PG-DAC

CDAC Mumbai

PAGE No
DATE / /

Assignment No-3

1) Explain the components of the JDK

→ Java Development Kit is a software development environment provided by Oracle Corporation, enabling developers to build Java application and applets. It includes various tools, libraries, and executables required for Java development. The JDK forms the foundation for Java Plateform, Standard Edition (Java SE), which is the platform used to develop and deploy Java applications for desktops, servers and embedded environments.

Major Components of JDK :

1) Java Compiler (javac)

The Java Compiler (javac) is a key component of JDK that transforms Java source code (.java files) into bytecode (.class files). The generated bytecode can be executed on any platform with a Java Virtual Machine (JVM) installed, ensuring the "write once, run anywhere" philosophy of Java.

2) Java Virtual Machine (JVM)

The Java Virtual Machine (JVM) is the runtime engine that executes Java bytecode. It provides an abstraction layer between the Java application and the underlying operating system. The JVM enables Java programs to run independently of the hardware and operating system, enhancing portability and security.

3) Java Runtime Environment (JRE)

The Java Runtime Environment (JRE) is a subset of JDK that includes the JVM and essential class libraries. It is required to run Java Applications on end-user systems without the need for development tools. Users can execute Java applications using the JRE, ensuring a seamless experience.

PAGE No.
DATE / /

4) Java API Libraries
Java API libraries provide a vast collection of pre-built classes and methods that simplify common programming tasks. These libraries cover areas such as input/output, networking, database connectivity, GUI development, and more. Developers can leverage these APIs to accelerate application development and improve code quality.

5) Java Debugger (jdb)
The Java Debugger (jdb) is a powerful tool for debugging Java applications. It allows developers to set breakpoints, inspect variables, and step through the code to identify and fix issues during development. The debugger significantly aids in understanding the program flow and identifying logical errors

6) Java Documentation Generator (Javadoc)
Java Documentation Generator (Javadoc) automatically generates documentation from Java source code. It helps in creating comprehensive API documentation, making it easier for developers to understand and use the classes and methods provided by the application's codebase.

Differen

JDK
1) JDK is abbrevia Java D Kit
2) JDK softwa kit i kit d in J JR com de (3)

Why USE

Difference between: JDK, JRE

	JDK	JRE	JVM
1)	JDK is an abbreviation for Java Development Kit	The JRE is an abbreviation for Java Runtime Environment	The JVM is an abbreviation for Java Virtual Machine
2)	JDK is a software development kit that ~~developers~~ kit develops apps in JAVA. Along with JRE, the JDK also consists of various development tools (JAVA Debugger, JAVADOC, compiler.)	The JRE is an implementation of JVM. It is a type of software package that provides class libraries of Java, JVM, and various other components for running the applications written in Java programming.	The JVM is a platform-independent abstract machine that has three notions in the form of specification. This document describes the requirement of JVM implementation.
3)	The JDK primarily assists in executing codes. It primarily functions in develop.	JRE has a major responsibility for creating an environment for the execution of code.	JVM specifies all of the implementations. It is responsible for providing all of these implementations to the JRE.
Why USE 4)	• It consists of various tools required for writing Java Prog.	• If a user wants to run the Java applets, then they must install JRE on their system.	• It provides its users with a platform-indepent way for executing the Java source code.
	• JDK also consists JRE for executing Java programs.	• The JRE consists of class libraries along with JVM and its supporting files. It has no other tools like a compiler or a debugger or a debugger for a Java development	• JVM consists of various tools, libraries, and multiple frameworks

PAGE No. 1
DATE / /

Features	Features of JDK	JRE	JVM
	• It has all the features that JRE does ~~• It has all the features that JRE~~ • JDK enables a user to handles multiple extensions in only one catch block. • It basically provide an environment for developing and executing the Java source code.	• It is a set of tools that actually helps the JVM to run • The JRE also consists of deployment technology. It includes Java Plug-in and Java Web Start as well	• The JVM enables a user to run apps on their device or in a cloud environment. • It helps in Converting the bytecode into machine-specific code.
	• It includes an Appletviewer, Java app launcher, compiler, etc.	• JRE uses crucial package classes like util, math, awt, lang, and various runtime libraries	• The JVM also comes with a JUST-in-Time (JIT) compiler for converting the Java source code into a low level machine language.

PAGE No.
DATE / /

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

→ Java Virtual Machine, or JVM, loads, verifies, and runs Java bytecode. It is known as the interpreter or the core of the Java programming language because it runs Java programming.

Role of JVM in Java

JVM is responsible for converting bytecode to machine-specific code and is necessary in both JDK and JRE. It is also platform-dependent and performs many functions, including memory management and security. In addition, JVM can run programs that are written in other programming languages that have been converted to Java bytecode.

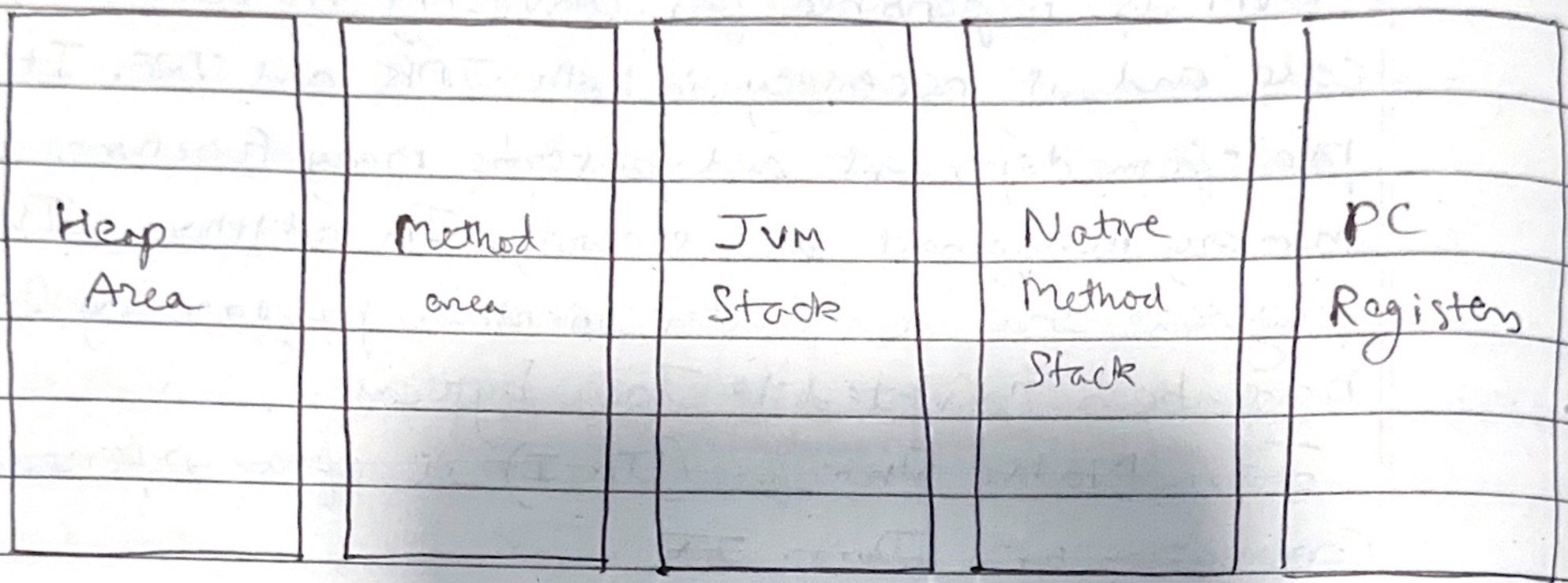
~~Java Native Interface (JNI) is often referred to in connection with JVM. JNI~~

Not Java Develop. K
(JDK 1.8)

PAGE No.
DATE / /

4) Explain the memory management system of the JVM.

→ 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 treads that are used in a program. However, memory area created by JVM is destroyed only when the JVM exits. The data areas of thread are created during the installation and destroyed when thread exit



Heap

- It is a shared runtime data area and stores the actual object in a memory. It is instantiated during the virtual machine startup
- This memory is allocated for all class instances and array. Heap can be of fixed or dynamic size depending upon the system's
- JVM provides the user control to initialize or vary the size of heap as per the requirement.

Method Area

- It is a logical part of the heap area and is created on virtual machine startup.
- This memory is allocated for class structure, method data and constructor field data. and also for interfaces or special meth used in class.

PAGE No.
DATE / /

JVM Stacks

- A stack is created at the same time when a thread is created and is used to store data and partial results which will be needed while returning value for method and performing dynamic linking.
- Stack can either be of fixed or dynamic size. The size of a stack can be chosen independently when it is created.
- The memory for stack needs not to be contiguous.

Native method Stacks

Also called as C stacks, native method stacks are not written in Java language. This memory is allocated for each thread when its created

Program counter (PC) registers

Each JVM thread which carries out the task of a specific method has a program counter register associated with it. The non native method, the value of program counter is undefined. PC register is capable of storing the return address or a native pointer on specific platform.

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

→ JIT (Just-In-Time) compiler is a component of the runtime environment that improves the performance of Java app. by compiling bytecodes to native machine code at run time Java program consists of classes, which contain platform-neutral bytecodes that can be interpreted by a JVM on many different computer architectures. At run time, the JVM loads the class files, determines the semantics of each individual bytecode, and perform the appropriate computation. The additional processor and memory usage during interpretation means that a Java application performs more slowly that a native application. The JIT complier helps improve the performance of Java programs by compiling bytecodes into native machine code at runtime.

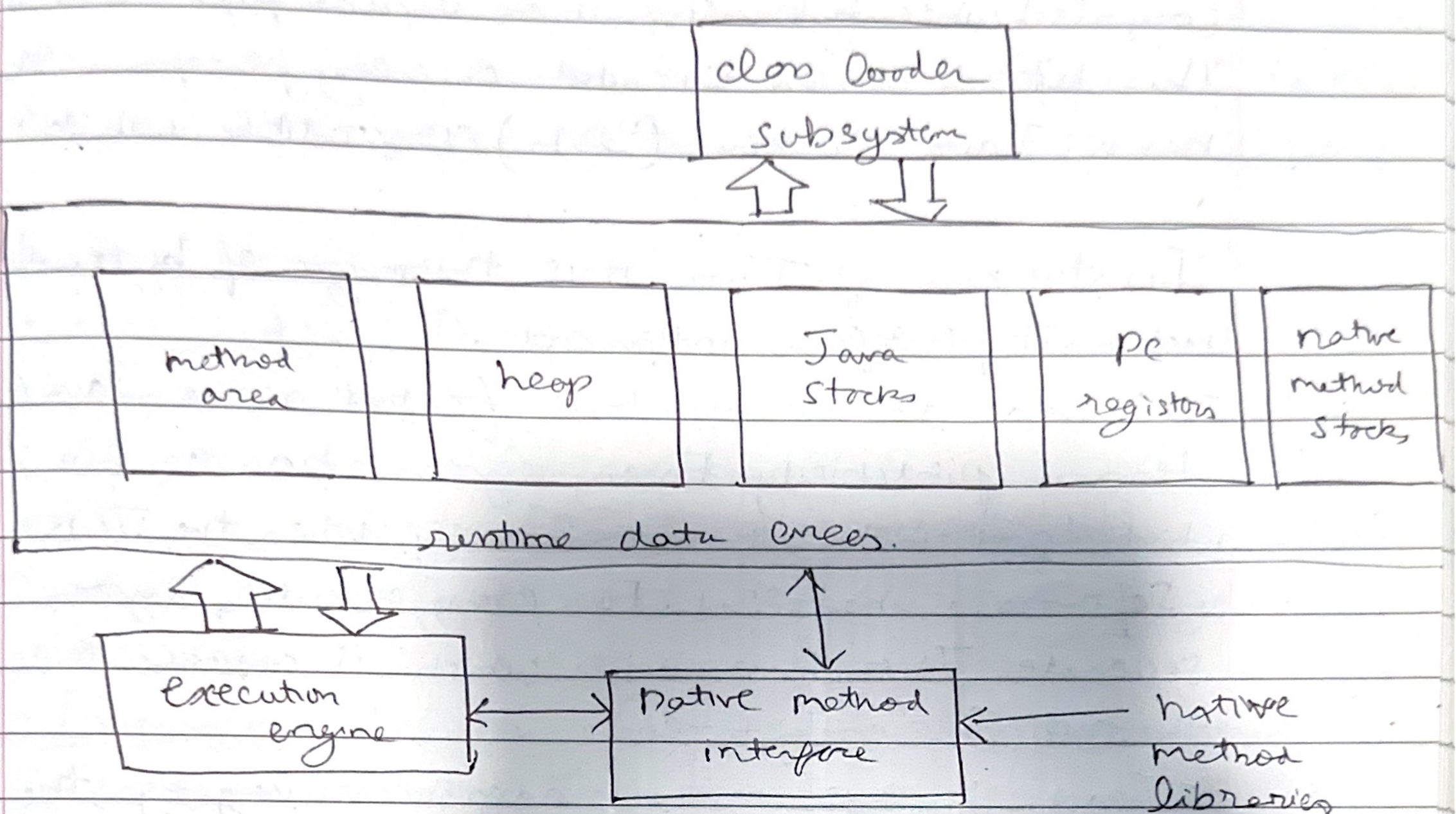
The JIT Compiler is enabled by default. When a method has been compiled, the JVM calls the compiled code of that method directly instead of interpreting it. Theoritically, if compilation did not require processor time and memory usage, compiling every method could allow the speed of the Java program to match that of a native application.

For each method, the JVM maintains an invocation count, which starts at a predefined compilation threshold value and is decremented everytime the method is called. When the invocation count reaches zero, a just-in-time compilation for the method is triggered. Therefore, often-used methods are compiled soon after the JVM has started, and less-method are complied much later, or not ~~at~~ at all.

Javabyte code is the instruction set for the Java Virtual Machine. It acts similar to an assembler which is an close representation of C++ code As Soon as a Java program is complied, java bytecode is generated. In more Apt terms java byte code is the machine code in the form of .a.class file. With the help of java byte code we achieve platform independance in java.

PAGE No.
DATE / /

6) Describe the architecture of the JVM
JVM stands for Java Virtual Machine. It is a crucial component of the Java platform and serves as an execution environment for Java bytecode.



Class loader: It is responsible for loading Java class files into the JVM at runtime. It performs tasks such as locating and reading class files, verifying their bytecode.

Runtime Data Area: The Runtime Data Area is the memory area where the JVM manages data during program execution. It consists of several components.

Method Area: The method Area stores class-level data including the bytecode of the method, constant pool, static variables.

Heap: The heap is the runtime data area where objects are allocated. It is divided into young gen and new gen.

Java Stack: Each thread in the JVM has a Java Stack that stores method specific data, including local variables, method arguments and method invocation records.

PAGE No.
DATE / /

7) What does Java achieve plateform independence through the JVM?

→ Java is plateform-independent because it uses "Write Once, Run anywhere" approach. Java source code is compiled into bytecode, which is plateform-neutral. This bytecode can be executed on any plateform that has a Java Machine (JVM) compatible with that bytcod

In the case of Java, it is the magic of bytecode that makes it plateform-indpendant.
This adds to an important feature is the Java language toward portability. Every system has its own JVM which gets installed automatically when the JDK Software is installed. For every operating System seperate JVM is available which is capable to read the .class file or byte code.
Java is a plateform-independent language, the JVM is platform-dependant.

PAGE No
DATE / /

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

→ The Class loader in Java is ~~called~~ crucial component of the Java Runtime Environment (JRE) and plays a central role in loading classes at runtime. It is responsible for locating and loading class files into memory so that they can be executed by the Java Virtual Machine (JVM)

Role of classLoader

- Class loading - The primary role of the ClassLoader is to load Java Class and interfaces into the JVM when they are needed for execution.
- Dynamic Loading - Class loading occurs dynamically as classes are referenced or needed during program execution.
- Namespace Isolation - ClassLoader helps maintain namespace isolation. Each ClassLoader creates a seperate namespace for loaded classes, preventing naming conflicts betⁿ classes loaded by different ClassLoader.
- Security - ClassLoader play a crucial role in Java's security model. They enable the enforcement of access control and security polices by controlling which classes can be loaded and executed.

Garbage collection: is a key feature for developers who build and compile Java programs on a Java Virtual Machine or JVM. Java objects are created on the heap, which is a section of memory dedicated to a program. When objects are no longer needed, the garbage collector finds and tracks these unused objects and deletes them to free up space. Without garbage collection, the heap would eventually run out of memory, leading to a runtime OutOfMemoryError.

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

→ In Java, access modifiers are keywords used in object-oriented programming to set the accessibility level for classes, variables, methods, and constructors.

1) Private: The private access modifier is the most restrictive level of access control. When a member is declared private, it can only be accessed within the same class. No other class, including subclasses in the same package or different packages can access the member. This modifier is typically used for sensitive data that should be hidden from external code.

2) Default (no modifier)

When no access modifier is specified, Java uses a default access level, often called package-private. This means the member is accessible only within classes in the same package. It is less restrictive than private but more restrictive than protected and public.

3) protected

The protected access modifier is less restrictive than private and default. Members declared as protected can be accessed within the same package or in subclasses in different packages. This is particularly useful in cases where you want to hide a member from the world but still makes it available to child classes.

4) public

The public access modifier is the least restrictive and specifies that the member can be accessed from any other class anywhere, whether within or in a different package. This access level is typically used for methods and variables that must be available consistently to all other classes.

10) What is d

→ Acces

~~Modifier~~ Mod

publ

prot

no

pri

Public

It

cl

Prote

with

the

cl

Def

o

a

Pr

PAGE No.
DATE / /

10) What is difference between public, protected and default acc.

→ Access Levels

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

Public : The access level of a public modifier is everywhere It can be accessed from within the class, outside the class, within the package and outside the package

Protected : The access level of protected modifier is within the package from the outside the package through the 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. If you do not specify any access level, it will be the default.

Private : The access level of private modifier is only within the class. It cannot be accessed from outside the class.

PAGE NO.
DATE / /

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

→ One cannot override a method which already has restrictive access modifier in a subclass.

The following rules for inherited method are enforced :-

- Method declared in public in a superclass also must be public in all subclasses.
- Method declared protected in a superclass must either be protected in a superclass must either be protected or public in subclasses; they cannot be private
- Methods declared private are not inherited at all, so there is no rule for them.

PAGE No
DATE / /

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

→ Default :-

If no modifiers are used, it is treated as default. The default is accessible only within package. It cannot be accessed from outside the package. It provides more accessibility than private. But it is more restrictive than protected, and public

Protected

The protected access modifier is accessible within package and outside the package but through inheritance only. The protected access modifier can be applied on the data It provides more accessibility than the default modifier.

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

→ We cannot declare top level class as private.
Java allows only public and default modifiers for top level classes in java. Inner classes can be private.
If ~~you~~ we declare a class as public, that class name should be file name.
We can declare a class with private access specifier. But not in main class.

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

→ No, a top-level class in Java cannot be declared 'protected' or 'private'. Top-level classes can only have two possible access modifiers.

- public: The class is accessible from any other, regardless of the package.
- default: (no modifier, also known as package-private) The class is accessible only to other classes within the same package.

Java restricts top-level classes to either public or default access to maintain clear visibility and accessibility rules. The language design enforces this so that top-level classes are either globally accessible (public) or package-restricted (default access).

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

→ In Java, if you declare a variable or method as private in a class, it is only accessible within that class. This means that any attempt to access the private variable or method from another class, even if the classes are in the same package, will result in a compile-time error.

Key points

- 'private' members of a class are not accessible outside the class in which they are declared, regardless of whether the other class is in the same package or a different one.
- Access to private members is restricted to the declaring class only.

PAGE NO.
DATE / /

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

→ In Java, "package-private" (also known as "default" access) refers to the access level when no explict access modifier is specified. When a class, method, or variable is declared without an access modifier, it is only accessible within the same package. It cannot be accessed from classes outside the package, even if those classes are in a subclass or in the same project.