U.S.N.										

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

Autonomous Institute Affiliated to VTU

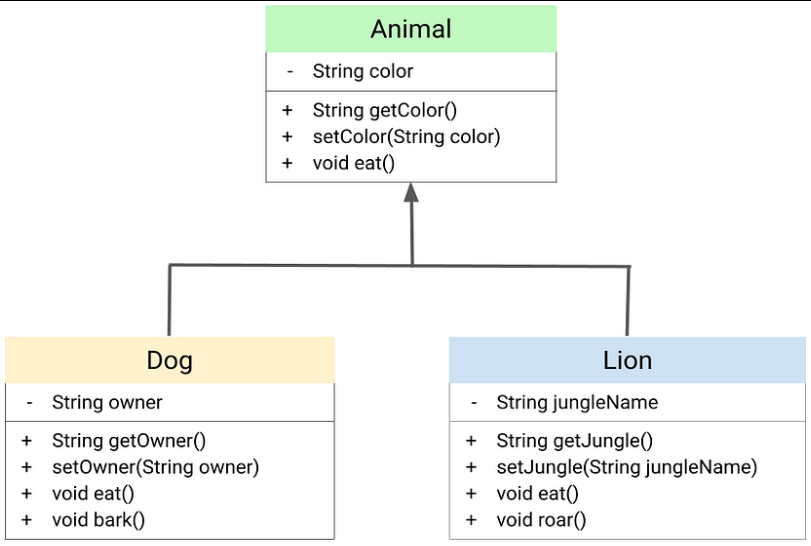
June 2025 Semester End Main Examinations

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.

		UNIT - I	*CO*	*PO*	**Marks**
1	a)	Explain the fundamental principles of Object-Oriented Programming (OOP) and how they are applied when modeling physical objects in Python.	*CO1*	*PO1*	**06**
	b)	Analyze the relationships between Classes and Objects in a given code snippet, identifying how they interact with each other?	*CO3*	*PO2*	**06**
	c)	Build a Python class called MathOperations to perform basic math calculations such as addition, subtraction, multiplication and division using the following methods: i. Static method ii. Instance method Create an instance of MathOperations and call each method to verify its functionality. For the static methods, show how to call them using both the class name and an instance of the class.	*CO2*	*PO1*	**08**
		OR			
2	a)	Explain the concept of abstract machine and interpreter with a neat diagram.	*CO1*	*PO1*	**06**
	b)	Build a Python class called Person that models information about a person. Create a method display() that prints the person's name and age.Test the class by: i. Creating a Person object using the default constructor. ii. Creating another Person object using the parameterized constructor. iii. Creating a third Person object using the non-parameterized constructor. iv. Calling the display() method for each object to confirm the values of the attributes.	*CO2*	*PO1*	**08**

		c)	Create a visual representation of an abstract machine that illustrates how it processes code.	*CO3*	*PO2*	**06**
			UNIT - II			
	3	a)	List and explain the benefits of Encapsulation with functions.	*CO1*	*PO1*	**05**
		b)	Outline the steps to create an Abstract Base Class in Python. Provide an example illustrating the use of the abc module.	*CO1*	*PO1*	**05**
		c)	Design a class called BankAccount to model a simple bank account. To protect sensitive data, ensure that certain attributes, like the account balance, cannot be accessed or modified directly from outside the class. And perform the following tasks: i. Create a class BankAccount with the following private instance variablesbalance and account_number ii. Implement methods in BankAccount to deposit, withdraw and get_balance iii. Test the class by creating a BankAccount instance and attempting to directly access and modify the private attributes. Then, use the provided methods to manage the account balance.	*CO3*	*PO2*	**10**
			OR			
	4	a)	Describe the role of abstract base classes in Object-oriented programming.	*CO1*	*PO1*	**05**
		b)	Differentiate between Encapsulation with functions and encapsulation with objects. What are the advantages of each?	*CO1*	*PO1*	**05**
		c)	Develop a Python application for a transportation system that involves different types of vehicles, each with unique characteristics. With this requirement, i. Create an abstract class called Vehicle with two abstract methods: start_engine() and stop_engine(). ii. Implement two concrete classes, Car and Bike, that inherit from Vehicle. Each class should provide specific implementations for start_engine() and stop_engine(). iii. Write code that creates instances of Car and Bike and calls their start_engine() and stop_engine() methods.	*CO3*	*PO2*	**10**
			UNIT - III			
	5	a)	Explain how the **super** keyword is used to access methods from a parent class.	*CO1*	*PO1*	**06**
		b)	Evaluate the role of polymorphism in reducing code duplication. Provide examples to support your argument.	*CO2*	*PO1*	**06**
		c)	Develop a Python program to demonstrate the given scenario:	*CO3*	*PO2*	**08**

			 Animal - String color + String getColor() + setColor(String color) + void eat() Dog - String owner + String getOwner() + setOwner(String owner) + void eat() + void bark() Lion - String jungleName + String getJungle() + setJungle(String jungleName) + void eat() + void roar()			
			OR			
	6	a)	Explain how polymorphism allows for different classes to be treated as instances of the same class through a common interface.	*CO1*	*PO1*	**06**
		b)	Using polymorphism develop a pygame which includes the shapes such as circle, square and triangle. In each shapes are represented as classes. For each shapes, compute the area and draw the corresponding shapes. Write the expected output for the program.	*CO3*	*PO2*	**08**
		c)	Write a program that uses magic methods to enable operator overloading for a custom class.	*CO2*	*PO1*	**06**
			UNIT - IV			
	7	a)	Explain the stages in the thread life cycle with a suitable diagram.	*CO1*	*PO1*	**06**
		b)	Given a scenario where multiple threads need to access a shared resource. Illustrate it using thread synchronization.	*CO2*	*PO1*	**06**
		c)	Illustrate the occurance of deadlock in operating system. And also develop Python code to implement synchronization during multithreading.	*CO3*	*PO2*	**08**
			OR			
	8	a)	Describe the role of thread pools in managing resources and improving performance.	*CO1*	*PO1*	**06**
		b)	Assess the implications of using daemon threads in a long-running application. What are the potential risks and benefits?	*CO2*	*PO1*	**06**
		c)	Consider 3 classes: Class A:Prints Fibonacci numbers from 1-15. Class B: perform linear search on the given list for a key element.	*CO3*	*PO2*	**08**

			Class C: prints special characters (! @ # $ %). Create threads to handle each class independently. Write the expected output.			
			UNIT - V			
	9	a)	Explain how reference counting contributes to the management of object lifetime.	*CO1*	*PO1*	**05**
		b)	Using countdown timer, implement a program that takes a user input for the timer duration and then counts down to zero, printing a message when the timer expires.	*CO3*	*PO2*	**10**
		c)	Illustrate the counting frames approach for Implementing Timers with code snippet.	*CO2*	*PO1*	**05**
			OR			
	10	a)	Describe the garbage collection process in Python and how it differs from manual memory management?	*CO1*	*PO1*	**05**
		b)	How to declare class variable constants?. Also write the python code using Class variables for counting how many objects initiated from a class.	*CO2*	*PO1*	**05**
		c)	Develop the python code related to managing memory used by objects for the following concepts: i. Object lifetime. ii. Reference count (Incrementing and decrementing). iii. Managing memory slots	*CO3*	*PO2*	**10**

******