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: Computer Science and Engineering **Duration: 3 hrs.**

Course Code: 23CS3PCOOJ / 19CS3PCOOJ **Max Marks: 100**

Course: Object Oriented Java 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)	Discuss the key features of Java that differentiate it from other programming languages.	*CO1*	*PO1*	**5**
	b)	Write a Java program that demonstrates the use of static variables, static methods, and final variables by implementing a simple Student class. Use a static variable to keep track of the total number of students created. Use a final variable to store the roll number of a student (which should be assigned only once). Use a static method to display the total number of students.	*CO1*	*PO1*	**8**
	c)	Discuss how garbage collection in Java improves memory management.	*CO1*	*PO1*	**7**
		OR			
2	a)	Create a MatrixOperations class with overloaded methods for adding matrices. Each overloaded addMatrices method should handle: Two 2x2 matrices represented as 2D arrays of int, Two 3x3 matrices represented as 2D arrays of int, Two matrices of any dimensions (where both matrices must have the same dimensions) represented as 2D arrays of int	*CO1*	*PO1*	**7**
	b)	Explain the different types of constructors in Java with suitable examples.	*CO1*	*PO1*	**5**
	c)	Write a Java program that demonstrates the concept of objects as parameters by implementing a BankAccount class. The class should have attributes: accountHolder (String), balance (double). Create a deposit() method that takes another BankAccount object as a parameter and transfers an amount from one account to another. Ensure that the method checks if the sender has a sufficient balance before transferring. Display the account details before and after the transaction.	*CO1*	*PO1*	**8**

			UNIT - II			
	3	a)	Create a parent class Employee with a method work() that prints "Employee is working." Derive two child classes Manager and Developer, overriding the work() method to print "Manager is managing" and "Developer is coding," respectively. Store objects of Employee, Manager, and Developer in an array of type Employee[]. Iterate through the array and call the work() method on each object to observe polymorphism in action.	*CO1*	*PO1*	**8**
		b)	Write a Java program that: Accepts a string from the user. Extracts a substring from index 2 to 6. Accepts another string from the user and compares both strings to check if they are equal.	*CO1*	*PO1*	**6**
		c)	Create a generic class named Storage<T>, where T represents the type of object stored. The class should have: A constructor that initializes the stored object. A method to retrieve and display the object value. Demonstrate the working of this generic class by creating objects of Storage<Integer>, Storage<String>, and Storage<Double>, then displaying their values.	*CO1*	*PO1*	**6**
			OR			
	4	a)	Create an abstract class Appliance with an abstract method turnOn() and a concrete method turnOff() that prints "Appliance is turned off." Implement two concrete classes, WashingMachine and Refrigerator, that override the turnOn() method to print "Washing Machine is now running" and "Refrigerator is now cooling," respectively. Use dynamic method dispatch to call the turnOn() and turnOff() methods for both appliances through a reference of the Appliance class	*CO1*	*PO1*	**8**
		b)	Write a Java program that: Accepts a string input from the user. Converts the string to uppercase and lowercase. Finds and displays the length of the string.	*CO1*	*PO1*	**6**
		c)	Write a Java program to implement a generic class that can store and display different types of objects. Create a generic class named Box<T>. The class should have: A constructor that initializes the object. A method to display the object value. Create objects of Box<Integer> and Box<String> and display their values.	*CO1*	*PO1*	**6**
			UNIT - III			
	5	a)	Create a package named shapes. Add two classes: Rectangle with protected methods for calculating area and perimeter. Circle with a public method to calculate the circumference. Write a main program in a different package to demonstrate access control.	*CO2*	*PO2*	**8**
		b)	Write a program that handles multiple exceptions using try-catch-finally block.	*CO2*	*PO2*	**6**
		c)	Explain the role of private and static methods in Java interfaces.	*CO2*	*PO2*	**6**

			OR			
	6	a)	Create a package named exceptions. Inside it, define a custom exception class InvalidInputException. Write another package main where you use this exception to validate user input (e.g., input must be a positive integer).	*CO2*	*PO2*	**7**
		b)	Explain the terminologies: try, catch, throw, throws, finally with example.	*CO2*	*PO2*	**6**
		c)	Develop a Java program that defines an interface for geometric shapes and implements it for different shapes like Circle, Rectangle, and Triangle.	*CO2*	*PO2*	**7**
			UNIT - IV			
	7	a)	Write a Java program where A thread reverses a given string. The main thread waits for the child thread to finish reversing the string and then prints the result.	*CO3*	*PO3*	**8**
		b)	Explain the concept of thread synchronization and how it helps in avoiding race conditions.	*CO3*	*PO3*	**6**
		c)	Write a Java program to demonstrate file handling using Java I/O streams.	*CO3*	*PO3*	**6**
			OR			
	8	a)	Write a Java program where One thread prints odd numbers from 1 to 10. Another thread prints even numbers from 1 to 10. Both threads alternate in printing numbers to ensure proper order.	CO3	PO3	8
		b)	Write a Java program that demonstrates the use of Byte Streams (FileInputStream and FileOutputStream) for reading and writing binary data to a file.	*CO3*	*PO3*	**6**
		c)	Discuss the differences between isAlive() and join() methods in thread management.	*CO2*	*PO2*	**6**
			UNIT - V			
	9	a)	Differentiate between "Event Source" and "Event Listener" in Java's event-handling mechanism with an example.	*CO2*	*PO2*	**8**
		b)	Explain the event delegation model and its benefits in event handling.	*CO1*	*PO1*	**6**
		c)	Discuss the differences between MouseListener and MouseMotionListener interfaces.	*CO2*	*PO2*	**6**
			OR			
	10	a)	Explain the delegation event model with a neat diagram.	*CO1*	*PO1*	**6**
		b)	Write a java program using the abstract window toolkit to draw a circle using only arcs and the circumference is blue in colour.	*CO3*	*PO3*	**6**
		c)	Write a java program that can incorporate any five MouseListener methods to develop a cohesive program.	*CO3*	*PO3*	**8**

****