Unit II

Java Programming Fundamentals

Tutor: Pema Galey

Learning Outcomes

In this session, you will learn about:

- Java programming language concept
- A simple Java program: syntax
- Java JVM, JRE and JDK
- Java Variables and Literals
- Keywords
- Java Data types

Java Hello World Program

Java "Hello, World!" Program

```
// Your First Program
class HelloWorld {
      public static void main(String[] args) {
            System.out.println("Hello World!");
      }
}
```

Output:

```
Hello World!
```

Note: If you have compiled the exact code, you need to save the file as HelloWorld.java. It's because the name of the class and file name should match in Java.

How Java "Hello, World!" Program Works?

1. `// Your First Program`
   - ✓ In Java, any line starting with// is a comment.

2. `class HelloWorld {...}`
   - ✓ In Java, every application begins with a class definition.
   - ✓ Java application has a class definition, and the name of the class should match the filename in Java.

How Java "Hello, World!" Program Works?

3. `public static void main(String[] args)    {...}`
   1. This is the main method. Every application in Java must contain the main method. The Java compiler starts executing the code from the main method.
   2. Keywords: ***public, static, void***

4. `System.out.println("Hello World!");`
   1. It is a print statement. It prints the text **`Hello World!`** to standard output (your screen). The text inside the quotation marks is called String in Java.
   2. Notice the print statement is inside the main function, which is inside the class definition.

Basic Java Concepts

- Every valid Java Application must have a class definition (that matches the filename).
- The main method must be inside the class definition.
- The compiler executes the codes starting from the main method/function.

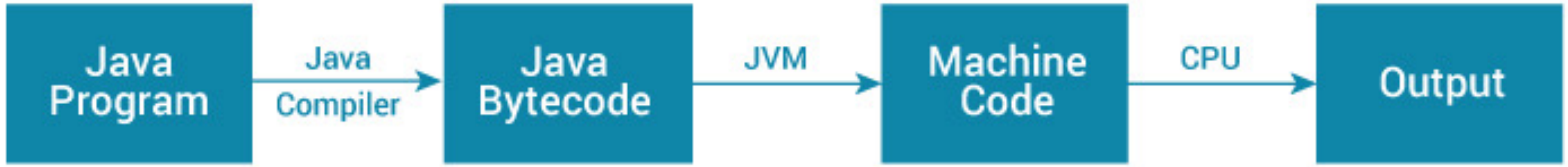
Java Program Compilation

1. Example Java file:
   - ✓ ***HelloWord.java***
2. Java Compilation
   - ✓ ***javac HelloWorld.java***
3. Java Interpretation (Run)
   - ✓ ***java HelloWorld***

Java JDK, JRE and JVM

What is JVM?

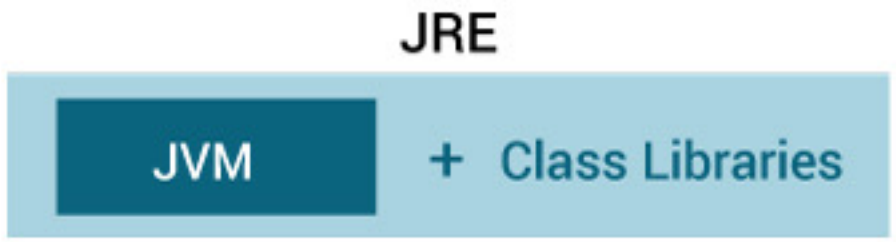
- JVM (Java Virtual Machine) is an abstract machine that enables your computer to run a Java program.
- When you run the Java program, Java compiler first compiles your Java code to bytecode. Then, the JVM translates bytecode into native machine code.



Java JDK, JRE and JVM

What is JRE?

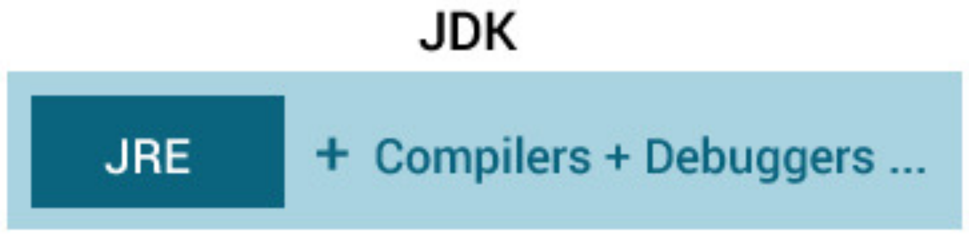
- JRE (Java Runtime Environment) is a software package that provides Java class libraries, Java Virtual Machine (JVM), and other components that are required to run Java applications.
- JRE is the superset of JVM.



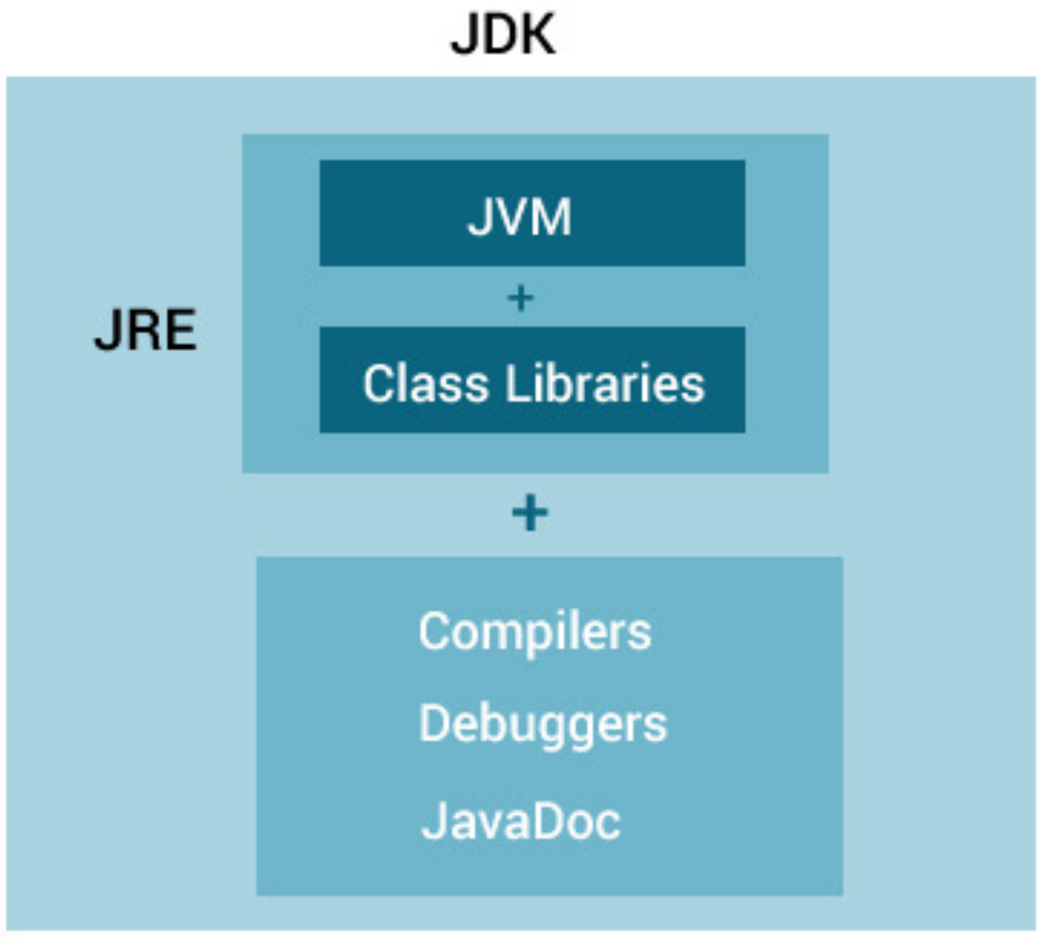
Java JDK, JRE and JVM

What is JDK?

- JDK (Java Development Kit) is a software development kit required to develop applications in Java. When you download JDK, JRE is also downloaded with it.
- In addition to JRE, JDK also contains a number of development tools (compilers, JavaDoc, Java Debugger, etc).



Relationship between JVM, JRE, and JDK



Java Variables

- A variable is a location in memory (storage area) to hold data.
- To indicate the storage area, each variable should be given a unique name (identifier).

- **Create Variables in Java:**
  - Here's how we create a variable in Java,

    int age = 20;
  - Here, ***age*** is a variable of int data type and we have assigned value 20 to it.
  - The int data type suggests that the variable can only hold integers.

Java Variables

- **Create Variables in Java:**
  - You can declare variables and assign variables separately.

```
int age;
age = 20;
```

- **Change values of variables**
  - The value of a variable can be changed in the program, hence the name variable.

```
int age = 20;
age = 25;
```

Rules for Naming Variables in Java

Java programming language has its own set of rules and conventions for naming variables:

1. Java is case sensitive. Hence age and AGE are two different variables.
2. Variables must start with either a **letter** or an **underscore, _** or a **dollar, $** sign.
3. Variable names cannot start with numbers.
4. Variable names can't use whitespace.
5. When creating variables, choose a name that makes sense.
6. If you choose one-word variable names, use all lowercase letters.

Rules for Naming Variables in Java

- There are 4 types of variables in Java programming language:
    1. Instance Variables (Non-Static Fields)
    2. Class Variables (Static Fields)
    3. Local Variables
    4. Parameters

Java Literals

- Literals are data used for representing fixed values. They can be used directly in the code.
- For example,

```
int a = 1;
float b = 2.5;
char c = 'F';
```

- Here, 1, 2.5, and 'F' are literals.

Different Types of Literals

1. Boolean Literals
   - In Java, boolean literals are used to initialize boolean data types. They can store two values: true and false. For example,

```
boolean flag1 = false;
boolean flag2 = true;
```

2. Integer Literals
   - An integer literal is a numeric value(associated with numbers) without any fractional or exponential part. There are 4 types of integer literals in Java:
     - *i. binary (base 2)*
     - *ii. decimal (base 10)*
     - *iii. octal (base 8)*
     - *iv. hexadecimal (base 16)*

Different Types of Literals

3. Floating-point Literals

- A floating-point literal is a numeric literal that has either a fractional form or an exponential form.

4. Character Literals

- Character literals are unicode character enclosed inside single quotes. For example,

```
char letter = ‘a’;
```

5. String literals

- A string literal is a sequence of characters enclosed inside double-quotes. For example,

```
String str1 = "Java Programming";
```

Keywords in Java

abstract	boolean	break	byte
case	catch	char	class
const	continue	default	do
double	else	extends	final
finally	float	for	goto
if	implements	import	instanceof
int	interface	long	native
new	package	private	protected
public	return	short	static
strictfp	super	switch	synchronized
this	throw	throws	transient
try	void	volatile	while
enum	assert		

Java Data Types

- Data types are divided into two groups:
  1. Primitive data types (Eight types)
     - includes *byte, short, int, long, float, double, boolean* and *char*

  1. Non-primitive data types
     - such as String, Arrays and Classes

Java Data Types (Primitive)

	Data Type	Size	Description
1	byte	1 byte	Stores whole numbers from -128 to 127
2	short	2 bytes	Stores whole numbers from -32,768 to 32,767
3	int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
4	long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
5	float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
6	double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
7	boolean	1 bit	Stores true or false values
8	char	2 bytes	Stores a single character/letter or ASCII values

Java Data Types (Primitive)

1. boolean type

- The boolean data type has two possible values, either true or false.
- Default value: false.
- They are usually used for **true/false** conditions.

- Example:

```
class Main {
    public static void main(String[] args) {
        boolean flag = true;
        System.out.println(flag); // prints true
    }
}
```

Java Data Types (Primitive)

2. byte type

- The byte data type can have values from -128 to 127 (8-bit signed two's complement integer).
- If it's certain that the value of a variable will be within -128 to 127, then it is used instead of int to save memory.
- Default value: 0

- Example:

```
class Main {
    public static void main(String[] args) {
        byte range;
        range = 124;
        System.out.println(range); // prints 124
    }
}
```

Java Data Types (Primitive)

3. short type

- The short data type in Java can have values from **-32768** to **32767** (16-bit signed two's complement integer).
- If it's certain that the value of a variable will be within -32768 and 32767, then it is used instead of other integer data types (int, long).
- Default value: 0
- Example:

```
class Main {
    public static void main(String[] args) {
        short temp;
        temp = -200;
        System.out.println(temp); // prints 124
    }
}
```

Java Data Types (Primitive)

4. int type

- The int data type can have values from **-2^{31}** to **$2^{31}-1$** (32-bit signed two's complement integer).
- If you are using Java 8 or later, you can use an unsigned 32-bit integer. This will have a minimum value of 0 and a maximum value of $2^{32}-1$. Default value: 0

5. long type

- The long data type can have values from **-2^{63}** to **$2^{63}-1$** (64-bit signed two's complement integer).
- If you are using Java 8 or later, you can use an unsigned 64-bit integer with a minimum value of **0** and a maximum value of **$2^{64}-1$**. Default value: 0

Java Data Types (Primitive)

6. double type

- The double data type is a double-precision 64-bit floating-point.
- It should never be used for precise values such as currency. Default value: 0.0 (0.0d)

7. float type

- The float data type is a single-precision 32-bit floating-point.
- It should never be used for precise values such as currency. Default value: 0.0 (0.0f)
- Example: `float number = -42.3f;`

Java Data Types (Primitive)

8. char type

- It's a 16-bit Unicode character.
- The minimum value of the char data type is '\u0000' (0) and the maximum value of the is '\uffff'.
- Default value: '\u0000’
- Example:

```
class Main {
    public static void main(String[] args) {
            char letter = '\u0051’; // Unicode value
            System.out.println(letter); // prints Q
            char letter1 = '9’; // prints Q
            System.out.println(letter1); // prints 9
} }
```

Java Data Types (Non-Primitive)

String type

- Java also provides support for character strings via java.lang.String class.
- Strings in Java are not primitive types. Instead, they are objects.
- For example,

```
String myString = "Java Programming";
```

  - Here, myString is an object of the String class.

String Concatenation

- It is the process of adding two strings together using the (+) plus operator.

- Example 1:

```
System.out.print(“Hello." + " How are you?”);
```

- Example 2:

```
int studentID = 2020001;
String name = ”Tashi";
System.out.println("This student ID " +   studentID + "
      belongs to Mr." + name);
```

Thank you!