SHREE SWAMI ATMANAND SARASWATI INSTITUTE OF TECHNOLOGY

COMPUTER ENGINEERING DEPARTMENT

Practical List

Subject Name & Code: Object Oriented Programming (BE04000231)
Year (Semester): 2nd Year (4th Semester)
Academic Year: Jan-2026. **Term Duration:** 01/01/2026 to 06/05/2026

Sr. No.	List of Experiment
	Practical Set-1 (Basics of JAVA) : CO 1
1	Simulate a simple ATM or cashier. Given an integer amount to be dispensed (e.g., 787), calculate and display the minimum number of currency notes of denominations 100, 50, 10, 5, 2, and 1 that would be given to the user.
2	Write a Java program that accepts a five-digit integer from the keyboard. Your program should then create a new number by adding one to each digit of the input number. For example, if the input is 12391, the output should be 23402 (note: 9+1=10, so it becomes 0 with a carry)
3	Write a Java program that accepts two numbers as command-line arguments. Convert these arguments to appropriate numeric types (e.g., int or double), perform a simple calculation (e.g., sum or product), and print the result to the console.
	Practical Set-2 (Basic OOP concepts-1) : CO 2
4	Design a class Time with hours (int) and minutes (int) as data members. Include method setTime(int h, int m) to initialize the time and displayTime() to display the time. Implement a method addTime(Time t1, Time t2) that takes two Time objects as arguments, adds their hours and minutes, and stores the result in the calling object. Do not use constructors for initialization in this specific practical.
5	Create a class BankAccount with accountId (String), accountHolderName (String), and balance (double) as instance variables. Include methods assignValues() (for initialization) and displayValues(). Implement a search function that takes an accountId as input and, if found within an array of BankAccount objects, displays the details of that specific account. In your main method, create an array of at least five BankAccount objects and demonstrate adding, displaying, and searching for accounts.
6	Write a program for billing system for a shopping mall. Create a class BillGenerator that uses method overloading to generate bills based on customer type: generateBill(int itemTotal): For regular customers, apply no discount. generateBill(int itemTotal, int discount): For privileged customers, apply flat discount in rupees. generateBill(int itemTotal, double discountPercent): For festive offers, apply percentage discount. Write a program to display the final bill amount using appropriate overloaded method based on customer category.
7	A bank wants to offer a facility to calculate EMI (Equated Monthly Installment) for different types of loans. Design a class LoanCalculator with the following overloaded methods: calculateEMI(int principal, int time, float rate): For home loans calculateEMI(double principal, int time, double rate): For vehicle loans calculateEMI(int principal, int time): For short-term personal loans with a fixed interest rate of 10% Demonstrate the use of all three methods in the main method by calculating EMIs for different loan types.
	Practical Set-3 (Basic OOP concepts-1) : CO 2
8	Create a Java class named University with a static data member totalStudents to keep track of the number of student objects created. Implement a static method getTotalStudents(). Also, include a static block to initialize a static variable (e.g., universityName) and an instance block to print a message when an object is created. Demonstrate their execution order.
9	Create a Java method isValidPassword(String password) that checks if a given string is a valid password based on the following rules: ● It must have at least eight characters. ● It must consist only of letters and digits. ● It must contain at least two digits. The program should prompt the user to enter a password and display "Valid Password" or "Invalid

	Password" accordingly.
Practical Set-4 (Inheritance & Polymorphism) : CO 2	
10	Create a base class named Vehicle that contains common attributes such as vehicleNumber, brand, and fuelType. Include a constructor to initialize these fields and a method displayDetails() to print them. Derive a subclass Car from Vehicle which adds attributes such as numberOfSeats and ACavailable (boolean). Override the displayDetails() method to include the car-specific details, and use the super keyword to invoke the parent class constructor and methods. Further, derive another subclass ElectricCar from Car that includes attributes such as batteryCapacity and chargingTime, and again override the displayDetails() method to include electric car-specific details. Demonstrate constructor chaining, method overriding, use of protected access specifier for inherited members, and instanceof operator to check object type at runtime. In the main() method, create objects of all three classes and display their details using overridden methods. Also, use upcasting (Vehicle v = new Car(...)) and downcasting with instanceof check to access subclass-specific features.
11	Write a Java program to demonstrate method overriding in an online payment system. Create a superclass Payment with a method processPayment(int amount). In the subclass CreditCardPayment, override the method to print "Payment of Rs.<amount> done using Credit Card". In the subclass UPIPayment, override the method to print "Payment of Rs.<amount> done using UPI". Accept user choice and amount, and display the payment result using method overriding.
Practical Set-5 (Interface, Abstract Class and Package) : CO 2	
12	Design a Java program for an Online Order Processing System using partial interface implementation. Create an interface Order with three methods: placeOrder(String item, int qty) ,cancelOrder(int orderId) ,generateBill() Create an abstract class PartialOrder that implements the Order interface but provides implementation only for placeOrder() (storing order details). Create a concrete class FinalOrder that extends PartialOrder and implements the remaining methods cancelOrder() and generateBill(). Accept user input for order details and allow user to either generate a bill or cancel the order.
13	Write a Java program using four different packages to demonstrate the use of access specifiers. ● Package apack: o Define a class A with three variables: ▪ public int pubVar ▪ protected int protVar ▪ private int privVar o Provide a constructor to initialize them. ● Package bpack: o Define a class B that extends A. o Create a display() method that tries to access variables of A using inheritance. ● Package cpack: o Define a class C with a display() method. o Inside display(), create an object of class A and try to access its variables. ● Package dpack: o Define a class ProtectedDemo with main(). o Create objects of class B and class C. o Call their respective display() methods to show which variables are saccessible and which are not.
Practical Set-6 (Exception Handling) : CO 3	
14	Write a Java application that defines a method average(String[] values) which: ● Accepts an array of strings as an argument. ● Converts each string element into a double and computes the average. ● If any array element is null, the method should throw a NullPointerException. ● If any element is not a valid number (e.g., "abc"), it should throw a NumberFormatException. ● Include the throws clause in the method declaration. ● In the main method, demonstrate the working of this program with valid and invalid inputs using try-catch-finally
15	Write a Java program to create a Banking Application using classes and exception handling. Create a class BankAccount with: ● A constructor to initialize the balance with 1000.00. ● A method deposit(double amount) to add money to the account.

	● A method withdraw(double amount) that subtracts money from the account. ● If withdrawal is more than the available balance, it should throw a custom exception called NotSufficientFundException. In the main method: ● Deposit 1000.00. ● Perform three withdrawals: 400.00, 300.00, and 500.00. ● The last withdrawal should throw the exception with the message "Not Sufficient Fund".
Practical Set – 7 (Concurrency Control – Threading) : CO 3	
16	Write a Java program where: ● Thread 1 computes the sum of numbers from 1 to 1000. ● Thread 2 computes the sum of numbers from 1001 to 2000. ● Both threads should run in parallel. ● Each thread should return its computed sum to the main method (e.g., using a getter method or storing result in a shared variable). ● The main method should wait for both threads to finish using join(), then combine the two partial sums and print the final result.
Practical Set – 8 (File I/O) : CO 3	
17	Write a Java program that takes file name(s) and commands from the command line arguments and performs the following operations: ● Copy a file from source to destination. ● Delete a given file. ● Rename a file. After performing the operation, the program should print the following file properties: ● File name ● Absolute path ● File size (in bytes) ● Whether the file is readable/writable ● Last modified date Use File class methods for properties, and handle exceptions such as IOException and FileNotFoundException.
Practical Set – 9 (Collection Framework and Generics) : CO 5	
18	Create a generic class Box<T> with a method addItem(T item) that stores items in an ArrayList<T>. Demonstrate the class by: ● Creating a Box<String> for names. ● Creating a Box<Integer> for roll numbers. ● Display stored items for both.
19	Write a generic method sortList(List<T> list) that sorts elements of an ArrayList<T>, where T extends Comparable<T>. Demonstrate with: ● An ArrayList<Integer> of numbers. ● An ArrayList<String> of names. ● Display the list before and after sorting.
Practical Set – 10 (JAVA FX) : CO 4	
20	Create a JavaFX app that shows a circle which continuously bounces (left↔right) inside the scene. Provide: ● A Start and Stop button. ● A Slider to control the ball speed (min: slow, max: fast). ● When window is resized, the ball should keep bouncing within new bounds. UI / Classes to use: Pane or AnchorPane, Circle, TranslateTransition or AnimationTimer, Button, Slider. Following behavior is expected: ● Press Start → ball moves horizontally and reverses on boundaries. ● Move Slider while running → ball speed updates. ● Press Stop → animation pauses.