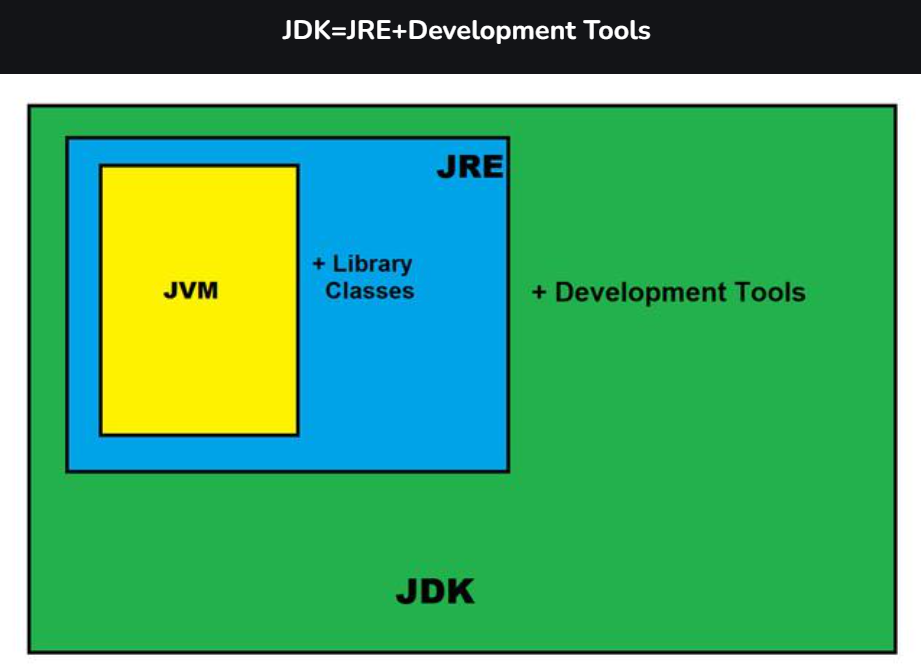
In programming languages like C and C++, the code is first compiled into platform-specific machine code. These languages are called *compiled languages*.

On the other hand, in languages like JavaScript and Python, the computer executes the instructions directly without having to compile them.

**converts each high-level program statement into machine code** .These languages are called *interpreted languages*.

Java uses a combination of both techniques. Java code is first compiled into byte code to generate a *class* file. This *class* file is then interpreted by the Java Virtual Machine for the underlying platform. The same *class* file can be executed on any version of JVM running on any platform and operating system.

*Similar to virtual machines,* **the JVM creates an isolated space on a host machine. This space can be used to execute Java programs irrespective of the platform or operating system of the machine.**



### **JRE**

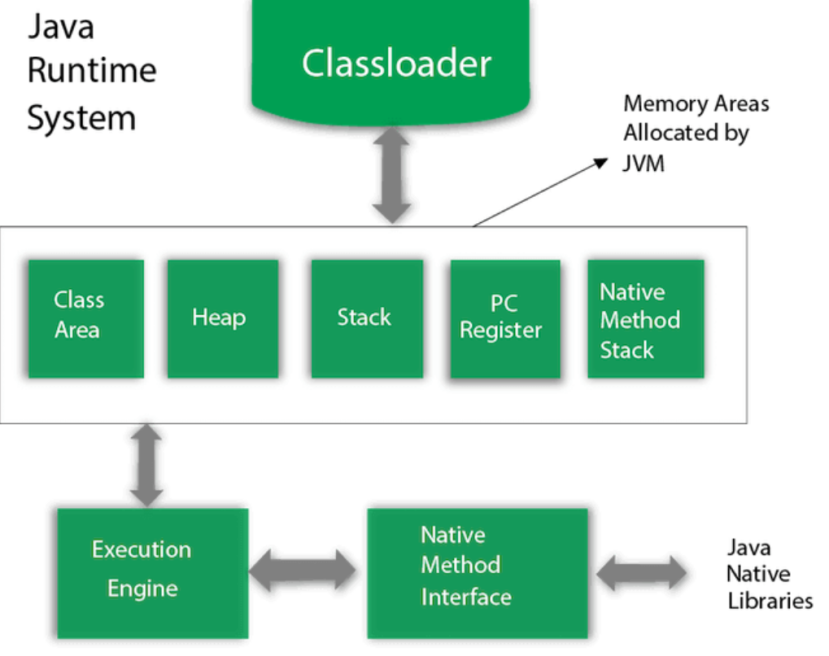
 It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.

The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and [applets](https://www.javatpoint.com/java-applet). It physically exists. It contains JRE + development tools.

The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.

**JDK contains:**

* Private JVM,
* An interpreter/loader (Java),
* A compiler (javac),
* An archiver (jar), Documentataion generator(Javadoc) etc.

****

### **1) Classloader**

Classloader is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the classloader. There are three built-in classloaders in Java.

Bootstrap , Extension , application.

### **2) Class(Method) Area**

Class(Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

### **3) Heap**

It is the runtime data area in which objects are allocated.

### **4) Stack**

Java Stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return.

Each thread has a private JVM stack, created at the same time as thread.

A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

### **5) Program Counter Register**

PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.

### **6) Native Method Stack**

It contains all the native methods used in the application.

### **7) Execution Engine**

It contains:

1. **A virtual processor**
2. **Interpreter:** Read bytecode stream then execute the instructions.
3. **Just-In-Time(JIT) compiler:** It is used to improve the performance. JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here, the term "compiler" refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

### **8) Java Native Interface**

Java Native Interface (JNI) is a framework which provides an interface to communicate with another application written in another language like C, C++, Assembly etc. Java uses JNI framework to send output to the Console or interact with OS libraries.

**Java Syntax**

public class Main {

public static void main(String[] args) {

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

}

}

How to run with cmd =>

**C:\Users\Your Name>javac Main.java**

**C:\Users\Your Name>java Main**

**In Java, every application begins with a class name, and that class must match the filename.**

**A class should always start with an uppercase first letter.**

**Java is case-sensitive: "MyClass" and "myclass" has different meaning.**

## Java Variables

Variables are containers for storing data values.

In Java, there are 5 different **types** of variables, for example:

* String
* int -
* float - stores floating point numbers, with decimals, such as 19.99 or -19.99 **float mynum = 5.99f; double mynum =19.99d;**
* char -
* boolean -

Data types are divided into two groups:

* Primitive data types - includes byte, short, int, long, float, double, boolean and char
* Non-primitive data types - such as [String](https://www.w3schools.com/java/java_strings.asp), [Arrays](https://www.w3schools.com/java/java_arrays.asp) and [Classes](https://www.w3schools.com/java/java_classes.asp) (you will learn more about these in a later chapter)

Non-primitive data types are called **reference types** because they refer to objects.

A primitive data type specifies the size and type of variable values, and it has no additional methods.

* A primitive type has always a value, while non-primitive types can be null.
* A primitive type starts with a lowercase letter, while non-primitive types starts with an uppercase letter.

## Java Type Casting

Type casting is when you assign a value of one primitive data type to another type.

In Java, there are two types of casting:

* **Widening Casting** (automatically) - converting a smaller type to a larger type size  
  byte -> short -> char -> int -> long -> float -> double
* **Narrowing Casting** (manually) - converting a larger type to a smaller size type  
  double -> float -> long -> int -> char -> short -> byte
* int myInt = (int) myDouble; // Manual casting: double to int

System.out.print() => will print in same line

System.out.println() => always prints in same line

\n => for new line

* Math.max(x,y)
* Math.min(x,y)
* Math.abs(x)
* Math.sqrt(x)
* Math.random()

**The Math.abs(x) method returns the absolute (positive) value**

**Math.random() returns a random number between 0.0 (inclusive), and 1.0 (exclusive)**

**ternary operator**

variable *= (*condition*) ?* expressionTrue *:*  expressionFalse*;*

## For-Each Loop

String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

for (String i : cars) {

System.out.println(i);

}

## Java Switch Statements

int day = ;

switch (day) {

case 6:

System.out.println("Today is Saturday");

break;

case 7:

System.out.println("Today is Sunday");

break;

default:

System.out.println("Looking forward to the Weekend");}

## Java User Input

The Scanner class is used to get user input, and it is found in the java.util package.

To use the Scanner class, create an object of the class and use any of the available methods found in the Scanner class documentation. In our example, we will use the nextLine() method, which is used to read Strings:

import java.util.Scanner; // Import the Scanner class

import java.util.\*;

class Main{

public static void main(String[] args){

System.out.println("Hi");

Scanner sc = new Scanner(System.in);

String name;

System.out.println("Enter your Name");

name = sc.nextLine();

System.out.println("Your Name is: " + name);

}

}

nextInt() = for taking next int as input.

**String to int**

  String str1 = "5";

**int** result = Integer.parseInt(str1); // Using Integer.parsrInt()

        System.out.println(result);

**Int to String**

**int** x = 5;

        //1st way

        String str = Integer.toString(x); // using Integer.toString()

        System.out.println(str);

        //2nd way  ( use this mostly)

        String str2 = String.valueOf(x); // using String.valueOf()

        System.out.println(str2);

Documention Comments

The documentation comments are placed between /\*\* and \*/.

****Syntax:****

1. /\*\*
2. \*
3. \*We can also use HTML tags
4. \*
5. \*/

# Java Garbage Collection

In java, garbage means unreferenced objects.

Garbage Collection is process of reclaiming the runtime unused memory automatically. In other words, it is a way to destroy the unused objects.

To do so, we were using free() function in C language and delete() in C++. But, in java it is performed automatically. So, java provides better memory management.

## How can an object be unreferenced?

There are many ways:

* By nulling the reference
* By assigning a reference to another
* By anonymous object etc.

The finalize() method is invoked each time before the object is garbage collected. This method can be used to perform cleanup processing.

The gc() method is used to invoke the garbage collector to perform cleanup processing

Lets Recall

1. How java works vs other language
2. Explain all jdk
3. What are jdk contains
4. Explain java runtime system
5. What is JNI
6. What are data types and primitive vs non primitive
7. Java user input write code
8. Type Casting Sequence
9. Int to string and vice versa (code)