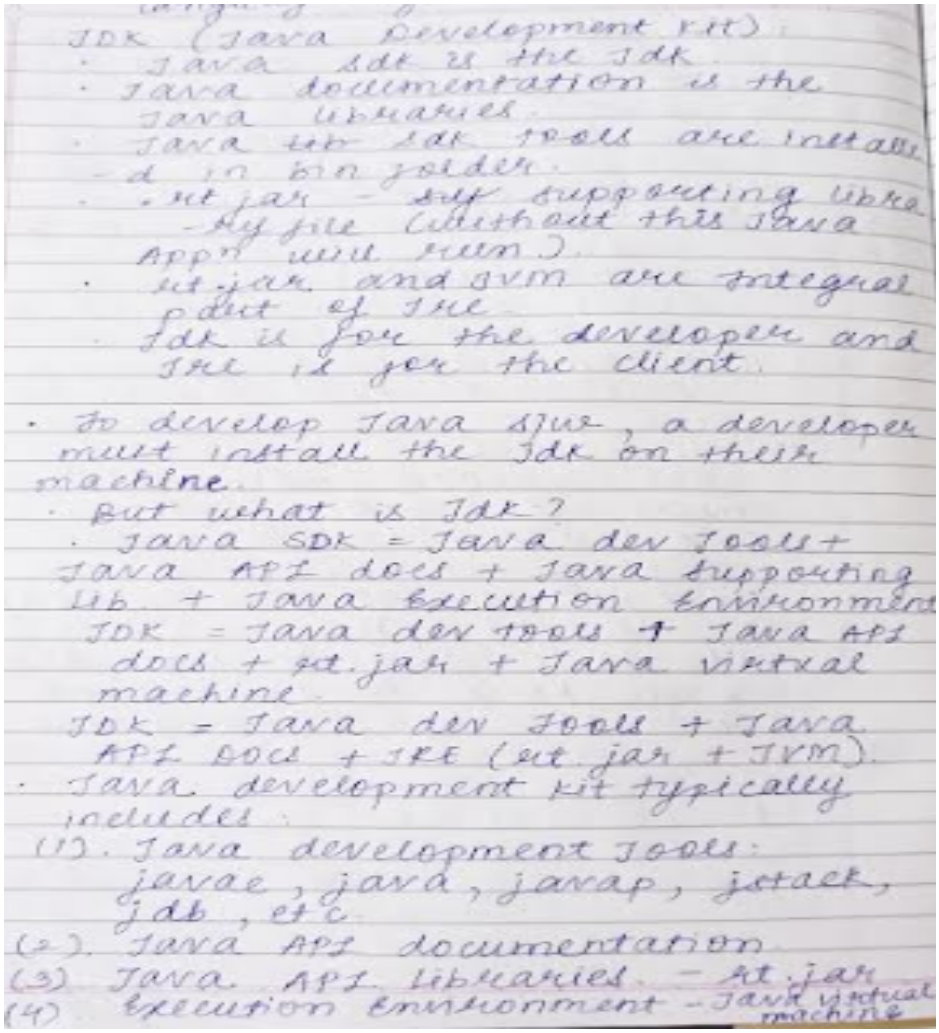
Name: Prashansa Nalawade

Assignment 3: Interview questions

1) Explain the components of the JDK.


JDK (Java Development Kit).
- Java sdk is the JDK.
- Java documentation is the Java libraries.
- Java lib sdk tools are installed in bin folder.
- rt.jar - all supporting libraries file (without this Java Appn will not run).
- rt.jar and JVM are integral part of JRE.
- Sdk is for the developer and JRE is for the client.

- To develop Java pgm, a developer must install the Jdk on their machine.
- But what is JDK?
- Java SDK = Java dev tools + Java API docs + Java supporting lib + Java execution environment
JDK = Java dev tools + Java API docs + rt.jar + Java virtual machine.
JDK = Java dev tools + Java API Docs + JRE (rt.jar + JVM).
- Java development kit typically includes:
(1). Java development tools: javac, java, javap, jstack, jdb, etc.
(2). Java API documentation.
(3) Java API libraries. - rt.jar
(4) Execution environment - Java virtual machine

2) Differentiate between JDK, JVM, and JRE.

Components of JVM: (Java 7) are seperate

(1). Bootstrap class loader: used to load java API from rt.jar file.

(2). Extension class loader: used to load java API from ext directory. It loads .jar file from HD to JVM.

(3). System class loader: To load .class file into main memory.

(4). Custom class loader: A custom class loader can maintain a separate class-loader for each developer's servlet.

· Runtime data areas:

(1). Method area: It stores class data and other information related to class and their methods.

(2). Heap: It is used to store objects during execution

(3) Java stacks: It is a memory used for the execution of a thread. They contain method-specific values that are short lived and references to other objects in the heap that is getting referenced from

(5) Native method stacks: All the process of function activation is stored here.

(4). PC register: Each thread has its own PC register to hold the addr^n of the currently executing JVM execution. It's value

. Execution engine: get updated

(1). Interpreter: To execute the .java file or .class file, it depends on the JVM (Interpreter). It reads the byte code and interprets (converts) in the machine code (native code) and executes them in a sequential manner.

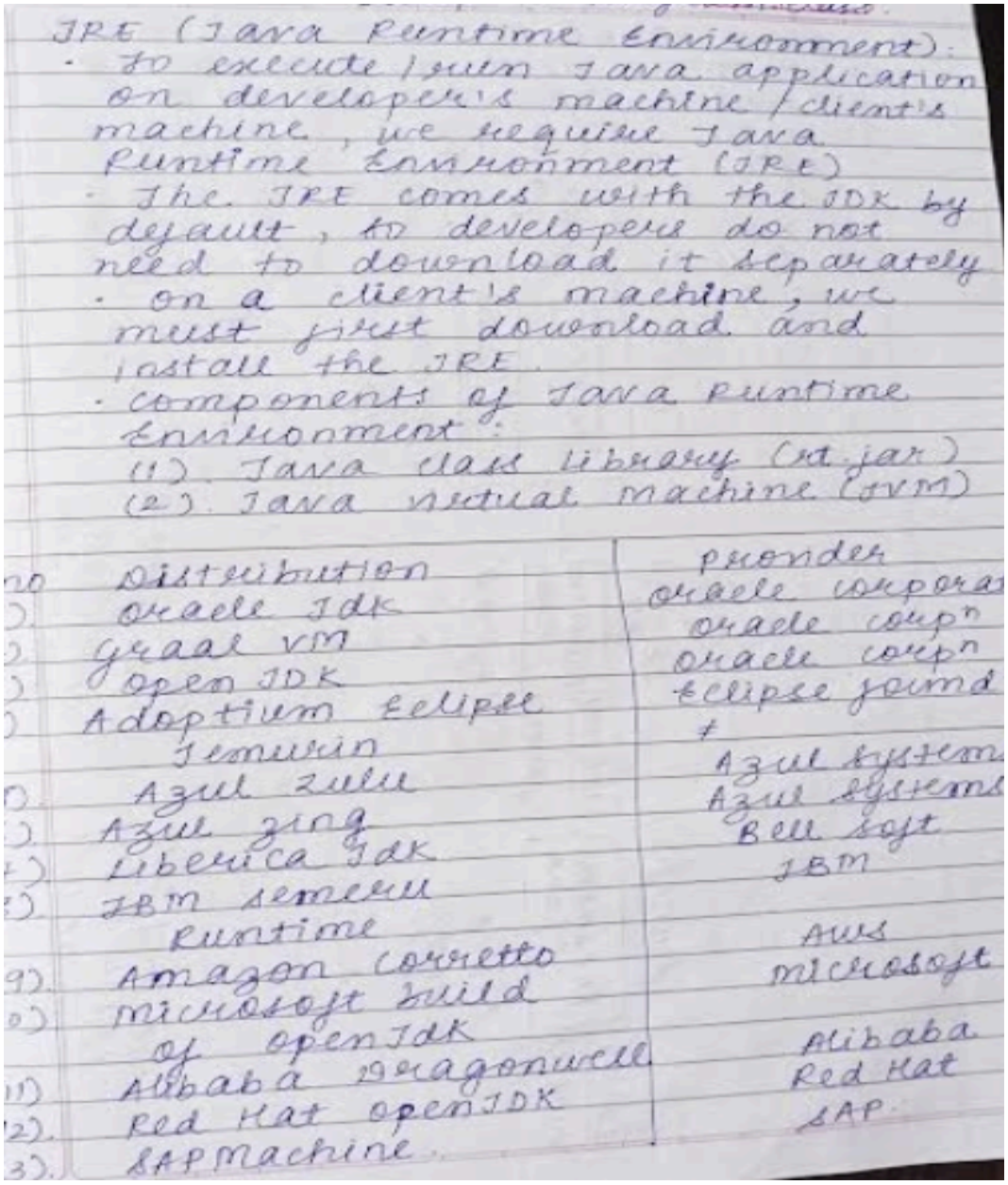
(2). Just In Time compiler: At runtime, the JVM loads the class files, the seman-tic of each is determined and appropriate computations are performed. The additional processor & memory usage during interpretation makes a Java application perform ~~so~~ slowly as compared to native application.

· The JIT compiler aids in improving the performance of Java programs by compiling bytecode into native machine code at runtime.

(3). Garbage collector: This is a program in Java that manages the memory automatically. It is a daemon thread which always runs in the background. This frees up the heap mem^n by destroying unreachable methods.

JDK (Java Development Kit).
- Java sdk is the JDK.
- Java documentation is the Java libraries.
- Java tools jar tools are installed in bin folder.
- rt.jar - std supporting libs (My file (without this Java App will run).
- rt.jar and JVM are integral part of JRE.
- JDK is for the developer and JRE is for the client.

- To develop Java apps, a developer must install the JDK on their machine.
- But what is JDK?
- Java SDK = Java dev tools + Java API docs + Java supporting lib + Java execution environment
JDK = Java dev tools + Java API docs + rt.jar + Java virtual machine
JDK = Java dev tools + Java API docs + JRE (rt.jar + JVM).
- Java development kit typically includes:
(1). Java development tools: javac, java, javap, jstack, jdb, etc.
(2). Java API documentation.
(3) Java API libraries. - rt.jar
(4) Execution environment - Java virtual machine



JRE (Java Runtime Environment):
- To execute/run Java application on developer's machine/client's machine, we require Java Runtime Environment (JRE)
- The JRE comes with the JDK by default, so developers do not need to download it separately
- on a client's machine, we must first download and install the JRE.
- components of Java Runtime Environment:
  (1). Java class library (.rt.jar)
  (2). Java virtual machine (JVM)

no	Distribution	provider
).	Oracle JDK	Oracle corporation
).	Graal VM	Oracle corp^n
).	Open JDK	Oracle corp^n
)	Adoptium Eclipse Temurin	Eclipse found +
).	Azul Zulu	Azul system
).	Azul Zing	Azul systems
).	Liberica JDK	Bell Soft
).	IBM semeru Runtime	IBM
9).	Amazon Corretto	AWS
0).	Microsoft build of openJDK	Microsoft
1).	Alibaba Dragonwell	Alibaba
2).	Red Hat openJDK	Red Hat
3).	SAP Machine.	SAP.

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

Components of JVM: (Java 7) are seperate

(1). Bootstrap class loader: used to load java API from rt.jar file.

(2). Extension class loader: used to load java API from ext directory. It loads .jar file from HD to JVM.

(3). System class loader: To load .class file into main memory.

(4). Custom class loader: A custom class loader can maintain a separate class-loader for each developer's servlet.

· Runtime data areas:

(1). Method area: It stores class data and other information related to class and their methods.

(2). Heap: It is used to store objects during execution of program

(3) Java stacks: It is a memory used for the execution of a thread. They contain method-specific values that are short lived and references to other objects in the heap that is getting referenced from

(5) Native method stacks: All the process of function activation is stored here.

(4). PC register: Each thread has its own PC register to hold the addr^n of the currently executing JVM execution. It's value

. Execution engine: get updated

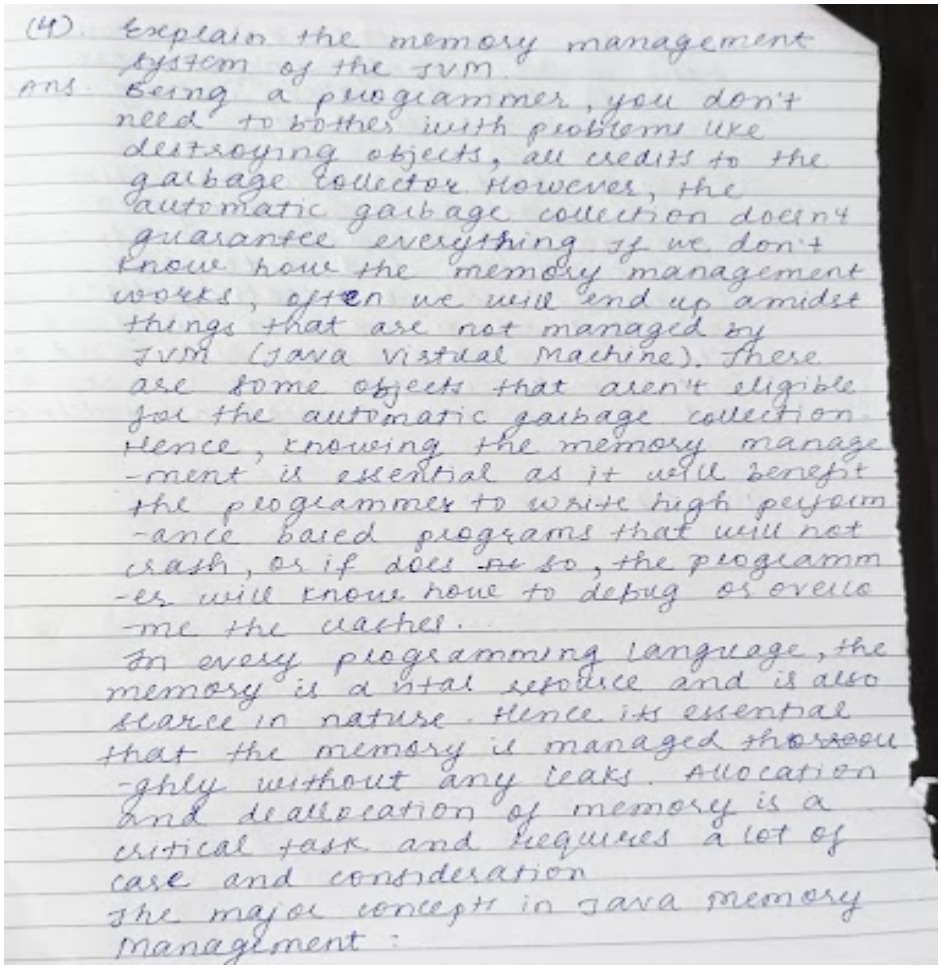
(1). Interpreter: To execute the .java file or .class file, it depends on the JVM (Interpreter). It reads the byte code and interprets (converts) in the machine code (native code) and executes them in a sequential manner.

(2). Just In Time compiler: At runtime, the JVM loads the class files, the seman-tic of each is determined and appropriate computations are performed. The additional processor & memory usage during interpretation makes a Java application perform ~~so~~ slowly as compared to native application.

· The JIT compiler aids in improving the performance of Java programs by compiling bytecode into native machine code at runtime.

(3). Garbage collector: This is a program in Java that manages the memory automatically. It is a daemon thread which always runs in the background. This frees up the heap mem^n by destroying unreachable methods.

4) Explain the memory management system of the JVM.



(4). Explain the memory management system of the JVM.

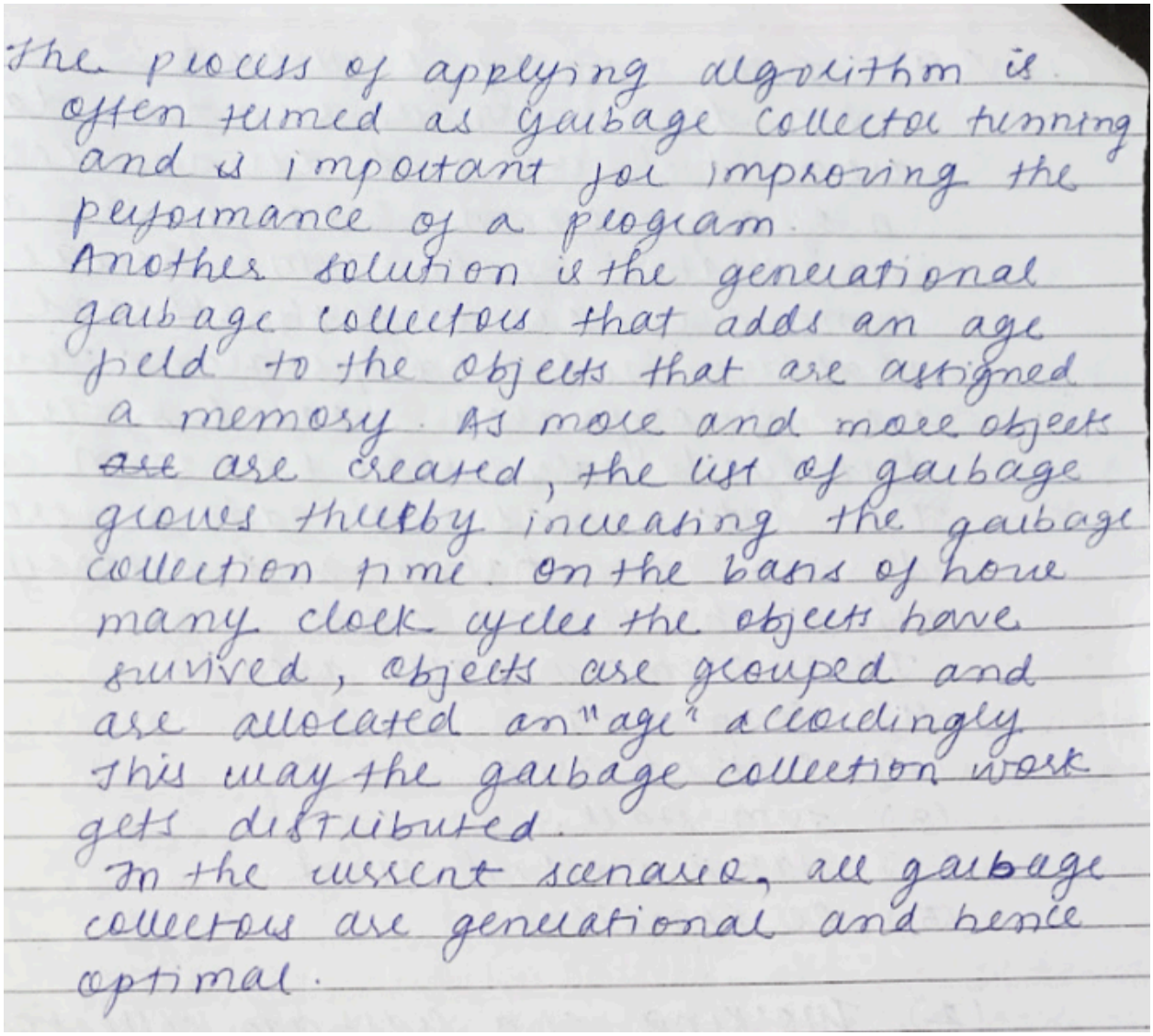
ans. Being a programmer, you don't need to bothers with problems like destroying objects, all credits to the garbage collector However, the automatic garbage collection doesn't guarantee everything if we don't know how the memory management works, often we will end up amidst things that are not managed by JVM (Java Virtual Machine). There are some objects that aren't eligible for the automatic garbage collection. Hence, knowing the memory manage-ment is essential as it will benefit the programmer to write high perform-ance based programs that will not crash, or if does ~~or~~ so, the programm-er will know how to debug or overlo-me the caches.

In every programming language, the memory is a vital resource and is also scarce in nature. Hence its essential that the memory is managed thoroughly without any leaks. Allocation and deallocation of memory is a critical task and requires a lot of care and consideration

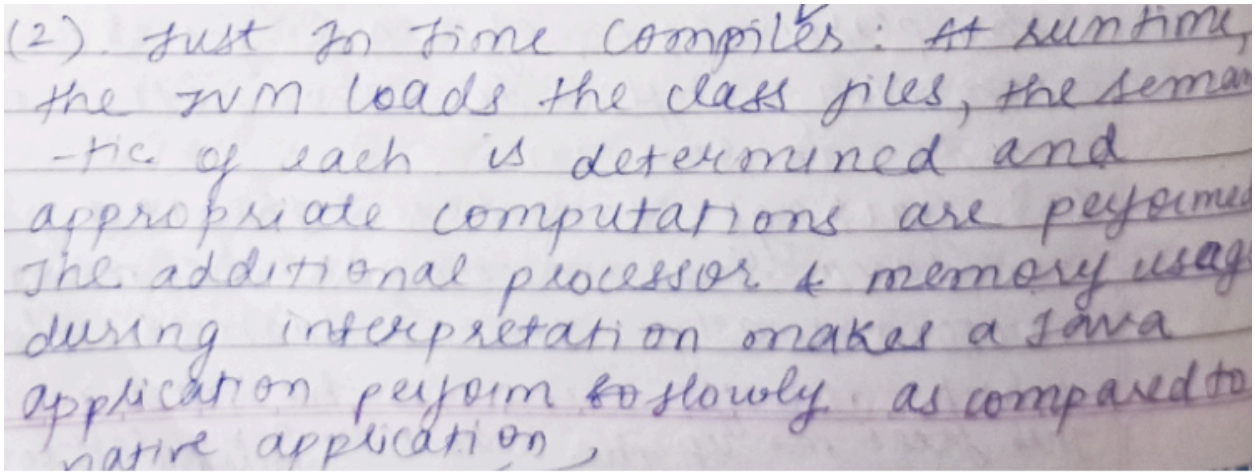
The major concepts in Java memory management :

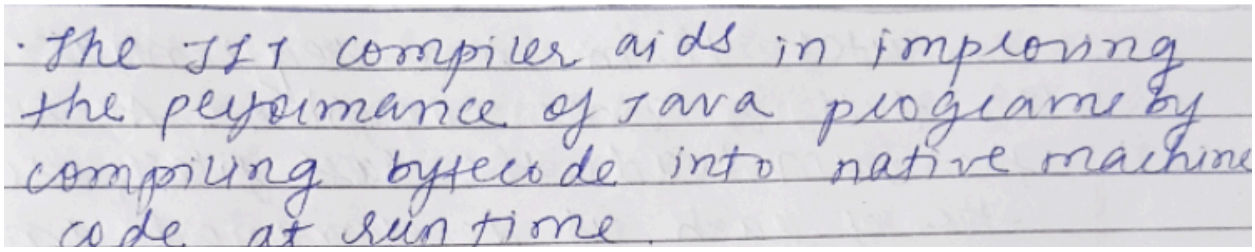
(1) JVM memory structure:
JVM defines various runtime data area which are used during execution of a program some of the areas are created by the JVM whereas some are created by the threads that are used in a program However, the memory area created by JVM is destroyed only when the JVM exits. The data areas of thread are created during instantiation and destroyed when the thread exits.
JVM memory parts are:
(1). Heap area
(2). Method area
(3). JVM stack
(4). Native method stack
(5). PC Registers

(2). Working of a garbage collector:
JVM triggers this process and as per the JVM garbage collection process is done or else withheld It reduces the burden of programmer by automatically performing the allocation or deallocation of memory.
Garbage collection process causes the rest of the processes or threads to be paused and thus is costly in nature This problem is unacceptable for the client but can be eliminated by applying several garbage collector based algorithms.


The process of applying algorithm is often turned as Garbage collector running and is important for improving the performance of a program.
Another solution is the generational garbage collectors that adds an age field to the objects that are assigned a memory. As more and more objects ~~are~~ are created, the list of garbage grows thusby increasing the garbage collection time. On the basis of how many clock cycles the objects have survived, objects are grouped and are allocated an "age" accordingly. This way the garbage collection work gets distributed.
In the current scenario, all garbage collectors are generational and hence optimal.

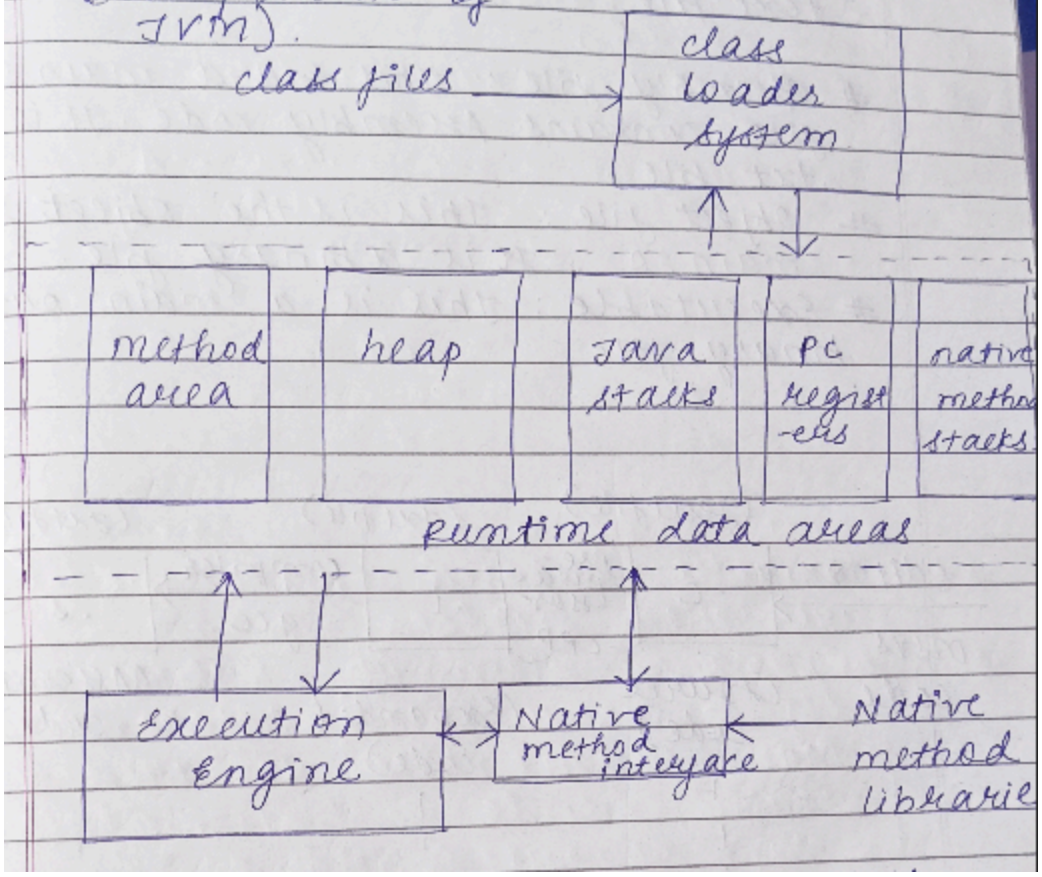
5) What are the JIT compiler and its role in the JVM? What is the bytecode and why is it important for Java?


(2) Just In Time compilers: At runtime, the JVM loads the class files, the semantic of each is determined and appropriate computations are performed. The additional processor & memory usage during interpretation makes a Java application perform slowly as compared to native application.


· The JIT compiler aids in improving the performance of Java programs by compiling bytecode into native machine code at runtime.

6) Describe the architecture of the JVM.

Overview of JVM Architecture:
Main components of JVM Architn are class loader subsystem, runti-me data areas and execution engine.
(Components of JVM).



Components of JVM: (Java + Seperate)

(1). Bootstrap class loader: used to load java API from rt.jar file.

(2). Extension class loader: used to load java API from ext directory. It loads .jar file from HD to JVM.

(3). System class loader: To load .class file into main memory.

(4). Custom class loader: A custom class loader can maintain a separate class-loader for each developer's servlet.

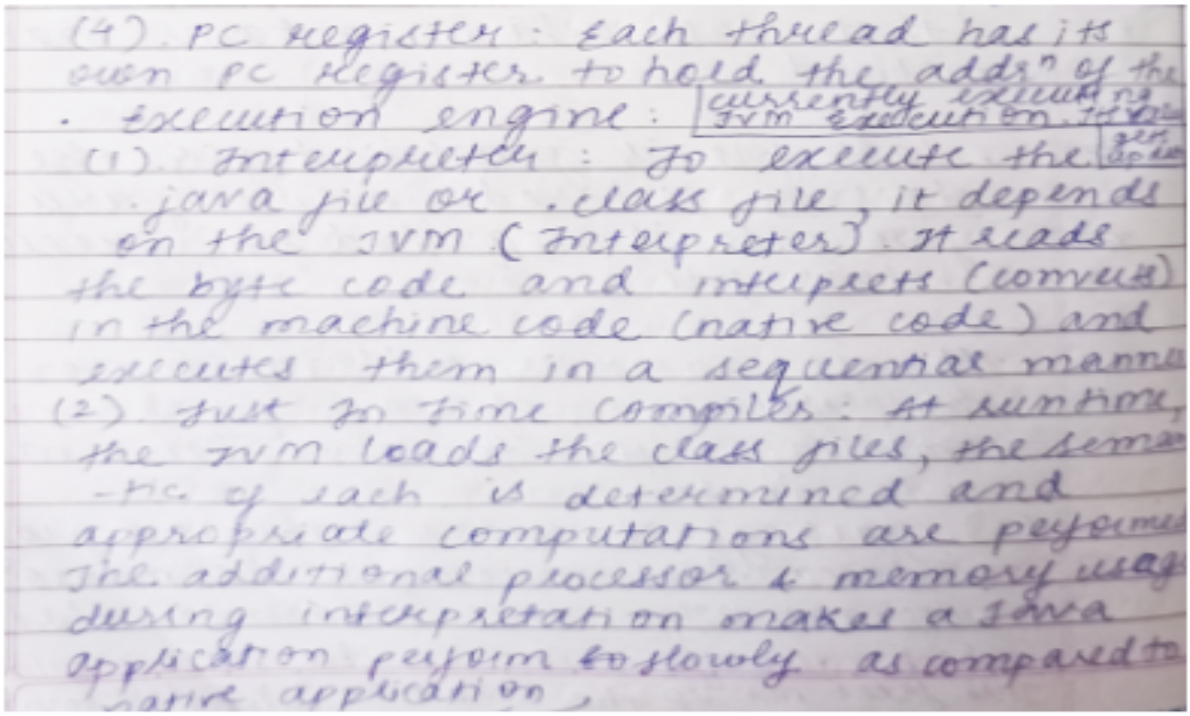
· Runtime data areas:

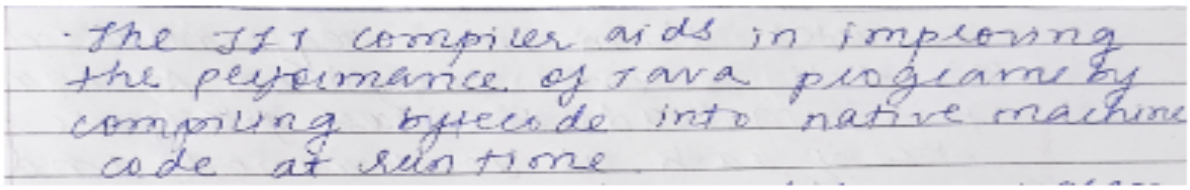
(1). Method area: It stores class data and other information related to class and their methods.

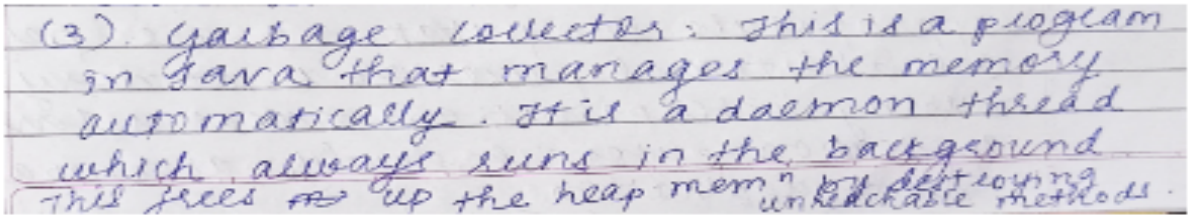
(2). Heap: It is used to store objects during execution.

(3) Java stacks: It is a memory used for the execution of a thread. They contain method-specific values that are short lived and references to other objects in the heap that is getting referenced from.

(5) Native method stacks: All the process of function activation is stored here.


(4). PC register : Each thread has its own PC register to hold the addr" of the currently executing JVM execution.
. Execution engine :
(1). Interpreter : To execute the .java file or .class file, it depends on the JVM (Interpreter). It reads the byte code and interprets (converts) in the machine code (native code) and executes them in a sequential manner
(2). Just In Time Compiler : At runtime, the JVM loads the class files, the semantics of each is determined and appropriate computations are performed. The additional processor & memory usage during interpretation makes a Java application perform too slowly as compared to native application.


· The JIT compiler aids in improving the performance of Java programs by compiling bytecode into native machine code at runtime.


(3). Garbage collector : This is a program in Java that manages the memory automatically. It is a daemon thread which always runs in the background. This frees up the heap mem" by destroying unreachable methods.

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

(Q). How does Java achieve platform independence through JVM?

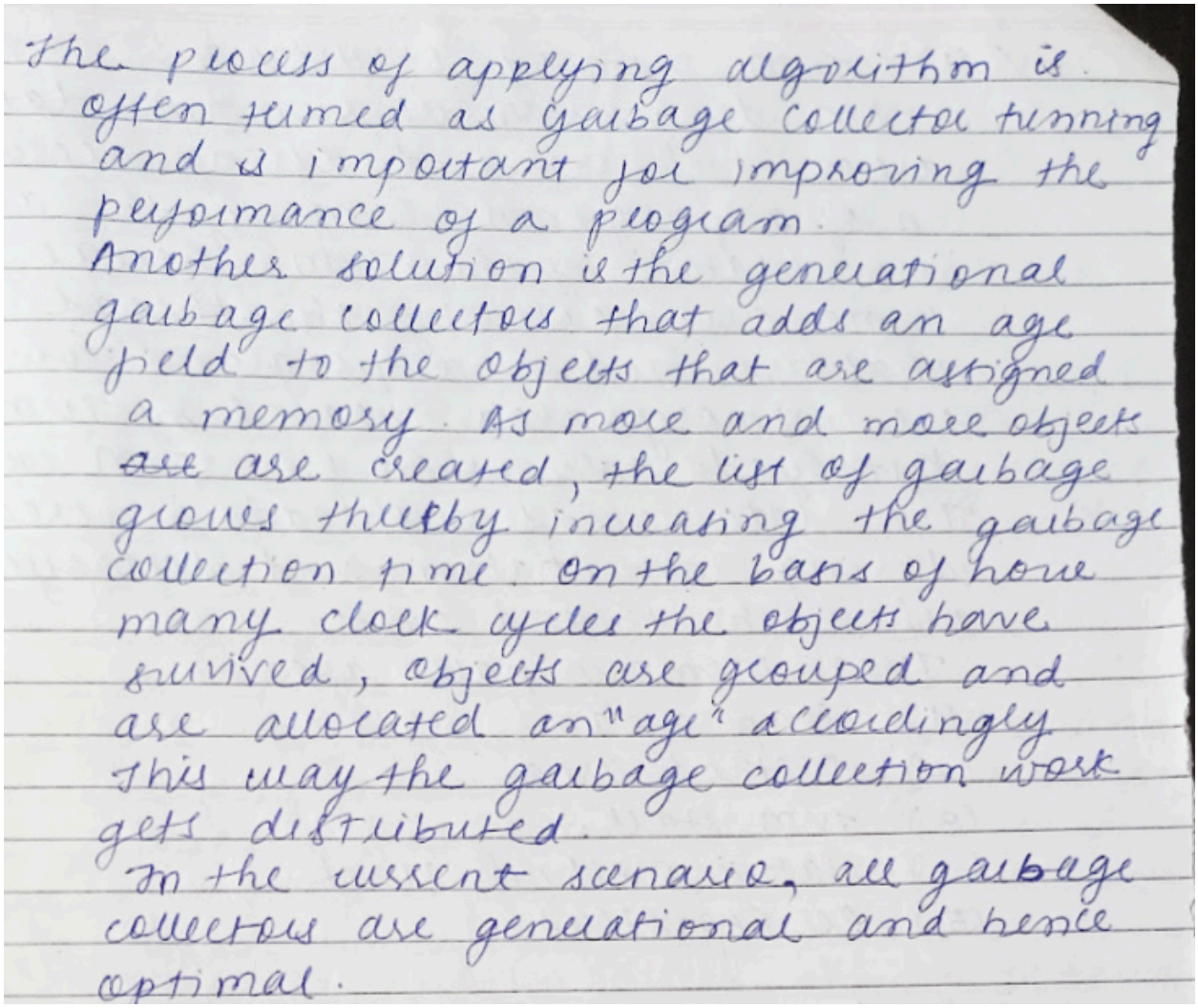
Ans - The meaning of Java platform-independent is that the Java compiled code (byte code) can run on all operating systems. A program is written in a language that is human-readable. It may contain words, phrases, etc which the machine does not understand. For the source code to be understood by the machine, it needs to be in a language understo- understood by machines, typically a machine-level language. So here comes the role of a compiler. The compiler converts the high-level language (human language) into a format understood by the machines.

Therefore, a compiler is a program that translates the source code for the another program from a programming language into executable code. This executable code may be an intermedia-te representation that is interpreted by a virtual machine. This intermedi-te representation in Java is the Byte code.

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

Components of JVM: (Java 7) 3 seperate
(1). Bootstrap class loader: used to load java APIs from rt.jar file.
(2). Extension class loader: used to load java APIs from ext directory. It loads .jar file from HD to JVM.
(3). System class loader: To load .class file into main memory.
(4). Custom class loader: A custom class loader can maintain a separate class-loader for each developer's servlet

(2). Working of a garbage collector:
JVM triggers this process and as per the JVM garbage collection process is done or else withheld. It reduces the burden of programmer by automatically performing the allocation or deallocation of memory.
Garbage collection process causes the rest of the processes or threads to be paused and thus is costly in nature. This problem is unacceptable for the client but can be eliminated by applying several garbage collector based algorithms.


The process of applying algorithm is often termed as Garbage collector running and is important for improving the performance of a program.
Another solution is the generational garbage collectors that adds an age field to the objects that are assigned a memory. As more and more objects ~~are~~ are created, the list of garbage grows thereby increasing the garbage collection time. On the basis of how many clock cycles the objects have survived, objects are grouped and are allocated an "age" accordingly. This way the garbage collection work gets distributed.
In the current scenario, all garbage collectors are generational and hence optimal.

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

what are the four acces modifiers in java, and how do they differ from each other?

In java, Acess modifiers helps to restrict the scope of a class, constructor, variable, method or data member. The It provides security, accessibility etc to the user depending upon the acess modifier used with the element.

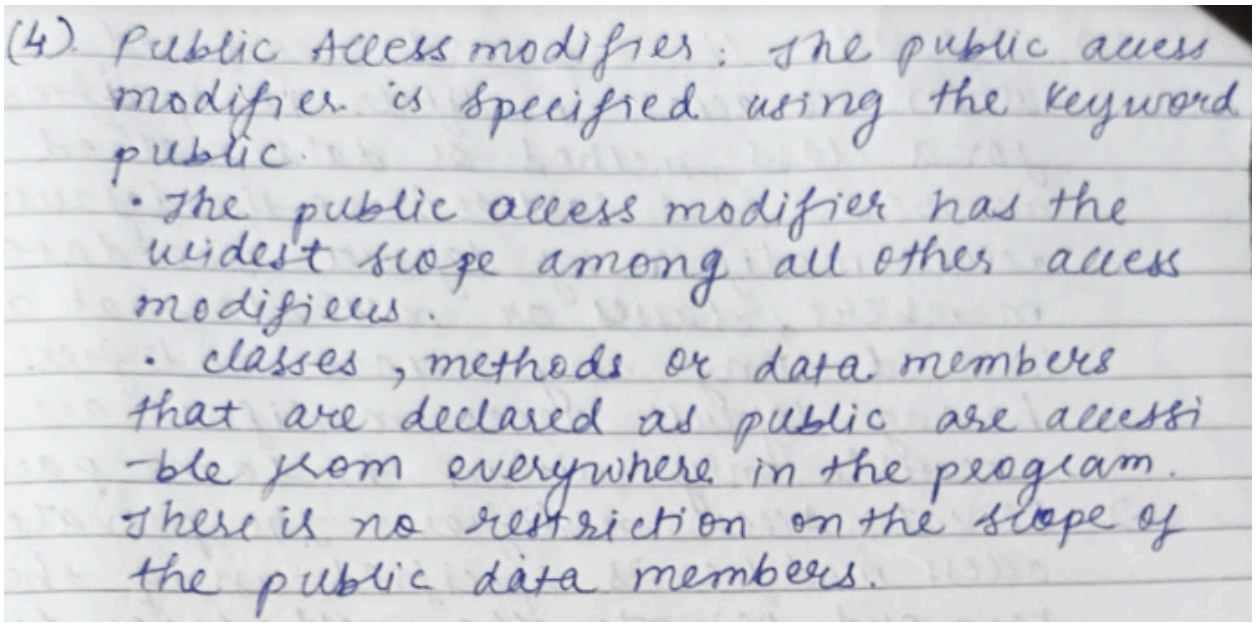
~~Let-us~~

Types of acess modifiers in java:

(1). Default : No keyword required
When no access modifier is specified for a class, method or data method member - It is said to have the default access modifier by default. The data members, classes or methods that are declared using any access modifiers i.e having default access modifiers are accessible only within the same package

(2). Private Access modifier: The private access modifier is specified using the keyword private. The methods or data members declared as private are accessible only within the class in which they are declared.
- Any other class of the same package will not be able to access these members.
- Top level classes or interfaces cannot be declared as private because,
  - Private means "only visible within the enclosing class".
  - protected means "only visible within the enclosing class and any subclasses
  Hence, these modifiers in terms of application to classes, apply only to nested classes and not on top-level classes.

(3). Protected access ~~specifier~~ modifier: The protected access ~~specifier~~ modifier is specified using the keyword protected. The methods or data members declared as protected are accessible within the same package or subclasses in different packages.


(4) Public Access modifier: The public access modifier is specified using the keyword public.
- The public access modifier has the widest scope among all other access modifiers.
- classes, methods or data members that are declared as public are accessible from everywhere in the program. There is no restriction on the scope of the public data members.

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

(1). Default : No keyword required
When no access modifier is specified for a class, method or data method member - It is said to have the default access modifier by default. The data members, classes or methods that are declared using any access modifiers i.e having default access modifiers are accessible only within the same package

(2). Private Access modifier: The private access modifier is specified using the keyword private. The methods or data members declared as private are accessible only within the class in which they are declared.
- Any other class of the same package will not be able to access these members.
- Top level classes or interfaces cannot be declared as private because,
  - Private means "only visible within the enclosing class".
  - protected means "only visible within the enclosing class and any subclasses
  Hence, these modifiers in terms of application to classes, apply only to nested classes and not on top-level classes.

(3). Protected access ~~specifier~~ modifier: The protected access ~~specifier~~ modifier is specified using the keyword protected. The methods or data members declared as protected are accessible within the same package or subclasses in different packages.

(4). Public Access modifier: The public access modifier is specified using the keyword public.

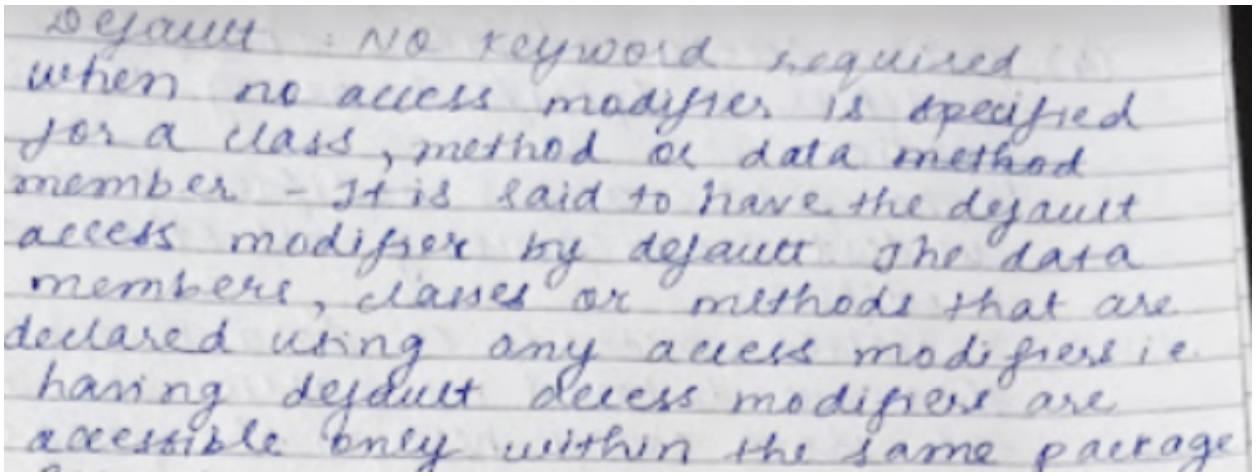
- The public access modifier has the widest scope among all other access modifiers.
- classes, methods or data members that are declared as public are accessible from everywhere in the program. There is no restriction on the scope of the public data members.

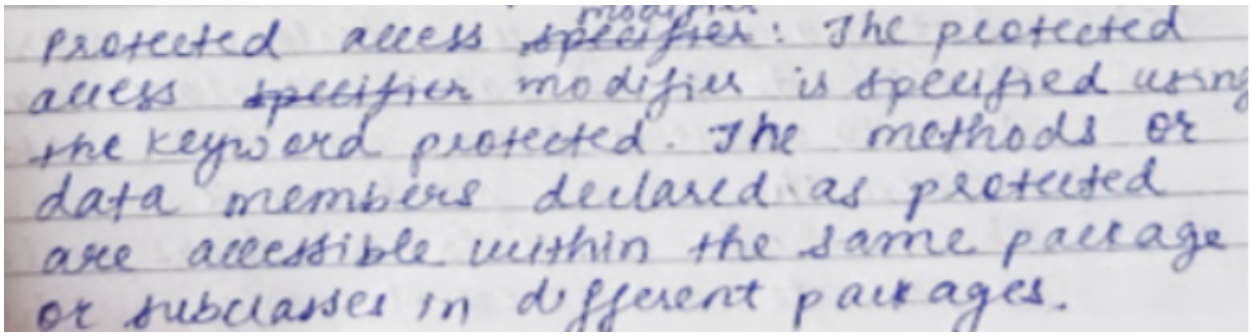
11) Can you override a method with a different access modifier in a subclass? For example, can a protected method in a superclass be overridden with a private method in a subclass? Explain.

(9). can you override a method with a different access modifier in ~~java~~ a subclass? for eg, can a protected method in a superclass be overridden with a private method in a subclass? Explain.

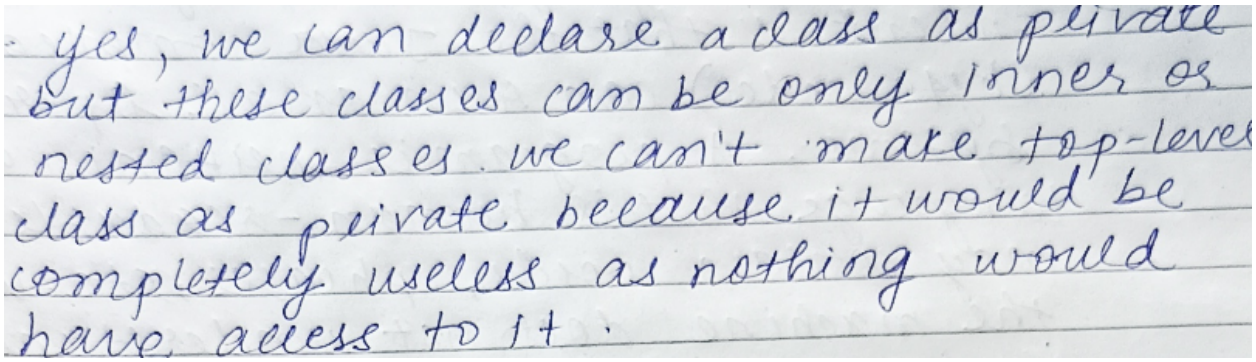
Ans. yes, an overridden method can have a different access modifier but it cannot lower the access scope. Methods declared public in a superclass also must be public in all subclasses. Methods declared protected in a superclass must either be protected or public in subclasses, they cannot be private.

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

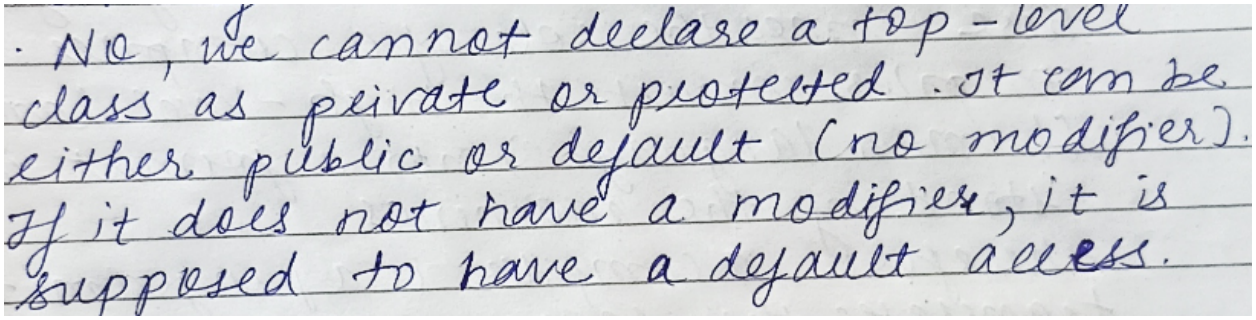

default : No keyword required
when no access modifier is specified for a class, method or data method member - It is said to have the default access modifier by default The data members, classes or methods that are declared using any access modifiers i.e having default access modifiers are accessible only within the same package


protected access ~~specifier~~ modifier: The protected access ~~specifier~~ modifier is specified using the keyword protected. The methods or data members declared as protected are accessible within the same package or subclasses in different packages.

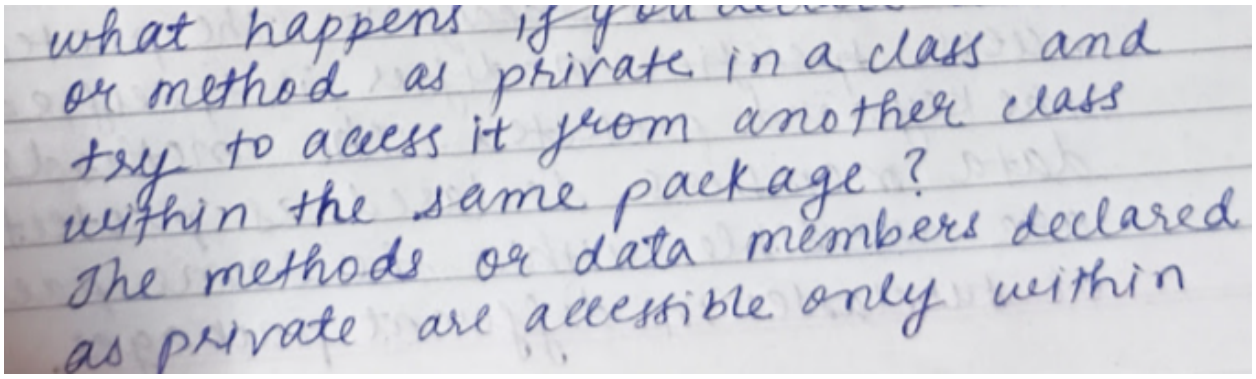
13) Is it possible to make a class private in Java? If yes, where can it be done, and what are the limitations?

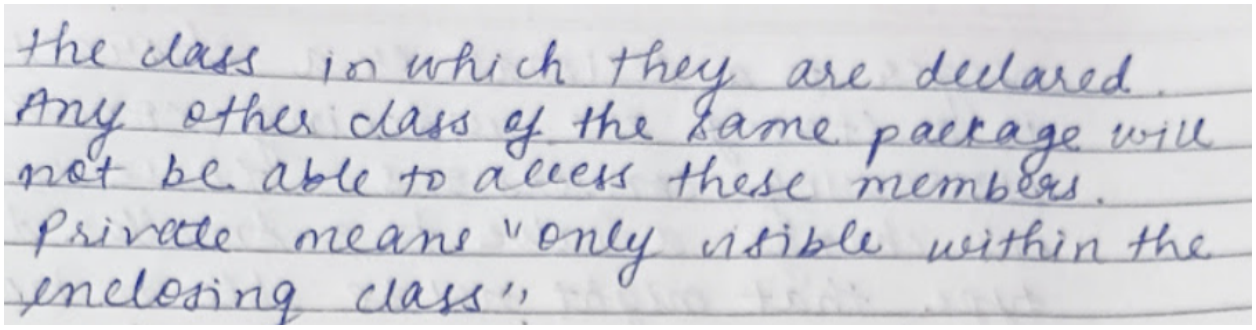

- yes, we can declare a class as private but these classes can be only inner or nested classes. we can't make top-level class as private because it would be completely useless as nothing would have access to it.

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?

Default : No keyword required when no access modifier is specified for a class, method or data method member - It is said to have the default access modifier by default The data members, classes or methods that are declared using any access modifiers i.e having default access modifiers are accessible only within the same package