Assignment - 3

Page No.
Date

Q.1) Components of the JDK

ans - JDK stands for Java development kit
- used for developing Java programming Applications
- It contains tools such as, Java compiler which convertr Java source code to the byte code.
- Basically the bytecode generate by Java Compiler can run on any platform with the help of JVM (Java Virtual machine)

.class	.class	.class
mac OS	Linux	windows

- The above .class files are in various platforms, basically we can use/execute the class file which is generated by the compiler on any platform

Components :

1] JRE (Java RunTime Environment) :
- It's a Component in JDK which includes JVM, Core classes & Supporting classes
- It's used to run the Java applications

2] Javac (Java compiler) :
- Convert Java source code into byte code that can be understood by the JVM

3] Java Debugger :
- A tool which is used to identify & fix errors in java code

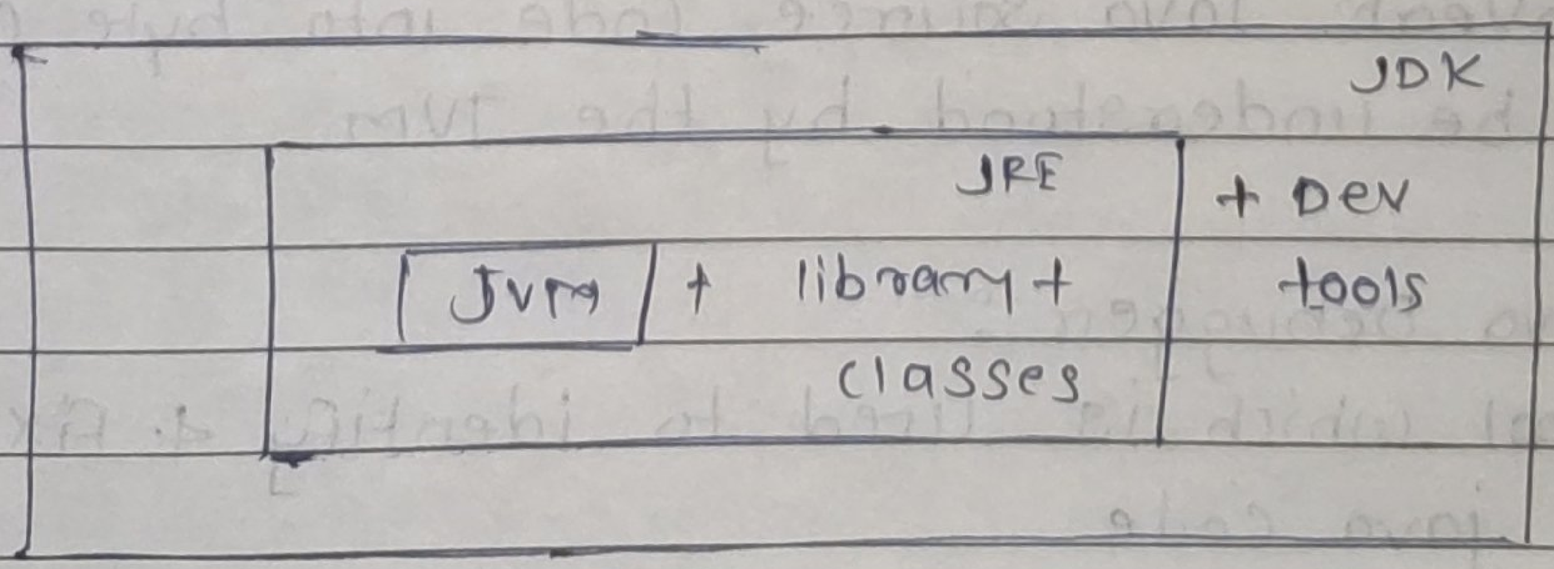
Page No.
Date

Q.2 Difference betn JDK, JRE, JVM

ans - JDK (Java Development Kit) :
purpose : used for developing Java applications
Components : JVM + JRE + compiler + debuggert other development tools
functionality : provides comprehensive set of tools & libraries necessary for developing, compiling, testing & deploying Java Application

- JRE (Java RunHime Environment):
purpose : It is used to compile the Java applicaHons
components : JVM + Java class library
functionality : provide Java applicaHons can run on different operating systems without modificaHon

- JVM (Java Virtual Machine) :
purpose : It interprets & executes Java bytecode
functionality : JIT (just in Time) compilaHon
optimizes Java bytecode for faster execuHon by translaHng it into machine code at runHme
- Automatically allocates & deallocates memory (Garbage collecHon)



Q.3 Role of JVM in Java ? How does JVM executes the Java code ?

→ - main role of JVM is, when a Java compiler compiles, Java

Source code —Javac→ Byte code

The byte code is converted to machine specific code with JIT compiler

So,

Byte code —JIT compiler (JVM)→ Machine specific code

- JIT compiler is used to executes parts of code, in a faster way compared to the CPU.
- JVM automatically manage memory, allocation & deallocation with the help of garbage collector
- It helps to prevent memory leaks & improves application performance
- JVM provides thread management where it can create or manage the threads, allow Java applications to perform multiple tasks concurrently
- JVM allows Java program to run on any platform that has a compatible JVM installed as per the platform like, windows, macos, Linux

Page No.
Date

Q.4 Memory Management System of JVM.

ans - JVM automatically manages memory using garbage collection
- It identifies & reclaims the unused memory
- when an object is created, it acquires some space IF an object is no longer reachable, its considered as a garbage value
- Some garbage collectors may also perform compaction, which involves moving live objects together in memory to reduce fragmentation & improve memory utilization

Q.5 what are JIT compiler & its role in JVM? what is the bytecode & why is it important for Java?

ans - A JIT compiler basically converts a byte code into machine specific code
- It is a part of JVM which is used to analyze & compiles the code and executes in no. of parts which makes it faster executable than CPU.
- Basically, bytecode is an intermediate language that is generated when Java source code is compiled.
- As JIT is platform independent means it can be executed on any system that has a compatible JVM

Q.6) Architecture of JVM :

- JVM is a complex piece of S/W that can be divided into several key components :
1) class loader : loads the class into memory

2] Bytecode verifier : checks the validity of the loaded bytecode

3] Interpreter : Executes the validity of the loaded bytecode

4] JIT compiler : optimizes bytecode by converting it to machine code at runtime

5] Garbage collector : manages memory allocation & deallocation

6] Execution engine :
Executes the machine code generated by the JIT compiler

7] Native Interface :
Allows Java code to interact with native libraries

Q.7) How Java achieves platform independents through the JVM ?

ans - Java achieves platform independence through the JVM. The JVM acts as an abstract machine that translates Java byte code into machine specific code.
- Some Java bytecode can be executed on any platform that has a compatible JVM

Page No.
Date

Q. 8) what is the significance of class loader in Java? what is the process of garbage collection in Java?

Ans. Significance of class loader:

- It is an internal part of the Java Runtime Environment (JRE) that dynamically loads Java classes into the JVM.
- class loader controls loading of classes from different resources
- Helps to prevent Malicious code execution
- Also developers can create custom load classes to implement specific loading strategies

process of Garbage Collection:

Step-1 : It is controlled by a thread known as Garbage Collector

Step-2 : Java provides two methods System.gc() & Runtime.gc() that sends request to the JVM for garbage collection

Step-3 : If the heap memory is fully (where the objects are stored), the JVM will not allow to create a new object & throws an error Java.lang.outofmemoryError

Step-4 : when garbage collector, removes the object from memory. First, the garbage collector thread calls the finalize() method of that object & then remove

Q 9 what are four access modifiers in Java? How they differ from each other?

ans - Access modifiers are used to control the visibility of the members of the class/enum/interface

- There are 4 access modifiers in Java:
  1) private
  2) package level private (default)
  3) protected
  4) public

public:
visibility - Accessible from anywhere within the project including other classes, packages, modules
usage - used for public API's, classes, method & variables that need to be accessible to external code

private:
visibility - Accessible only within the same class
usage - used for methods, & variable that should be encapsulated within a class and not directly accessible from outside

protected:
visibility: Accessible within the same class. its subclasses and other classes within the same package

usage :- used for methods and variables that need to be accessible to subclass but should not be directly accessible to unrelated classes

Default (package private) :
visibility :- Accesible only within the same package

Q.11 can you overide a method with different access Modifier in subclass? ex. can a protected method in a superclass be overriden with a private method in subclass? explain

→ yes, it is possible to overide a method with a different access modifier in subclass
- public - can be overriden by public, protected, default or private
- protected - can be overriden by public, protected or default access modifiers
- private - cant be overriden
- Default (package-private) - can be overiden by public, protected or default

Q.12) Difference betn protected and default (package private) in java?

ans protected -
visibility : Accesible within the same class, its subclasses & other classes within the same package

Default :
visibility : Accessible only within the same package

Page No.
Date

Q.13) It is possible to make a class private in Java? If yes, where can it be done, and what are the limitations?

ans - It's not possible to make a class private in Java.
- Because, classes in Java are inherently public by default
- This means they are accessible from anywhere in the project, including classes, packages, Modules

Q.14) Can a top-level class in Java be declared as protected or private? why or why not?

— - No, a top-level class in Java cannot be declared as protected or private
- Top-level classes, which are not nested within any other class, must be declared as public or default.
- protected & private access modifier limit the visible of a class to specific contextr

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

ans - compile error will occur
- In Java, private access modifier restrics the visibility of a variable or method to within the class where it is declared

Page No.
Date

Q.16) package private (default) access ?

ans - package private or default access in java, limits visibility of class member to the same package.

- If no access modifier is specifies, itr assumed to be package private
- this provides a middle ground bet" public & private acces, allowr encapsulation