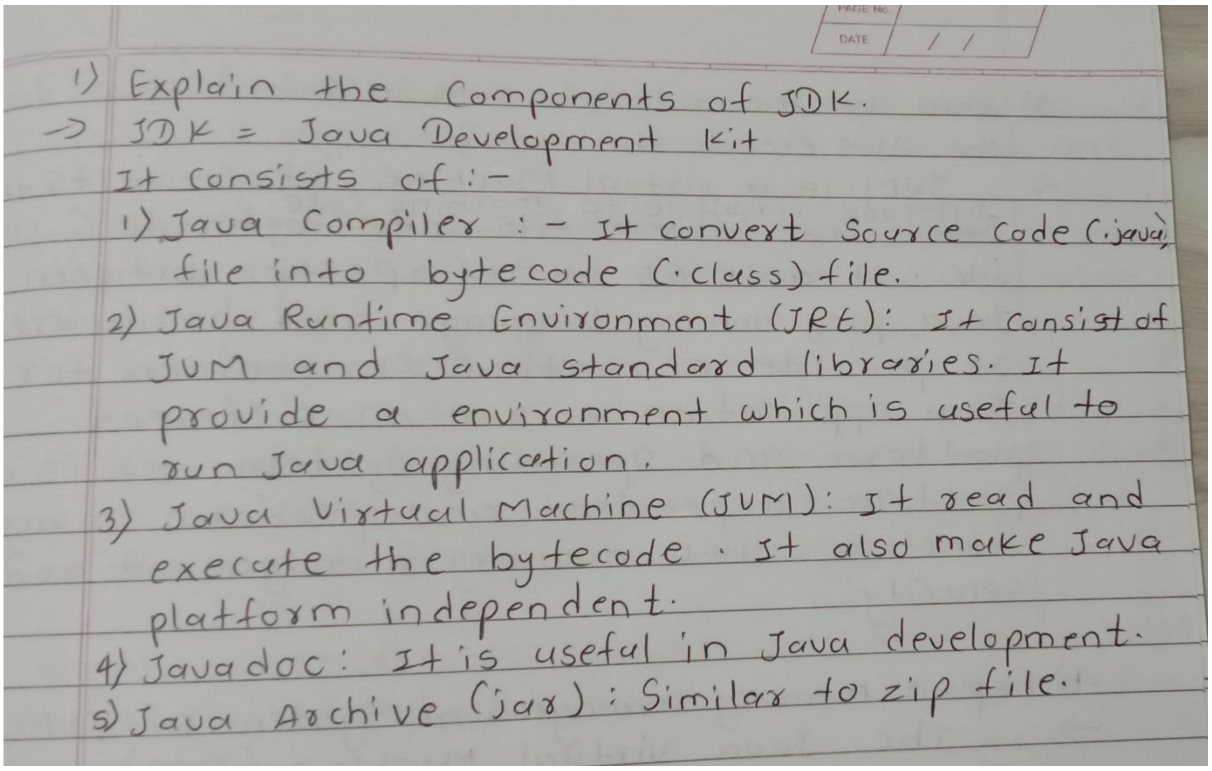
Assignment 3



PAGE NO.
DATE / /

1) Explain the Components of JDK.

→ JDK = Java Development Kit

It consists of :-

1) Java Compiler :- It convert Source code (.java) file into bytecode (.class) file.

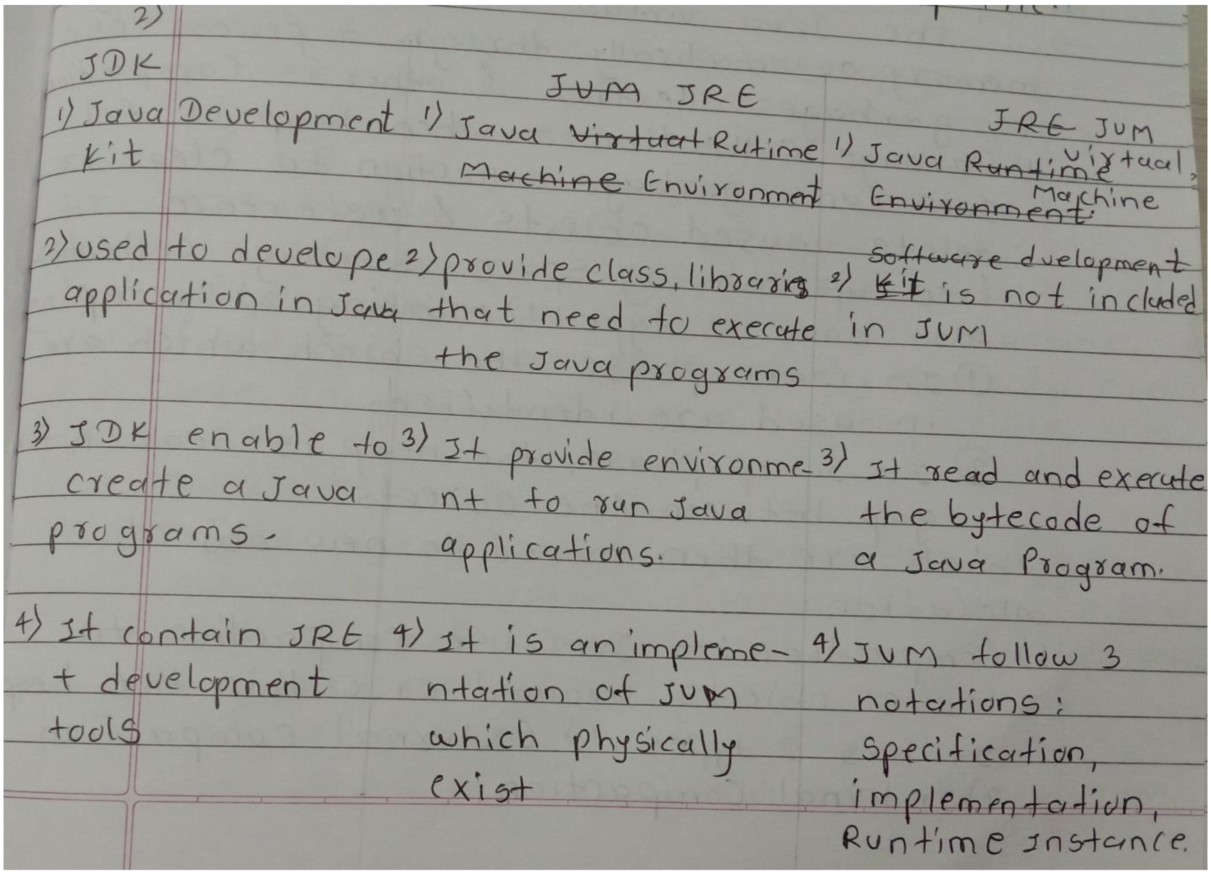
2) Java Runtime Environment (JRE): It consist of JVM and Java standard libraries. It provide a environment which is useful to run Java application.

3) Java Virtual Machine (JVM): It read and execute the bytecode. It also make Java platform independent.

4) Javadoc: It is useful in Java development.

5) Java Archive (jar): Similar to zip file.

2) Differentiate between JDK, JVM, and JRE.



2)

JDK	~~JVM~~ JRE	~~JRE~~ JVM
1) Java Development Kit	1) Java ~~Virtual~~ Rutime ~~Machine~~ Environment	1) Java ~~Runtime~~ Virtual, ~~Environment~~ Machine
2) used to develope application in Java	2) provide class, librariès that need to execute the Java programs	2) ~~Software dvelopment~~ ~~kit~~ is not included in JVM
3) JDK enable to create a Java programs.	3) It provide environment to run Java applications.	3) It read and execute the bytecode of a Java Program.
4) It contain JRE + development tools	4) It is an implementation of JVM which physically exist	4) JVM follow 3 notations: Specification, implementation, Runtime Instance.

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

-> JVM is a virtual Machine. It converts the
bytecode (.class) file to machine code. ~~Source code (.java) file into bytecode (.class)~~
file. It acts as an interpreter between
Java programming language and hardware.
It provide a runtime environmen for
Java application to run on different
platform and operating system. It is also
platform dependent and perform many
functions like memory management and
security.

4) Explain memory management system of JVM.

-> The Java virtual Machine (JVM) manges
memory automatically through a process: one
is garbage collection & other is Compaction

1) Garbage Collection :- when heap is full,
JVM runs garbage collection to clear or
delete unused objects & reference. It
free ~~up~~ memory space. Garbage collection
also have 2 type: 1) Marking 2) Sweeping

① In mark phase, the objech which are
in used are identified.

② In sweep phase, the heap is search for
gap between the objects.

which are then moved to new object
allocation.

2) Compaction :- JVM use compaction to
move objects around or within the heap
It has 2 types 1) External compaction
2) Internal Compaction,

PAGE NO.
DATE / /

① In External compaction, objects are moved out of compaction area & down to the heap.
② In Internal compaction, objects are move to close together within the compaction area.

JVM also used stack & heap
- Stack used for local variable & method parameters. Stack grow or shrinks when code block enters or exists automatically.
- Heap used for dynamic memory.

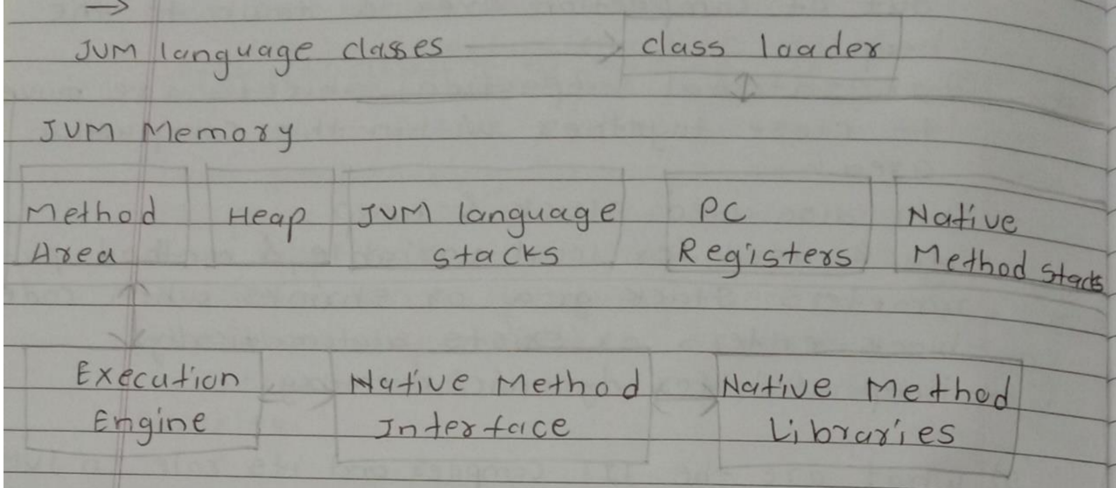
5) What are the JIT compiler and its role in JVM? What is bytecode and why is it important for Java?

→ Just-In-Time Compiler improves the performance of Java program by compiling bytecode into machine code.

JIT Compiler
In converts bytecode into machine code at run time, which is specific to machine hardware. It allows same bytecode to execute on any platform with compatible JVM.

Bytecode
It is a low level representation of code. It use virtual machine instead of CPU. It is a compact form of data where each instruction is represented by a single byte. It is portable & secure. It can run on any platform with compatible/appropriate JVM.

6) Describe architecture of JVM.
→



1) class loader Subsystem
Loading :- It loads .class file into Memory which is generated by compiler after compiling .Java file.
Linking :- This involves verifying bytecode, preparing static variable and assigning them to a default value.
Initialization :- Static variables are assigned their correct values & Static block are executed.

2) Runtime Data Areas
Method Area :- Stores class level data like class structure, method data, Constant pool, field data and method bytecode.
Heap :- objects are allocated to heap. used for dynamic memory.
Stack :- used for local variable & method parameter

PC (Programme Counter) Registers:- Each thread has its own pc register that hold address of currently executing JVM instruction.
Native Method Stack:- used for native method execution, when native method are written in languages like c, c++ & Java.

3) Execution Engine:-
Interpreter:- Read and execute byte code instructions one by one. It is simple but slow because it does not optimize bytecode.
Just-In-Time (JIT):- Convert bytecode to native machine code at run time, optimizes performance. Once compiled, native code is executed directly, it speeds up execution.
Garbage Collector-: when heap is full, JVM run Garbage collector, it clear ~~use~~ unused object & references. & free up the space.

4) Native Method Interface:- It allows JVM to interact with native applications which are written in c, c++. It provide necessary framework to execute native method.

5) Native Method Libraries:- These libraries are actual native code implementation that are referenced via the JNI.

PAGE No.
DATE / /

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

→ The Java compiler convert the program into bytecode.

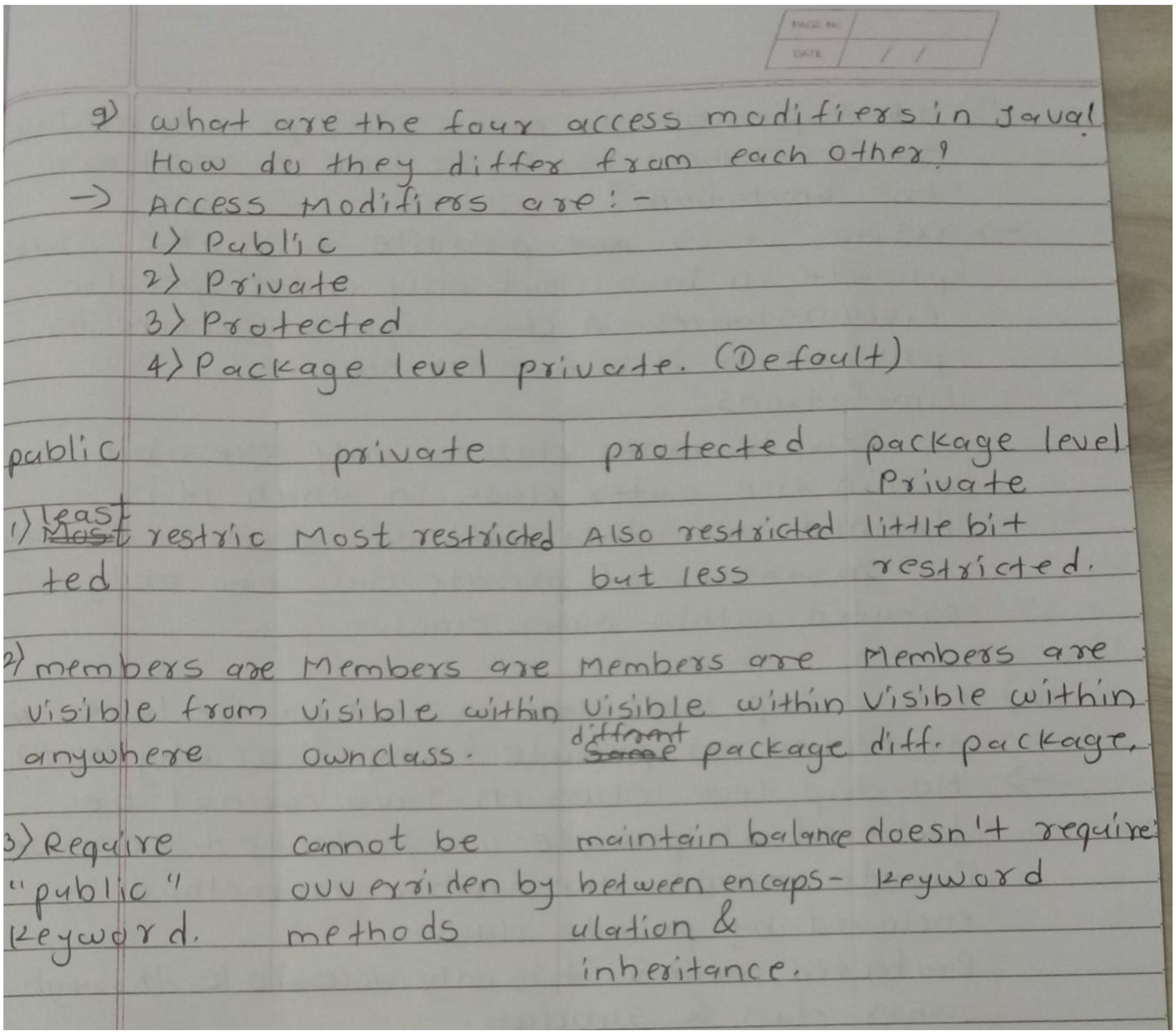
JVM convert bytecode to machine code for operating system. JVM recognize platform & convert byte code to machine code that can be read & executed.

This allows Java application to run on any device with JVM, regardless of hardware configuration and operating system.

8) what is significance of class loader in Java? what is process of garbage collector in Java?

→ The class loader is a tool that Java uses to load different classess. into the memory of our computer when Java program is running. It is like a librarian who help computer to find & read the right book or classes that the program need to use.

Garbage collection have a ability to automatically manage memory, prevent memory leak, enable dynamic memory allocation, improve performance & optimize memory usage. Garbage collector can help developer to write better, more efficient programs.



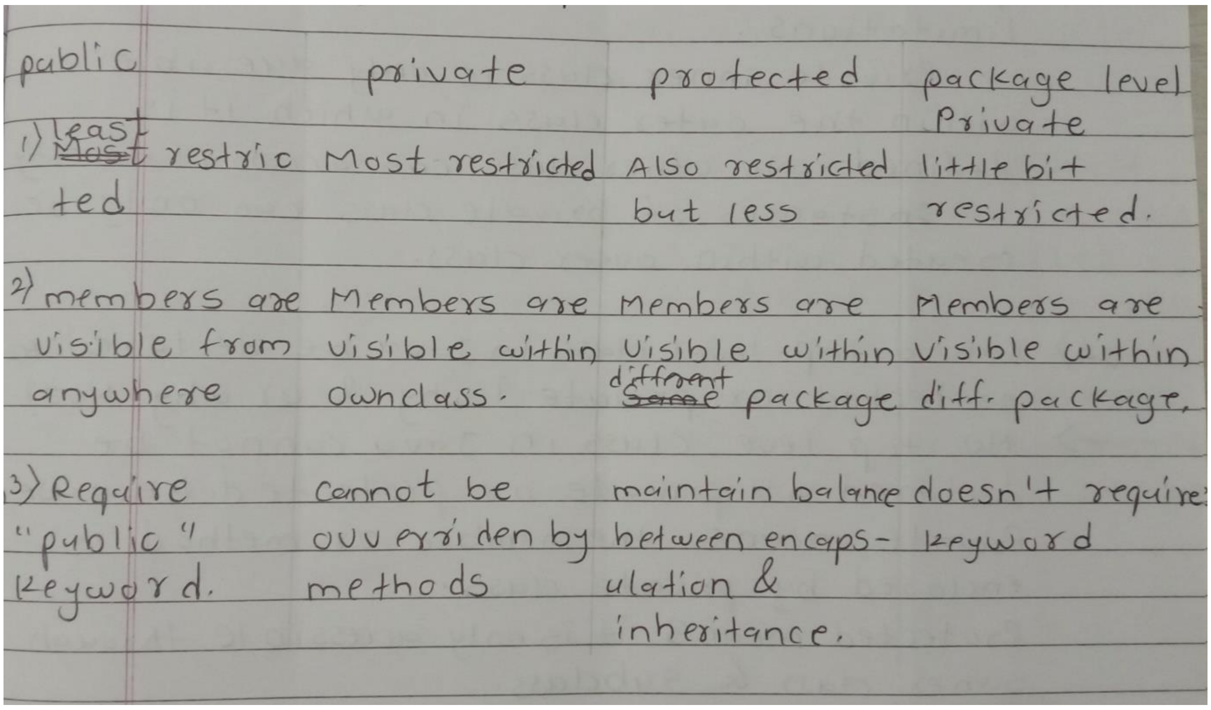
9) What are the four access modifiers in Java? How do they differ from each other?

→ Access modifiers are:-

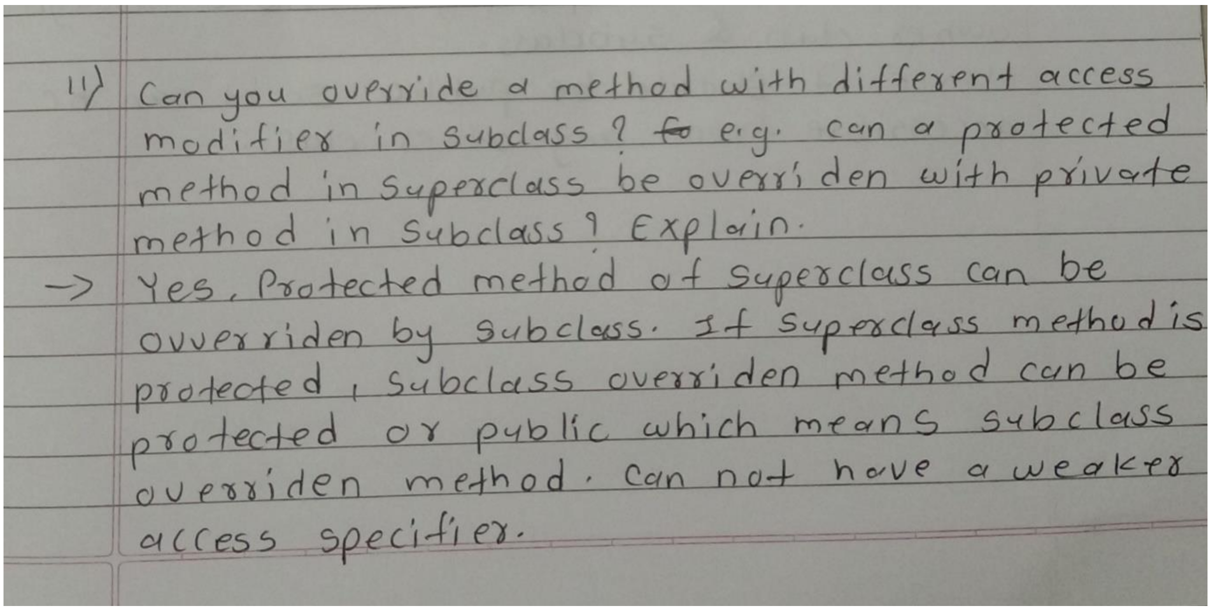
1) Public
2) Private
3) Protected
4) Package level private. (Default)

public	private	protected	package level Private
1) least ~~Most~~ restricted	Most restricted	Also restricted but less	little bit restricted.
2) members are visible from anywhere	Members are visible within ownclass.	Members are visible within different ~~Same~~ package	Members are visible within diff. package.
3) Require "public" keyword.	cannot be ouverriden by methods	maintain balance between encaps-ulation & inheritance.	doesn't require keyword

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



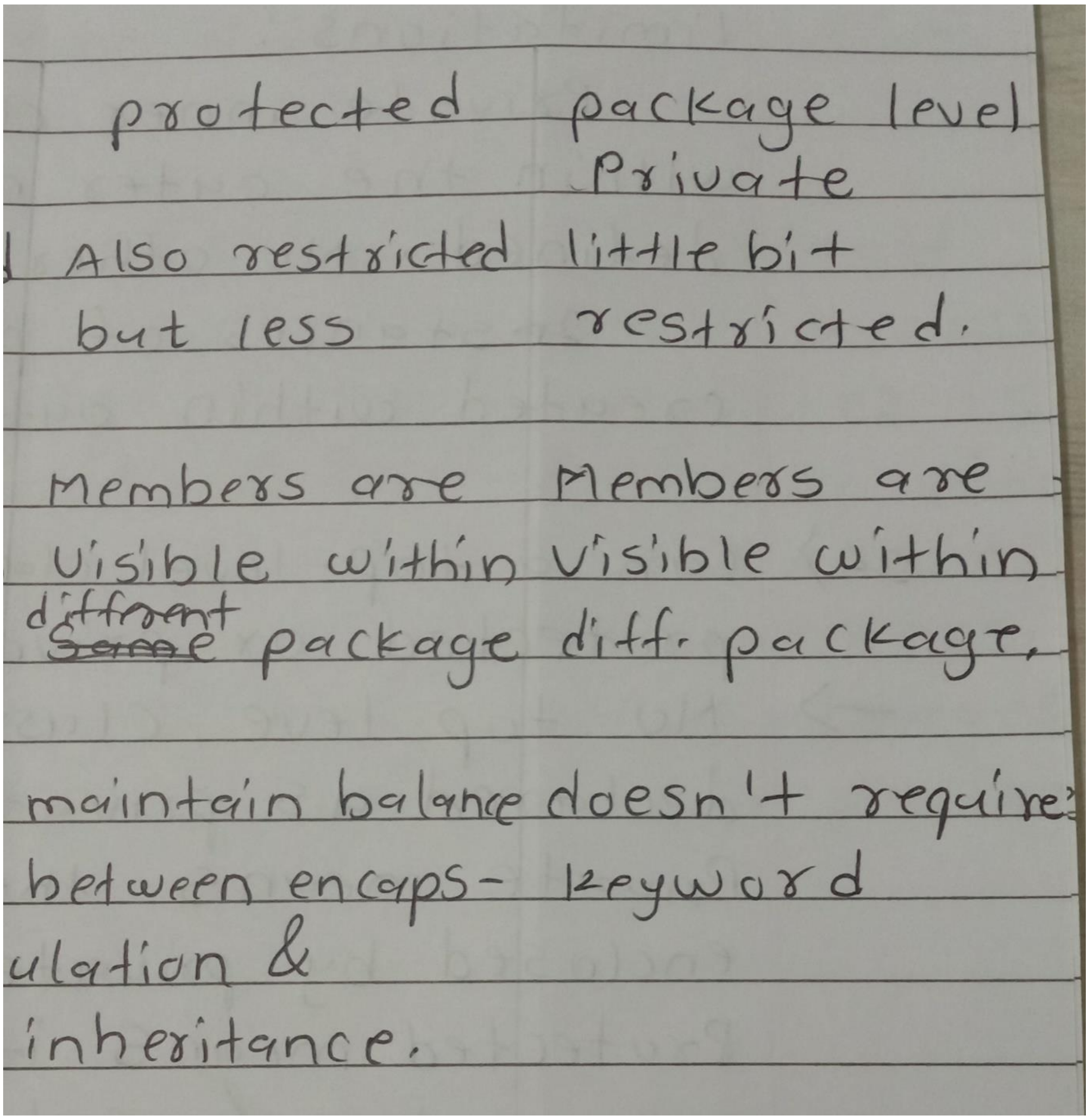
public	private	protected	package level Private
1) least ~~Most~~ restricted	Most restricted	Also restricted but less	little bit restricted.
2) members are visible from anywhere	Members are visible within ownclass.	Members are visible within different ~~Same~~ package	Members are visible within diff. package.
3) Require "public" keyword.	cannot be ouverriden by methods	maintain balance between encaps-ulation & inheritance.	doesn't require keyword



11) Can you override a method with different access modifier in subclass ? ~~fo~~ e.g. can a protected method in superclass be overriden with private method in subclass ? Explain.

→ Yes, Protected method of superclass can be ovverriden by subclass. If superclass method is protected, subclass overriden method can be protected or public which means subclass overriden method. Can not have a weaker access specifier.

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



protected	package level Private
Also restricted but less	little bit restricted.
Members are visible within ~~Same~~ diffrent package	Members are visible within diff. package.
maintain balance between encaps-ulation & inheritance.	doesn't require keyword

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

→ Yes~~No~~, it is ~~not~~ possible to make class private in Java. but only under specific circumstances. A class can be declare as private if it is a inner class.

Limitations:

Private inner class is only accessible within the outer class in which it is defined. No other class can access it directly.

Instances of private class can only be created within outer class.

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

→ No top leve class in Java cannot be declared as private or protected. Because, Private means the member or method is enclosed by private class.
Protected means it is only accessible through owner class & subclass.
Hence it should be public so it can be accessible from any other class.

15) If we declare a variable or method as private in a class & then ~~to~~ access it from another class within same package we will get a compilation error. the private access modifier restrict access to variable or method strictly to class in which it is declared means it cannot be accessed from any other class even if the ~~pa~~ class is in the same package.

16) Package - Private is also known as default access. it is access level assigned to a class member when no explicit access modifier is specified. In Java, it is accessible only within the same package. Any ~~package~~ class in a same package can access these members. but not visible to class in other packages It allows for encapsulation within a package