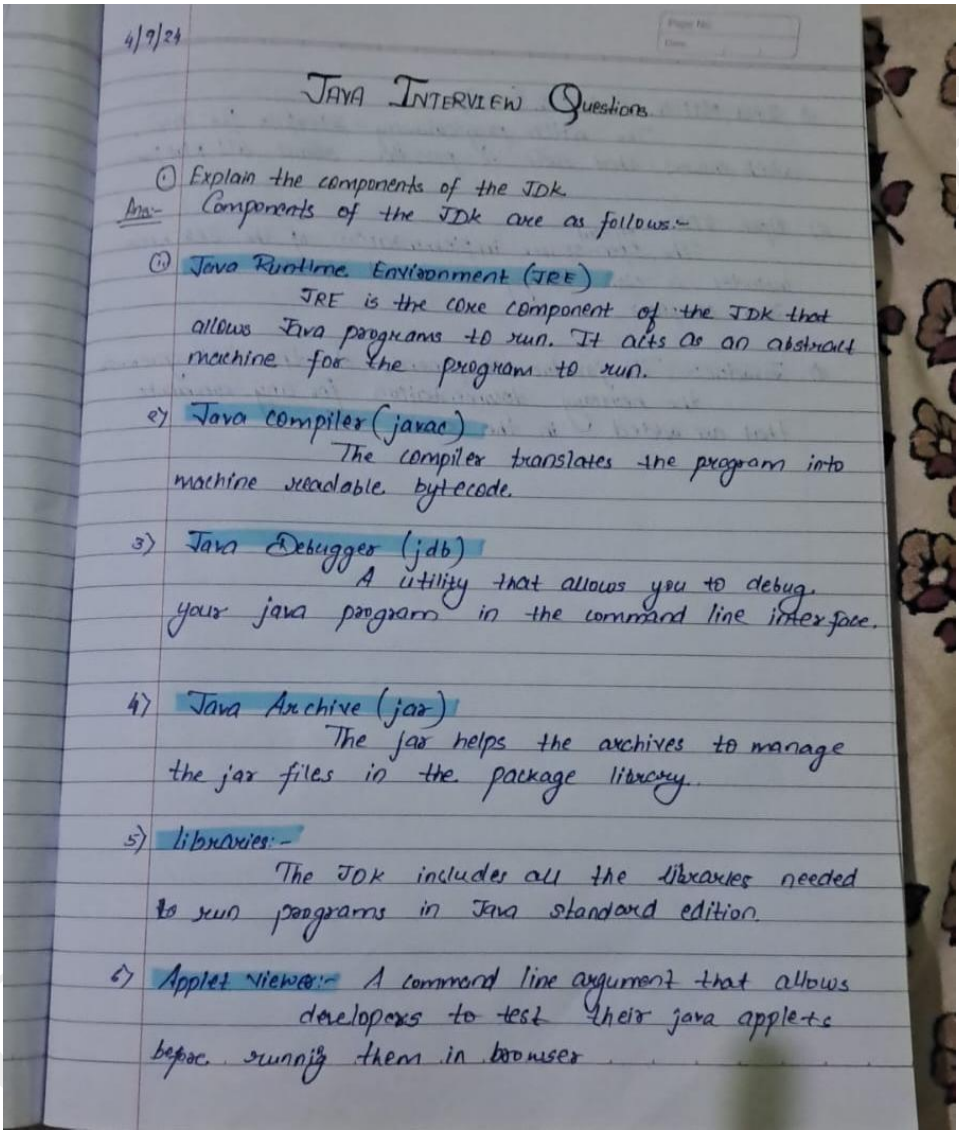
CDAC Mumbai PG-DAC AUGUST 24
Assignment No- 3

Note: Write down this Interview questions & answers in your notebook take a screenshorts, make word file & upload on Github.

1) Explain the components of the JDK.



4/7/24

JAVA INTERVIEW Questions

① Explain the components of the JDK.

Ans:- Components of the JDK are as follows:-

(i) Java Runtime Environment (JRE)
JRE is the core component of the JDK that allows Java programs to run. It acts as an abstract machine for the program to run.

2) Java compiler (javac)
The compiler translates the program into machine readable bytecode.

3) Java Debugger (jdb)
A utility that allows you to debug your java program in the command line interface.

4) Java Archive (jar)
The jar helps the archives to manage the jar files in the package library.

5) libraries:-
The JDK includes all the libraries needed to run programs in Java standard edition.

6) Applet Viewer:- A command line argument that allows developers to test their java applets before running them in browser.

7) Java Native Interface (JNI):-
The native programming interface for Java, which ensures that code is portable across all platforms

8) Open JDK:-
The official Open-source implementation of the JDK which includes a class library a virtual machine and a Java compiler

9) Javadoc:- The javadoc components of the JDK generates the necessary documentation for any comments that are added in the source code.

2) Differentiate between JDK, JVM, and JRE.

Q.2) Differentiate between JDK, JVM & JRE

Ans:-

1) JVM (Java Virtual Machine)
Provides Runtime environment in which java bytecode can be executed
Task of JVM - loads code
verifies code
Execute code
Provide Runtime environment
- Jvm is platform dependent ie for each software and hardware we have different Jvm configuration
- Jvm does not exists physically. It is an environment which is abstract in natwre.

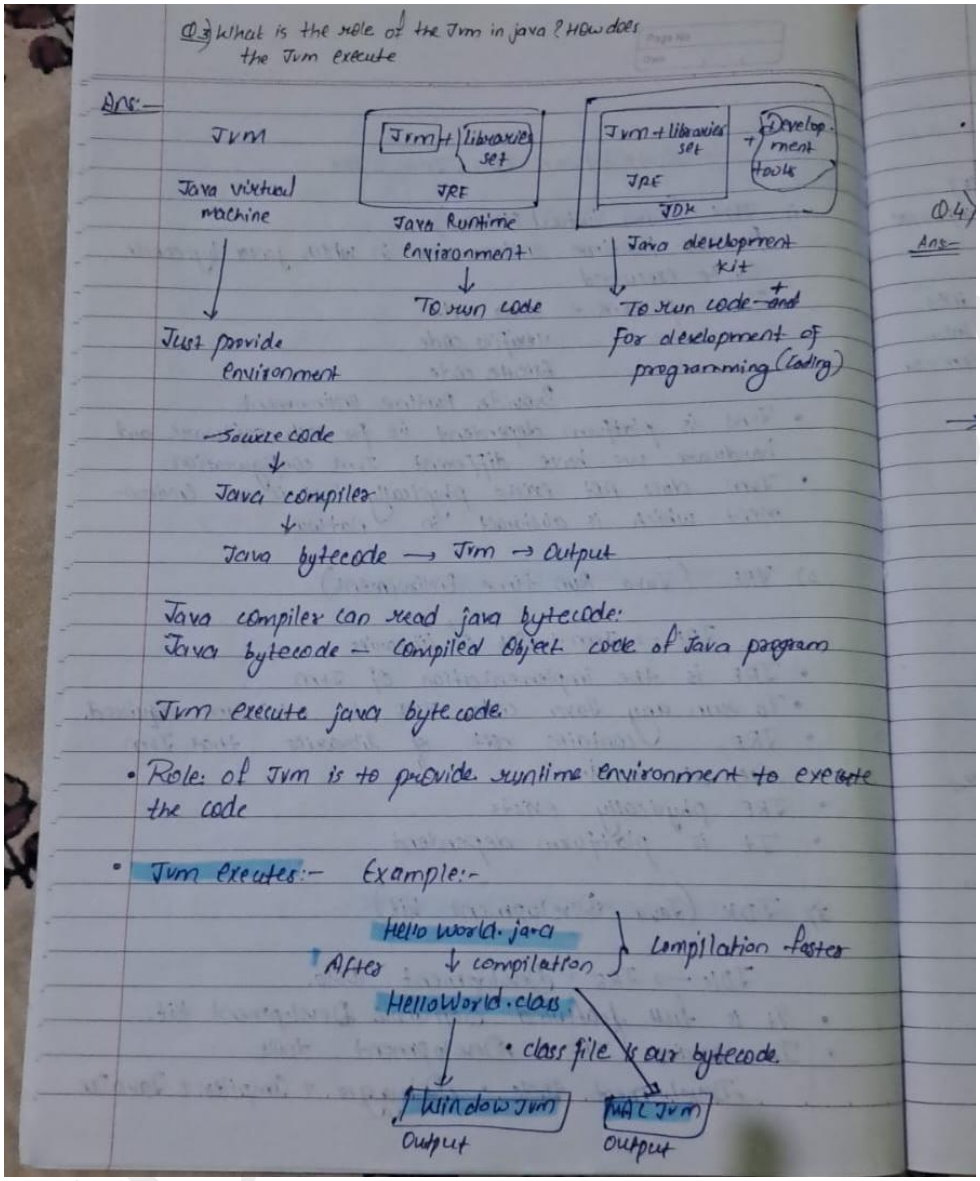
2) JRE (Java Run time Environment)

JRE → Jvm + Set of libraries
- JRE is the implementation of Jvm
- To run any Java code. JRE is minimum required.
- JRE contains set of libraries that Jvm uses at runtime
- JRE physically exists
- It is platform dependent

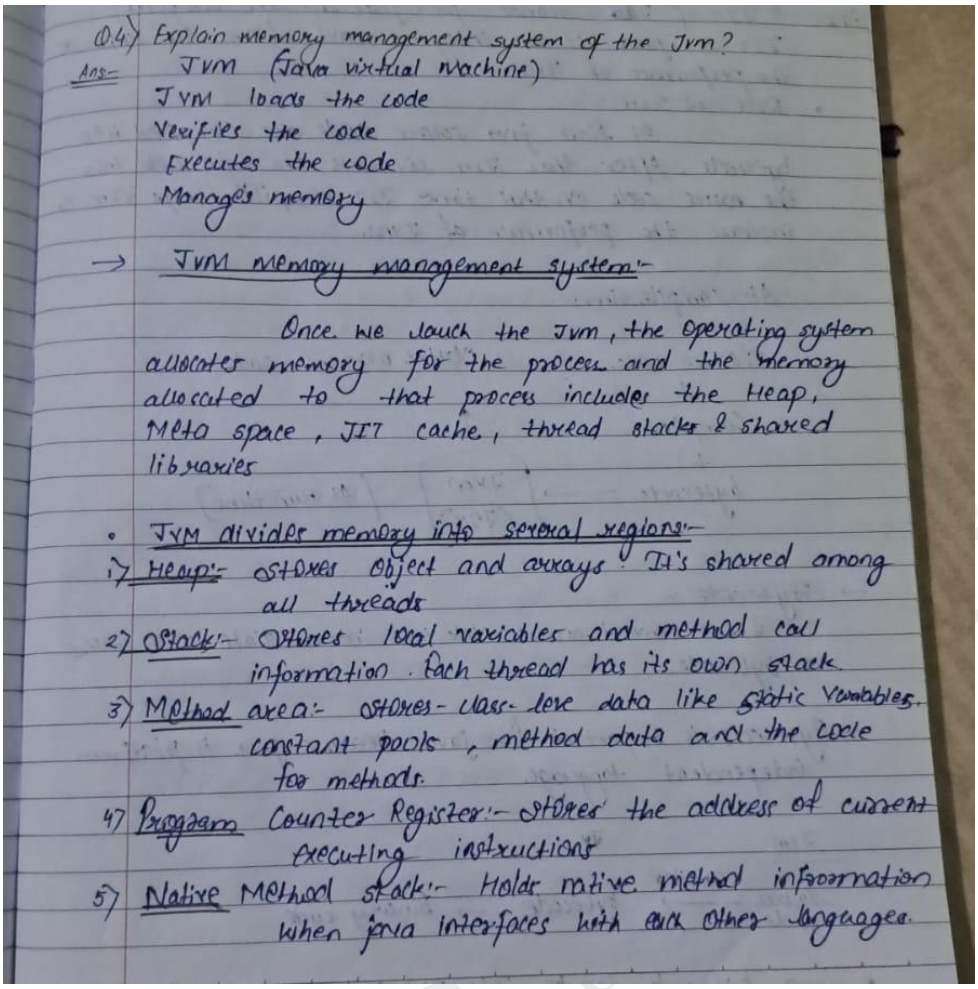
3) JDK (Java Development kit)

JDK → JRE + Development Tools.
- It is full featured software Development kit.
- It contains JRE + Development tools
Development tools • Debugger + Compiler + JavaDoc

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



4) Explain the memory management system of the JVM.



Q.4) Explain memory management system of the Jvm?

Ans- JVM (Java virtual machine)

JVM loads the code
Verifies the code
Executes the code
Manages memory

→ JVM memory management system:-

Once we launch the JVM, the operating system allocates memory for the process and the memory allocated to that process includes the Heap, Meta space, JIT cache, thread stacks & shared libraries

- JVM divides memory into several regions:-

1) Heap:- stores Object and arrays. It's shared among all threads
2) Stack:- Stores local variables and method call information. Each thread has its own stack.
3) Method area:- stores class level data like static variables, constant pools, method data and the code for methods.
4) Program Counter Register:- stores the address of current executing instructions
5) Native Method stack:- Holds native method information when java interfaces with and other languages.

5) What are the JIT compiler and its role in the JVM? What is the bytecode and why is it important for Java?

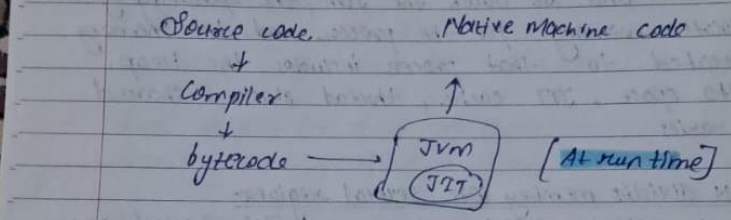
Q.5) What are JIT compiler and its role in the Jvm? What is the bytecode and why it is important for java?

Ans:- JIT (JIT stands for Just-In-Time Compiler)
JIT is java is an integral part of Jvm. It improves the performace of Jvm.

- **Role of Jvm:-**

At first java source code are compiled into bytecode. After that Jvm converts the byte code into the native code on that time JIT compiler helps Jvm to increase the performace of Jvm.

At compile time:-



→ **Bytecode** :-
Jvm convert source code into intermediate code called as bytecode.

Bytecode is important for java because java is platform independent language.

Jvm

Source code → compiler → bytecode → Binary code

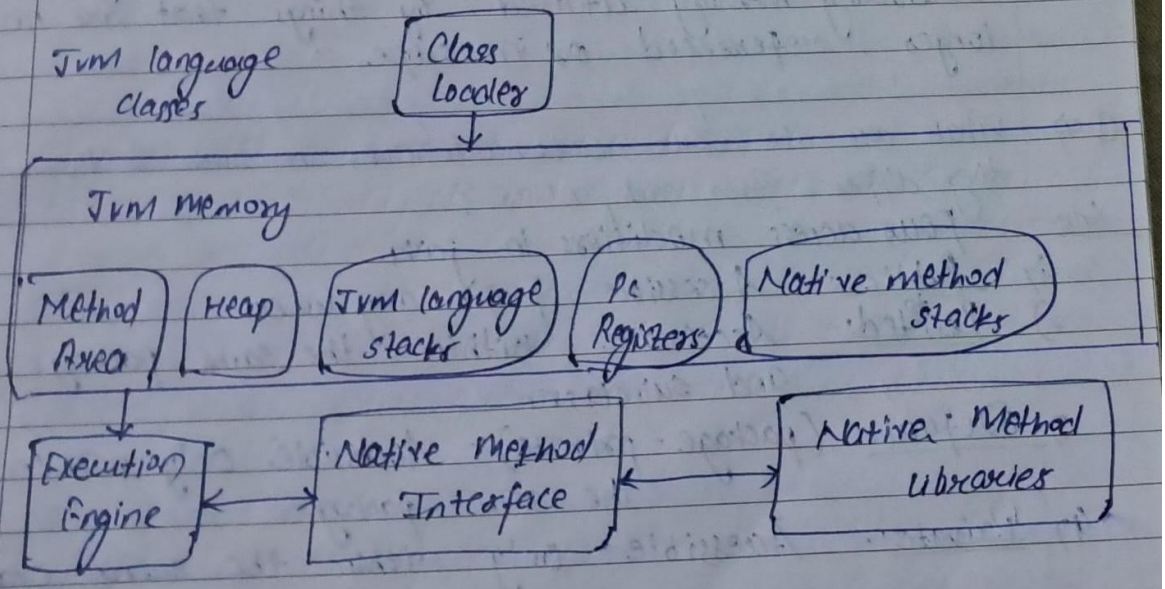
- Java bytecode can be interpreted on any system that provides Jvm

6) Describe the architecture of the JVM.

Q.6) Describe the architecture of JVM.

Ans:- JVM (Java Virtual machine) runs Java applications as a run-time engine. JVM is the one that calls the main method present in Java code.

When we compile a .java file, .class file (contains bytes code with the same name present in .java file are generated by the java compiler. This class file goes into various steps when we run it. These steps together describe JVM.



Class loader subsystem:- loading, linking, Initialization

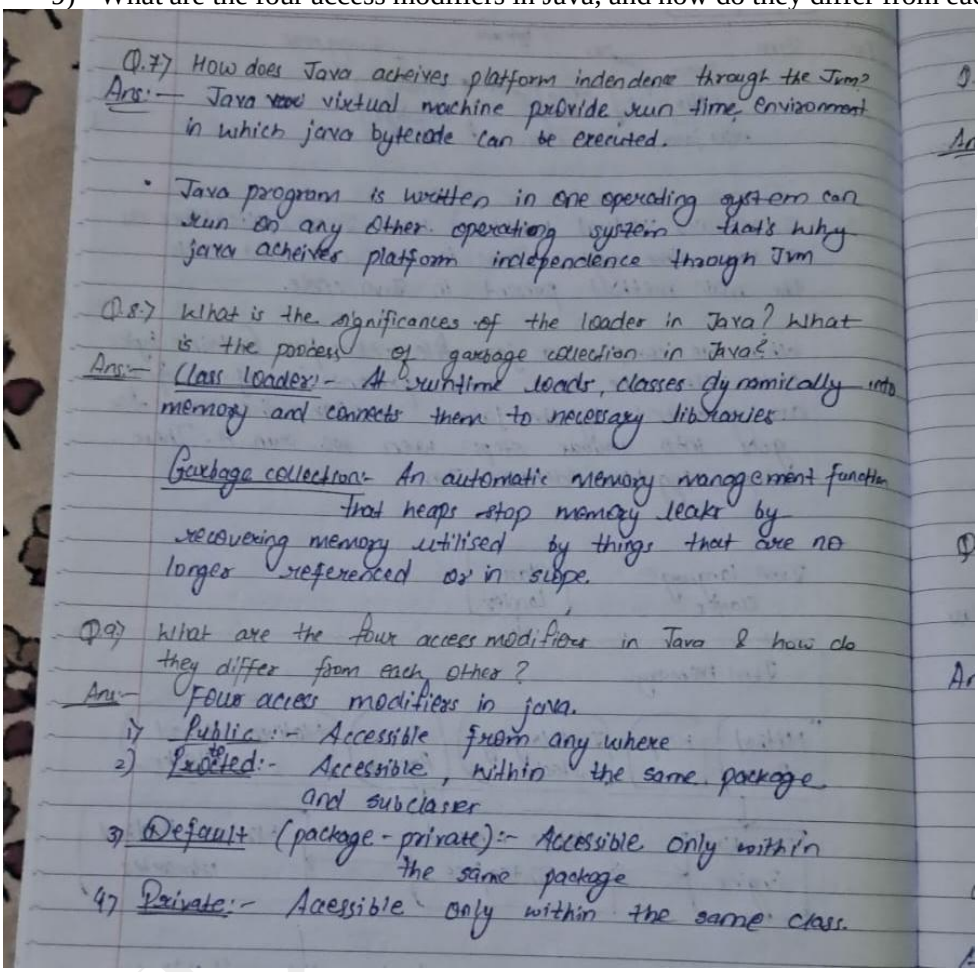
6) JVM architecture includes:

1. Class loader:- loads class file into memory
2. Memory area:- Divides memory into regions like heap, stacks, method area etc.
3. Execution Engine:- Executes the bytecode, includes the interpreter, JIT compiler, & garbage collector.
4. Native Interface :- Interacts with native libraries
5. Native Method libraries:- Libraries written in other languages (eg C or C++)

7) How does Java achieve platform independence through the JVM?

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

9) What are the four access modifiers in Java, and how do they differ from each other?



Q.7) How does Java acheives platform indendence through the Jvm?
Ans:— Java virtual machine provide run time environment in which java bytecode can be executed.

- Java program is written in one operating system can run on any other operating system that's why java acheives platform independence through Jvm

Q.8) What is the significances of the loader in Java? What is the process of garbage collection in Java?
Ans:— Class loader:- At runtime loads, classes dynamically into memory and connects them to necessary libraries

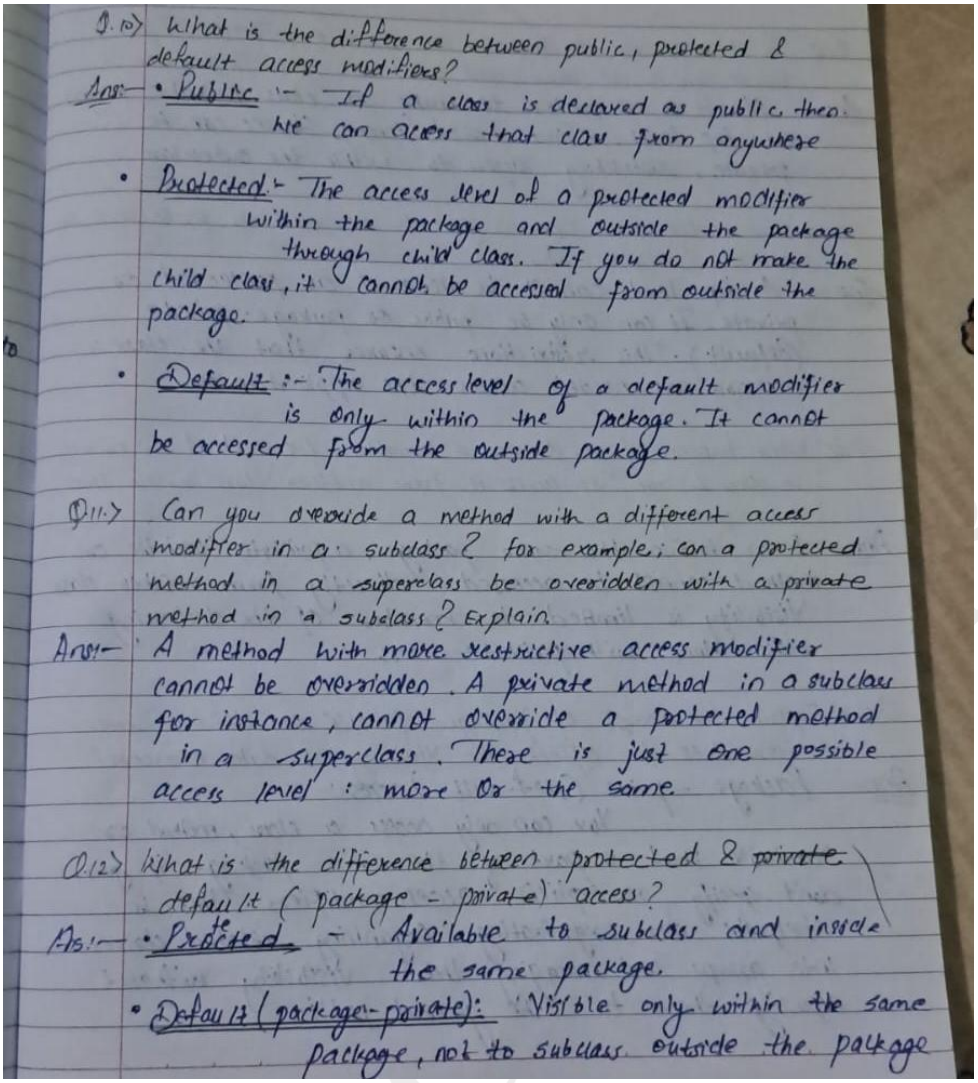
Garbage collection:- An automatic memory management functio that heaps stop memory leaks by recovering memory utilised by things that are no longer referenced or in scope.

Q9) What are the four access modifiers in Java & how do they differ from each other?
Ans:— Four access modifiers in java.
1) Public :- Accessible from any where
2) Protected:- Accessible, within the same package and subclasses
3) Default (package-private):- Accessible only within the same package
4) Private:- Acessible only within the same class.

10) What is the difference between public, protected, and default access modifiers?

11) Can you override a method with a different access modifier in a subclass? For example, can a protected method in a superclass be overridden with a private method in a subclass? Explain.

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



Q.10) What is the difference between public, protected & default access modifiers?

Ans:- • Public :- If a class is declared as public, then we can access that class from anywhere

• Protected :- The access level of a protected modifier within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.

• Default :- The access level of a default modifier is only within the package. It cannot be accessed from the outside package.

Q.11) Can you override a method with a different access modifier in a subclass? for example, can a protected method in a superclass be overridden with a private method in a subclass? Explain.

Ans:- A method with more restrictive access modifier cannot be overridden. A private method in a subclass for instance, cannot override a protected method in a superclass. There is just one possible access level : more or the same.

Q.12) What is the difference between protected & private default (package - private) access?

Ans:- • Protected - Available to subclass and inside the same package.

• Default (package-private): Visible only within the same package, not to subclass outside the package

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

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

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

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

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

Ans:- No it is not possible to make a class private in java at top level. Since nested (inner) classes can be private, restricting access to within the outerclass

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

Ans:- A top level cannot be declared as protected or private. It can only be public or package-private (default). This restrictions ensures that the class is accessible as needed across the application.

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

Ans:- Even within the same package, a private variable or method cannot be accessed directly from another class Visibility is limited to member of the declaring class alone with private access

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

Ans:- Package - prive (default) access:-

You can only access a class, method or variable that is part of the same package if you don't specify an explicit access modifiers. This is helpful for organizing organising related classes into groups and managing their visibility without exposing them to the general public.

6)
1.
2.
3.
4.
5.