# Java Basics

- **JVM ->** A blueprint of how to provide a runtime environment in which Java bytecode can be executed.
- **JRE ->** Actual implementation of JVM. Provided by different companies eg:- Sun, Oracle, etc.
- **JDK ->** JRE + development tools(compiler, interpreter, archiver(jar),etc)

- **Java Classloader**
  - Is a part of the **JRE** that dynamically loads Java classes into the **JVM.**
  - Java classes aren't loaded into memory all at once, but when required by an application. At this point, the **Java ClassLoader** is called by the **JRE** and these ClassLoaders load classes into memory dynamically.
  - If any of these classes are not found then it returns a **NoClassDefFoundError** or **ClassNotFoundException**.
    - **BootStrap** or **Primodial ClassLoader ->** loads classes from the location ***rt.jar***
    - **Extension ClassLoader ->** loads files from ***jre/lib/ext*** directory or any other directory pointed by the system property.
    - **System** or **Application ClassLoader ->** loads classes from **classpath**.
    - **Priority ->** BootStrap > Extension > Application

- **Bitwise Operator**
  - **Left Shift -> multiply**
    - (10<<2) -> 10*2^2 -> 40
    - (5<<3) -> 5*2^3 -> 40
  - **Right Shift -> Divide**
    - (10>>2) -> 10/2^2 -> 2
    - (50>>5) -> 50/2^5 -> 1

- **Constructors**
  - Called when instance of class is created.
  - Memory for object is allocated.
  - It can't be **static** because static means it belongs to class and not to objects and the job of constructor is to initialize objects.
  - **Final** methods can't be overidden. Since **constructers** cannot be inherited by the child class, writing **final** before constructor makes no sense.
  - If we declare a constructor as **abstract** we have to implement it in a child class, but we know a constructor is called implicitly,so it can't lack a body. Also, if we make a constructor abstract then we have to provide the body later but constructor can not be overridden so providing body

    - is impossible. Hence, what we will do with this abstract constructor when we can not provide implementation to it.
    - Abstract classes can have constructors which can be called from sub classes using constructor chaining.

- Interfaces have **public**, **static** & **final** variables and **public abstract** methods. Hence there is no point of having a constructor in interfaces as static final fields needs to be initialized at the time of declaration.
- If you are overriding any method, overridden method (i.e. declared in subclass) must not be more restrictive.

- To create Immutable class
    - declare class as final -> so that child classes cannot be created
    - data members should be private -> direct access is not allowed
    - data members should be final(optional) -> value can be changed after object creation
    - No setters -> not give an option to change the value
    - Provide all-arg constructor -> to initialize data members at the time of object creation
    - Initializing all non-primitive mutable fields via constructor by performing a deep copy
    - Performing cloning of the returned non-primitive mutable object in getter methods
-