Name : Shivam Relan

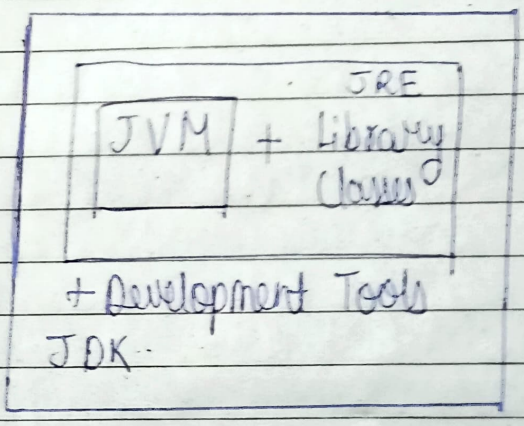
Assignment 3

1) Components of JDK:
→ JDK - Java Development Kit.
a) JDK is a cross-platform environment which provides tools and libraries for development of java programs and applications.

b) JRE (Java Runtime Environment) & JVM (Java Virtual Machine) are integral parts of JDK.

c) As JRE only provides runtime environment it is sufficient enough for client side application execution.

d) Diagram



e) JVM is the virtual machine of JDK and it runs the Java applications as runtime engine.

Page No.
Date / /

2.) Difference between JDK, JVM & JRE

- JDK is the development environment that facilitates the development & execution of the Java application.
- AND JRE is the integral part of JDK which provides runtime environment for java application. But it doesn't have development tools provided by JDK.
- A client can only download JRE inorder to Run java application on their device, if they are not in any need of development.
- But for Developers JDK as a whole is required, as it provides development tools.
- JRE → for clients. [Run not Develop]
  JDK → for users/developers.
  [Develop and execute]
- JVM – JVM acts as runtime engine for both JDK & JRE as it is responsible for executing the applications & programs.
  - It is inbuild for both JDK & JRE.
  - JVM is also responsible for executing the code line by line hence also known as interpreter.

3) What is the role of JVM in Java? & How does JVM execute Java code?

* JVM – Java Virtual Machine

- JVM is a runtime engine.
- JVM is responsible for calling main method in java program and executing it.
- JVM is part of JRE ~~and JDK~~.
- Java application property "Write Once Run anywhere" [WORA] is possible because of JVM.
- The .class file that we get after we compile the .java is executed by JVM.

Thus JVM takes byte code and convert it into machine code. Thus JVM play very important role in making java programs portable.

b) Java code execution.

When the .class file code enters the JVM, first it enters the class loader.

* ClassLoader has three main responsibilities
  - Loading
  - Linking
  - Initialization.

Page No.:
Date. / /

Bootstrap Class Loader loads: core Java API classes.

Extention Class Loader: loads platform specific extention's from JDKs module system.

- System Class loader: It is the child of Extention Class loader and loads the class from application class path.

- Next it enters the JVM memory Areas where:

1.) Memory area: all class level info is stored like class name, parent class name, methods & variables, etc.

2.) Heap area: Information of all Objects is stored in Heap area.

3.) Stack area: JVM creates one run-time stack for each thread, it is stored here.

4.) PC Register: stores all the execution thread instruction of a thread.

Execution Engine:

It executes .class file (bytecode) line by line

It first reads bytecode line by line and uses data and info present in various memory area and executes the instructions.

It converts byte code to machine code.

Page No.
Date: / /

5) What is JIT compiler in JVM and it's role in JVM? What is byte code and why is it important for java.

Ans → • JIT → Just-In-Time compiler is used in JVM to increase the efficiency of interpreter.
• It compiles the entire byte code and changes it to native code.
So when the interpreter encounters repeated method calls, JIT provides a direct native code thus re-interpretation is not required thus saving lot of time.

B) Bytecode:

→ The java compiler compiles the java code in bytecode in .class file.
- Now this .class file is portable in nature.
- Because the bytecode can be executed on any os containing JDK or JRE.
Bytecode is not os dependent, thus it's execution is same for all os.
- Thus the phrase "write once run anywhere" was introduced.

6) Describe the architecture of JVM.
→

Ans: Components of JVM:

A) Class loader subsystem
1. Bootstrap class loader
2. Extension class loader / Platform class loader
3. System class loader / Application class loader
4. Custom class loader.

B) Runtime data areas
1. Method area
2. Heap
3. Java stacks
4. PC Register
5. Native method stacks.

C) Execution Engine
1. Interpreter
2. Just In Time Compiler
3. Garbage Collector

→ Class Loader is used for:
Loading linking & initialization purposes.

Bootstrap → Loads core Java APIs
Extension → Loads platform specific extentions from JDK's module.
System → Child of Extention class loader and load class from applications class path.
Custom → It is customizable class loader.

Page No.
Date / /

→ Runtime data areas are used to store different components of program.

1) Method area → All class level information is stored including class name, parent classname, methods, variables, etc.

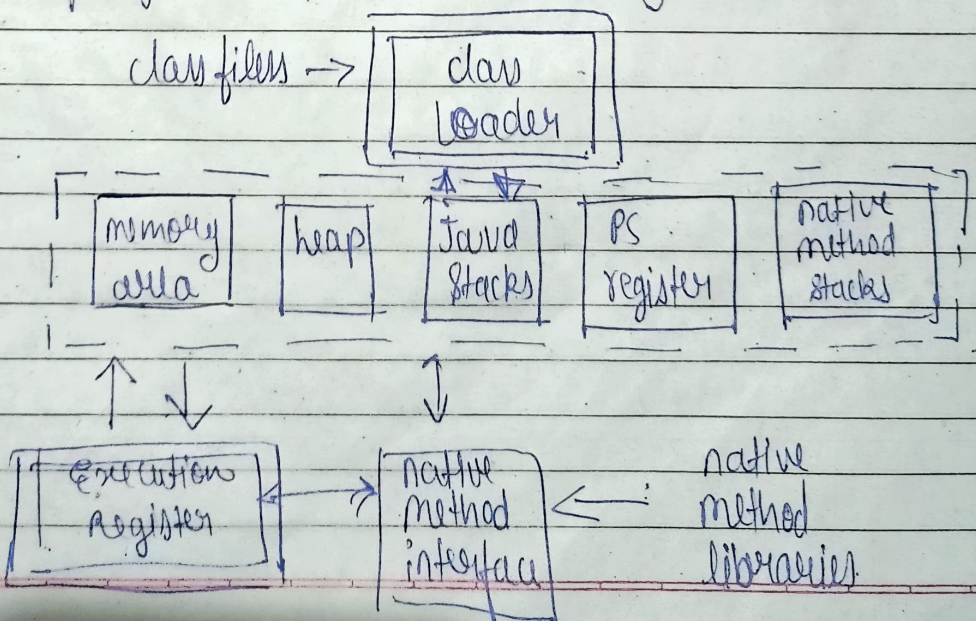
2) Heap → Info of all objects is stored in heap area.

3) Stack → For each thread, JVM creates one runtime stack which is stored here.

4) PC Register: Stores all execution thread instruction of a thread.

5) Native Method Stack → useful to include other native language progs like cpp in java.

→ Execution Engine: Here the bytecode is converted into native machine and the program is executed line by line.



7) How did java achive platform independence through the JVM ?

- meaning of platform independence is that - java compiled code (byte code) can run on all operating systems.

- The java program is compiled into .class file by javac compiler.
- This .class file is a bytecode file and it is a non-executable code does need an interpreter to execute on a machine.
- JVM is this interpreter, thus bytecode in converted to machine code by JVM and executed line by line.
- JVM initiates execution by accessing main() function from main.class.
- JVM is platform dependent for each OS. But bytecode is executable on every OS ~~with~~ because of JVM.
Thus → Write Once Run anywhere.
- Thus, this is how JVM helps in achiving platform independence.

8) Significance of class loader in java.

Class loader in JVM are responsible for:

8) What is significance of class Loader in Java? What is process of Garbage collection in Java.

→ In JVM Class Loader has three responsibilities:
a) Loading
b) Linking
c) Initialization.

Class Loader reads .class file and generates the corresponding binary data and save it in method area. For each ".class" file, JVM stores the following info in method area.
- Name of class & parent name.
- Whether ".class" file related to Class or Enum or Interface.
- Modifiers, method and variable information, etc.

→ It has four parts:
- Bootstrap loader.
- Extension class loader
- System class loader
- Custom class Loader.

B) Garbage collector: Garbage collection [GC] is one of the most important features of Java.
It is used to de-allocate unused memory, which is nothing but clear the space consumed by unused objects.

Q) What are four access modifier in java and how do they differ from each other.

A) There are four access modifier in java:

1) Private:
Private members are only accessible in same class they are declared in.

2) Package level private (default):
When no access modifier is specified to a class or method or data members then they have default access modifier.
Default access members are only accessible within the same package.

3) Protected:
Protected data members are accessible in same package or in subclasses in different packages

4) Public: There is no restriction on the scope of public data members.
It is accessible in same or different packages as.

	Same package			Different package	
	Same class	Subclass	Non subclass	Subclass	Non subclass
Private	A	NA	NA	NA	NA
Default	A	A	A	NA	NA
Protected	A	A	A	A	NA
Public	A	A	A	A	A

10.) What is difference b/w public, protected & default access modifiers.

→ Answer same as question 9.

11.) Can you override a method with a different access modifier in a subclass?

→ Method overriding is one of the ways java achieves runtime polymorphism.

Method overriding in a subclass does allow overriding a method with different access modifier But.

The overriding method can allow only more access but not less access than the overridden method.
i.e. we cannot use more restricted access modifier.

(eg.) protected instance method in subclass can be made public.
But public protected instance method cannot be made private.

Page No.
Date. / /

12) What is the difference between protected and default (package-private) acces

- protected access: It protected members are accessible in the same package as well as in a subclass in other package.
- Default: It allows default members to only be accessed anywhere in current package only.
- So the major difference is that protected access range is beyond the package it can be accessed in other package by a subclass of current class but default access modifier's range is within the package itself, It can be accessed outside the package.

13) Is it is accessible to make a class private in Java? If yes then where can it be done and what are limitations.

→ Yes we can declare classes as private class but only inner or nested class not the outer top level class or main class.

14) Can top level class can be declared as private or public.

→ • We cannot declare a top level class as private or protected.
• It should be declared as public or private.
• A top level class as private will completely be useless because nothing would have access to it

15) What happens if you declare a variable or method as private in a class & try to access them from another class within the same package?

→ Private methods are only accessible in same class, any other class of same package will not be able to acces their members.

16) Explain the concept of "package-private" or "default" access. How does it affect the visibility of class members?

→ The "package-private" means that the class or data members are accessible with the package but outside the package they act as private for class which are not in their same package. And it is the default access modifier. Thus class members are only visible to the classes and members inside the same package.