Welcome to Technical Training for Placement Preparation Masterclass by

TalentBattle

20

Hrs.

overview

What to expect.

Exactly how does this go down?

JAVA
[20 Hrs.]

☆ In-depth training.

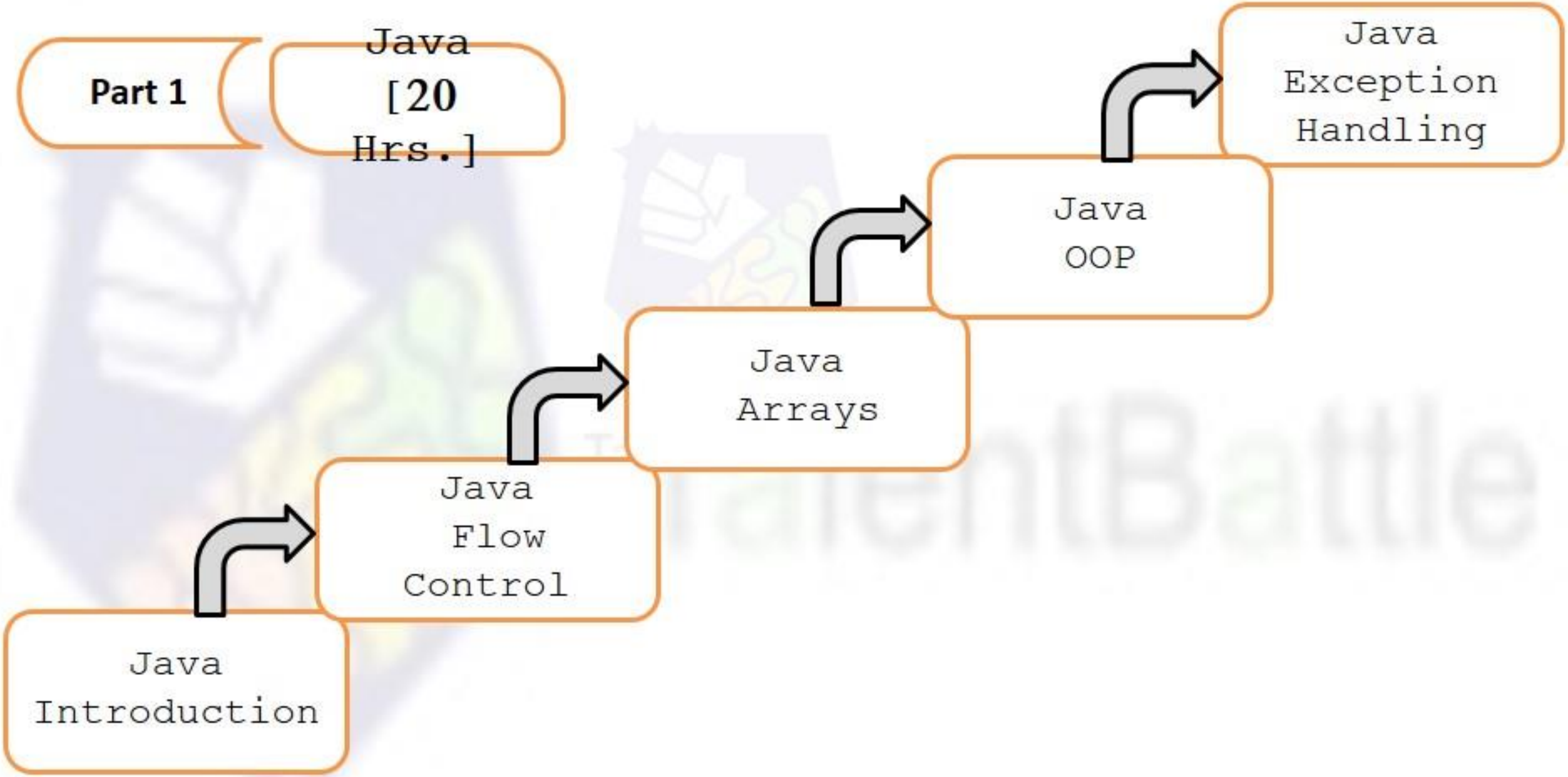
☆ Competitive programming.

☆ Hands-on sessions.

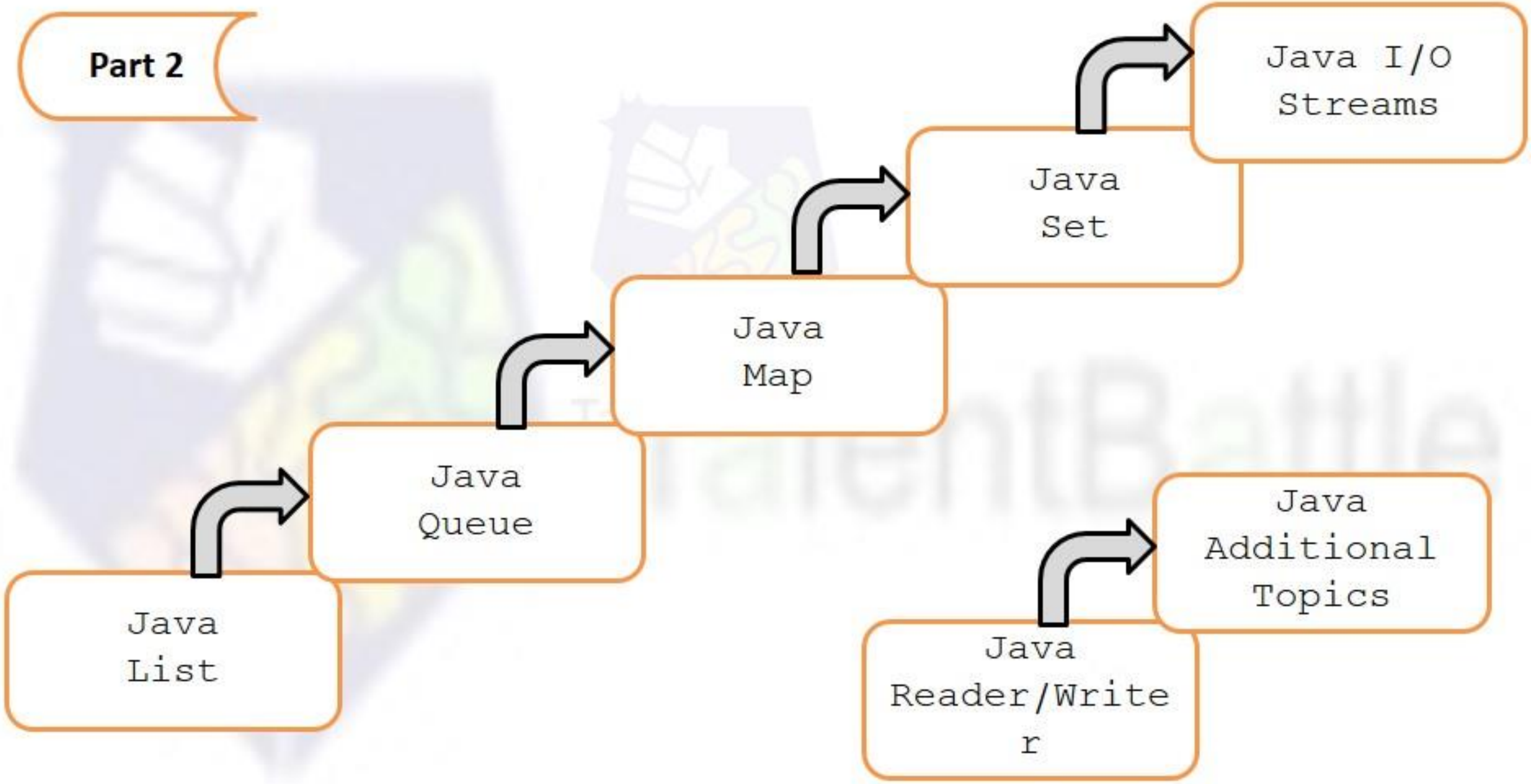
Skills you will gain…

- Programming Language
- Concepts
- Problem Solving

Part 1 Java [20 Hrs.]



Part 2


Java List
Java Queue
Java Map
Java Set
Java I/O Streams
Java Reader/Write r
Java Additional Topics

About Java Programming

•**Platform independent** - We can write Java code in one platform (operating system) and run on another platform without any modification.

•**Object-oriented** - Java is an object-oriented language. This helps to make our Java code more flexible and reusable.

•**Speed** - Well optimized Java code is nearly as fast as lower-level languages like C++ and much faster than Python, PHP, etc.

Java Hello World Program

```
// Your First Program

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

How Java "Hello, World!" Program Works?

// Your First Program

In Java, any line starting with // is a comment. Comments are intended for users reading the code to understand the intent and functionality of the program. It is completely ignored by the Java compiler (an application that translates Java program to Java bytecode that computer can execute).

class HelloWorld { ... }

In Java, every application begins with a class definition. In the program, HelloWorld is the name of the class.

For now, just remember that every Java application has a class definition, and the name of the class should match the filename in Java.

public static void main(String[] args) { ... }

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.

For now, just remember that the main function is the entry point of your Java application, and it's mandatory in a Java program.

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

The code above 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.

•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 function.

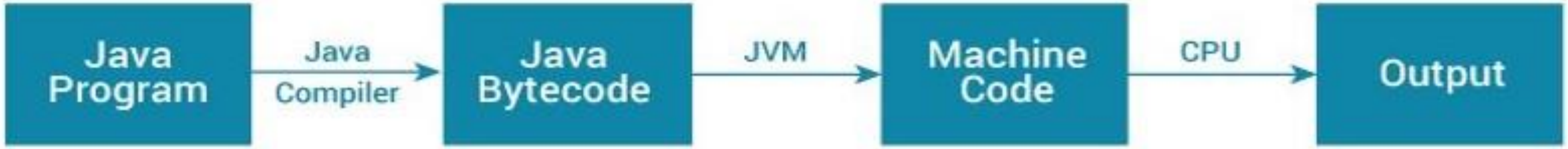
Java JDK, JRE and JVM

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 (set of instructions that a computer's CPU executes directly).

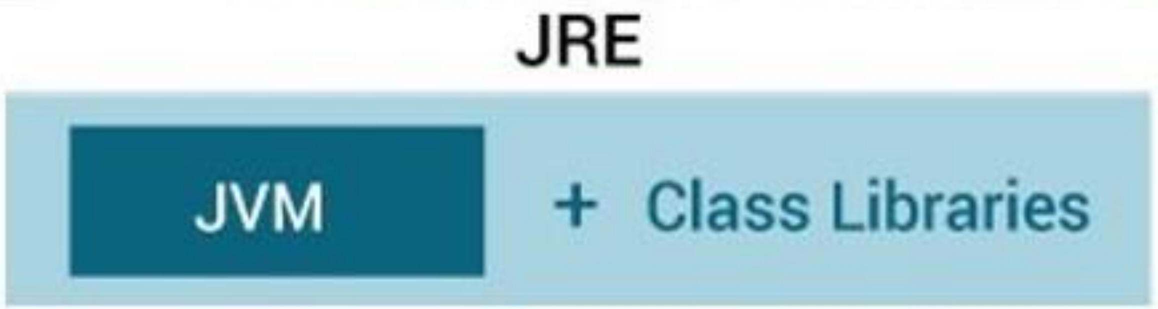
Java is a platform-independent language. It's because when you write Java code, it's ultimately written for JVM but not your physical machine (computer). Since JVM executes the Java bytecode which is platform-independent, Java is platform-independent.



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.

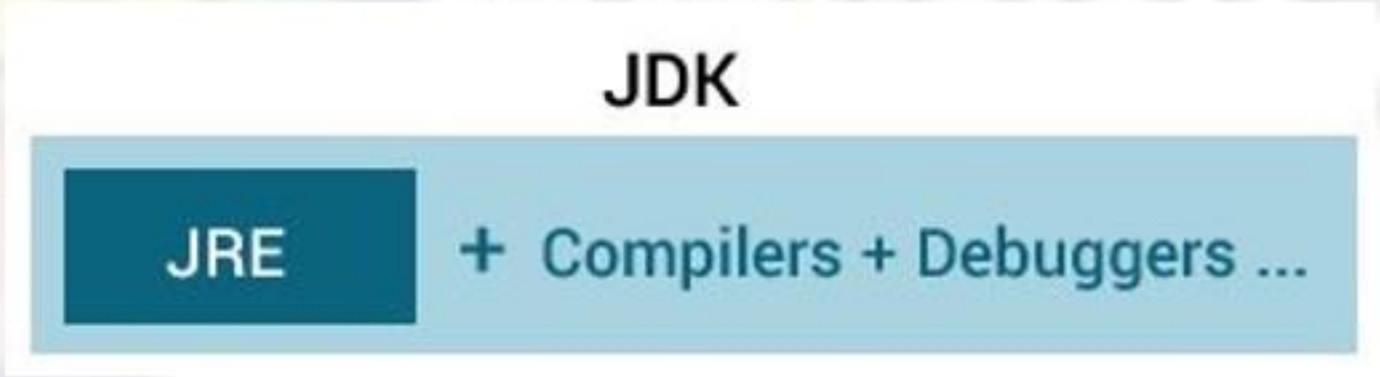
JRE is the superset of JVM.



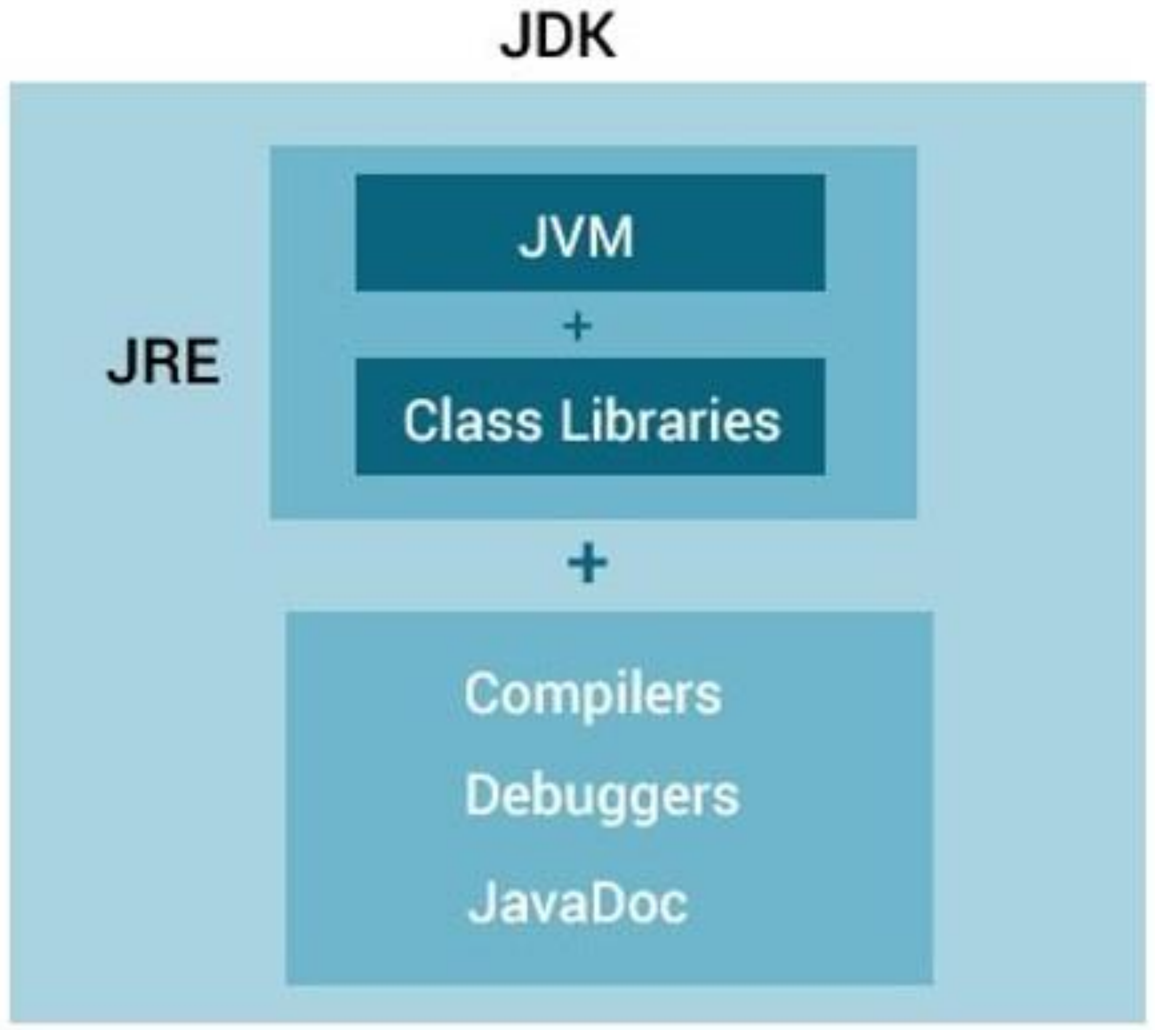
What is JDK?

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 and Literals

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 speedLimit = 80;
```

Here, speedLimit is a variable of int data type and we have assigned value 80 to it.

Note: Java is a statically-typed language. It means that all variables must be declared before they can be used.

Rules for Naming Variables in Java

Java is case sensitive. Hence, age and AGE are two different variables.

Variables must start with either a **letter** or an **underscore, _** or a **dollar, $** sign.

Variable names cannot start with numbers.

Variable names can't use whitespace.

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.

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;
```

Here, false and true are two boolean literals.

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:

1. binary (base 2)
2. decimal (base 10)
3. octal (base 8)
4. hexadecimal (base 16)

For example:

```
// binary
int binaryNumber = 0b10010;
// octal
int octalNumber = 027;
// decimal
int decNumber = 34;
// hexadecimal
int hexNumber = 0x2F; // 0x represents hexadecimal
// binary
int binNumber = 0b10010; // 0b represents binary
```

In Java, binary starts with 0b, octal starts with 0, and hexadecimal starts with 0x.

3. Floating-point Literals

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

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

    double myDouble = 3.4;
    float myFloat = 3.4F;

    // 3.445*10^2
    double myDoubleScientific = 3.445e2;

    System.out.println(myDouble);  // prints 3.4
    System.out.println(myFloat);    // prints 3.4
    System.out.println(myDoubleScientific);   // prints 344.5
  }
}
```

4. Character Literals

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

```
char letter = 'a';
```

Here, a is the character literal.

We can also use escape sequences as character literals. For example, \b (backspace), \t (tab), \n (new line), etc.

5. String literals

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

```
String str1 = "Java Programming";
String str2 = “Talent Battle";
```

Here, **Java Programming and Talent Battle** are two string literals.

Java Data Types (Primitive)

There are 8 data types predefined in Java programming language, known as primitive data types.
Note: In addition to primitive data types, there are also referenced types (object type).

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 1: Java boolean data type

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

    boolean flag = true;
    System.out.println(flag);    // prints true
  }
}
```

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 2: Java byte data type

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

   byte range;
   range = 124;
   System.out.println(range);   // prints 124
 }
}
```

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 3: Java short data type

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

   short temperature;
   temperature = -200;
   System.out.println(temperature);  // prints -200
 }
}
```

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

Example 4: Java int data type

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

   int range = -4250000;
   System.out.println(range);  // print -4250000
 }
}
```

5. long type

The long data type can have values from -263 to 263-1 (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 264-1.
Default value: 0

Example 5: Java long data type

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

   long range = -42332200000L;
   System.out.println(range);    // prints -42332200000
  }
}
```

Notice, the use of L at the end of -42332200000. This represents that it's an integral literal of the long type.

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)

Example 6: Java double data type

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

    double number = -42.3;
    System.out.println(number);  // prints -42.3
  }
}
```

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 7: Java float data type

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

    float number = -42.3f;
    System.out.println(number);  // prints -42.3
  }
}
```

Notice that, we have used -42.3f instead of -42.3in the above program. It's because -42.3 is a double literal.

To tell the compiler to treat -42.3 as float rather than double, you need to use f or F.

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 8: Java char data type

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

   char letter = '\u0051';
   System.out.println(letter);  // prints Q
 }
}
```

Here, the Unicode value of Q is \u0051. Hence, we get Q as the output.

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.