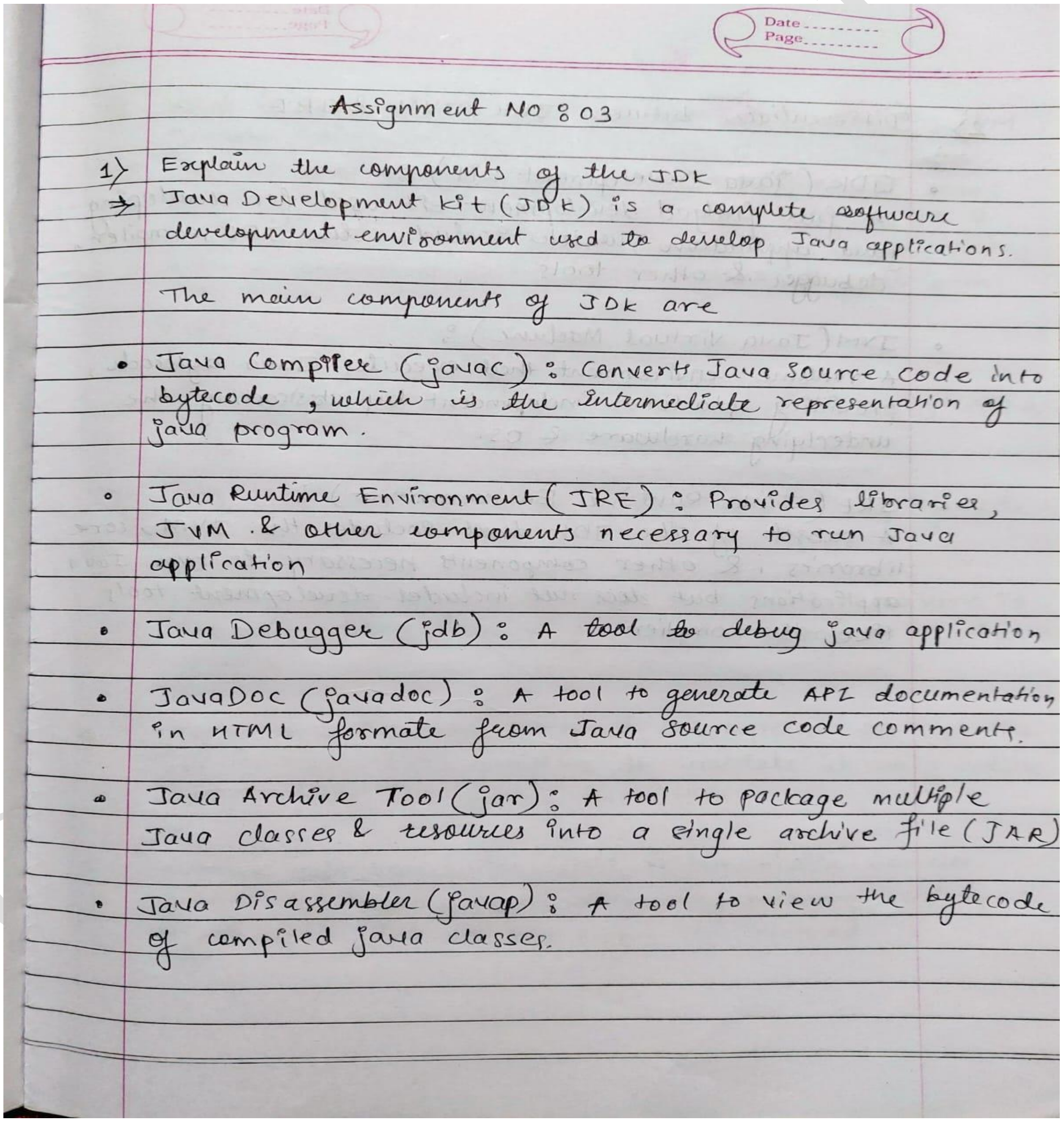
CDAC Mumbai PG-DAC AUGUST 24
Assignment No- 3

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.



Date.........
Page.........

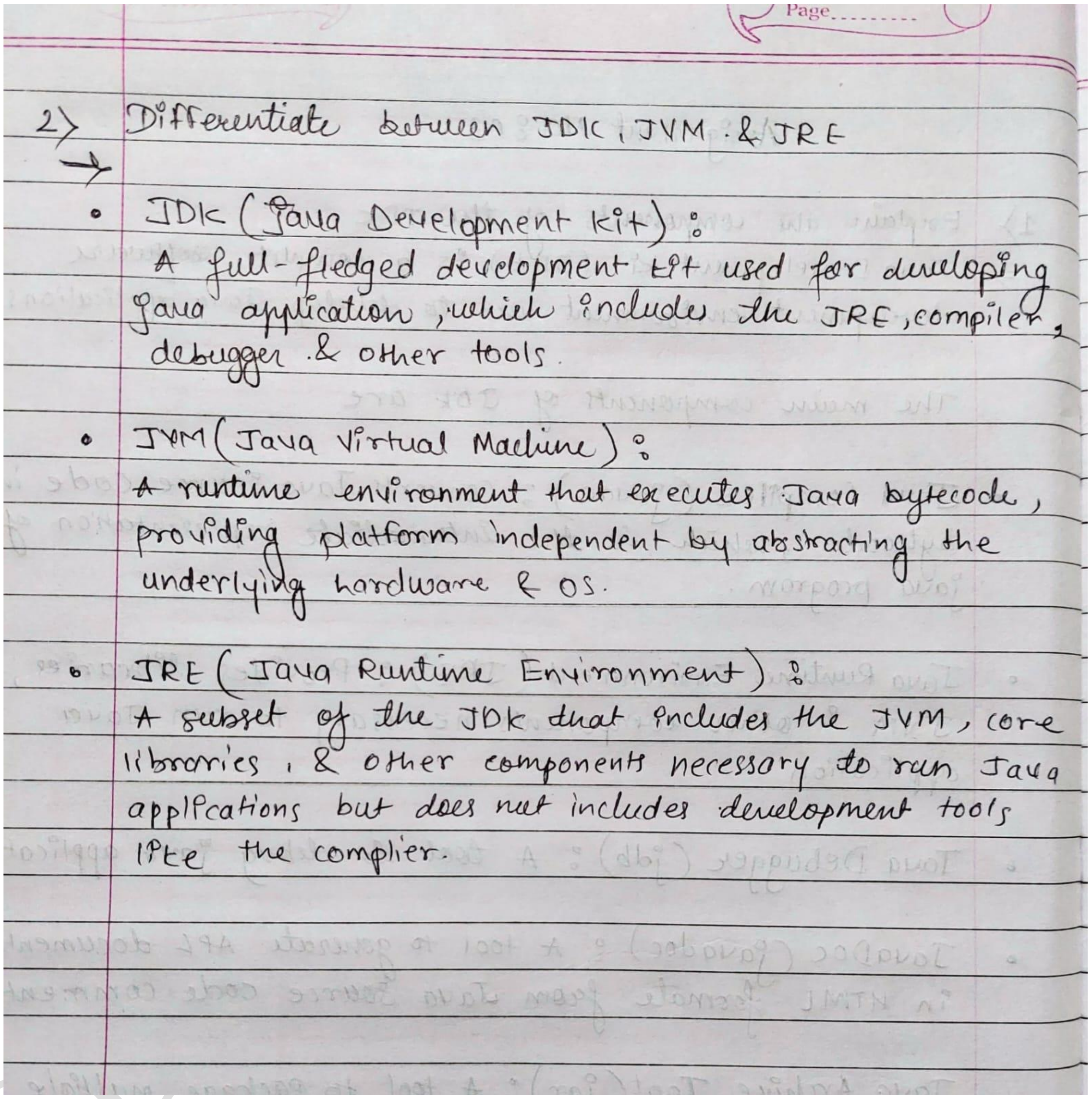
Assignment No : 03

1> Explain the components of the JDK
⇒ Java Development kit (JDK) is a complete software development environment used to develop Java applications.

The main components of JDk are

- Java Compiler (javac) : Converts Java source code into bytecode, which is the intermediate representation of java program.
- Java Runtime Environment (JRE) : Provides libraries, JVM & other components necessary to run Java application
- Java Debugger (jdb) : A tool to debug java application
- JavaDoc (javadoc) : A tool to generate API documentation in HTML formate from Java source code comments.
- Java Archive Tool (jar) : A tool to package multiple Java classes & resources into a single archive file (JAR)
- Java Disassembler (javap) : A tool to view the bytecode of compiled java classes.

2) Differentiate between JDK, JVM, and JRE.



Page........

2) Differentiate between JDK, JVM & JRE

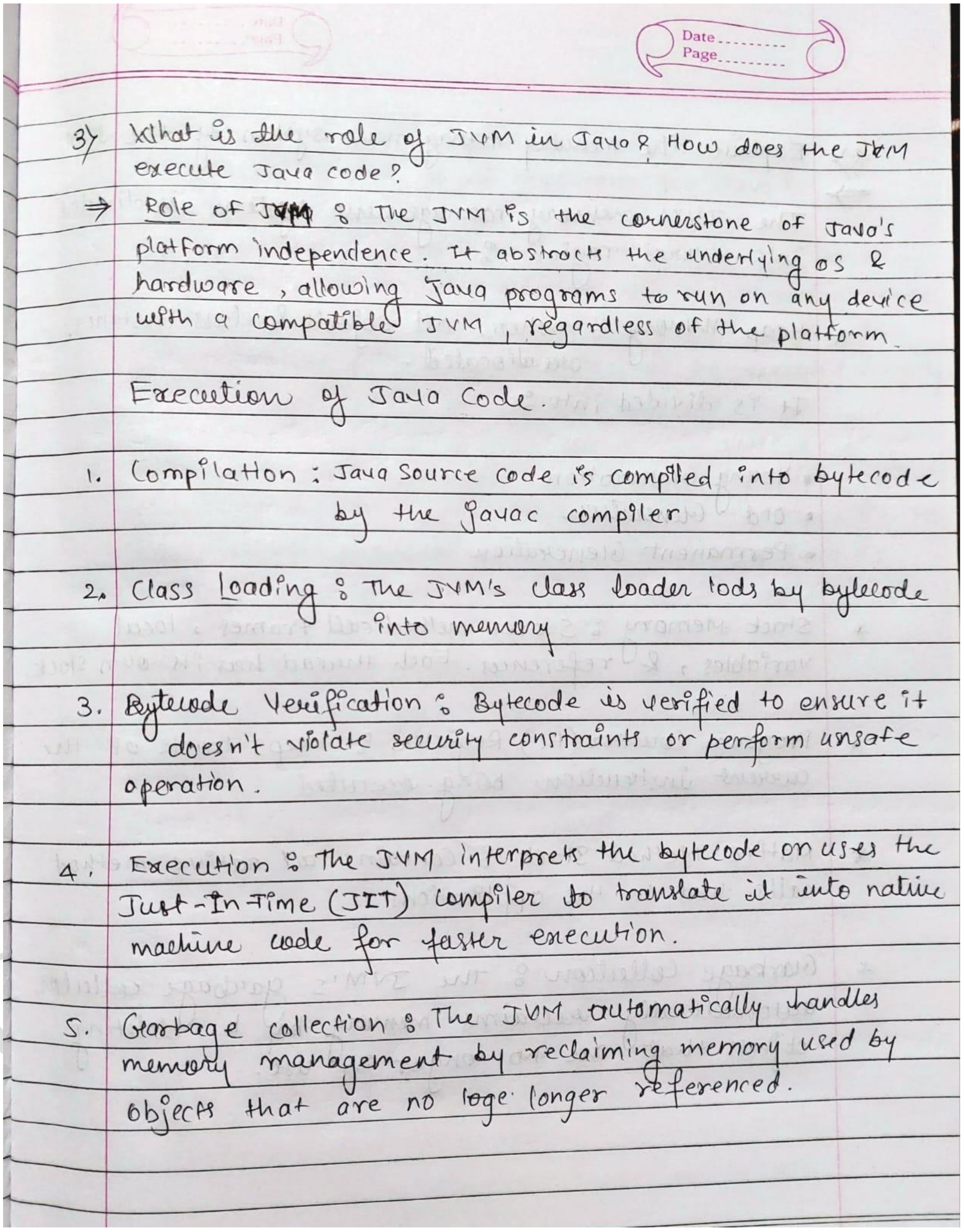
→

- JDK (Java Development kit) :
  A full-fledged development kit used for developing Java application, which includes the JRE, compiler, debugger & other tools.

- JVM (Java Virtual Machine) :
  A runtime environment that executes Java bytecode, providing platform independent by abstracting the underlying hardware & OS.

- JRE (Java Runtime Environment) :
  A subset of the JDK that includes the JVM, core libraries, & other components necessary to run Java applications but does not includes development tools like the compiler.

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



Date.........
Page.........

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

→ Role of JVM : The JVM is the cornerstone of Java's platform independence. It abstracts the underlying OS & hardware, allowing Java programs to run on any device with a compatible JVM, regardless of the platform.

Execution of Java Code.

1. Compilation : Java Source code is compiled into bytecode by the javac compiler.

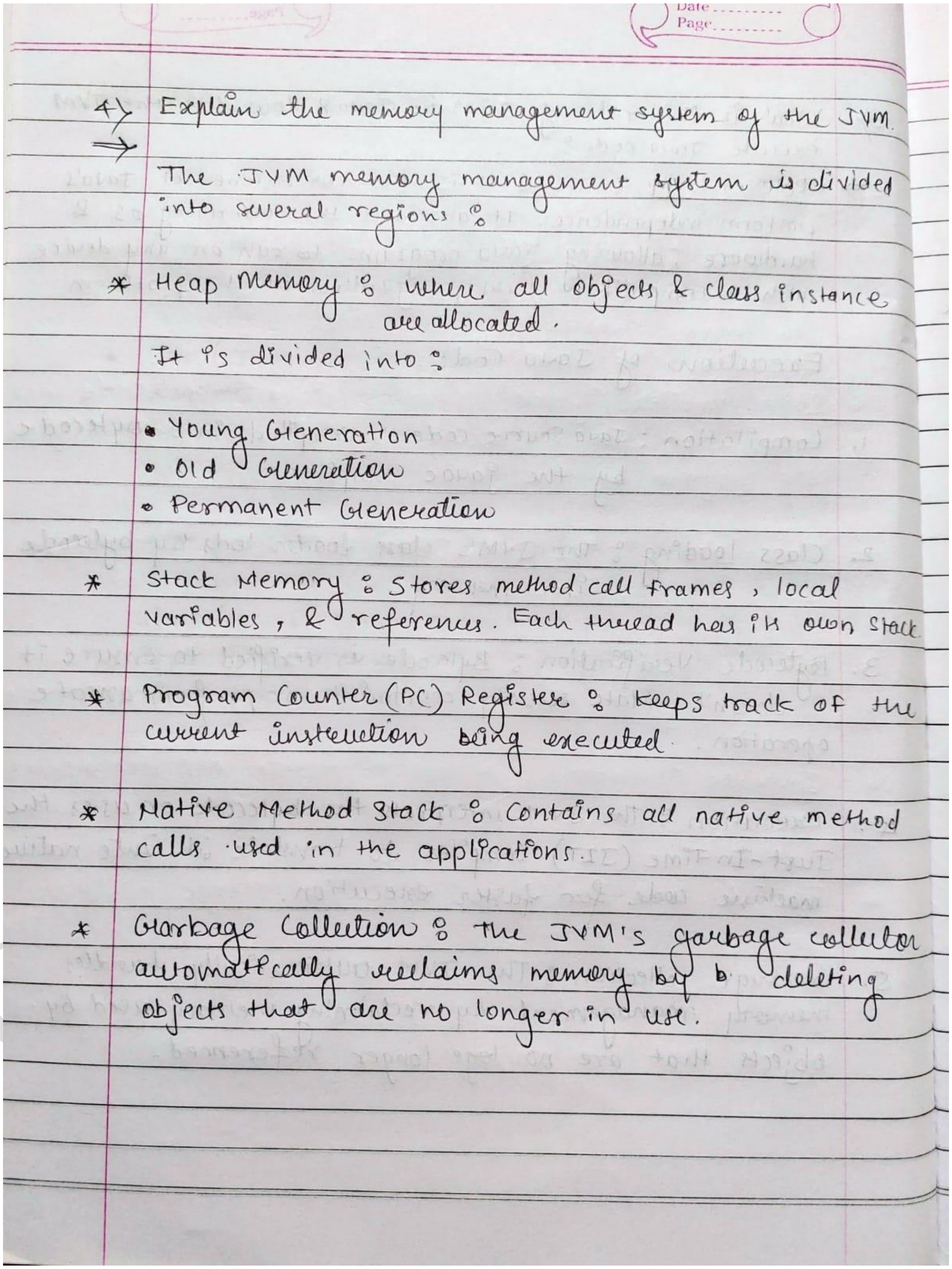
2. Class Loading : The JVM's class loader lods by bytecode into memory

3. Bytecode Verification : Bytecode is verified to ensure it doesn't violate security constraints or perform unsafe operation.

4. Execution : The JVM interprets the bytecode or uses the Just-In-Time (JIT) compiler to translate it into native machine code for faster execution.

5. Garbage collection : The JVM automatically handles memory management by reclaiming memory used by objects that are no ~~loge~~ longer referenced.

4) Explain the memory management system of the JVM.



Date ....

Page ....

4) Explain the memory management system of the JVM.

→

The JVM memory management system is divided into several regions :

* Heap Memory : where all objects & class instance are allocated.

It is divided into :

- Young Generation
- Old Generation
- Permanent Generation

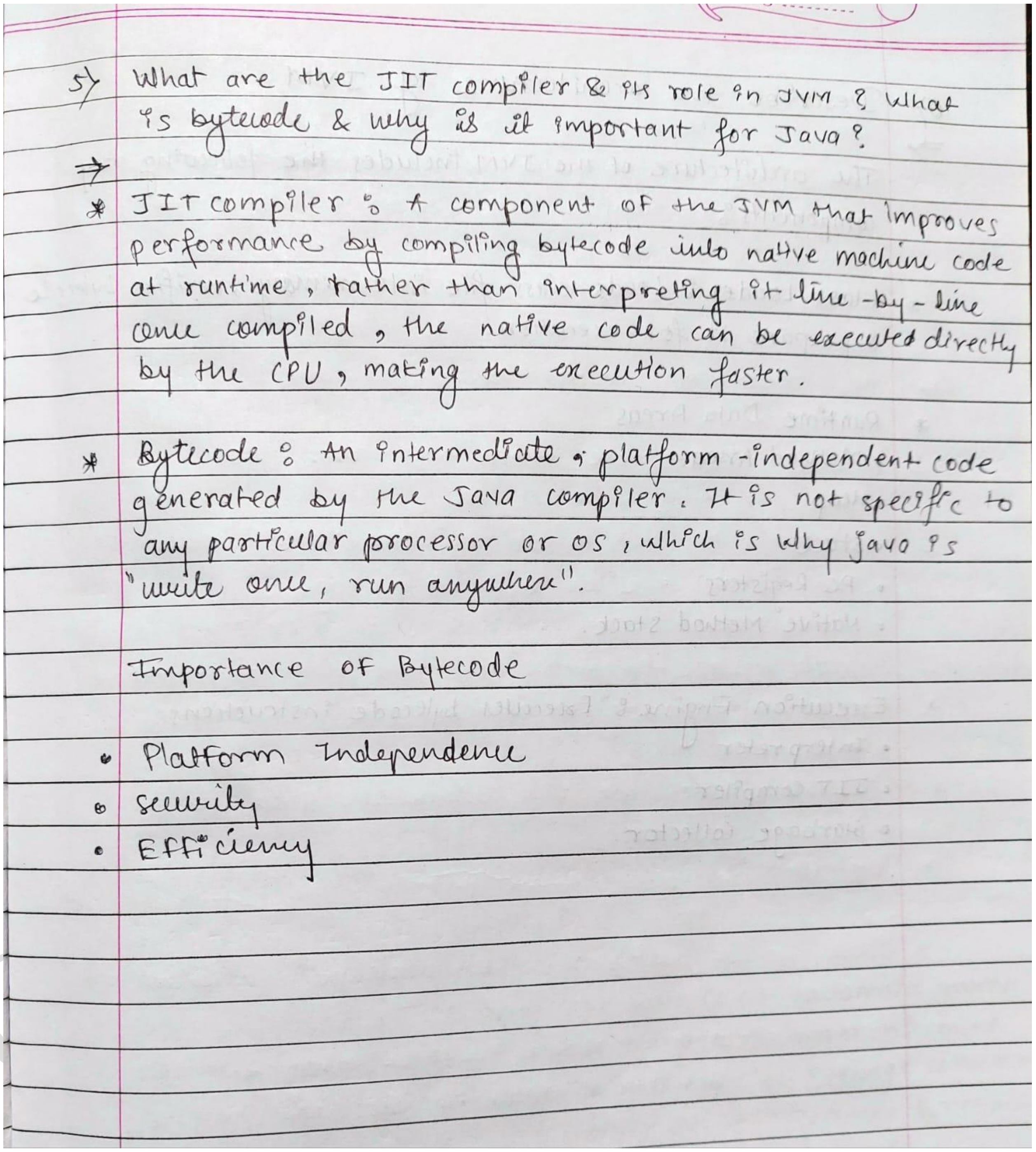
* Stack Memory : Stores method call frames, local variables, & references. Each thread has its own stack.

* Program Counter (PC) Register : Keeps track of the current instruction being executed.

* Native Method Stack : Contains all native method calls used in the applications.

* Garbage Collection : the JVM's garbage collector automatically reclaims memory by deleting objects that are no longer in use.

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



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

→

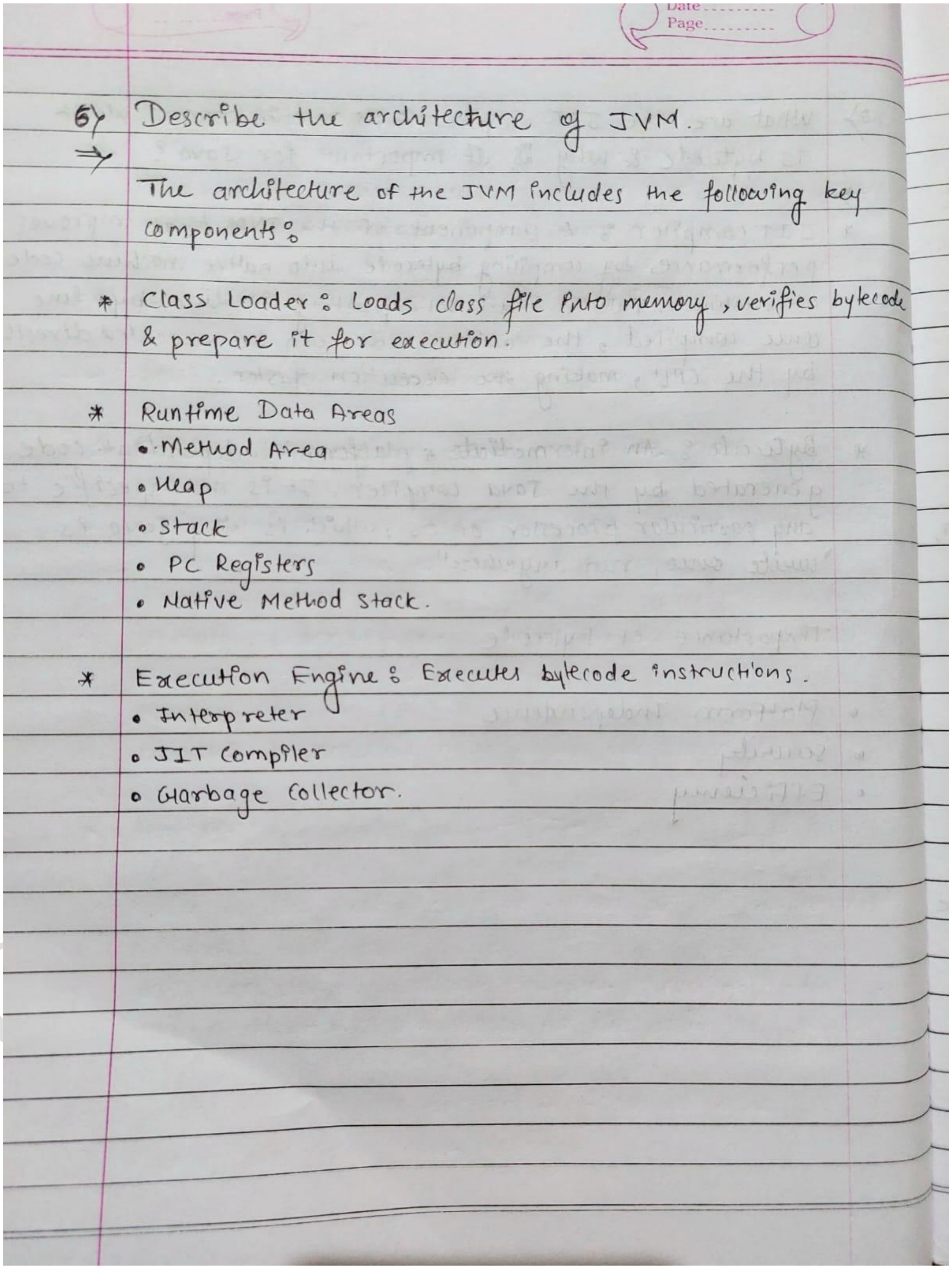
* JIT compiler : A component of the JVM that improves performance by compiling bytecode into native machine code at runtime, rather than interpreting it line-by-line once compiled, the native code can be executed directly by the CPU, making the execution faster.

* Bytecode : An intermediate, platform-independent code generated by the Java compiler. It is not specific to any particular processor or OS, which is why Java is "write once, run anywhere".

Importance of Bytecode

- Platform Independence
- Security
- Efficiency

6) Describe the architecture of the JVM.



Date.........
Page.........

6) Describe the architecture of JVM.

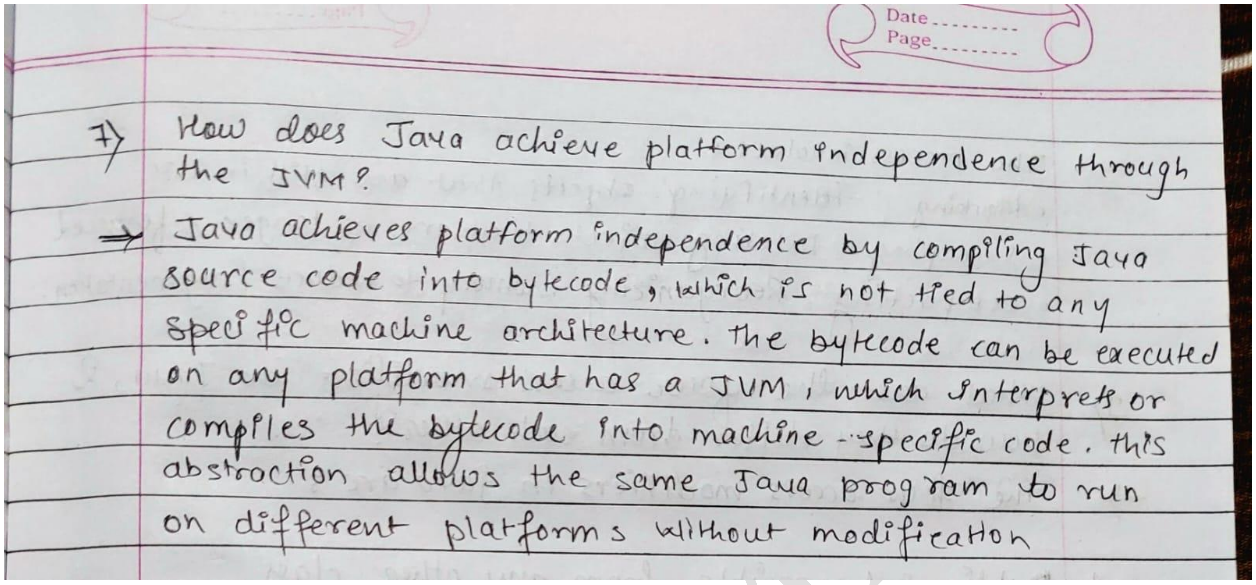
→ The architecture of the JVM includes the following key components:

* Class Loader: Loads class file into memory, verifies bytecode & prepare it for execution.

* Runtime Data Areas
  - Method Area
  - Heap
  - Stack
  - PC Registers
  - Native Method Stack.

* Execution Engine: Executes bytecode instructions.
  - Interpreter
  - JIT Compiler
  - Garbage Collector.

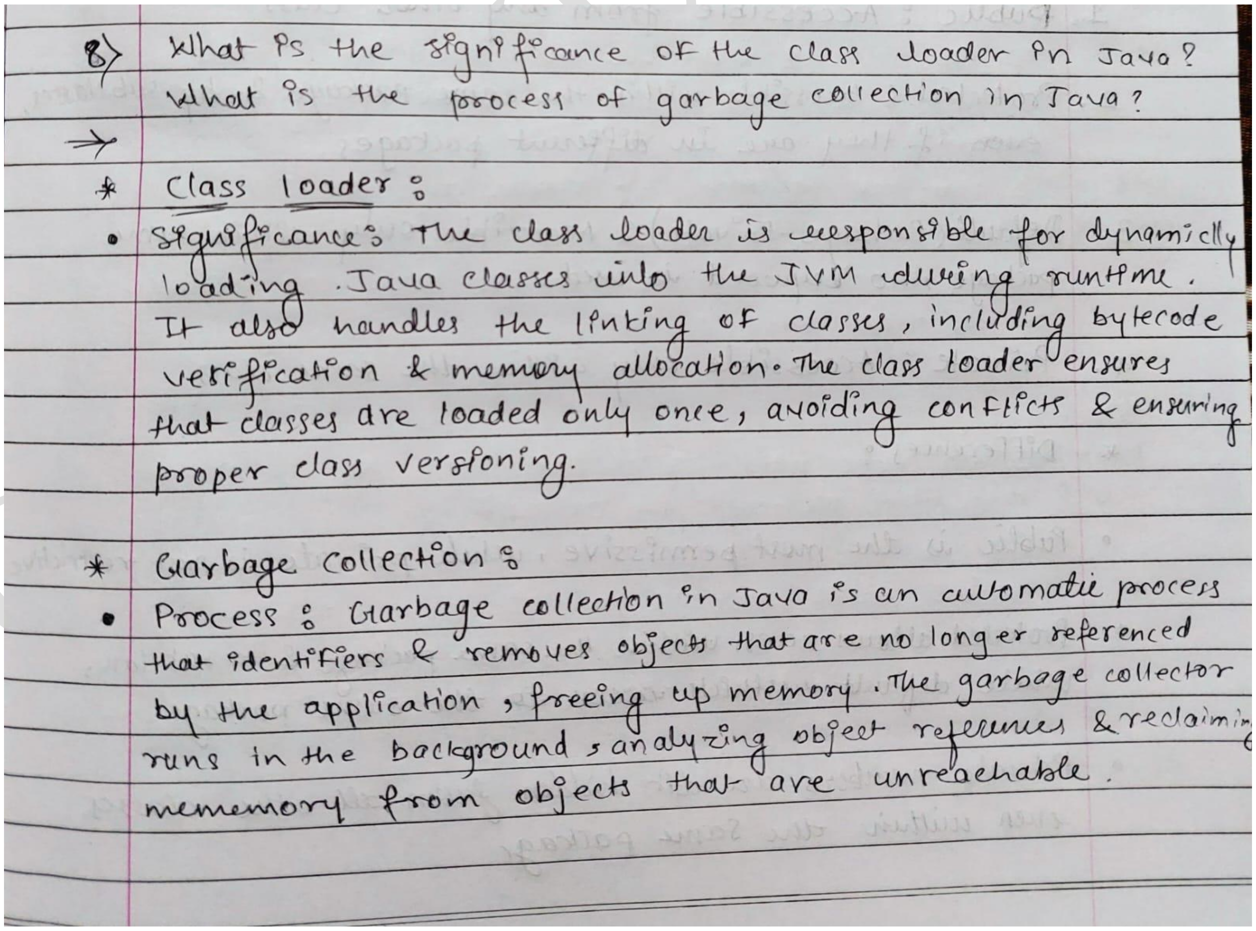
7) How does Java achieve platform independence through the JVM?



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

→ Java achieves platform independence by compiling Java source code into bytecode, which is not tied to any specific machine architecture. The bytecode can be executed on any platform that has a JVM, which interprets or compiles the bytecode into machine-specific code. this abstraction allows the same Java program to run on different platforms without modification

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



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

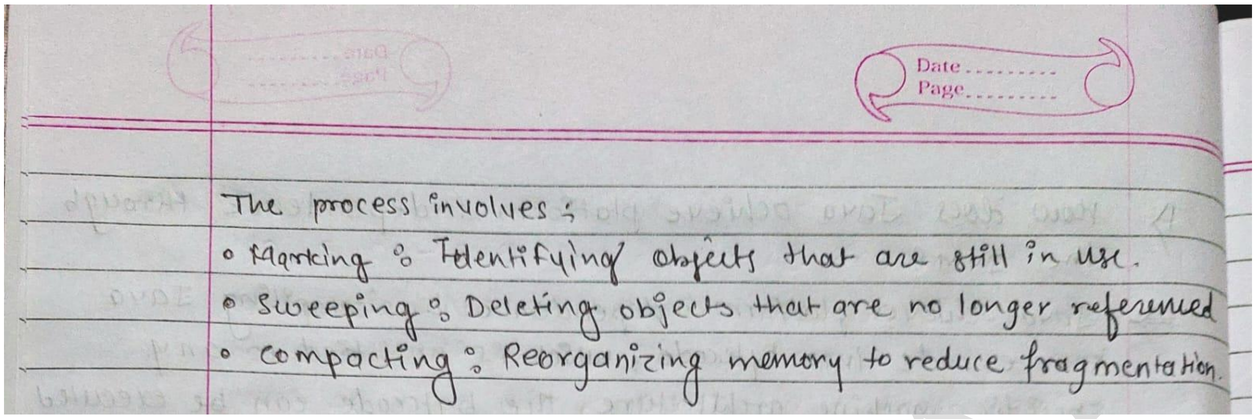
→

* Class loader :

- Significance: The class loader is responsible for dynamiclly loading Java classes into the JVM during runtime. It also handles the linking of classes, including bytecode verification & memory allocation. The class loader ensures that classes are loaded only once, avoiding conflicts & ensuring proper class versioning.

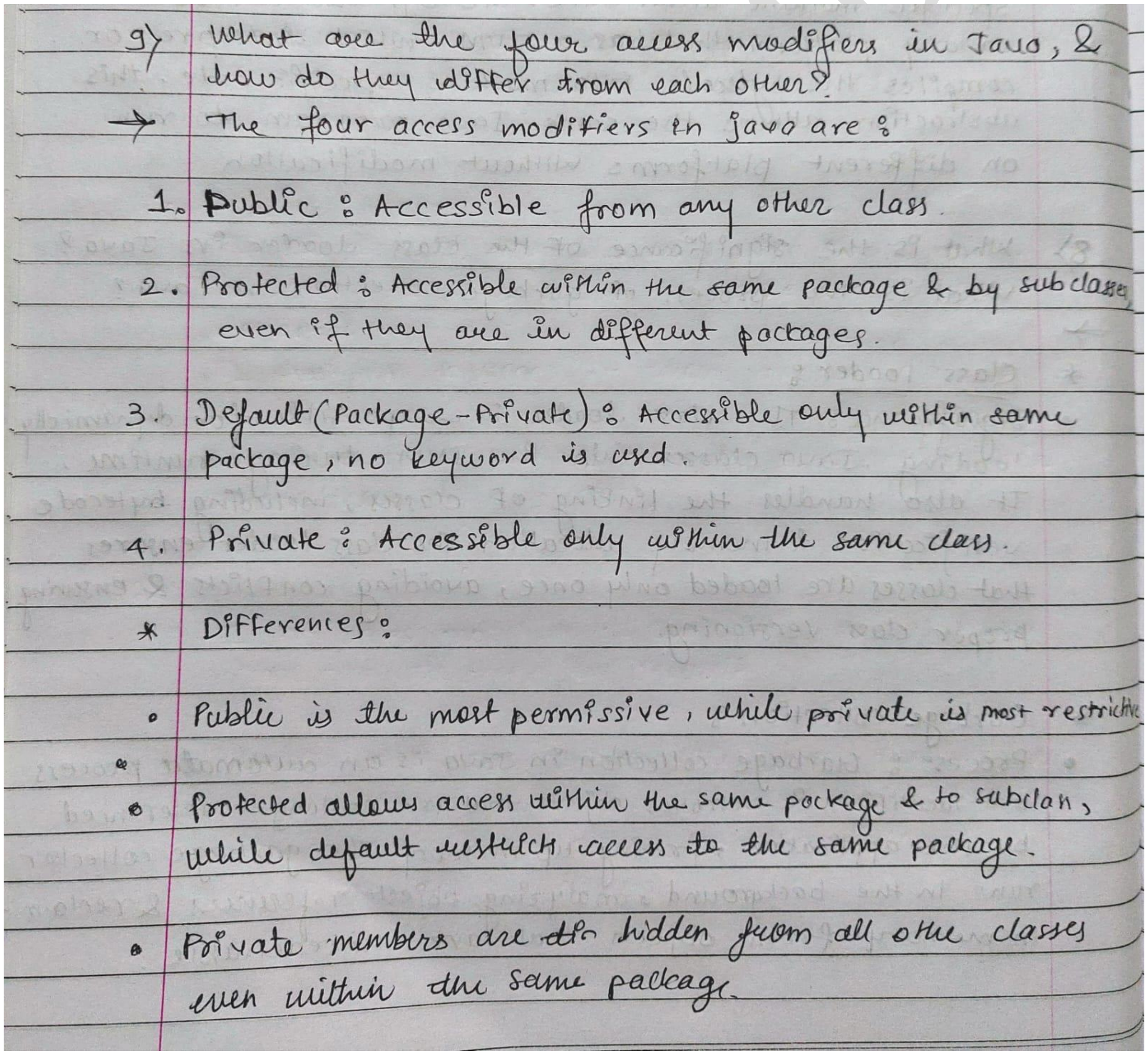
* Garbage collection :

- Process : Garbage collection in Java is an automatic process that identifiers & removes objects that are no longer referenced by the application, freeing up memory. The garbage collector runs in the background, analyzing object references & reclaiming mememory from objects that are unreachable.


Date........
Page........

The process involves :
- Marking : Identifying objects that are still in use.
- Sweeping : Deleting objects that are no longer referenced
- Compacting : Reorganizing memory to reduce fragmentation.

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


9) What are the four access modifiers in Java, & how do they differ from each other?
→ The four access modifiers in java are :

1. Public : Accessible from any other class.

2. Protected : Accessible within the same package & by subclasses even if they are in different packages.

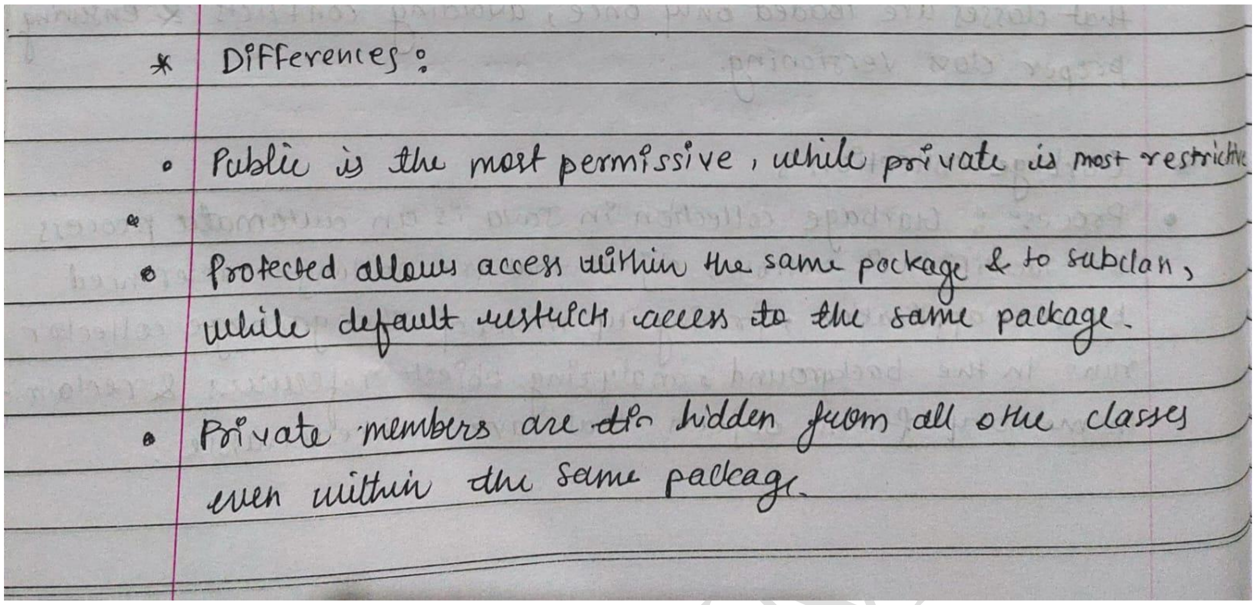
3. Default (Package-Private) : Accessible only within same package, no keyword is used.

4. Private : Accessible only within the same class.

* Differences :

- Public is the most permissive, while private is most restrictive
- Protected allows access within the same package & to subclass, while default restricts access to the same package.
- Private members are ~~dif~~ hidden from all other classes even within the same package.

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

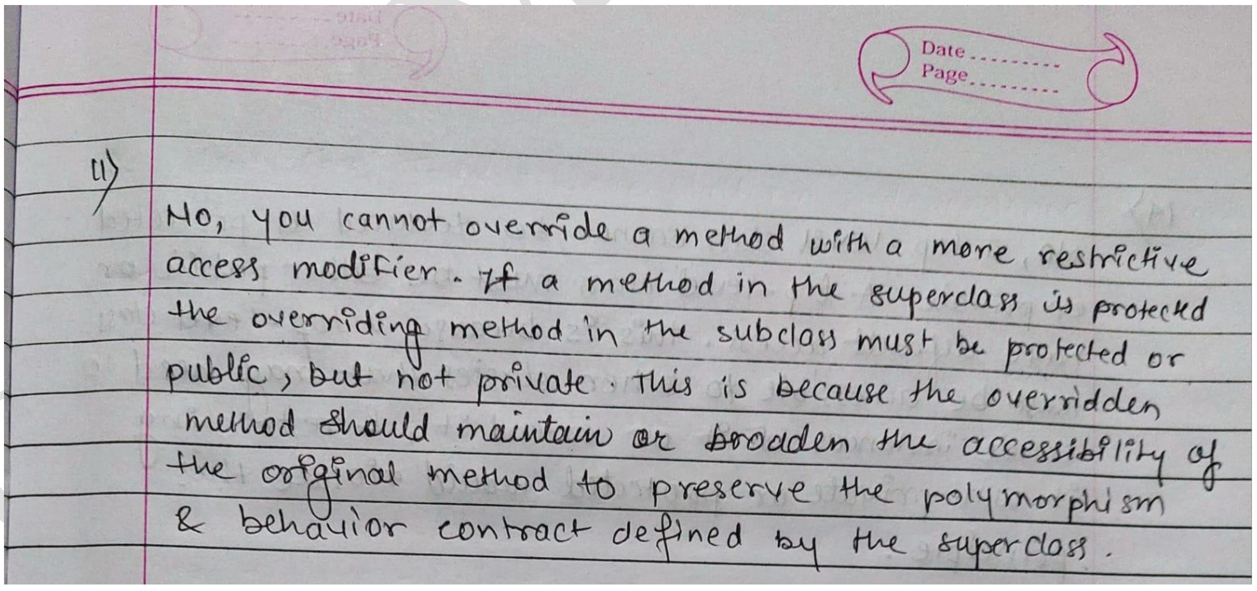


* Differences :

- Public is the most permissive, while private is most restrictive
- Protected allows access within the same package & to subclass, while default restricts access to the same package.
- Private members are ~~the~~ hidden from all other classes even within the same package.

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

.



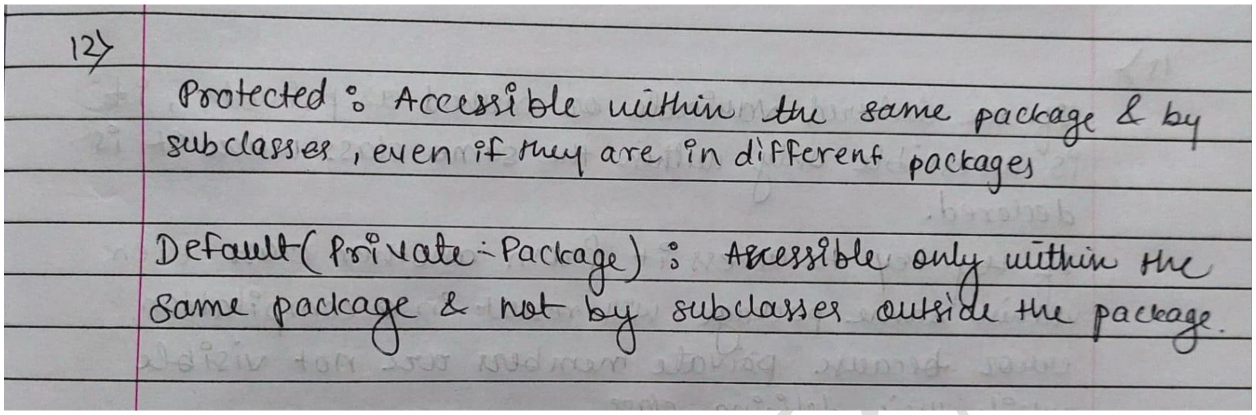
Date........

Page........

11)

No, you cannot override a method with a more restrictive access modifier. If a method in the superclass is protected the overriding method in the subclass must be protected or public, but not private. This is because the overridden method should maintain or broaden the accessibility of the original method to preserve the polymorphism & behavior contract defined by the superclass.

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

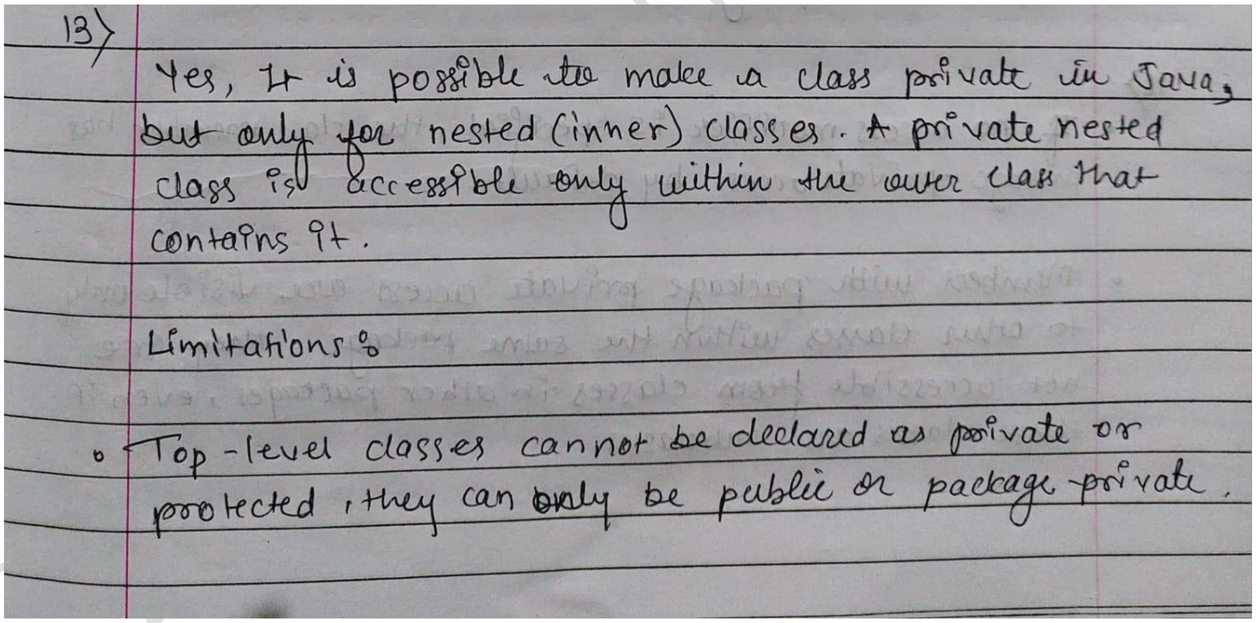


12>

Protected : Accessible within the same package & by subclasses, even if they are in different packages

Default (Private-Package) : Accessible only within the same package & not by subclasses 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?



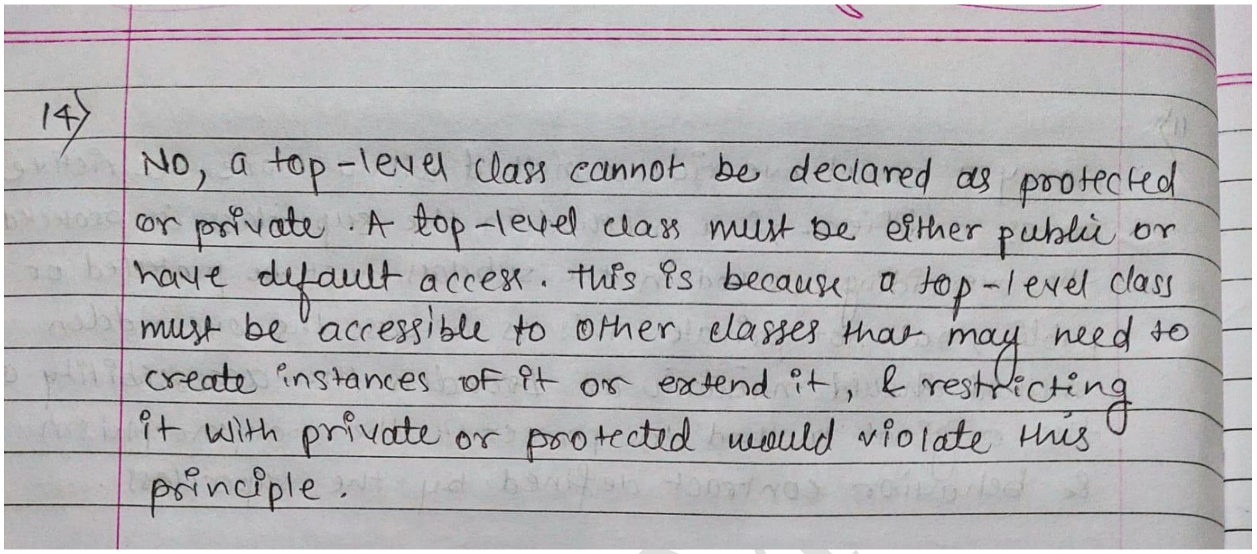
13>

Yes, It is possible to make a class private in Java, but only for nested (inner) classes. A private nested class is accessible only within the outer class that contains it.

Limitations :

- Top-level classes cannot be declared as private or protected, they can ~~only~~ be public or package-private.

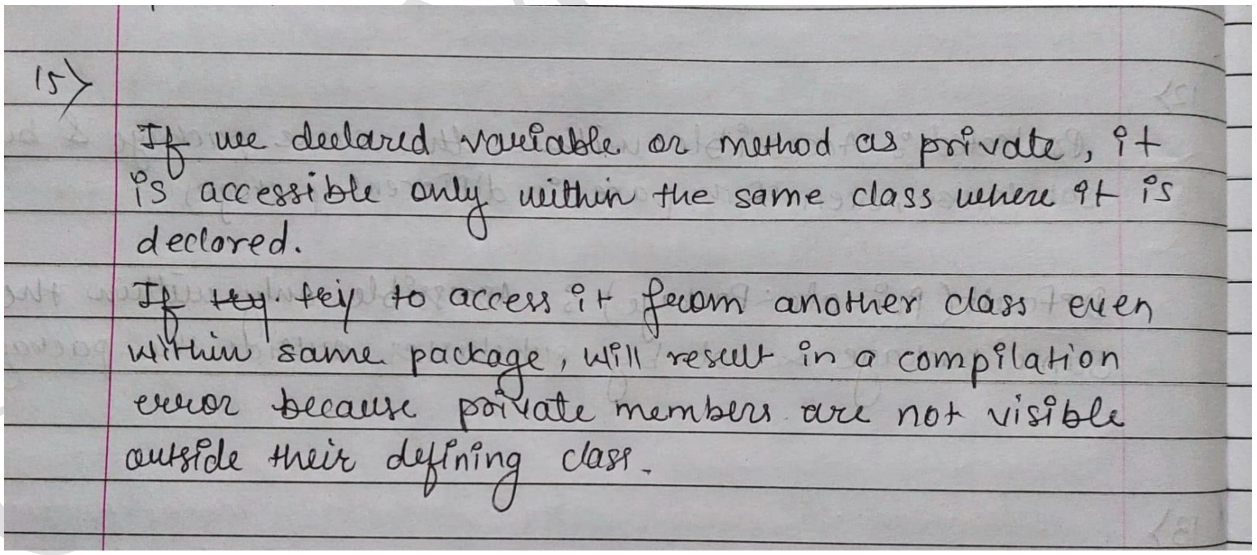
14) Can a top-level class in Java be declared as protected or private? Why or why not?



14)

No, a top-level class cannot be declared as protected or private. A top-level class must be either public or have default access. this is because a top-level class must be accessible to other classes that may need to create instances of it or extend it, & restricting it with private or protected would violate this principle.

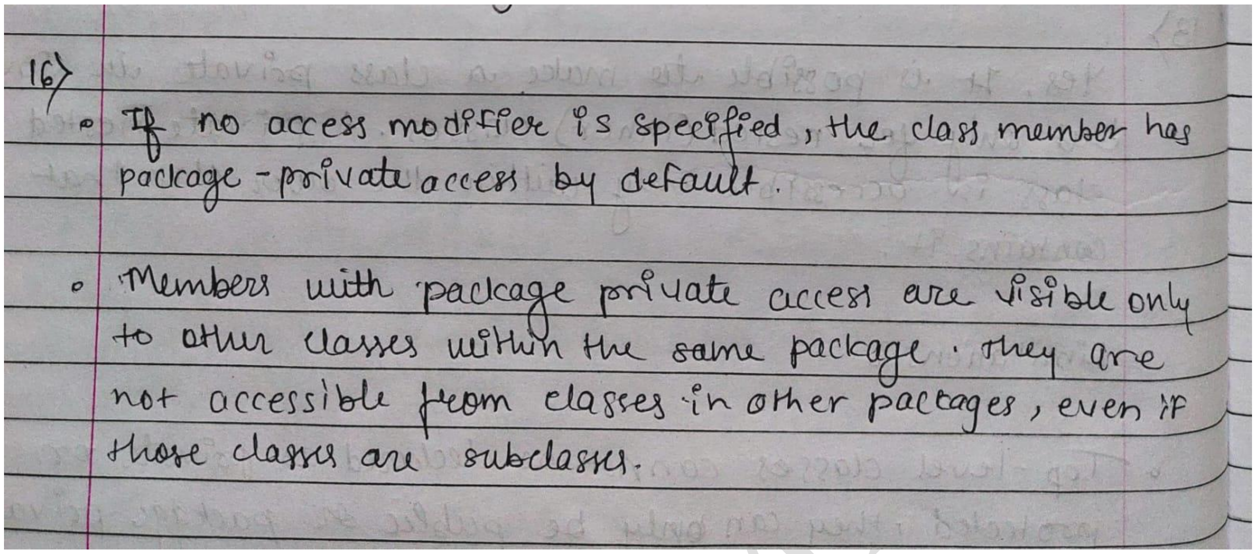
15) What happens if you declare a variable or method as private in a class and try to access it from another class within the same package?



15)

If we declared variable or method as private, it is accessible only within the same class where it is declared.
If ~~tey~~ tey to access it from another class even within same package, will result in a compilation error because private members are not visible outside their defining class.

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



16)

- If no access modiffier is specified, the class member has package-private access by default.
- Members with package private access are visible only to other classes within the same package. They are not accessible from classes in other pactages, even if those classes are subclasses.