We have 2 types of data types –

1. Primitive data type

There are 8 primitive types in java and they are mentioned as keywords in java.

These 8 primitive data types are

1 byte = 8 bit

* Byte – has memory size of 1 byte (8 bit).
* Short – has memory size of 2 byte (16 bit)
* Int – has memory size of 4 byte ()
* Long - has memory size of 8 byte ()
* Float - has memory size of 4 byte ()
* Double - has memory size of 8 byte ()
* Boolean - has true and false values
* Character – java stores ASCII values

1. Non-Primitive data type
2. Derived data types –

Ex: Array

1. User defined data types –

Ex: enum (e is in lower case alphabet)

Enums – enums are predefined set of values

Syntax – enum enum-name {enum1, enum2}

Every enum is final implicitly. When something is marked final it cannot be overridden or extend enum. This is purposely designed because enums are meant to have predefined values. If is marked final so that it cannot be overridden and modified.

There are different types of operations –

1. Arithmetic
2. Relational
3. Bitwise
4. Logical
5. Assignment
6. Miscellaneous

JVM fundamentals –

1. Virtual Machine
2. Types of Virtual Machine
3. Hardware based vm
4. Application based vm
5. Basic Architecture diagram of JVM
6. Classloader sub system
7. Loading
8. Linking
9. Initialization
10. Types of classloader
11. Bootstrap classloader
12. Extension classloader
13. Application classloader
14. How java classloader works?
15. Need of customized classloader?
16. Pseudo code to define customized classloader?
17. Various memory area of jvm
18. Method area
19. Heap area
20. Stack memory
21. PC registers area
22. Native method stacks area
23. Importance of runtime class
24. Program to display statistic of heap memory
25. Maxmemory
26. Totalmemory
27. freememory
28. How to set maximum and minimum heap size?
29. Execution engine
30. Interpreter
31. JIT compiler
32. Java native interface
33. Class file structure
34. Magic number
35. Minor version
36. Major version

What is virtual machine?

A software simuation of a machine which can perform operations like a physical machine. It does not have physical existance.

Types of VM

1. Hardware based VM / system based VM

It provides several logical based separation systems on the same computer with strong isolation from each other.

Advanatges: effective use of hardware resources

Ex: vmware, KVM (kernel virtual machine), xen, cloud computing

(system admin related terminologies not related to programmers)

1. Application based VM / process based VM

These virtual machines act as runtime engines to run a particular language application.

Ex: JVM(java virtual machine) acts as a runtime engine to run java applications.

PVM(parrot virtual machines) acts as a runtime engine to run perl(scripting) application.

CLR (Common language runtime) acts as a runtime engine to run .net applications.

Basic JVM Architecture:

JVM



JVM is a runtime engine to run java based .class files

JVM is present inside JRE, JRE is present inside JDK, JDK is the outermost element.

There are 3 components of jvm architecture:

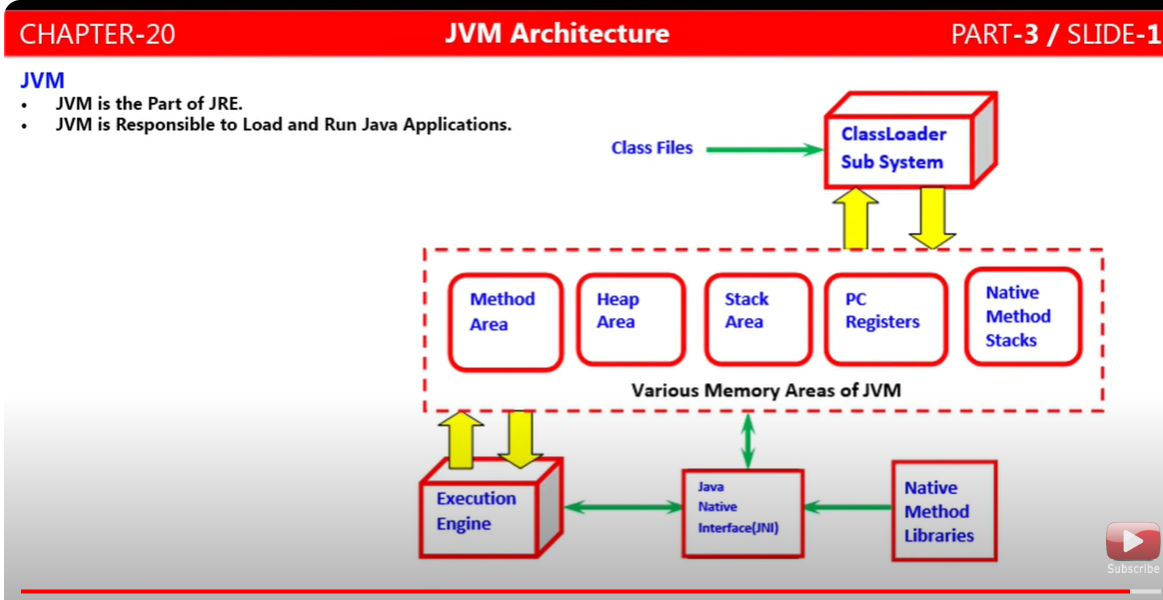
1 classloader sybsystem – responsible for loading .class files

2 jvm memory area – all the methods, variables are loaded in the memory area. There are 5 memoery areas 1. Method area, 2. Heap area, 3. Stack area, 4. PC registers, 5. Native method stacks

3 execution engine – responsible for exeuting line by line code from .class file

Java Native Interface – used to load native methods.

Hashcode, clone are native methods. They are written in C language.



ClassLoader subsystem –

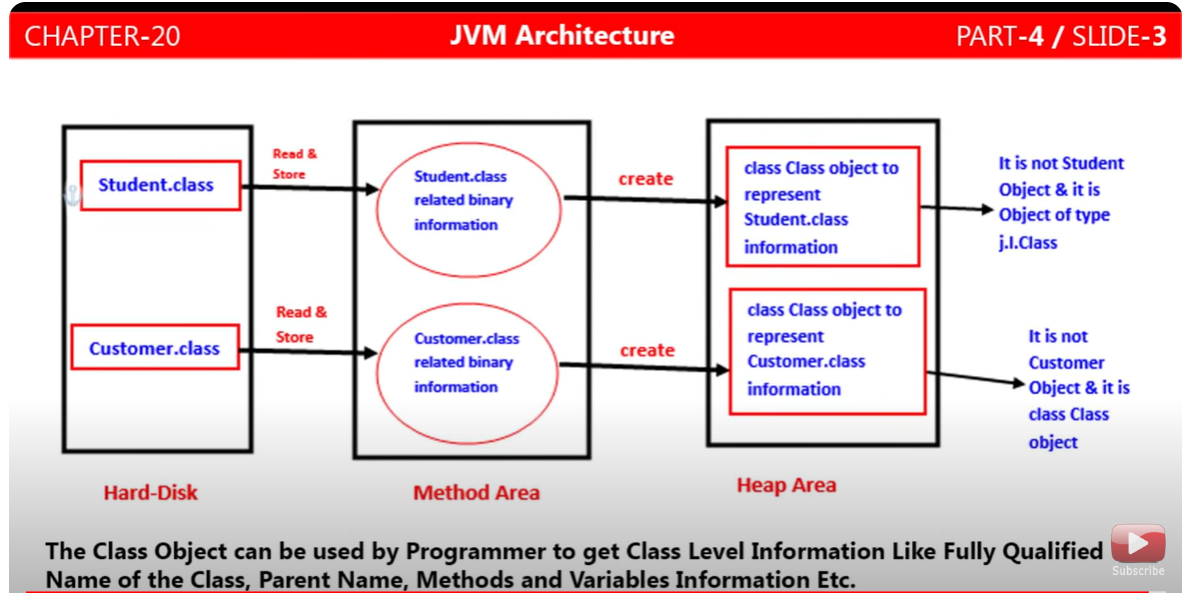
It is responsible for 3 activities:

1. Loading

Readinf .class file information and storing corresponding binary information inside memory area. For every .class file fully qualified name of class, method name, parent class name, variable informations, constructors information etc etc.

Complete information is saved in memory area

Loading process –



Consider you compiled your Student.java file and java compiler converted into .class file and saved that .class file in the harddisk. From hasddisk jvm will inform classloader subsystem to load .class files into its memoery area. Once all the .class information is loaded into memory area, it will create an object of class Class and hacve all the information loaded into that class Class object. This is not Student class object. It is class Class object storring all the binary information of the .class file loaded from memory area. Object of class Class will be stored in heap area.

Refer the above diagram for clarity.

We programmers can refer the class Class object to get the method names in that class, variable name in that class.

Please refer the program below –

Public class Student {

Int rollno;

String name;

Public int getRollNo(){

Return this/rollno;

}

Public void getRollNo(int no){

This.rollno = no;

}

Public string getName(){

Return this.name;

}

Public void setName(String name){

This.name = name;

}

}

Class Test{

Psvm(string[] args){

Class c = Class.forName(“student”);

Method[] methods = c.getdeclaredMethods();

// iterate methods to print method name

Field[] fileds = c.getDeclaredFields();

// iterate fields to print field names

// Method and Field are classes in java.lang.reflect package.

// with help of this program programmer can get thedetials of the methods varaiblaes, in the java files.

}

}

For every loaded .class file, only one class Class object will be creared by JVM, even though we are using that class object multiple times in our program.

Class Test{

Psvm(){

Student s1 = new Student();

Class c1 = s1.getClass();

Student s2 = new Student();

Class c2 = s2.getClass();

Syso(c1.hashCode());//1234

Syso(c2.hashcode());//1234

Syso(c1==c2);//true

Both sop will print same hashcode bcoz both are refering/poinitng to the same object in the memory

}

}

1. Linking
2. Verification
3. Preparation
4. Resolution
5. Initialization