**Practical List-**

1. Create a class “Student” having following instance variables and methods.

**Instance variables:** ID, Name, Branch and university

**Method:** setDetails() and showDetails().

The setDetails() method sets the values of ID, Name, Branch and University.

And showDetails() method shows the value of each field.

2. Write a Java Program to demonstrate the working of a banking-system

**Instance variables:** name, account\_no, amount

**Instance methods:**  deposit(), withdraw(), checkBalance(), insert() and display().

Here we can deposit and withdraw amount from our account using deposit() and withdraw() methods respectively.

The insert() method is to initialize state and display() method is to display state values.

3. Write a program to sum two numbers. Here inputs are provided through command line argument.

4. Create class Employee with following attributes and methods

ID, name, department and salary.

The setDetails() method sets the values of ID, name, department and salary.

And showDetails() method shows the value of each field.

**Note:** (i) Values must be entered through Scanner class.

(ii) Use proper constructor

(iii) Use “this” reference variable to avoid ambiguity.

5. Re-write program 1 with better memory management approach.

**Note:** use of static keyword

6. Apply following functions on the String "Java".

(i) Try to concat "Welcome" and write down your observation.

(ii) Find character at index 1

(iii) Find index of first 'a'.

(iv) Find index of second 'a'

(v) Compare "Java" to "JAVA"

(vi) Compare "Java" to "JAVA" ignoring the case

(vii) Find the index of first 'a' from last

7. Apply following functions on StringBuffer object "HELLO"

(i) Append "Java"

(ii) Insert "Java" at index 1

(iii) Replace with "Java" with characters between index 1 to 2

(iv) Delete characters between index 1 and 2

(v) Reverse the string "HELLO"

8. Create a class “Student” having following instance variables and methods.

Instance variables: ID, Name, Branch, city and university

While creating constructors with one, two, three, four and five arguments reuse the constructors by **construction chaining**

9. Create two dimensional integer array and insert, search and traverse this array.

**Note:** Use Scanner class to insert data.

10. Create a jagged array having three rows. Where 1st row contains 3 columns, 2nd row contains 4 columns and 3rd row contains 2 columns. Insert and traverse it.

11. Create a class “Shape” having area() method to calculate area. Overload the area() method for shapes like triangle, rectangle and circle.

12. Create a class “Bank” having method getRateOfInterest(). Create child classes as HDFC, SBI and PNB and override getRateOfInterest() and return interest rates as 4.0, 4.5 and 5% correspondingly.

**Use concept of Upcasting to implement this scenario.**

13. Create a package **pack1** having one class **C1** and one interface **I1**. Class **C1** has two methods **int sum(int, int)** and **int sub(int, int)**. The I1 has one method **int division(int, int)**. Create another package **pack2** having class **C2**. Reuse **C1** and **I1** in **C2** and show the results.

**Note: Use appropriate Access Modifiers as required.**

14. Write a program to divide two numbers with proper exception handlers.

15. Create LowBalanceException that occurs when user tries to withdraw some amount that is greater than his current bank balance. To withdraw you have to write a **void withdrawal(int amount)** method.

16. Write a program that reads from a text file byte by byte and writes in some another file. Write this program in an efficient way.

17. Write a program that reads from a text file char by char and writes in some another file. Write this program in an efficient way.

18. Write a program that reads from a text file line by line and writes on console.

19. Write a program that take your name from keyboard and writes in some text file.

20. Write a multithreaded program where three threads are there and printing the numbers from 1 to 10 concurrently.

21. Write a program to set and get the name of threads also set and get the priority of threads.

22. Write a class Display having void wish(String name) methods that wishes hello to given string name. Between printing hello and provided string name apply delay of 500 milliseconds. Suppose multiple threads are there and they are trying to access this wish() method concurrently on **same object** then irregular output will be there. Write this application in such a way so that output becomes regular.

23. Write a class Display having **synchronized void wish(String)** methods that wishes hello to given string name. Between printing hello and provided string name apply delay of 500 milliseconds. Suppose multiple threads are there and they are trying to access this wish() method concurrently on **different** **objects** then irregular output will be there. Write this application in such a way so that output becomes regular.

24. Write a class Customer having **balance** as a property and **void withdrawal(int amount)**, and **void deposit(int amount)** as instance methods. There are two threads, the first thread wants to withdrawal some amount and second thread wants to deposit some amount. Apply inter thread communication where, if withdrawal amount is higher than current balance then first thread will wait for second thread to deposit then resume the withdrawal.

25. Create a GUI for student’s information system. A GUI that asks all the relevant information’s related to a student.

26. Create a canvas having smiley face.

27. Write a JFrame having three textfields. The first two textfields refers to two numbers on which sum or subtraction will happen. The third textfield will show the result. There are two buttons “SUM” and “SUBTRACTION”. Once user will click on sum or subtraction buttons then the corresponding result will be displayed in result field.

28. Write a Java program that interacts with database. It enables to-

(a) Inserts the student name and roll number to database.

(b) Fetch records from table

(c) Modify the records

(d) Delete the records