

1. WAP that implements the concept of Encapsulation.

```
class Student {  
    private int roll;  
    private String name;  
  
    public void setData(int r, String n) {  
        roll = r;  
        name = n;  
    }  
  
    public void display() {  
        System.out.println("Roll: " + roll);  
        System.out.println("Name: " + name);  
    }  
}  
  
public class EncapsulationDemo {  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.setData(1, "Amit");  
        s.display();  
    }  
}
```

2. WAP to demonstrate concept of function overloading of Polymorphism.

```
class Add {  
    int sum(int a, int b) {  
        return a + b;  
    }  
    int sum(int a, int b, int c) {  
        return a + b + c;  
    }  
}  
  
public class FunctionOverloading {  
    public static void main(String[] args) {  
        Add a = new Add();  
        System.out.println(a.sum(10, 20));  
        System.out.println(a.sum(10, 20, 30));  
    }  
}
```

3. WAP to demonstrate concept of construction overloading of Polymorphism.

```
class Demo {
    Demo() {
        System.out.println("Default Constructor");
    }
    Demo(int a) {
        System.out.println("Parameterized Constructor: " + a);
    }
}

public class ConstructorOverloading {
    public static void main(String[] args) {
        new Demo();
        new Demo(100);
    }
}
```

4. WAP to use Boolean data type and print the Prime Number series up to 50.

```
public class PrimeSeries {
    public static void main(String[] args) {
        for (int i = 2; i <= 50; i++) {
            boolean prime = true;
            for (int j = 2; j < i; j++) {
                if (i % j == 0) {
                    prime = false;
                    break;
                }
            }
            if (prime)
                System.out.print(i + " ");
        }
    }
}
```

5. WAP to print first 10 number of the following series using do while loop  
0,1,1,2,3,5,8,13.

```
public class Fibonacci {
    public static void main(String[] args) {
        int a = 0, b = 1, c, i = 1;
        do {
            System.out.print(a + " ");
            c = a + b;
        }
```

```

        a = b;
        b = c;
        i++;
    } while (i <= 10);
}
}

```

6. WAP to check the given number is Armstrong or not.

```

import java.util.Scanner;

public class Armstrong {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int sum = 0, temp = n;

        while (n > 0) {
            int r = n % 10;
            sum += r * r * r;
            n /= 10;
        }

        if (sum == temp)
            System.out.println("Armstrong Number");
        else
            System.out.println("Not Armstrong");
    }
}

```

7. WAP to find the factorial of any given number.

```

import java.util.Scanner;

public class Factorial {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        long fact = 1;

        for (int i = 1; i <= n; i++)
            fact *= i;
    }
}

```

```

        System.out.println("Factorial = " + fact);
    }
}

```

8. WAP to sort the element of one dimensional array in ascending order.

```
import java.util.Arrays;
```

```

public class ArraySort {
    public static void main(String[] args) {
        int a[] = {5, 3, 1, 4, 2};
        Arrays.sort(a);
        for (int i : a)
            System.out.print(i + " ");
    }
}

```

9. WAP for matrix multiplication using input/output stream class.

```
import java.io.*;
```

```

public class MatrixMultiplication {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int a[][] = {{1,2},{3,4}};
        int b[][] = {{5,6},{7,8}};
        int c[][] = new int[2][2];

        for(int i=0;i<2;i++)
            for(int j=0;j<2;j++)
                for(int k=0;k<2;k++)
                    c[i][j] += a[i][k] * b[k][j];

        for(int i=0;i<2;i++){
            for(int j=0;j<2;j++)
                System.out.print(c[i][j]+" ");
            System.out.println();
        }
    }
}

```

10. WAP for matrix addition using input/output stream class.

```

public class MatrixAddition {
    public static void main(String[] args) {
        int a[][]={{1,2},{3,4}};
        int b[][]={{5,6},{7,8}};
    }
}

```

```

        for(int i=0;i<2;i++){
            for(int j=0;j<2;j++){
                System.out.print((a[i][j]+b[i][j])+" ");
                System.out.println();
            }
        }
    }
}

```

11. WAP for matrix transposes using input/output stream class.

```

public class MatrixTranspose {
    public static void main(String[] args) {
        int a[][]={{1,2},{3,4}};
        for(int i=0;i<2;i++){
            for(int j=0;j<2;j++){
                System.out.print(a[j][i]+" ");
                System.out.println();
            }
        }
    }
}

```

12. WAP for matrix transposes using input/output stream class.

```

import java.io.BufferedReader;
import java.io.InputStreamReader;

public class MatrixTranspose {
    public static void main(String[] args) throws Exception {

        BufferedReader br = new BufferedReader(
            new InputStreamReader(System.in));

        int a[][] = new int[3][3];

        System.out.println("Enter elements of 3x3 matrix:");

        // Input matrix
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                a[i][j] = Integer.parseInt(br.readLine());
            }
        }
    }
}

```

```

        System.out.println("\nOriginal Matrix:");
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                System.out.print(a[i][j] + " ");
            }
            System.out.println();
        }

        System.out.println("\nTranspose Matrix:");
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                System.out.print(a[j][i] + " ");
            }
            System.out.println();
        }
    }
}

```

13. WAP to check that the given string is palindrome or not.

```

import java.util.Scanner;

public class PalindromeString {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s = sc.next();
        String rev = new StringBuilder(s).reverse().toString();

        if (s.equals(rev))
            System.out.println("Palindrome");
        else
            System.out.println("Not Palindrome");
    }
}

```

14. WAP to arrange the string in alphabetical order

```

import java.util.Arrays;

public class SortString {
    public static void main(String[] args) {
        String s = "java";
        char ch[] = s.toCharArray();
    }
}

```

```

        Arrays.sort(ch);
        System.out.println(new String(ch));
    }
}

```

15. WAP for String class which perform all the methods of that class.

```

public class StringClassDemo {
    public static void main(String[] args) {

        String s1 = "Java Programming";
        String s2 = "JAVA";

        // length()
        System.out.println("Length: " + s1.length());

        // toUpperCase() and toLowerCase()
        System.out.println("Uppercase: " + s1.toUpperCase());
        System.out.println("Lowercase: " + s1.toLowerCase());

        // charAt()
        System.out.println("Character at index 5: " + s1.charAt(5));

        // substring()
        System.out.println("Substring: " + s1.substring(5));

        // equals() and equalsIgnoreCase()
        System.out.println("Equals: " + s1.equals(s2));
        System.out.println("Equals Ignore Case: " + s2.equalsIgnoreCase("java"));

        // compareTo()
        System.out.println("CompareTo: " + s1.compareTo(s2));

        // concat()
        System.out.println("Concat: " + s1.concat(" Language"));

        // contains()
        System.out.println("Contains 'Java': " + s1.contains("Java"));

        // replace()
        System.out.println("Replace: " + s1.replace("Java", "Core Java"));

        // indexOf()
    }
}

```

```

        System.out.println("Index of 'P': " + s1.indexOf('P'));

        // trim()
        String s3 = " Hello World ";
        System.out.println("Trimmed: " + s3.trim());

        // startsWith() and endsWith()
        System.out.println("Starts with 'Java': " + s1.startsWith("Java"));
        System.out.println("Ends with 'ing': " + s1.endsWith("ing"));
    }
}

```

16. WAP to calculate simple interest using the wrapper class.

```

import java.io.BufferedReader;
import java.io.InputStreamReader;

public class SimpleInterestWrapper {
    public static void main(String[] args) throws Exception {

        BufferedReader br = new BufferedReader(
            new InputStreamReader(System.in));

        // Using Wrapper Classes
        Integer principal;
        Double rate;
        Integer time;

        System.out.print("Enter Principal Amount: ");
        principal = Integer.valueOf(br.readLine());

        System.out.print("Enter Rate of Interest: ");
        rate = Double.valueOf(br.readLine());

        System.out.print("Enter Time (in years): ");
        time = Integer.valueOf(br.readLine());

        // Simple Interest formula

```



```
Double simpleInterest = (principal * rate * time) / 100;
```

```
System.out.println("Simple Interest = " + simpleInterest);
```

```
}
```

```
}
```

17. WAP to calculate simple interest using the wrapper class.

```
import java.io.BufferedReader;
```

```
import java.io.InputStreamReader;
```

```
public class SimpleInterestWrapper {
```

```
    public static void main(String[] args) throws Exception {
```

```
        BufferedReader br = new BufferedReader(  
            new InputStreamReader(System.in));
```

```
        // Wrapper class objects
```

```
        Integer principal;
```

```
        Double rate;
```

```
        Integer time;
```

```
        System.out.print("Enter Principal Amount: ");
```

```
        principal = Integer.valueOf(br.readLine());
```

```
        System.out.print("Enter Rate of Interest: ");
```

```
        rate = Double.valueOf(br.readLine());
```

```
        System.out.print("Enter Time (in years): ");
```

```
        time = Integer.valueOf(br.readLine());
```

```
        // Simple Interest calculation
```

```
        Double si = (principal * rate * time) / 100;
```

```
        System.out.println("Simple Interest = " + si);
```

```
}
```

```
}
```

18. WAP to calculate area of various geometrical figures using the abstract class.

```
import java.io.BufferedReader;
import java.io.InputStreamReader;

// Abstract class
abstract class Shape {
    abstract void area();
}

// Rectangle class
class Rectangle extends Shape {
    void area() {
        int l = 10, b = 5;
        System.out.println("Area of Rectangle = " + (l * b));
    }
}

// Circle class
class Circle extends Shape {
    void area() {
        double r = 7;
        System.out.println("Area of Circle = " + (3.14 * r * r));
    }
}

// Triangle class
class Triangle extends Shape {
    void area() {
        int base = 6, height = 4;
        System.out.println("Area of Triangle = " + (0.5 * base * height));
    }
}

// Main class
public class AreaAbstract {
    public static void main(String[] args) {
        Shape s;

        s = new Rectangle();
        s.area();
    }
}
```

```

        s = new Circle();
        s.area();

        s = new Triangle();
        s.area();
    }
}

```

19. WAP where single class implements more than one interface and with the help of interface reference variable user call the methods

```

// First interface
interface Printable {
    void print();
}

// Second interface
interface Showable {
    void show();
}

// Class implementing multiple interfaces
class Demo implements Printable, Showable {

    public void print() {
        System.out.println("Print method from Printable interface");
    }

    public void show() {
        System.out.println("Show method from Showable interface");
    }
}

// Main class
public class MultipleInterfaceDemo {
    public static void main(String[] args) {

        // Interface reference variable
        Printable p = new Demo();
        p.print();
    }
}

```

```

        Showable s = new Demo();
        s.show();
    }
}

```

20. WAP that use multiple catch statements within the try-catch mechanism

```
import java.util.Scanner;
```

```

public class MultipleCatchDemo {
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        try {
            System.out.print("Enter first number: ");
            int a = sc.nextInt();

            System.out.print("Enter second number: ");
            int b = sc.nextInt();

            int c = a / b; // May cause ArithmeticException
            System.out.println("Result = " + c);

        } catch (ArithmeticException e) {
            System.out.println("Error: Division by zero is not allowed");

        } catch (NumberFormatException e) {
            System.out.println("Error: Invalid number format");

        } catch (Exception e) {
            System.out.println("General Exception Occurred");
        }

        System.out.println("Program continues...");
    }
}

```

21. WAP where user will create a self-Exception using “throw” keyword.

```
import java.util.Scanner;
```

```
// User-defined exception
```

```
class InvalidAgeException extends Exception {  
  
    InvalidAgeException(String msg) {  
  
        super(msg);  
  
    }  
  
}
```

```
public class UserDefinedExceptionDemo {  
  
    public static void main(String[] args) {  
  
  
        Scanner sc = new Scanner(System.in);  
  
  
        try {  
  
            System.out.print("Enter your age: ");  
  
            int age = sc.nextInt();  
  
  
            if (age < 18) {  
  
                // throw user-defined exception  
  
                throw new InvalidAgeException("Age must be 18 or above");  
  
            }  
  
  
            System.out.println("You are eligible.");  
  
  
  
        } catch (InvalidAgeException e) {  
  
            System.out.println("Exception Caught: " + e.getMessage());  
  
        }  
  
}
```

```
}  
  
}
```

## 22. WAP for multithread using different methods of Thread class.

// Thread by extending Thread class

```
class MyThread1 extends Thread {  
    public void run() {  
        for (int i = 1; i <= 5; i++) {  
            System.out.println("Thread 1: " + i);  
            try {  
                Thread.sleep(500); // Pause for 0.5 seconds  
            } catch (InterruptedException e) {  
                System.out.println(e);  
            }  
        }  
    }  
}
```

// Thread by extending Thread class

```
class MyThread2 extends Thread {  
    public void run() {  
        for (int i = 1; i <= 5; i++) {  
            System.out.println("Thread 2: " + i);  
            try {  
                Thread.sleep(700); // Pause for 0.7 seconds  
            } catch (InterruptedException e) {  
                System.out.println(e);  
            }  
        }  
    }  
}
```

```
public class MultithreadDemo {  
    public static void main(String[] args) {
```

```
        MyThread1 t1 = new MyThread1();  
        MyThread2 t2 = new MyThread2();
```

```
        // Setting thread names  
        t1.setName("First Thread");  
        t2.setName("Second Thread");
```

```
// Starting threads
```

```
t1.start();
```

```
t2.start();
```

```
// Display thread details using Thread class methods
```

```
System.out.println("Thread 1 is alive: " + t1.isAlive());
```

```
System.out.println("Thread 2 is alive: " + t2.isAlive());
```

```
System.out.println("Thread
```

### 23. WAP for multithreading using synchronized () method of Thread class.

```
class Table {
```

```
    // Synchronized method to prevent thread interference
```

```
    synchronized void printTable(int n) {
```

```
        for (int i = 1; i <= 5; i++) {
```

```
            System.out.println(n + " x " + i + " = " + (n * i));
```

```
            try {
```

```
                Thread.sleep(400);
```

```
            } catch (InterruptedException e) {
```

```
                System.out.println(e);
```

```
            }
```

```
        }
```

```
    }
```

```
}
```

```
// Thread class 1
```

```
class MyThread1 extends Thread {
```

```
    Table t;
```

```
    MyThread1(Table t) {
```

```
        this.t = t;
```

```
    }
```

```
    public void run() {
```

```
        t.printTable(5);
```

```
    }
```

```
}
```

```
// Thread class 2
```

```
class MyThread2 extends Thread {
```

```
    Table t;
```

```
    MyThread2(Table t) {
```

```
        this.t = t;
```

```

    }
    public void run() {
        t.printTable(10);
    }
}

public class SynchronizedThreadDemo {
    public static void main(String[] args) {
        Table t = new Table();

        MyThread1 t1 = new MyThread1(t);
        MyThread2 t2 = new MyThread2(t);

        t1.start();
        t2.start();
    }
}

```

24. WAP for multithread for setting the priority of the thread and the print their

```

class MyThread extends Thread {
    MyThread(String name) {
        super(name);
    }
    public void run() {
        System.out.println(getName() + " is running with priority " + getPriority());
    }
}

public class ThreadPriorityDemo {
    public static void main(String[] args) {

        MyThread t1 = new MyThread("Thread-1");
        MyThread t2 = new MyThread("Thread-2");

        // Set priority
        t1.setPriority(Thread.MIN_PRIORITY); // 1
        t2.setPriority(Thread.MAX_PRIORITY); // 10

        t1.start();
        t2.start();
    }
}

```



25. WAP to create a package using command and one package will import another package

```
class MyThread extends Thread {
    MyThread(String name) {
        super(name);
    }
    public void run() {
        System.out.println(getName() + " is running with priority " + getPriority());
    }
}

public class ThreadPriorityDemo {
    public static void main(String[] args) {

        MyThread t1 = new MyThread("Thread-1");
        MyThread t2 = new MyThread("Thread-2");

        // Set priority
        t1.setPriority(Thread.MIN_PRIORITY); // 1
        t2.setPriority(Thread.MAX_PRIORITY); // 10

        t1.start();
        t2.start();
    }
}
```

26. WAP for reading and writing bytes to a file.

```
import java.io.*;

public class FileByteDemo {
    public static void main(String[] args) throws IOException {
        FileOutputStream fout = new FileOutputStream("data.txt");
        String s = "Hello World!";
        byte b[] = s.getBytes(); // Convert string to bytes
        fout.write(b);
        fout.close();

        FileInputStream fin = new FileInputStream("data.txt");
        int i;
        System.out.println("File Content:");
        while ((i = fin.read()) != -1) {
```

```

        System.out.print((char)i);
    }
    fin.close();
}
}

```

## 27. WAP for demonstrate the switch statement, continue and break

```

import java.util.Scanner;

public class SwitchContinueBreak {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        for (int i = 1; i <= 5; i++) {
            System.out.print("Enter a number (1-3): ");
            int choice = sc.nextInt();

            switch (choice) {
                case 1:
                    System.out.println("You chose 1");
                    break;
                case 2:
                    System.out.println("You chose 2");
                    continue; // skips the rest and goes to next iteration
                case 3:
                    System.out.println("You chose 3");
                    break;
                default:
                    System.out.println("Invalid choice");
            }
            System.out.println("End of iteration " + i);
        }
    }
}

```

## 28.WAP for creating a text file and perform write operation using FileWriter class.

```

import java.io.*;

public class FileReaderDemo {
    public static void main(String[] args) throws IOException {
        FileReader fr = new FileReader("data.txt");
        int i;
    }
}

```

```

        System.out.println("File Content:");
        while ((i = fr.read()) != -1) {
            System.out.print((char)i);
        }
        fr.close();
    }
}

```

## 29.WAP for perform read operation using FileReader class.

```

import java.io.FileReader;
import java.io.IOException;

public class FileReaderDemo {
    public static void main(String[] args) {
        FileReader fr = null;
        try {
            // Create FileReader object to read from file
            fr = new FileReader("data.txt"); // Make sure data.txt exists

            int i;
            System.out.println("File Content:");
            // Read character by character
            while ((i = fr.read()) != -1) {
                System.out.print((char) i);
            }
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            try {
                if (fr != null) fr.close(); // Close the file
            } catch (IOException e) {
                System.out.println(e);
            }
        }
    }
}

```

## 30.WAP for JDBC to display the records from the existing table

```

import java.sql.*;

public class JDBCSelectDemo {
    public static void main(String[] args) {
        try {
            // Load JDBC driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Establish connection
            Connection con = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/yourDB", "root", "password"
            );

            // Create statement
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM students");

            // Display records
            System.out.println("ID\tName\tAge");
            while (rs.next()) {
                System.out.println(rs.getInt(1) + "\t" +
                    rs.getString(2) + "\t" +
                    rs.getInt(3));
            }

            con.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}

```