**DESIGN INTRODUCTION**

**What Is Java?**

* Developed in **1995**.
* Earlier, Java was known as **Oak**. Name changed as Oak was already a registered company.
* **Platform:** Any hardware/software environment in which a program runs.
* Java’s platform is JRE.

**API:** Application programming interface.

**Types Of Java Applications**

* **Standalone Application:** General purpose applications. Java uses AWT (abstract window toolkit) for GUI and swing for GUI & backend developed by Sun Microsystems.
* **Web Applications:** Applications which uses servers constantly and creates dynamic pages. **For this we use:-** Servlet , JSP, Struts, Spring, Hibernate, JSF etc in Java right now.
* **Enterprise Applications:** Advantages includes **high-level security, load balancing, clustering** etc. For Java we use API called **EJB**.
* **Mobile Applications:** Java currently uses **Android Studio** and **Java ME** for now.

**Java Editions**

* Java SE
* Java EE
* Java ME
* JavaFX (For internet rich apps)

**JVM**

* Doesn’t physically exist.
* Converts bytecode to Java executable code or vice versa.
* Or simply saying, compiles.
* Java command on CMD creates JVM instance.

**JRE**

* Environment to run Java.
* Contains JVM and Java libraries.

**JDK**

* Framework and platform, set of tools.

JVM provides environment, JRE implements it.

**ClassLoaders**

**Bootstrap:** Is superclass, loads ***rt.jar*** file. Contains ***java.lang*,*net*,*util*,*io*,*sql*** packages.

**Extension:** Loads jar files at **$JAVA\_HOME/jre/lib/ext**.

**System/Application:** Loads class files from ***classpath*** (**current directory**), change directory using ***-cp*** or ***-classpath***.

Write ***c.getClassLoader()*** to access class loader of the object.

**Class Area**

* Contains data about methods in class.
* Also contains per class runtime information.

**Program Counter Register**

* Also called **PC Register**.
* Contains address of JVM instruction, executing currently.

**Execution Engine**

* Contains:
  + A virtual processor
  + Interpreter
  + JIT compiler

**Java Native Interface**

* Used to communicate with other languages.
* Java uses **JNI** framework.

**OOPs**

* ***Simula*** is first OOP language.
* ***Smalltalk*** is first **truly** OOP language.
* **s** in OOPS is **system**.

**Paradigm:** A **pattern** followed in writing of certain linguistic/programming language.

**Extra OOPs Concepts**

* **Coupling:** One class depending on another class, using it in itself (**weak/strong**).
* **Cohesion:** ***java.util*** is weakly cohesive as it splits tasks into multiple components, whereas ***java.io*** is…
* **Association:** 4 relations between objects (one to one, one to many etc).
* **Aggregation:** Represents weak relationships among objects.
* **Composition:** Represents strong relationships among objects.