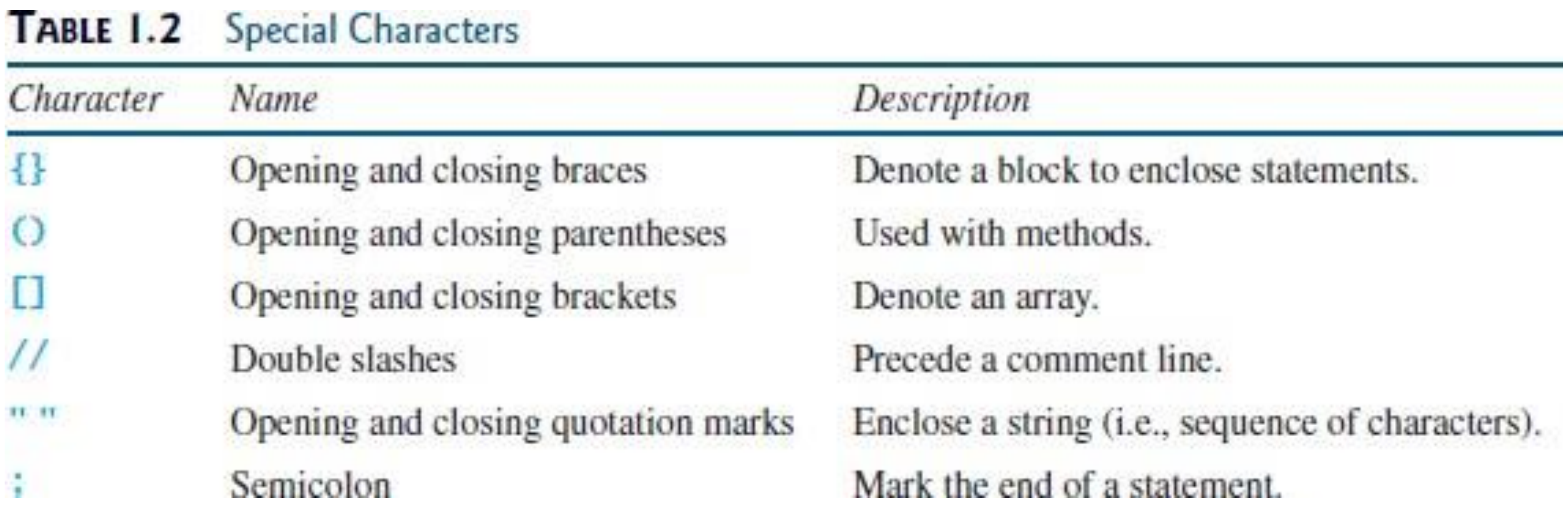


TABLE 1.2 Special Characters

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

Displaying More Messages to the Console

```
public class WelcomeWithThreeMessages {
  public static void main(String[] args) {
    System.out.println("Programming is fun!");
    System.out.println("Fundamentals First");
    System.out.println("Problem Driven");
  }
}
```

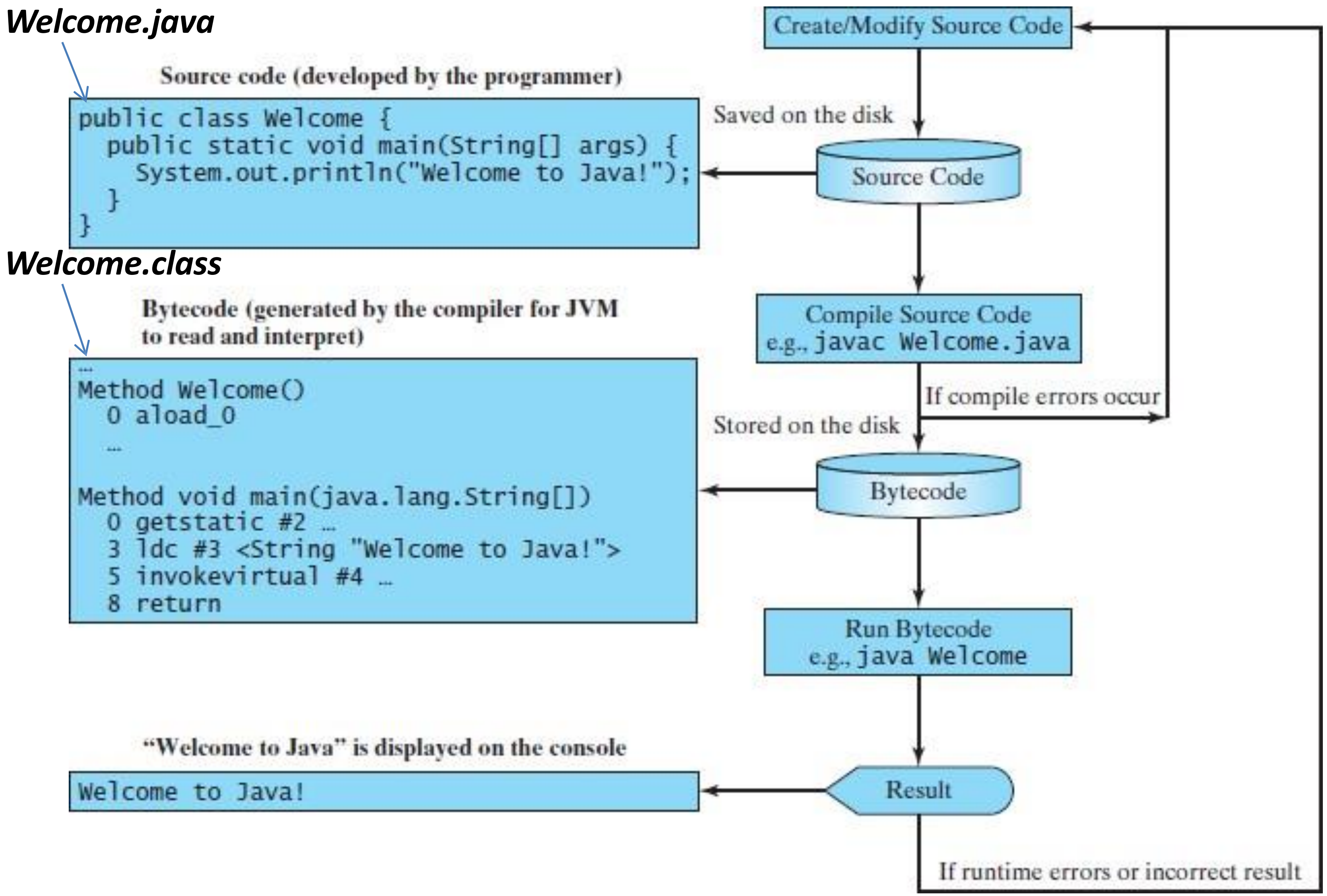
```
Programming is fun!
Fundamentals First
Problem Driven
```

Displaying the Result of a Mathematical Computation

```
public class ComputeExpression {
  public static void main(String[] args) {
    System.out.println((10.5 + 2 * 3) / (45 - 3.5));
  }
}
```

```
0.39759036144578314
```

Creating, Compiling, and Executing a Java Program



Welcome.java
Source code (developed by the programmer)
Welcome.class
Bytecode (generated by the compiler for JVM to read and interpret)
"Welcome to Java" is displayed on the console
Create/Modify Source Code
Saved on the disk
Source Code
Compile Source Code e.g., javac Welcome.java
If compile errors occur
Stored on the disk
Bytecode
Run Bytecode e.g., java Welcome
Result
If runtime errors or incorrect result

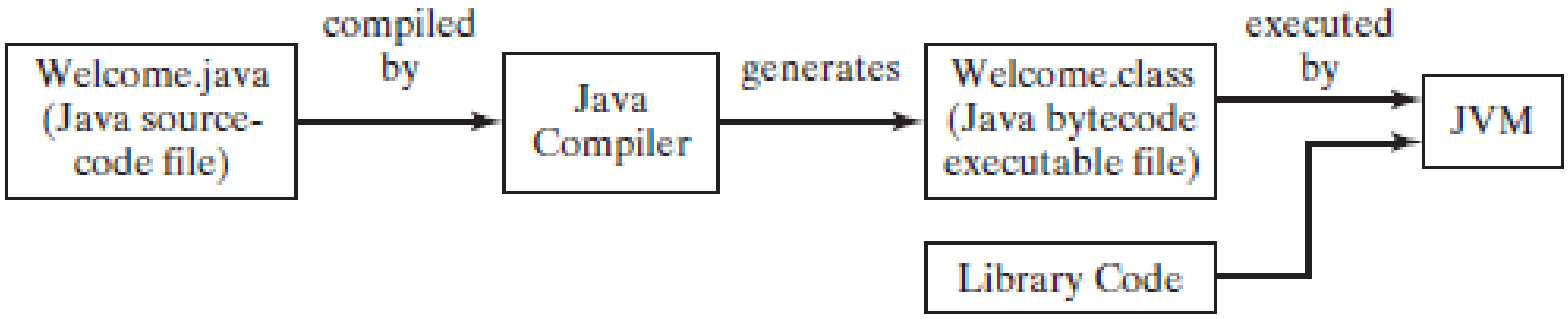
```
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

```
...
Method Welcome()
  0 aload_0
  ...

Method void main(java.lang.String[])
  0 getstatic #2 …
  3 ldc #3 <String "Welcome to Java!">
  5 invokevirtual #4 …
  8 return
```

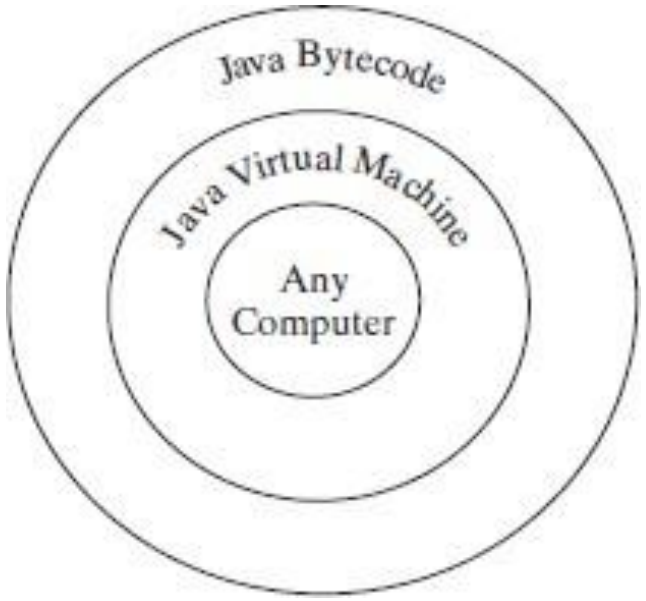
```
Welcome to Java!
```

Creating, Compiling, and Executing a Java Program (Cont.)


Welcome.java (Java source-code file)
compiled by
Java Compiler
generates
Welcome.class (Java bytecode executable file)
executed by
JVM
Library Code

- Java source code is compiled into Java *bytecode*.
- Your Java code may use the code in the Java library.
- The JVM is an *interpreter*, which translates individual instructions in the *bytecode* into the target machine language code and executes it immediately.

Creating, Compiling, and Executing a Java Program (Cont.)



- The *bytecode* is similar to machine instructions, but is *architecture neutral* and can run on any platform that has a *Java Virtual Machine* (*JVM*).

Creating, Compiling, and Executing a Java Program (Cont.)

- When executing a Java program, the JVM first loads the bytecode of the class to memory using a program called the *class loader*.
  - If your program uses other classes, the class loader dynamically loads them just before they are needed.
- After a class is loaded, the JVM uses a program called the *bytecode verifier* to check the validity of the bytecode and to ensure that the bytecode does not violate Java's security restrictions.
  - Java enforces strict security to make sure that Java class files are not tampered with and do not harm your computer.

Programming Errors

- Syntax errors.
  - Detected by the compiler.
  - Result from errors in code construction.
- Runtime errors.
  - Cause a program to terminate abnormally.
  - Occur while the program is running if the environment detects an operation that is impossible to carry out.
  - Examples include input errors and division by zero.
- Logic errors.
  - Occur when a program does not perform the way it is intended to.

Java code with a syntax error:

```
public class SyntaxErrorExample {
    public static void main(String[] args) {
        int x = 5
        System.out.println("The value of x is: " + x);
    }
}
```

The corrected version of the code:

```
public class SyntaxErrorExample {
    public static void main(String[] args) {
        int x = 5; // Corrected: added semicolon at the end
        System.out.println("The value of x is: " + x);
    }
}
```

An example of Java code that compiles without syntax errors but encounters a runtime error (also known as an exception):

```
public class RuntimeErrorExample {
    public static void main(String[] args) {
        int [] numbers = {1, 2, 3};
        System.out.println("Accessing element at index 3: " + 
numbers[3]);
    }
}
```

```
System.out.println("Accessing element at index 3: " + numbers[3]);
```

```
Exception in thread
"main“java.lang.ArrayIndexOutOfBoundsException: Index 3 out of
bounds for length 3 at
RuntimeErrorExample.main(RuntimeErrorExample.java:5)
```

An example of Java code that compiles without syntax errors and runs without throwing exceptions, but has a logic error:

```
public class LogicErrorExample {
    public static void main(String[] args) {
        int num1 = 10;
        int num2 = 5;

        int result = multiply(num1, num2); // Function call with incorrect method name

        System.out.println("Multiplication result: " + result);
    }

    // Incorrect method name: should be multiply instead of add
    public static int add(int a, int b) {
        return a * b; // Incorrect operation: should be a + b
    }
}
```