U.S.N.										

B.M.S.College of Engineering, Bengaluru-560019

Autonomous Institute Affiliated to VTU

January / February 2025 Semester End Main Examinations

Programme: B.E. **Semester:III**

Branch: Artificial Intelligence and Machine Learning **Duration: 3 hrs.**

Course Code: 24AM3PCOOP **Max Marks: 100**

Course: Object Oriented Programming

Instructions: 1.Answer any FIVE full questions, choosing one full question from each unit.
2. Missing data, if any, may be suitably assumed.

Important Note: Completing your answers, compulsorily draw diagonal cross lines on the remaining blank pages. Revealing of identification, appeal to evaluator will be treated as malpractice.

		UNIT - I	*CO*	*PO*	Marks
1	a)	Define the terms "class" and "object" in object-oriented programming. Provide a simple example of each.	*CO1*	*PO1*	**06**
	b)	Write a program to define a class Student with attributes name, roll_number, and marks. And Implement the following: i. A constructor to initialize the attributes. ii. A method to calculate the grade based on marks (e.g., A for marks ≥ 90, B for 75 ≤ marks < 90, etc.). iii. Create multiple student instances, initializing their details, and calculating their grades. iv. A method to display the student details and their grades. v. Write the expected output of the program.	*CO2*	*PO1*	**10**
	c)	Discuss the role of the self keyword in Python class methods. Provide an example to access instance attributes.	*CO1*	*PO1*	**04**
		OR			
2	a)	Explain the concept of an abstract machine and its role in programming language design.	*CO1*	*PO1*	**06**
	b)	Consider a class BankAccount with attributes account_number, holder_name, and balance. And implement methods for the following: i. Deposit an amount. ii. Withdraw an amount (check for sufficient balance). iii. Display account details. iv. Create multiple accounts and demonstrate deposit, withdrawal, and balance display operations.	*CO2*	*PO1*	**10**
	c)	Differentiate between Procedural oriented and Object-oriented programming approaches.	*CO1*	*PO1*	**04**
		UNIT - II			
3	a)	Explain how encapsulation improves data security and code readability in Object-oriented programming with an example.	*CO1*	*PO1*	**06**

		b)	Consider a banking system where customer account details like balance must be securely handled. With this answer the following: i. Design a class BankAccount using encapsulation to protect the balance variable and provide controlled access to deposit and withdraw money. ii. Justify why private instance variables are critical in scenarios involving sensitive data.	*CO2*	*PO1*	**10**
		c)	Differentiate between Encapsulation and Data hiding. Why is encapsulation considered a broader concept than data hiding?	*CO3*	*PO2*	**04**
			OR			
	4	a)	Discuss the importance of abstraction in large-scale software development. Illustrate your answer with a practical scenario.	*CO1*	*PO1*	**06**
		b)	A vehicle manufacturing company uses an abstraction to define different types of vehicles. With this answer the following: i. Create an abstract class Vehicle with abstract methods like start() and stop(). ii. Extend this class in Car and Bike to provide specific implementations. iii. Explain how abstraction helps in this scenario.	*CO2*	*PO1*	**10**
		c)	Explain the significance of abstraction in designing large-scale systems. How does it help in managing complexity?	*CO3*	*PO2*	**04**
			UNIT - III			
	5	a)	Explain the purpose of the super keyword in Python. Illustrate its use with an example of method overriding in a derived class.	*CO1*	*PO1*	**06**
		b)	Write a program to demonstrate multiple inheritance: i. Define a base class Vehicle with attributes brand and speed. ii. Create derived classes Car and Bike with additional attributes like num_wheels and methods such as display_details().	*CO2*	*PO1*	**06**
		c)	Implement a class called Shape that has methods to draw different shapes (e.g., rectangle, circle) on the Pygame window. Each shape should have attributes for position and color. Demonstrate how to create instances of this class and draw them on the screen.	*CO3*	*PO2*	**08**
			OR			
	6	a)	Define multiple inheritance and explain the diamond problem in Python. Use an example to demonstrate how Python resolves it using the Method Resolution Order (MRO).	*CO1*	*PO1*	**06**
		b)	Create a Python program to show polymorphism with classes: i. Define a base class Shape with a method area() that returns 0. ii. Create two derived classes, Rectangle and Circle, that override the area() method to calculate and return the area of the respective shapes.	*CO2*	*PO1*	**06**
		b)	Write a Python program to create a base class Employee and a derived class Manager. Override a method to calculate bonuses for employees and managers using the super keyword.	*CO3*	*PO2*	**08**

			UNIT - IV			
	7	a)	Explain the difference between processes and threads in a multitasking environment. How do processes and threads interact with the operating system to achieve concurrency?	*CO1*	*PO1*	**06**
		b)	Consider 3 classes: Class A:Prints Fibonacci numbers from 1-15. Class B:Perform linear search on the given list for a key element. Class C:Prints special characters (! @ # $ %). Create threads to handle each class independently. Write the expected output.	*CO2*	*PO1*	**08**
		c)	Describe the various states in the thread life cycle. How does a thread transition between these states, and what events trigger each transition?	*CO3*	*PO2*	**06**
			OR			
	8	a)	Explain the thread pool for creating threads in an efficient way. Write the Python program to demonstrate the same.	*CO1*	*PO1*	**06**
		b)	Describe the occurrence of deadlock in operating system. And also develop Python code to implement synchronization during multithreading.	*CO2*	*PO1*	**08**
		c)	Illustrate the inter-thread communication in brief. Develop Python code for inter-thread communication using queue.	*CO3*	*PO2*	**06**
			UNIT - V			
	9	a)	In brief, explain different approaches to implement timers in Python.	*CO1*	*PO1*	**06**
		b)	Develop the python code related to managing memory used by objects for the followings concepts: i. Object lifetime. ii. Reference count (Incrementing and decrementing). iii. Managing memory slots	*CO3*	*PO2*	**08**
		c)	How to declare class variable constants. Also write the python code using Class variables for counting how many objects initiated from a class.	*CO3*	*PO2*	**06**
			OR			
	10	a)	Illustrate the counting frames approach for implementing Timers with a code snippet.	*CO1*	*PO1*	**06**
		b)	Analyze how Python's reference counting mechanism influences the object lifecycle. Provide an example demonstrating the release of memory when an object's reference count drops to zero.	*CO3*	*PO2*	**06**
		c)	Write a Python program to display time using countup timer which starts at zero and counts up indefinitely.	*CO3*	*PO2*	**08**

****