

## 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 **Primordial 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**
    - ◆  $(10 \ll 2) \rightarrow 10 * 2^2 \rightarrow 40$
    - ◆  $(5 \ll 3) \rightarrow 5 * 2^3 \rightarrow 40$
  - **Right Shift -> Divide**
    - ◆  $(10 \gg 2) \rightarrow 10 / 2^2 \rightarrow 2$
    - ◆  $(50 \gg 5) \rightarrow 50 / 2^5 \rightarrow 1$
- **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 overridden. Since **constructors** 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
- String is immutable because
  - String Pool
  - Caching
  - Security
  - Thread-safe
- intern() is used to place an string created with new keyword in string constant pool.

## Java Memory Model

- Heap memory is divided into 2 parts :-
  - Young Generation
    - ◆ Eden Memory
    - ◆ Two Survivor Memory(S1, S2)
  - Old Generation

New object is created in Eden space. When Eden space is full, a minor GC is performed and the survivor objects are moved to one of the survivor space. Minor GC also moves survivor objects into one survivor space so that at a time one survivor space is always empty. Objects that survive many cycles of GC are moved to old generation. Old generation contains long lived objects. When old gen is full, major GC runs. It takes more time to run major GC.

- Non-Heap memory
  - Permanent Generation
    - ◆ It stores per class structures such as runtime constant pool, field and method data, and the code for methods and constructors, as well as interned Strings.
- Stack Memory
  - Is used for static memory allocation and the execution of a thread. It contains primitive values that are specific to a method and references to objects that are in a heap, referred from the method.