**Java Basics**

### **What is JVM?**

Java virtual Machine(JVM) is a virtual Machine that provides runtime environment to execute java byte code. The JVM doesn't understand Java typo, that's why you compile your \*.java files to obtain \*.class files that contain the bytecodes understandable by the JVM.

JVM control execution of every Java program. It enables features such as automated exception handling, Garbage-collected heap.

### JVM Architecture

**Class Loader :** Class loader loads the Class for execution.

**Method area :** Stores pre-class structure as constant pool.

**Heap :** Heap is in which objects are allocated.

**Stack :** Local variables and partial results are store here. Each thread has a private JVM stack created when the thread is created.

**Program register :** Program register holds the address of JVM instruction currently being executed.

**Native method stack :** It contains all native used in application.

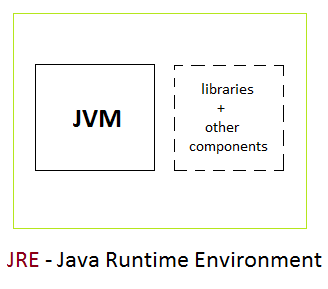
**Executive Engine :** Execution engine controls the execute of instructions contained in the methods of the classes.

**Native Method Interface :** Native method interface gives an interface between java code and native code during execution.

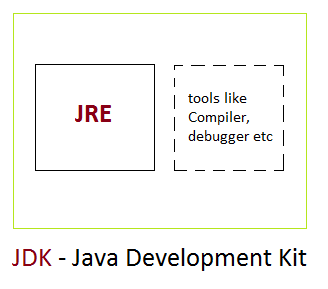
**Native Method Libraries :** Native Libraries consist of files required for the execution of native code.

#### Difference between JDK and JRE

**JRE** : The Java Runtime Environment (JRE) provides the libraries, the Java Virtual Machine, and other components to run applets and applications written in the Java programming language. JRE does not contain tools and utilities such as compilers or debuggers for developing applets and applications.



**JDK** : The JDK also called Java Development Kit is a superset of the JRE, and contains everything that is in the JRE, plus tools such as the compilers and debuggers necessary for developing applets and applications.



**Class and Object**

### **Object and Classes**

Since Java is an object oriented language, complete java language is build on classes and object. Java is also known as a strong **Object oriented programming language**(OOPS).

OOPS is a programming approach which provides solution to problems with the help of algorithms based on real world.

#### **Main Features of OOPS**

* Inheritence
* Polymorphism
* Encapsulation
* Abstraction

### **Class**

In Java everything is encapsulated under classes. Class is the core of Java language. Class can be defined as a template/ blueprint that describe the behaviors /states of a particular entity. A class defines new data type. Once defined this new type can be used to create object of that type. Object is an instance of class. You may also call it as physical existence of a logical template class.

A class is declared using **class** keyword. A class contain both data and code that operate on that data. The data or variables defined within a **class** are called **instance variables** and the code that operates on this data is known as **methods**. Thus, the instance variables and methods are known as class members. **class** is also known as a user defined datatype.

**A class and an object can be related as follows:** Consider an ice tray(like of cube shape) as a class. Then ice cubes can be considered as the objects which is a blueprint of its class i.e of ice tray.

#### Rules for Java Class

* A class can have only public or default(no modifier) access specifier.
* It can be either abstract, final or concrete (normal class).
* It must have the class keyword, and class must be followed by a legal identifier.
* It may optionally extend one parent class. By default, it will extend java.lang.Object.
* It may optionally implement any number of comma-separated interfaces.
* The class's variables and methods are declared within a set of curly braces {}.
* Each **.java** source file may contain only one public class. A source file may contain any number of default visible classes.
* Finally, the source file name must match the public class name and it must have a .java suffix.

#### **A simple class example**

Suppose, Student is a **class** and student's name, roll number, age will be its property. Lets see this in Java syntax

class Student.

{

String name;

int rollno;

int age;

}

When a reference is made to a particular student with its property then it becomes an **object**, physical existence of Student class.

Student std=new Student();

After the above statement **std** is instance/object of Student class. Here the **new** keyword creates an actual physical copy of the object and assign it to the **std** variable. It will have physical existence and get memory in heap area. **The new operator dynamically allocates memory for an object**

