**22CS203 - OBJECT-ORIENTED**

**PROGRAMMING THROUGH JAVA**

**List of Lab Programs**

**Module 1**

**Unit – 1 Programs on OOP basics**

1. Create a class Rectangle. The class has attributes length and width. It should have methods that calculate the perimeter and area of the rectangle. It should have read Attributes method to read length and width from user.

Hint: Area of rectangle = length \* width, Perimeter of rectangle = 2\*(length+width).

1. Develop a Java application to generate Electricity bill. Create a class with the followingmembers: Consumer no., consumer name, previous month reading, current month reading,type of EB connection (i.e domestic or commercial). Compute the bill amount using thfollowing tariff.

If the type of the EB connection is domestic, calculate the amount to be paid as follows:

First 100 units – Rs. 1 per unit

101-200 units – Rs. 2.50 per unit

201 -500 units – Rs. 4 per unit

> 501 units – Rs. 6 per unit

If the type of the EB connection is commercial, calculate the amount to be paid as follows:

First 100 units – Rs. 2 per unit

101-200 units – Rs. 4.50 per unit

201 -500 units – Rs. 6 per unit

> 501 units – Rs. 7 per unit

1. Write a JAVA program to search for an element in a given list of elements using binarysearch mechanism.
2. Create a class Fact with attribute n and a method factorial. Develop a JAVA code to read n from user and find the factorial of a given number.
3. Write a program to display the employee details using Scanner class.
4. Develop a java application to implement currency converter (Dollar to INR, EURO toINR, Yen to INR and vice versa), distance converter (meter to KM, miles to KM and viceversa) , time converter (hours to minutes, seconds and vice versa) using switch case.
5. Implement a Java Program that reads a line of integers, and then displays each integer, and the sum of all the integers (use StringTokenizer class).
6. Implement a java program to print all tokens of a string on the bases of multiple separators (use StringTokenizer class).
7. Develop a java application with Employee class with Emp\_name, Emp\_id, Address,Mail\_id, Mobile\_no as members. Inherit the classes, Programmer, Assistant Professor,Associate Professor and Professor from employee class. Add Basic Pay (BP) as the memberof all the inherited classes with 97% of BP as DA, 10 % of BP as HRA, 12% of BP as PF,0.1% of BP for staff club fund. Generate pay slips for the employees with their gross and netsalary.

**Unit – 2**

**Inheritance, Interface & Packages**

1. Write a Java program to create a vehicle class hierarchy. The base class should be Vehicle, with subclasses Truck, Car and Motorcycle. Each subclass should have properties such as make, model, year, and fuel type. Implement methods for calculating fuel efficiency, distance traveled, and maximum speed. Use the super keyword to initialize the superclass attributes and invoke superclass methods where appropriate. Apply the principles of inheritance and method overriding.

**Constraints:**

**Fuel Efficiency:**

* + Truck: 15 miles per gallon
  + Car: 30 miles per gallon
  + Motorcycle: 50 miles per gallon

**Distance Traveled**:

Calculated as fuelConsumed \* calculateFuelEfficiency()

**Maximum Speed**:

* Truck: 120 mph
* Car: 150 mph
* Motorcycle: 180 mph

1. Develop a JAVA program to create an abstract class Shape with abstract methods calculateArea() and calculatePerimeter(). Create subclasses Circle and Triangle that extend the Shape class and implement the respective methods to calculate the area and perimeter of each shape.
2. You are required to implement a banking system with a hierarchy of classes representing different types of bank accounts. The base class should be Bank, containing a list of accounts and methods for adding new accounts. Create an Account interface with methods for depositing, withdrawing, calculating interest, and viewing balances.

Implement two types of accounts: SavingsAccount and CurrentAccount, both implementing the Account interface. Each account type should have its own unique methods and behaviors. Use the principles of interfaces, inheritance, and method overriding to create a comprehensive banking system.

**Constraints:**

**Bank Class**:

Should maintain a list of accounts.

Methods for adding accounts and viewing all accounts.

**Account Interface**:

* Methods: deposit(double amount), withdraw(double amount), calculateInterest(), viewBalance()

**SavingsAccount**:

* Interest rate: 3% per annum
* Additional method: applyInterest()

**CurrentAccount**:

* Overdraft limit: $500
* Additional method: checkOverdraft()

1. Create a package ‘student.Fulltime.Btech‘ in your current working directory

a. Create a default class student in the above package with the following attributes: Name, age, sex.

b. Have methods for storing as well as displaying.

1. You are tasked with designing a Java application to manage a library system. The system involves different types of library items such as Books and DVDs, each with specific attributes and behaviors. Implement inheritance and interface implementation using the extends and implements keywords to model these entities.

**Requirements**

**Library Item Interface:**

* + Create an interface LibraryItem with the following methods:
    - void checkout() - to handle the checkout process for the item.
    - void returnItem() - to handle the return process for the item.

**Book Class:**

* + Create a class Book that implements the LibraryItem interface.
  + Attributes:
    - String title (title of the book)
    - String author (author of the book)
  + Methods:
    - Implement the checkout() and returnItem() methods to specify how a book is checked out and returned.

**DVD Class:**

* + Create a class DVD that also implements the LibraryItem interface.
  + Attributes:
    - String title (title of the DVD)
    - String director (director of the DVD)
  + Methods:
    - Implement the checkout() and returnItem() methods to specify how a DVD is checked out and returned.

1. You are given an interface AdvancedArithmetic which contains a method signature int divisor\_sum(int n). You need to write a class called MyCalculator which implements the interface.

divisorSum function just takes an integer as input and return the sum of all its divisors. For example divisors of 6 are 1, 2, 3 and 6, so divisor\_sum should return 12. The value of n will be at most 1000.

**Sample Input**

**6**

**Sample Output**

I implemented: AdvancedArithmetic

12

1. Implement a Java program for the following

a) Creation of simple package.

b) Accessing a package.

**Exception Handling**

1. a). Implement a Java program to create a method that takes an integer as a parameter and throws an exception if the number is odd.

b). Implement a Java program that reads a list of integers from the user and throws an exception if any numbers are duplicates.

1. a.) Implement a Java program to create a method that takes a string as input and throws an exception if the string does not contain vowels.

b). Create a class which accepts a number as choice and returns the Month of the year. If the entered number is greater than 12 or less than 1 throw InvalidChoiceException. If the entered option is not a number throw NotANumberException.

1. Create an Animal class which have four methods (eat, sleep, swim, walk). Extend a class Fish that overrides walk method. Extend class Dolphin from Fish that overrides the walk method. If a walk method is called form Dolphin class, generate an exception (as fish can’t walk).
2. Implement a java program in which you have two Rides (Car, Boat) and two locations (Land, Water). Ask from the user about the ride and the location. If user selects ride a car and location water, then throw an exception. Similarly, if user selects ride a boat and location water, then also throw an exception.
3. There are N number of bottles and M number of glasses that will be filled from these bottles. Each bottle can fill up 5 glasses. You have to Implement JAVA code that will ask user to input N and M, and check if the bottles are enough to fill M glasses. If bottles are not enough throw an exception informing the user. (Total number of glasses that can be made from bottles = 5\*number of bottles).
4. Create a three-level hierarchy of exceptions. Now create a base-class A with a method that throws an exception at the base of your hierarchy. Inherit B from A and override the method so it throws an exception at level two of your hierarchy. Repeat by inheriting class C from B. In main (), create a C and up cast it to A, then call the method.
5. Implement a java program using multiple catch blocks. Create a class CatchExercise inside the try block declare an array a[] and initialize with value a[5] =30/5; . In each catch block show Arithmetic exception and ArrayIndexOutOfBoundsException.
6. Implement a java program that count how many prime numbers between minimum and maximum values provided by user. If minimum value is greater than or equal to maximum value, the program should throw a InvalidRange exception and handle it to display a message to the user on the following format: Invalid range: minimum is greater than or equal to maximum. (For example, if the user provided 10 as maximum and 20 as minimum, the message should be: Invalid range: 20 is greater than or equal to 10).

**Module – 2**

**Multi-Threading**

1. Develop a multi-function utility application in Java that consists of the following threads:

Thread 1: Prints numbers from 1 to 10 as part of a counting utility.

Thread 2: Prints the alphabet from A to J to simulate a letter generation function.

Thread 3: Prints the first 10 prime numbers, simulating a prime number generator tool.

Ensure that each thread runs concurrently and independently. Include appropriate user interface elements (such as console outputs) to display the tasks performed by each thread.

1. Create a Java application that simulates different states of a thread:

Begin with creating a new thread (New state).

Set the thread to runnable (Runnable state) and start the thread to transition it to the running state (Running state).

Use Thread.sleep() and synchronized blocks to simulate a waiting or blocked state.

Terminate the thread and showcase its termination (Terminated state).

Provide detailed console logs to visualize and explain the transitions between different states of the thread life cycle.

1. Develop a task executor application where threads are created by extending the Thread class. Each thread should perform a specific task, such as printing a message indicating its execution.
2. Develop another version of the task executor application where threads are created by implementing the Runnable interface. Each thread should also perform a specific task such as printing a message indicating its execution.

Compare and discuss the two approaches, highlighting the differences in implementation and use-case scenarios.

1. Implement a Java application that uses multiple threads to increment a shared counter:

Implement the application without any synchronization to showcase the race condition and potential incorrect final value of the counter.

Modify the application by adding synchronization using the synchronized keyword, ensuring that the final value of the counter is accurate.

Include detailed console outputs and comments to explain the race condition and how synchronization resolves it.

1. Create a priority-based task scheduler application in Java:

Develop multiple threads with different priorities (e.g., MIN\_PRIORITY, NORM\_PRIORITY, MAX\_PRIORITY).

Run the threads and observe their execution order and completion time.

1. Develop a producer-consumer simulation in Java:

Create a SharedResource class that holds shared data.

Implement a Producer thread that generates data and puts it in the SharedResource.

Implement a Consumer thread that retrieves and processes the data from the SharedResource.

Utilize wait(), notify(), and notifyAll() methods to manage synchronization and communication between the producer and consumer threads. Provide detailed comments and console outputs to explain how data flow and synchronization are managed.

1. Develop a Java application that simulates a simple data processing pipeline with:

DataBuffer Class: A buffer to hold a single piece of data shared between the producer and consumer.

DataProducer Thread: A thread that reads data (simulates reading data from a file or sensor) and places it into the shared buffer.

DataProcessor Thread: A thread that processes the data from the shared buffer (simulates data processing such as transformation or analysis).

Synchronization: Utilize basic synchronization mechanisms to ensure thread-safe operations.

**Java Collection Framework**

1. Create a Java program that manages a music playlist using an ArrayList. Each song should have a title, artist, and duration (in seconds). Implement the following functionalities:

* Add a new song to the playlist.
* Remove a song from the playlist by title.
* Display the total duration of the playlist.
* List all songs.
* Shuffle the playlist (randomize the order of songs).

Write the necessary classes and methods to support these operations.

1. Implement a program that manages a shopping list using a LinkedList in Java. Each node in the list should store an item name and its quantity. The program should support the following operations:

* Add an item and its quantity to the end of the list.
* Remove an item from the list by specifying its name.
* Update the quantity of an existing item.
* Display the entire shopping list.

Ensure the program handles edge cases like empty lists and non-existent items gracefully.

1. Create a ticket reservation system for a concert using Java's Vector. Each ticket request must store: Fan Name, Number of Tickets

The system should allow organizers to:

* Add a ticket request to the queue.
* Process requests in the order they were received (first-come, first-served).
* Display the current queue of requests.

1. Develop a Java program to find the intersection of two sets of student names. The program should:

* Take two sets of student names as input.
* Find and display the names that are common to both sets.

1. Create a Java program to count the frequency of each word in a given text string using a HashMap. The program should:

* Take a string of text as input.
* Use a HashMap to store each unique word along with its frequency.
* Display each word and its corresponding frequency at the end.

1. Develop a Java program to implement a student grading system using a HashMap. The program should allow you to:

* Add a student's name and their grade.
* Remove a student by their name.
* Update the grade of an existing student.
* Retrieve the grade of a student by their name.
* Display all students and their grades.

**Unit – 2**

**Java Swings**

1. Create a Java program that handles various mouse events, including when the mouse enters, exits, clicks, presses, releases, drags, and moves within the client area. Demonstrate how each event can be used to perform specific actions, such as changing the background color, displaying coordinates, or drawing shapes, to enhance user interaction in a simple graphical application.
2. Implement a java program using swings to design a multiple choice question having three options (use radio button), display the message using dialog box “Your answer is wrong” if the user selects wrong option otherwise display, “Your answer is correct."
3. Develop a Java program that handles various key events, including when a key is pressed, released, and typed within a client area. Demonstrate how each event can be used to perform specific actions, such as updating a display with the pressed key to enhance user interaction.
4. Design a Java application that uses Swing components to create a simple calculator with buttons for digits (0-9), arithmetic operations (+, -, \*, /), and an equals (=) button. Utilize event listeners to handle button clicks and display the result of arithmetic operations in a text field.
5. Develop a Java application that includes two JButton components labeled "Enable" and "Disable". Implement functionality so that clicking the "Enable" button enables a third JButton labeled "Action" and clicking the "Disable" button disables the "Action" button.
6. Implement a Java application for selecting a gender using JRadioButton (Male and Female). Implement functionality so that selecting a radio button updates a JLabel to display the selected gender.
7. Create a Java program that displays a list of cities (Delhi, Hyderabad, Kolkata. Mumbai) using ‘JList’. Implement functionality so that selecting a city from the list updates a ‘JLabel’ with additional information about the city, such as population and country.
8. Develop a Java program that displays a ‘JComboBox’ with a list of colors (e.g., Red, Green, Blue). Implement functionality to print the selected color to the console when a user selects an item from the dropdown.