1. **Write a Java program that accepts four integers from the user and prints equal if all four are equal, and not equal otherwise.**

   Algorithm :

   1. Prompt the user to enter the first integer and read it into a variable num1.
   2. Prompt the user to enter the second integer and read it into a variable num2.
   3. Prompt the user to enter the third integer and read it into a variable num3.
   4. Prompt the user to enter the fourth integer and read it into a variable num4.
   5. If num1 equals num2 and num2 equals num3 and num3 equals num4, then print "equal".
   6. Otherwise, print "not equal".

   Code :

```
import java.util.Scanner;

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

      System.out.print("Enter first integer: ");
      int num1 = input.nextInt();

      System.out.print("Enter second integer: ");
      int num2 = input.nextInt();

      System.out.print("Enter third integer: ");
      int num3 = input.nextInt();

      System.out.print("Enter fourth integer: ");
      int num4 = input.nextInt();

      if (num1 == num2 && num2 == num3 && num3 == num4) {
         System.out.println("equal");
      } else {
         System.out.println("not equal");
      }
   }
}
```

2. **Write a program to create a class Student2 along with two methods getData(), printData() to get the value through argument and display the data in printData. Create the two objects s1, s2 to declare and access the values from class STtest.**

Algorithm :

Define the Student2 class with three private instance variables: name (String), roll (int), and marks (double).
Define a getData() method that takes three parameters (name, roll, and marks) and sets the corresponding instance variables to the given values.
Define a printData() method that prints the values of the instance variables (name, roll, and marks).
In the main() method:
a. Create two instances s1 and s2 of the Student2 class using the new keyword.
b. Call the getData() method on each instance to set the values of the instance variables.
c. Call the printData() method on each instance to print the values of the instance variables.

Code :

```
public class Student2 {
   private String name;
   private int roll;
   private double marks;

   public void getData(String name, int roll, double marks) {
      this.name = name;
      this.roll = roll;
      this.marks = marks;
   }

   public void printData() {
      System.out.println("Name: " + name);
      System.out.println("Roll: " + roll);
      System.out.println("Marks: " + marks);
   }

   public static void main(String[] args) {
      Student2 s1 = new Student2();
      s1.getData("John Doe", 101, 85.5);
      s1.printData();
```

```
        Student2 s2 = new Student2();
        s2.getData("Jane Smith", 102, 91.2);
        s2.printData();
    }}
```

3. **Write a java program to create class car, truck and motorcycle which extends the vehicle class (attribute registration_number, color, type of vehicle) with their own attribute like make, CC and fuel type. Input data from the user and print all the details.**

   Algorithm :

   Create a Vehicle class with attributes registrationNumber, color, and typeOfVehicle
   Create a constructor for Vehicle that takes in values for registrationNumber, color, and typeOfVehicle and sets the values of the corresponding attributes
   Create a Car subclass that extends Vehicle with additional attributes make, cc, and fuelType
   Create a constructor for Car that takes in values for all six attributes and sets the values of the corresponding attributes
   Create a Truck subclass that extends Vehicle with additional attributes make, cc, and fuelType
   Create a constructor for Truck that takes in values for all six attributes and sets the values of the corresponding attributes
   Create a Motorcycle subclass that extends Vehicle with additional attributes make, cc, and fuelType
   Create a constructor for Motorcycle that takes in values for all six attributes and sets the values of the corresponding attributes
   In the main method, prompt the user to input data for a car, a truck, and a motorcycle
   Store the input data for each vehicle in variables
   Create a new Car object with the input data for the car, a new Truck object with the input data for the truck, and a new Motorcycle object with the input data for the motorcycle
   Print out the details for each vehicle, including all attributes, using the get methods for each attribute in each object
   End the program

   Code :

```
class Vehicle
{
        String regno;
        String color;
        String type_of_vehicle;
        Vehicle(String r, String cr, String t)
        {
```

```
                regno=r;
                color=cr;
      type_of_vehicle= t;
        }
        void display()
        {
                System.out.println("Registration no: "+regno);
                System.out.println("Color: "+color);
      System.out.println("Type: "+type_of_vehicle);
        }
}

class Car extends Vehicle
{
        int make;
    int cc;
    String fuel;
        Car(String r, String cr, String t, int m, int c, String f)
        {
                super(r,cr,t);
                make=m;
      cc=c;
      fuel=f;
        }
        void display()
        {
                System.out.println("Car Details are:");
                super.display();
                System.out.println("Making Year : " +make);
      System.out.println("Cubic capacity: " +cc);
      System.out.println("Fuel Type:\n" +fuel);
        }
}

class Truck extends Vehicle
{
         int make;
    int cc;
    String fuel;
        Truck(String r, String cr, String t, int m, int c, String f)

        {
                super(r,cr,t);
```

```
                make=m;
        cc=c;
        fuel=f;
           }
           void display()
           {
                System.out.println("Truck Details");
                super.display();
                System.out.println("Making Year : " +make);
        System.out.println("Cubic capacity: " +cc);
        System.out.println("Fuel Type:\n " +fuel);

           }
}

class Motorcycle extends Vehicle
{
           int make;
    int cc;
    String fuel;
           Motorcycle(String r, String cr, String t, int m, int c, String f)
           {
                super(r,cr,t);
                make=m;
        cc=c;
        fuel=f;
           }
           void display()
           {
                System.out.println("Motorcycle Details");
                super.display();
                System.out.println("Making Year : " +make);
        System.out.println("Cubic capacity: " +cc);
        System.out.println("Fuel Type:\n " +fuel);

           }
}
public class VehicleDemo
{
           public static void main(String arg[])
           {
                Car c1;
                Truck t1;
```

```
                Motorcycle m1;
                c1=new Car("TN7412345", "Blue","Four
wheeler",2022,900,"Petrol" );
                t1=new Truck("TN74126745","Blue","Six
wheeler",2009,38000,"Diesel");
                m1=new Motorcycle ("TN741442345","Blue","Two wheeler", 2012,
110,"Petrol");
                c1.display();
                t1.display();
                m1.display();
        }
}
```

4. **Write a java program to create a class Student with data „name, city and age‟ along with method addData and printData to input and display the data. Create the two objects s1, s2 to declare and access the values.**

   Algorithm :

   Create a Student class with attributes name, city, and age
   Create a void method addData() that:
   a. Creates a new Scanner object
   b. Prompts the user to input a value for name
   c. Stores the input in the name attribute
   d. Prompts the user to input a value for city
   e. Stores the input in the city attribute
   f. Prompts the user to input a value for age
   g. Stores the input in the age attribute
   Create a void method printData() that:
   a. Prints out the value of name attribute
   b. Prints out the value of city attribute
   c. Prints out the value of age attribute
   In the main() method:
   a. Create a Student object s1
   b. Call the addData() method for s1
   c. Print a message to display the details of s1
   d. Call the printData() method for s1
   e. Create a Student object s2
   f. Call the addData() method for s2
   g. Print a message to display the details of s2
   h. Call the printData() method for s2
   End the program.

   Code :

```
import java.util.Scanner;

public class Student {
   String name;
   String city;
   int age;

   public void addData() {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter name: ");
      name = scanner.nextLine();
      System.out.print("Enter city: ");
      city = scanner.nextLine();
      System.out.print("Enter age: ");
      age = scanner.nextInt();
   }

   public void printData() {
      System.out.println("Name: " + name);
      System.out.println("City: " + city);
      System.out.println("Age: " + age);
   }

   public static void main(String[] args) {
      Student s1 = new Student();
      s1.addData();
      System.out.println("\nDetails of s1:");
      s1.printData();

      Student s2 = new Student();
      s2.addData();
      System.out.println("\nDetails of s2:");
      s2.printData();
   }
}
```

5. **5. Write a java program that implements a multi-thread application that has three threads. First thread generates a random integer every 1 second and if the value is even, the second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of the cube of the number.**

   Algorithm :
   Create a Random object to generate random integers.

Create the first thread t1 with an infinite loop that generates a new random integer every 1 second.
If the integer is even, create the second thread t2 to compute the square of the number and print it.
If the integer is odd, create the third thread t3 to compute the cube of the number and print it.
Use lambda expressions to implement the threads for more concise code.
Use Thread.sleep(1000) to pause the execution of the first thread for 1 second before generating the next random integer.

Code :

```
import java.util.Random;

public class MultiThreadExample {
    public static void main(String[] args) {
        Random random = new Random();

        Thread t1 = new Thread(() -> {
            while (true) {
                int num = random.nextInt(100);
                System.out.println("Generated number: " + num);
                if (num % 2 == 0) {
                    Thread t2 = new Thread(() -> {
                        int square = num * num;
                        System.out.println(num + " squared is " + square);
                    });
                    t2.start();
                } else {
                    Thread t3 = new Thread(() -> {
                        int cube = num * num * num;
                        System.out.println(num + " cubed is " + cube);
                    });
                    t3.start();
                }
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });

        t1.start();
```

```
    }
}
```

6. **Write a java program to create an abstract class named Shape that contains two integers and an empty method named print Area (). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contains only the method print Area () that prints the area of the given shape.**

   Algorithm :
   Create an abstract class named Shape.
   Define two integer variables length and breadth in the Shape class.
   Define an abstract method named printArea() in the Shape class.
   Create a class named Rectangle that extends the Shape class.
   Define two integer variables length and breadth in the Rectangle class.
   Implement the printArea() method in the Rectangle class to compute and print the area of a rectangle.
   Create a class named Triangle that extends the Shape class.
   Define two integer variables length and breadth in the Triangle class.
   Implement the printArea() method in the Triangle class to compute and print the area of a triangle.
   Create a class named Circle that extends the Shape class.
   Define two integer variables length and breadth in the Circle class.
   Implement the printArea() method in the Circle class to compute and print the area of a circle.
   In the main method, create objects of the Rectangle, Triangle, and Circle classes, and call the printArea() method for each object to print the area of the respective shape.

   Code :

```
abstract class Shape {
    protected int length;
    protected int breadth;

    public Shape(int length, int breadth) {
        this.length = length;
        this.breadth = breadth;
    }

    public abstract void printArea();
}

class Rectangle extends Shape {
    public Rectangle(int length, int breadth) {
```

```
        super(length, breadth);
    }

    public void printArea() {
        int area = length * breadth;
        System.out.println("Area of Rectangle is: " + area);
    }
}

class Triangle extends Shape {
    public Triangle(int length, int breadth) {
        super(length, breadth);
    }

    public void printArea() {
        double area = 0.5 * length * breadth;
        System.out.println("Area of Triangle is: " + area);
    }
}

class Circle extends Shape {
    public Circle(int length, int breadth) {
        super(length, breadth);
    }

    public void printArea() {
        double radius = length / 2.0;
        double area = Math.PI * radius * radius;
        System.out.println("Area of Circle is: " + area);
    }
}

public class Main {
    public static void main(String[] args) {
        Shape shape1 = new Rectangle(5, 6);
        shape1.printArea();

        Shape shape2 = new Triangle(8, 10);
        shape2.printArea();

        Shape shape3 = new Circle(12, 0);
        shape3.printArea();
    }
```

}

7. **Write a java program to create a calculator which performs addition, subtraction and multiplication of numbers for different types like integer, float and complex numbers using single function add(), sub() and multi().**

Algorithm :
Start by asking the user to input the first number.
Store the input as variable A.
Ask the user to input the second number.
Store the input as variable B.
Add the values of A and B together.
Store the result as variable C.
Display the value of C to the user.

Code :

```
import java.util.Scanner;
class Calculator {

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

      System.out.println("Enter type of numbers: 1 for integer, 2 for float, 3 for
complex");
      int type = sc.nextInt();

      switch (type) {
         case 1:
            System.out.print("Enter first integer number: ");
            int num1 = sc.nextInt();
            System.out.print("Enter second integer number: ");
            int num2 = sc.nextInt();

            System.out.println("Result of addition: " + add(num1, num2));
            System.out.println("Result of subtraction: " + sub(num1, num2));
            System.out.println("Result of multiplication: " + multi(num1, num2));
            break;

         case 2:
            System.out.print("Enter first float number: ");
            float num3 = sc.nextFloat();
            System.out.print("Enter second float number: ");
```

```
            float num4 = sc.nextFloat();

            System.out.println("Result of addition: " + add(num3, num4));
            System.out.println("Result of subtraction: " + sub(num3, num4));
            System.out.println("Result of multiplication: " + multi(num3, num4));
            break;

        case 3:
            System.out.print("Enter real part of first complex number: ");
            float real1 = sc.nextFloat();
            System.out.print("Enter imaginary part of first complex number: ");
            float img1 = sc.nextFloat();
            System.out.print("Enter real part of second complex number: ");
            float real2 = sc.nextFloat();
            System.out.print("Enter imaginary part of second complex number: ");
            float img2 = sc.nextFloat();

            Complex num5 = new Complex(real1, img1);
            Complex num6 = new Complex(real2, img2);

            System.out.println("Result of addition: " + add(num5, num6));
            System.out.println("Result of subtraction: " + sub(num5, num6));
            System.out.println("Result of multiplication: " + multi(num5, num6));
            break;

        default:
            System.out.println("Invalid input!");
    }

    sc.close();
}

public static int add(int a, int b) {
    return a + b;
}

public static int sub(int a, int b) {
    return a - b;
}

public static int multi(int a, int b) {
    return a * b;
}
```

```
    public static float add(float a, float b) {
        return a + b;
    }

    public static float sub(float a, float b) {
        return a - b;
    }

    public static float multi(float a, float b) {
        return a * b;
    }

    public static Complex add(Complex a, Complex b) {
        return a.add(b);
    }

    public static Complex sub(Complex a, Complex b) {
        return a.sub(b);
    }

    public static Complex multi(Complex a, Complex b) {
        return a.multiply(b);
    }

}

class Complex {
    private double real, imag;

    public Complex(double r, double i) {
        real = r;
        imag = i;
    }

    public Complex add(Complex c) {
        return new Complex(real + c.real, imag + c.imag);
    }

    public Complex sub(Complex c) {
        return new Complex(real - c.real, imag - c.imag);
    }
```

```
    public Complex multiply(Complex c) {
        double r = real * c.real - imag * c.imag;
        double i =
real * c.imag + imag * c.real;
return new Complex(r, i);
}
}
```

8. **Create a simple application of Java AWT in which shows an awt component button by setting its placement and window frame size**

Algorithm :

Import the necessary AWT classes and create a new class for the application.
Create a new instance of the Frame class to create the window frame.
Set the title of the window using the setTitle() method.
Set the size of the window using the setSize() method.
Set the layout of the window using the setLayout() method.
Create a new instance of the Button class to create the button.
Set the label of the button using the setLabel() method.
Add the button to the frame using the add() method.
Set the placement of the button within the frame using the setBounds() method.
Make the frame visible using the setVisible() method

Code :

```
import java.awt.*;
import java.awt.event.*;

public class MyAWTApp extends Frame {

  public MyAWTApp() {
    // Set the window title
    setTitle("My AWT App");

    // Set the size of the window
    setSize(300, 200);

    // Set the layout of the window
    setLayout(new BorderLayout());
```

```
    // Create a button
    Button myButton = new Button("Click Me!");

    // Add the button to the window frame
    add(myButton, BorderLayout.CENTER);

    // Show the window frame
    setVisible(true);

    // Handle closing the window
    addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        System.exit(0);
      }
    });
  }

  public static void main(String[] args) {
    MyAWTApp myApp = new MyAWTApp();
  }
}
```

9. **Create a student table with fields roll number, name, percentage. Insert values in the table. Display all the details of the student table in a tabular format on the screen (using swing).**

   Algorithm :

   Import the necessary packages (java.sql., javax.swing., java.util.*)
   Create a database connection using the DriverManager.getConnection() method and specifying the database URL, username, and password.
   Create a Statement object using the createStatement() method of the Connection object.
   Execute an SQL statement to create a table named "students" with fields roll number, name, and percentage using the executeUpdate() method of the Statement object.
   Insert data into the students table using an SQL INSERT statement and the executeUpdate() method of the Statement object.

Create a JTable object and set its model to a new DefaultTableModel object with the data and column names retrieved from the database using the executeQuery() method of the Statement object.
Create a new JScrollPane object and add the JTable to it.
Add the JScrollPane to a new JFrame object and set its size and visibility.
Close the Statement object, the Connection object, and any other relevant resources using the close() method.

Code :

```
import java.awt.BorderLayout;
import java.sql.*;
import javax.swing.JFrame;
import java.awt.EventQueue;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;


public class StudentTable extends JFrame {

   private static final long serialVersionUID = 1L;
   private JPanel contentPane;
   private JTable table;

   public static void main(String[] args) {
      EventQueue.invokeLater(new Runnable() {
         public void run() {
            try {
               StudentTable frame = new StudentTable();
               frame.setVisible(true);
            } catch (Exception e) {
               e.printStackTrace();
            }
         }
      });
   }

   public StudentTable() {
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setBounds(100, 100, 450, 300);
      contentPane = new JPanel();
      contentPane.setBorder(null);
```

```
        contentPane.setLayout(new BorderLayout(0, 0));
        setContentPane(contentPane);

        // Database connection
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/mydb", "root", "root");

            Statement stmt = con.createStatement();

            // Create student table if not exists
            stmt.executeUpdate("CREATE TABLE IF NOT EXISTS student(rollno INT,
name VARCHAR(20), percentage FLOAT)");

            // Insert data into the student table
            stmt.executeUpdate("INSERT INTO student(rollno, name, percentage)
VALUES(1,'John',87.5)");
            stmt.executeUpdate("INSERT INTO student(rollno, name, percentage)
VALUES(2,'Alice',94.2)");
            stmt.executeUpdate("INSERT INTO student(rollno, name, percentage)
VALUES(3,'Bob',68.8)");

            // Get data from student table
            ResultSet rs = stmt.executeQuery("SELECT * FROM student");

            // Create a table model
            String[] columnNames = { "Roll No.", "Name", "Percentage" };
            Object[][] data = new Object[100][3];
            int i = 0;
            while (rs.next()) {
                data[i][0] = rs.getInt("rollno");
                data[i][1] = rs.getString("name");
                data[i][2] = rs.getFloat("percentage");
                i++;
            }
            Object[][] finalData = new Object[i][3];
            for (int j = 0; j < i; j++) {
                finalData[j] = data[j];
            }

            table = new JTable(finalData, columnNames);
            JScrollPane scrollPane = new JScrollPane(table);
```

```
            contentPane.add(scrollPane, BorderLayout.CENTER);

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

10. **Write a java program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green with radio buttons. On selecting a button, an appropriate message with “stop” or “ready” or “go” should appear above the buttons in a selected color. Initially there is no message shown.**
    Algorithm :

    Create a JFrame window
    Create three radio buttons with labels "Red", "Yellow", and "Green"
    Create a JLabel to display the message
    Add action listeners to each radio button to listen for selection events
    In each action listener, set the text of the JLabel to the appropriate message and change its color based on the selected radio button
    Add the radio buttons and JLabel to the JFrame window
    Set the properties of the JFrame window such as size, layout, and visibility

    Code :

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class TrafficLight implements ActionListener {
    private JFrame frame;
    private JRadioButton redButton, yellowButton, greenButton;
    private JLabel messageLabel;

    public TrafficLight() {
        // create a new JFrame
        frame = new JFrame("Traffic Light");

        // set the layout of the frame
        frame.setLayout(new BorderLayout());

        // create a new JPanel for the radio buttons
```

```
    JPanel buttonPanel = new JPanel(new FlowLayout());

    // create the radio buttons and add them to the button panel
    redButton = new JRadioButton("Red");
    yellowButton = new JRadioButton("Yellow");
    greenButton = new JRadioButton("Green");

    buttonPanel.add(redButton);
    buttonPanel.add(yellowButton);
    buttonPanel.add(greenButton);

    // add an action listener to the radio buttons
    redButton.addActionListener(this);
    yellowButton.addActionListener(this);
    greenButton.addActionListener(this);

    // create a new JPanel for the message label
    JPanel messagePanel = new JPanel(new FlowLayout());

    // create the message label and add it to the message panel
    messageLabel = new JLabel("");
    messageLabel.setFont(new Font("Serif", Font.BOLD, 20));
    messagePanel.add(messageLabel);

    // add the button panel and message panel to the frame
    frame.add(buttonPanel, BorderLayout.CENTER);
    frame.add(messagePanel, BorderLayout.SOUTH);

    // set the size of the frame and make it visible
    frame.setSize(300, 150);
    frame.setVisible(true);
}

public void actionPerformed(ActionEvent e) {
    // determine which radio button was selected
    if (e.getSource() == redButton) {
        messageLabel.setText("Stop");
        messageLabel.setForeground(Color.RED);
    } else if (e.getSource() == yellowButton) {
        messageLabel.setText("Ready");
        messageLabel.setForeground(Color.YELLOW);
    } else if (e.getSource() == greenButton) {
        messageLabel.setText("Go");
```

```
            messageLabel.setForeground(Color.GREEN);
        }
    }

    public static void main(String[] args) {
        new TrafficLight();
    }
}
```

11. **Write a java program that connects to a database using JDBC and does add, deletes, modify, and retrieve operations.**

    Algorithm :

    Initialize a variable 'max' to the first element of the array
    Loop through the array from the second element to the last element
    If the current element is greater than the 'max' variable, set 'max' to the current element
    After looping through the entire array, 'max' will contain the largest element in the array
    Return 'max'

    Code :

```
import java.sql.*;

public class DatabaseOperations {
    // JDBC driver name and database URL
    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/your_database_name";

    // Database credentials
    static final String USER = "your_username";
    static final String PASS = "your_password";

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try {
            // Register JDBC driver
            Class.forName(JDBC_DRIVER);
```

```
            // Open a connection
            System.out.println("Connecting to database...");
            conn = DriverManager.getConnection(DB_URL, USER, PASS);

            // Create a statement
            System.out.println("Creating statement...");
            stmt = conn.createStatement();

            // Perform CRUD operations
            // Insert operation
            String sql = "INSERT INTO your_table_name (id, name, age) " +
                         "VALUES (1, 'John Doe', 30)";
            stmt.executeUpdate(sql);

            // Update operation
            sql = "UPDATE your_table_name SET age = 40 WHERE id = 1";
            stmt.executeUpdate(sql);

            // Delete operation
            sql = "DELETE FROM your_table_name WHERE id = 1";
            stmt.executeUpdate(sql);

            // Retrieve operation
            sql = "SELECT * FROM your_table_name";
            ResultSet rs = stmt.executeQuery(sql);

            // Display retrieved data
            while (rs.next()) {
                int id = rs.getInt("id");
                String name = rs.getString("name");
                int age = rs.getInt("age");
                System.out.println("ID: " + id + ", Name: " + name + ", Age: " + age);
            }

            // Clean-up environment
            rs.close();
            stmt.close();
            conn.close();
        } catch (SQLException se) {
            // Handle errors for JDBC
            se.printStackTrace();
        } catch (Exception e) {
            // Handle errors for Class.forName
```

```
            e.printStackTrace();
        } finally {
            // Close resources
            try {
                if (stmt != null) stmt.close();
            } catch (SQLException se2) {
            } // nothing we can do
            try {
                if (conn != null) conn.close();
            } catch (SQLException se) {
                se.printStackTrace();
            }
            System.out.println("Database connection closed.");
        }
    }
}
```

12. Write a program that implements the concept of Encapsulation.

Algorithm :

Initialize a variable 'sum' to 0
Loop through the array from the first element to the last element
Add the current element to 'sum'
After looping through the entire array, 'sum' will contain the sum of all the elements in the array
Return 'sum'

Code :

```
public class Student {

    private String name;

    private int age;

    private double gpa;


    public Student(String name, int age, double gpa) {

        this.name = name;

        this.age = age;

        this.gpa = gpa;
```

```
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public int getAge() {
    return age;
}

public void setAge(int age) {
    this.age = age;
}

public double getGpa() {
    return gpa;
}

public void setGpa(double gpa) {
    this.gpa = gpa;
}

public static void main(String[] args) {
    Student student = new Student("John Doe", 20, 3.5);
```

```
        System.out.println("Name: " + student.getName());
        System.out.println("Age: " + student.getAge());
        System.out.println("GPA: " + student.getGpa());
        student.setGpa(3.8);
        System.out.println("Updated GPA: " + student.getGpa());
    }
}
```

13. **Write a program to demonstrate the concept of function overloading of Polymorphism.**

    Algorithm :

    Create a Java class and name it MathUtils.
    Inside the MathUtils class, define three methods named add. The first method should take two integer parameters, the second method should take two double parameters, and the third method should take three integer parameters.
    Implement each add method so that it returns the sum of the parameters passed to it.
    In the main method of the MathUtils class, create an instance of the MathUtils class.
    Call each of the add methods with appropriate parameters.
    Print the result returned by each add method.
    Run the program and observe the output.

    Code :

```
public class MathUtils {
    public int add(int a, int b) {
        return a + b;
    }

    public double add(double a, double b) {
        return a + b;
    }

    public int add(int a, int b, int c) {
        return a + b + c;
    }
```

```
    public static void main(String[] args) {
        MathUtils mathUtils = new MathUtils();
        int sum1 = mathUtils.add(2, 3);
        double sum2 = mathUtils.add(2.5, 3.5);
        int sum3 = mathUtils.add(2, 3, 4);
        System.out.println("Sum of 2 and 3: " + sum1);
        System.out.println("Sum of 2.5 and 3.5: " + sum2);
        System.out.println("Sum of 2, 3, and 4: " + sum3);
    }
}
```

14. Write a program to demonstrate the concept of constructor overloading of Polymorphism.

Algorithm :

Define a class and name it Person.
Add two private instance variables to the class, name and age.
Define three constructors for the Person class:
a. A default constructor that initializes name to "Unknown" and age to 0.
b. A constructor that takes a String parameter name and initializes name to the provided value and age to 0.
c. A constructor that takes both a String parameter name and an int parameter age, and initializes both instance variables to the provided values.
Define getter methods for name and age.
Inside the main method, create three instances of the Person class using each of the constructors.
Print out the values of the name and age variables for each instance to verify that the constructors were called correctly.

Code :

```
public class Person {
    private String name;
    private int age;

    public Person() {
        this.name = "Unknown";
        this.age = 0;
    }
```

```
    public Person(String name) {
        this.name = name;
        this.age = 0;
    }

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return this.name;
    }

    public int getAge() {
        return this.age;
    }

    public static void main(String[] args) {
        Person person1 = new Person();
        Person person2 = new Person("John");
        Person person3 = new Person("Jane", 25);

        System.out.println(person1.getName() + " " + person1.getAge()); // "Unknown
0"
        System.out.println(person2.getName() + " " + person2.getAge()); // "John 0"
        System.out.println(person3.getName() + " " + person3.getAge()); // "Jane 25"
    }
}
```

15. **Write a program that uses Boolean data type and print the prime number series up to 50.**

    Algorithm :

    Define a class named PrimeNumbers.
    Inside the class, create the main method.
    Define a Boolean variable named isPrime.
    Create a loop that iterates from 2 to 50. Let the current number be represented by the variable i.
    Set isPrime to true at the beginning of the loop.

Create a nested loop that iterates from 2 to i/2. Let the current number be represented by the variable j.
Check if i is divisible by j. If it is, set isPrime to false and break out of the inner loop.
If isPrime is still true after the inner loop completes, print i to the console.
Continue the outer loop until it completes.
Run the program and verify that the prime numbers up to 50 are printed to the console.

Code :

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

16. Write a program to check the given number is Armstrong or not

Algorithm :

Define a class named ArmstrongNumber.
Inside the class, create the main method.
Declare an integer variable named number.
Read an integer input from the user and store it in the number variable.
Declare integer variables named digit, sum, and temp.
Initialize the sum variable to 0.
Set the value of temp to number.
Create a loop that iterates until temp is greater than 0.
Inside the loop, get the last digit of temp by using the modulus operator and store it in the digit variable.

Compute the cube of digit and add it to sum.
Remove the last digit of temp by dividing it by 10 and assigning the result back to temp.
Continue the loop until it completes.
Check if the value of sum is equal to number.
If sum is equal to number, then print "The given number is an Armstrong number".
If sum is not equal to number, then print "The given number is not an Armstrong number".
End the program.

Code :

```
import java.util.Scanner;

public class ArmstrongNumber {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter a number: ");
      int number = scanner.nextInt();

      int digit, sum = 0, temp;
      temp = number;

      while (temp > 0) {
         digit = temp % 10;
         sum += (digit * digit * digit);
         temp /= 10;
      }

      if (sum == number)
         System.out.println("The given number is an Armstrong number");
      else
         System.out.println("The given number is not an Armstrong number");

      scanner.close();
   }
}
```

17. Write a program to sort the element of One Dimensional Array in Ascending order

Algorithm :

Define a class named ArraySorter.
Inside the class, create the main method.
Declare an integer array variable named numbers.
Read the size of the array from the user.
Create an array of size n and store it in the numbers variable.
Create a loop that iterates from 0 to n-1.
Inside the loop, read an integer input from the user and store it in the current element of the numbers array.
Create another loop that iterates from 0 to n-1.
Inside the loop, create a nested loop that iterates from 0 to n-2.
Inside the nested loop, compare the current element of the array with the next element.
If the current element is greater than the next element, swap the elements.
Continue the nested loop until it completes.
Continue the outer loop until it completes.
Print the sorted array in ascending order.
End the program.

Code :

```
import java.util.Scanner;

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

      System.out.print("Enter the size of the array: ");
      int n = scanner.nextInt();

      int[] numbers = new int[n];

      for (int i = 0; i < n; i++) {
         System.out.print("Enter element " + (i + 1) + ": ");
         numbers[i] = scanner.nextInt();
      }

      for (int i = 0; i < n; i++) {
         for (int j = 0; j < n - 1; j++) {
            if (numbers[j] > numbers[j + 1]) {
               int temp = numbers[j];
               numbers[j] = numbers[j + 1];
               numbers[j + 1] = temp;
            }
```

```
            }
        }

        System.out.println("Sorted Array in Ascending Order:");
        for (int i = 0; i < n; i++) {
            System.out.print(numbers[i] + " ");
        }

        scanner.close();
    }
}
```

18. Write a program for matrix multiplication using input/output Stream.

Algorithm :

Define a class named MatrixMultiplication.
Inside the class, create the main method.
Declare integer variables for the size of the first matrix m, n, and the second matrix p, q.
Read the size of the first matrix from the user.
Create a two-dimensional integer array matrix1 of size m x n and read the matrix elements from the user using input streams.
Read the size of the second matrix from the user.
Create a two-dimensional integer array matrix2 of size p x q and read the matrix elements from the user using input streams.
Create a two-dimensional integer array result of size m x q to store the result of matrix multiplication.
Check if the number of columns in the first matrix is equal to the number of rows in the second matrix.
If not, display an error message and exit the program.
If the number of columns in the first matrix is equal to the number of rows in the second matrix, then perform matrix multiplication and store the result in the result array.
Display the result using output streams.

Code :

```
import java.io.*;

public class MatrixMultiplication {
```

```
   public static void main(String[] args) {
      BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
      PrintWriter pw = new PrintWriter(new OutputStreamWriter(System.out));

      int m, n, p, q;
      try {
         pw.println("Enter the number of rows and columns of the first matrix:");
         m = Integer.parseInt(br.readLine());
         n = Integer.parseInt(br.readLine());

         int[][] matrix1 = new int[m][n];
         pw.println("Enter the elements of the first matrix:");
         for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
               matrix1[i][j] = Integer.parseInt(br.readLine());
            }
         }

         pw.println("Enter the number of rows and columns of the second matrix:");
         p = Integer.parseInt(br.readLine());
         q = Integer.parseInt(br.readLine());

         int[][] matrix2 = new int[p][q];
         pw.println("Enter the elements of the second matrix:");
         for (int i = 0; i < p; i++) {
            for (int j = 0; j < q; j++) {
               matrix2[i][j] = Integer.parseInt(br.readLine());
            }
         }

         if (n != p) {
            pw.println("Error: Number of columns of first matrix is not equal to
number of rows of second matrix.");
            System.exit(0);
         }

         int[][] result = new int[m][q];
         for (int i = 0; i < m; i++) {
            for (int j = 0; j < q; j++) {
               for (int k = 0; k < n; k++) {
                  result[i][j] += matrix1[i][k] * matrix2[k][j];
               }
```

```
            }
        }

        pw.println("Result of matrix multiplication:");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < q; j++) {
                pw.print(result[i][j] + " ");
            }
            pw.println();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }

    pw.flush();
    pw.close();
  }
}
```

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

Algorithm :

Initialize an array of integers.
Initialize a variable index to 5.
Try to access the element at index index of the array.
If the index is less than 0 or greater than or equal to the length of the array, catch the ArrayIndexOutOfBoundsException and print a message saying that the index is out of bounds.
If the index is within the bounds of the array, print the value of the element at that index.
Catch any other exceptions that may occur and print an appropriate message.

Code :

```
public class MultipleCatchExample {

   public static void main(String[] args) {
      int[] numbers = { 1, 2, 3, 4, 5 };
      try {
         int result = numbers[5] / 0;
      } catch (ArithmeticException e) {
         System.out.println("Arithmetic exception occurred: " + e.getMessage());
```

```
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array index out of bounds exception occurred: " +
e.getMessage());
        } catch (Exception e) {
            System.out.println("Exception occurred: " + e.getMessage());
        }
    }
}
```

20. Write a program for multi thread using is Alive(), join() and synchronized() methods of thread class

Algorithm :

Define a class that extends Thread class
Implement the run() method for the thread that will be executed in a separate thread
Create an instance of the thread class and start the thread using start() method
Use isAlive() method to check if the thread is still alive
Use join() method to wait for the thread to finish executing before moving on
Use synchronized() method to synchronize access to shared data between threads

Code :

```
public class MultiThreadExample implements Runnable {
    private String threadName;
    private Thread thread;
    private static int count = 0;
    private static final int MAX_COUNT = 10;

    public MultiThreadExample(String name) {
        this.threadName = name;
        System.out.println("Creating thread: " + threadName);
    }

    public void run() {
        System.out.println("Running thread: " + threadName);
        try {
            for (int i = 0; i < MAX_COUNT; i++) {
                System.out.println(threadName + ": " + count);
                synchronized (this) {
```

```
            count++;
        }
        Thread.sleep(100);
      }
    } catch (InterruptedException e) {
      System.out.println("Thread " + threadName + " interrupted.");
    }
    System.out.println("Exiting thread: " + threadName);
}

public void start() {
    System.out.println("Starting thread: " + threadName);
    if (thread == null) {
      thread = new Thread(this, threadName);
      thread.start();
    }
}

public void join() {
    try {
      thread.join();
    } catch (InterruptedException e) {
      System.out.println("Thread " + threadName + " interrupted.");
    }
}

public boolean isAlive() {
    return thread.isAlive();
}

public static void main(String[] args) {
    MultiThreadExample thread1 = new MultiThreadExample("Thread 1");
    thread1.start();
    MultiThreadExample thread2 = new MultiThreadExample("Thread 2");
    thread2.start();

    while (thread1.isAlive() || thread2.isAlive()) {
      if (thread1.isAlive()) {
        thread1.join();
      }
      if (thread2.isAlive()) {
        thread2.join();
      }
```

```
        }
        System.out.println("Final count: " + count);
    }
}
```

21. **Write a program for JDBC to insert the values into the existing table by using prepared statement**

    Algorithm :

    Import the necessary JDBC packages.
    Set up the database connection by loading the driver, creating a connection object, and specifying the database URL, username, and password.
    Write the SQL query to insert the values into the table.
    Create a prepared statement object and pass the SQL query as a parameter.
    Set the values to be inserted into the prepared statement object using the set methods.
    Call the executeUpdate() method on the prepared statement object to insert the values into the table.
    Close the prepared statement object and the database connection.

    Code :

```
import java.sql.*;

public class JdbcInsertExample {

   // JDBC driver and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost/testdb";

   // Database credentials
   static final String USER = "username";
   static final String PASS = "password";

   public static void main(String[] args) {
      Connection conn = null;
      PreparedStatement stmt = null;

      try {
         // Register JDBC driver
         Class.forName(JDBC_DRIVER);
```

```
            // Open a connection
            conn = DriverManager.getConnection(DB_URL, USER, PASS);

            // Prepare a SQL statement
            String sql = "INSERT INTO students (id, name, age) VALUES (?, ?, ?)";
            stmt = conn.prepareStatement(sql);

            // Set values for prepared statement parameters
            stmt.setInt(1, 101);
            stmt.setString(2, "John Smith");
            stmt.setInt(3, 21);

            // Execute the SQL statement
            int rowsInserted = stmt.executeUpdate();
            System.out.println("Rows inserted: " + rowsInserted);

        } catch(SQLException se) {
            // Handle errors for JDBC
            se.printStackTrace();
        } catch(Exception e) {
            // Handle errors for Class.forName
            e.printStackTrace();
        } finally {
            // Close resources
            try {
                if(stmt != null) {
                    stmt.close();
                }
            } catch(SQLException se2) {
                // Nothing we can do
            }
            try {
                if(conn != null) {
                    conn.close();
                }
            } catch(SQLException se) {
                se.printStackTrace();
            }
        }
    }
}
```

22. WAP for JDBC to display the records from the existing table.

Algorithm :

```
import java.sql.*;
// create the connection object
Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/mydatabase","userna
me","password");
// create the statement object
Statement stmt = conn.createStatement();
// create the SQL query to select the records from the table
String sql = "SELECT * FROM mytable";
// execute the query and store the result set in a ResultSet object
ResultSet rs = stmt.executeQuery(sql);
// iterate through the rows in the ResultSet and retrieve the column values
while (rs.next()) {
python
Copy code
int id = rs.getInt("id");
arduino
Copy code
String name = rs.getString("name");
arduino
Copy code
double salary = rs.getDouble("salary");
arduino
Copy code
// print the column values to the console
bash
Copy code
System.out.println("ID: " + id + ", Name: " + name + ", Salary: " + salary);
}
// close the ResultSet, statement, and connection objects
rs.close();
stmt.close();
conn.close();
```

Code :

```
import java.sql.*;
```

```
public class DisplayRecords {
   public static void main(String[] args) {
      // Set up the database connection
      String url = "jdbc:mysql://localhost:3306/mydb"; // replace with your database
URL
      String username = "root"; // replace with your username
      String password = "password"; // replace with your password

      try {
         Connection connection = DriverManager.getConnection(url, username,
password);

         // Write the SQL query to retrieve the records from the table
         String query = "SELECT * FROM student";

         // Create a statement object and execute the query
         Statement statement = connection.createStatement();
         ResultSet resultSet = statement.executeQuery(query);

         // Display the results
         while (resultSet.next()) {
            int id = resultSet.getInt("id");
            String name = resultSet.getString("name");
            int age = resultSet.getInt("age");
            double gpa = resultSet.getDouble("gpa");
            System.out.printf("%d\t%s\t%d\t%.2f\n", id, name, age, gpa);
         }

         // Close the statement object and the database connection
         statement.close();
         connection.close();
      } catch (SQLException e) {
         System.out.println("Error: " + e.getMessage());
      }
   }
}
```

23. WAP for demonstration of switch statement ,continue and break.

Algorithm :

Start the program.

Take an input from the user for the switch case.
Implement a switch statement with different cases.
For each case, perform some action or print some output.
Use continue statement to skip a specific iteration of a loop.
Use break statement to exit a loop or switch statement.
End the program.

Code :

```
import java.util.Scanner;

public class SwitchDemo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number between 1 and 5: ");
        int num = scanner.nextInt();

        switch (num) {
            case 1:
                System.out.println("One");
                break;
            case 2:
                System.out.println("Two");
                break;
            case 3:
                System.out.println("Three");
                break;
            case 4:
                System.out.println("Four");
                break;
            case 5:
                System.out.println("Five");
                break;
            default:
                System.out.println("Invalid number entered");
                break;
        }

        for (int i = 1; i <= 10; i++) {
            if (i == 5) {
                System.out.println("Skipping iteration " + i);
                continue;
            }
```

```
            System.out.println("Iteration " + i);
            if (i == 8) {
                System.out.println("Breaking out of loop at iteration " + i);
                break;
            }
        }
    }
}
```

23. WAP to design a simple calculator using swings

Algorithm :

Create a JFrame to hold the calculator.

Create a JTextField to display the current number or calculation result.

Create a JPanel to hold the calculator buttons.

For each button (0-9, +, -, *, /, ., =), create a JButton with the specified label and add an ActionListener to handle button clicks.

If the button is a number or decimal point, append its label to the display text.

If the button is an operator (+, -, *, /), store the first number and the operator, and clear the display.

If the button is the equals sign, retrieve the second number, perform the calculation, and display the result.

Add each button to the button panel.

Add the display and button panel to the JFrame.

Set the JFrame visible.

Code :

```
Import javax.swing.*;

Import java.awt.*;


Public class Calculator extends JFrame {
```

```
Private JTextField display;
Private JPanel buttonPanel;
Private double num1, num2;
Private String operator;

Public Calculator() {
   setTitle("Calculator");
   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   setSize(300, 200);

   display = new JTextField("0", 20);
   display.setEditable(false);
   display.setHorizontalAlignment(JTextField.RIGHT);

   buttonPanel = new JPanel();
   buttonPanel.setLayout(new GridLayout(4, 4));

   addButton("7");
   addButton("8");
   addButton("9");
   addButton("/");
   addButton("4");
   addButton("5");
   addButton("6");
   addButton("*");
   addButton("1");
   addButton("2");
   addButton("3");
```

```
    addButton("-");
    addButton("0");
    addButton(".");
    addButton("=");
    addButton("+");

    getContentPane().add(display, BorderLayout.NORTH);
    getContentPane().add(buttonPanel, BorderLayout.CENTER);
}

Private void addButton(String label) {
    JButton button = new JButton(label);
    Button.addActionListener(e -> {
        String text = display.getText();
        If (text.equals("0")) {
            Display.setText(label);
        } else {
            Display.setText(text + label);
        }
    });

    If (label.equals("=")) {
        Button.addActionListener(e -> {
            Num2 = Double.parseDouble(display.getText());
            Double result = 0;
            Switch (operator) {
                Case "+":
                    Result = num1 + num2;
```

```
                Break;
            Case “-“:
                Result = num1 – num2;
                Break;
            Case “*”:
                Result = num1 * num2;
                Break;
            Case “/”:
                Result = num1 / num2;
                Break;
        }
        Display.setText(Double.toString(result));
    });
  } else if (label.equals(“+”) || label.equals(“-“) || label.equals(“*”) || label.equals(“/”)) {
    Button.addActionListener(e -> {
        Num1 = Double.parseDouble(display.getText());
        Operator = label;
        Display.setText(“0”);
    });
  }

  buttonPanel.add(button);
}

Public static void main(String[] args) {
  Calculator calc = new Calculator();
  Calc.setVisible(true);
}
```

}

25. WAP to implement multiple inheritance in java.

Algorithm :

Define two or more interfaces with one or more methods each.

Define a class that implements these interfaces.

Implement all the methods declared in the interfaces in the class.

Create an object of the class and call the methods of the interfaces on it.

The class will inherit the behavior of all the interfaces implemented by it.

Code :

```
Interface Shape {
   Double getArea();
}

Interface Color {
   String getColor();
}

Class Circle implements Shape, Color {
   Private double radius;
   Private String color;

   Public Circle(double radius, String color) {
      This.radius = radius;
      This.color = color;
```

```
    }

    @Override
    Public double getArea() {
        Return Math.PI * radius * radius;
    }

    @Override
    Public String getColor() {
        Return color;
    }
}

Public class MultiInheritanceDemo {
    Public static void main(String[] args) {
        Circle circle = new Circle(5.0, “red”);
        System.out.println(“Area: “ + circle.getArea());
        System.out.println(“Color: “ + circle.getColor());
    }
}
```

26. WAP to design an applet which accepts personal details

Algorithm :

Create an applet class that extends java.applet.Applet.

Declare the necessary instance variables for the applet, such as TextField and Label objects to accept and display personal details.

In the init method, create and initialize the instance variables and add them to the applet using the add method.

Implement the ActionListener interface in the applet class to handle button clicks.

In the actionPerformed method, retrieve the text entered in each field and perform any desired processing, such as validating the input or displaying the information in a message box or console.

In the HTML file, embed the applet using the <applet> tag, specifying the applet class and any desired attributes such as the width and height of the applet.

Code :

```
Import java.applet.*;
Import java.awt.*;
Import java.awt.event.*;

Public class PersonalDetailsApplet extends Applet implements ActionListener {
   Private TextField nameField, ageField, addressField, occupationField;
   Private Label nameLabel, ageLabel, addressLabel, occupationLabel;
   Private Button submitButton;

   Public void init() {
      nameLabel = new Label(“Name:”);
      nameField = new TextField(20);

      ageLabel = new Label(“Age:”);
      ageField = new TextField(3);

      addressLabel = new Label(“Address:”);
      addressField = new TextField(40);

      occupationLabel = new Label(“Occupation:”);
```

```
    occupationField = new TextField(20);

    submitButton = new Button(“Submit”);
    submitButton.addActionListener(this);

    add(nameLabel);
    add(nameField);
    add(ageLabel);
    add(ageField);
    add(addressLabel);
    add(addressField);
    add(occupationLabel);
    add(occupationField);
    add(submitButton);
}

Public void actionPerformed(ActionEvent e) {
    If (e.getSource() == submitButton) {
        String name = nameField.getText();
        String age = ageField.getText();
        String address = addressField.getText();
        String occupation = occupationField.getText();

        System.out.println(“Name: “ + name);
        System.out.println(“Age: “ + age);
        System.out.println(“Address: “ + address);
        System.out.println(“Occupation: “ + occupation);
    }
```

```
    }
}
```