Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2024-25

Class:	SE	**Semester:**	III
Course Code:		**Course Name:**	OOPJ

Name of Student:	AMAN MEHTAR
Roll No. :	32
Assignment No.:	01
Title of Assignment:	Fundamentals of OOPJ
Date of Submission:	16-08-24
Date of Correction:	

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Completeness	5	5
Demonstrated Knowledge	3	3
Legibility	2	2
Total	10	10

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Completeness	5	3-4	1-2
Demonstrated Knowledge Legibility	3	2	1
Legibility	2	1	0

Checked by

Name of Faculty : MS NEHA RAUT

Signature :

Date :

Q1 Differentiate between Object-oriented Programming and Procedure oriented Programming.

→ * Object-Orientd Programming (OOP)

1. Concept : OOP is based on the concept of objects, which are instances of classes. A class defines a blueprint for objets, encapsulating data (attributes) and behaviors (methods).

2. Structure : Code is organized around objects and classes. It emphasizes grouping related data and functions together.

3. Key Principles :
   - Encapsulation : Bundling data and methods that operate on the data within a single unit (class).
   - Abstraction : Hiding the complex implementation details and showing only the necessary features.
   - Inheritance : A way to form new classes that have (using classes) already been defined.
   - Polymorphism : The ability to redefine methods in derived classes, allowing for different behaviours.

4. Reusability : OOP promotes code reusability through inheritance and use of objects.

5. Data Handling : Focuses on data rather than functions, meaning data is tightly integrated with the functions that operate on it.

6. Examples : Common OOP languages include Java, C++, Python and C#.

Q2

* Procedure - Oriented Programming (POP)

1. Concept : POP is based on the concept of procedures , which are also known as routines or functions. These are blocks of code that perform a specific task.
2. Structure : Code is organized into functions , and the program is a ~~function~~ sequence of function calls.
3. Key characteristics :
   - Modularization : The program is divided into smaller , manageable sections called functions or procedures.
   - Sequential Execution : Functions are called in a sequence to perform tasks.
4. Reusability : Functions can be reused , but the reuse is limited as compared to OOP , as functions are independent and not associated with objects or classes.
5. Data Handling : Focuses on functions rather than data. Data is often passed from one function to another leading to less emphasis on data encapsulation.
6. Examples : Common POP languages include C, Pascal and fortran.

* Summary :

- OOP is centered around objects and classes, promoting encapsulation, inheritance, and Polymorphism, whereas POP is centred around functions or procedures, emphasizing sequential execution and modularization.

Q2. Explain different features of Java. and a role of JVM with a neat labelled diagram.

→ * Features of Java.

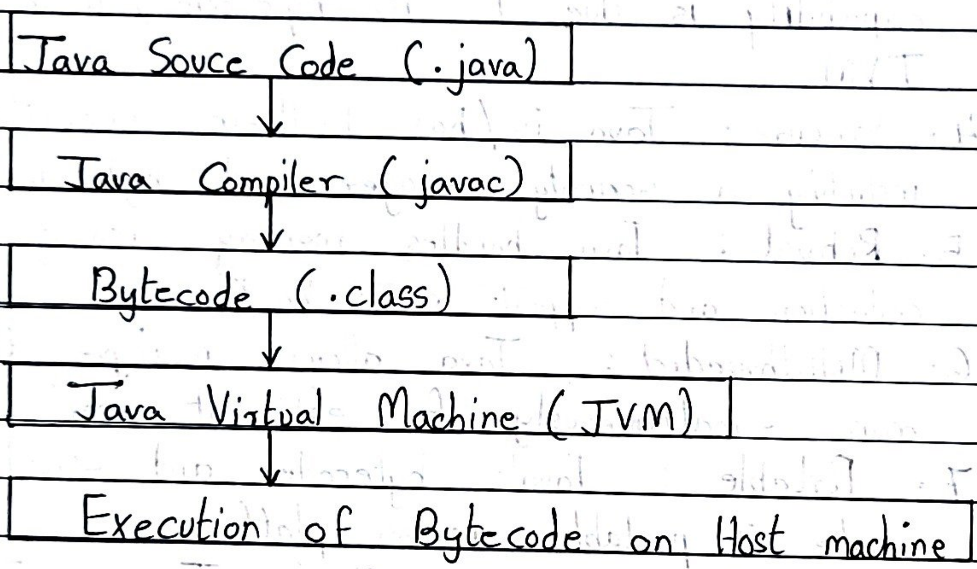
1. Simple : Java is easy to learn and uses a syntax similar to C++, but without complex features like pointers.
2. Object - Oriented : Everything in Java is an object, supporting OOP principles like inheritance and encapsulation.
3. Platform - Independent : Java's "Write Once, Run Anywhere" capability is due to its bytecode, which runs on any JVM.
4. Secure : Java is / has built-in security features, including a security manager and bytecode verification.
5. Robust : Java handles memory management with garbage collection and supports error handling.
6. Multithreaded : Java allows multiple threads to run simaltaneously for efficient multitasking.
7. Portable : Java's bytecode and standardized libraries make it portable across platforms.
8. High Performance : Java's Just - in - Time (JIT) compiler enhances performances by optimizing bytecode execution.
9. Distributed : Java supports distributed computing with built-in networking capabilities.
10. Dynamic : Java can dynamically load classes and manage runtime data.

* Role of Java Virtual Machine (JVM)

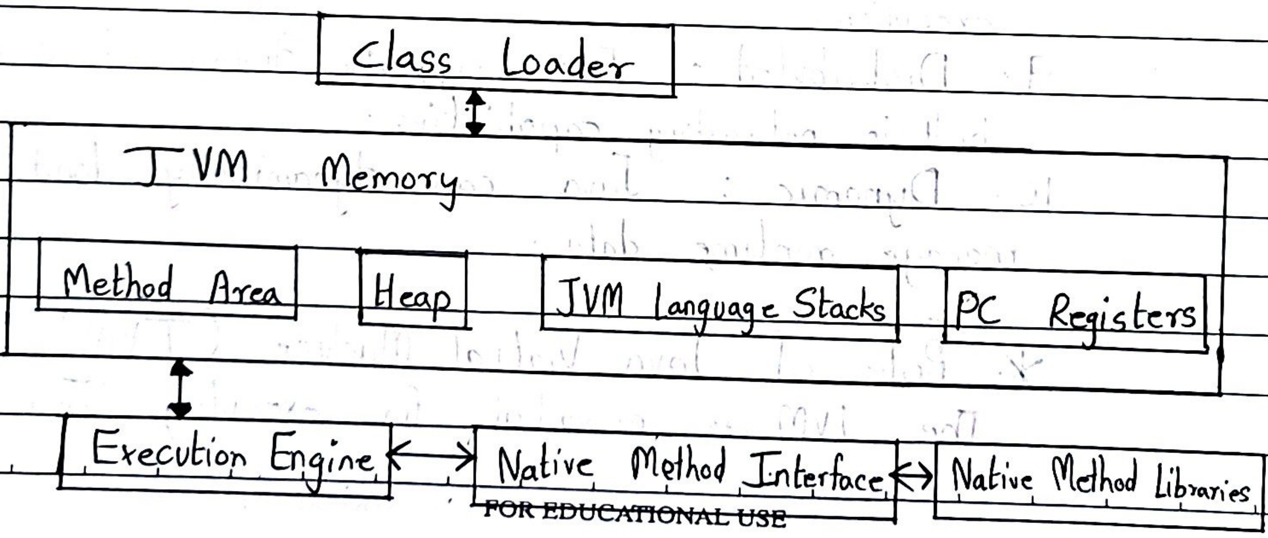
The JVM is essential for executing Java programs.

It interprets java bytecode, ensuring platform independ by allowing the same bytecode to run on any system with a compatible JVM. The JVM also manages memory via garbage collection, enforces security with bytecode verification, optimizes performance through Just-in-time (JIT) compilation.

* Diagram of Java Program Execution:


Java Souce Code (.java)
Java Compiler (javac)
Bytecode (.class)
Java Virtual Machine (JVM)
Execution of Bytecode on Host machine

* Diagram of Java Virtual Machine (JVM)



Q3) Some examples of data to be stored are listed below with the data type. Mention the data type that will be best suited for them.

→

(i) Age in Years.

⇒ Integer (int)

(ii) Rate of Interest.

⇒ Floating-point number (float / double)

(iii) Area of Circle.

⇒ Floating-point number (float / double)

(iv) Runs made by a batsman.

⇒ Integer (int)

(v) User input as 'true' or 'false'.

⇒ Boolean (boolean)