// 1.Write a program to create a class and implement a default, overloaded and copy constructor.

```
class Book {
    String title;
    String author;

    // Default Constructor
    public Book() {
        this.title = "Unknown Title";
        this.author = "Unknown Author";
    }

    // Overloaded Constructor
    public Book(String t, String a) {
        this.title = t;
        this.author = a;
    }

    // Copy Constructor
    public Book(Book anotherBook) {
        this.title = anotherBook.title;
        this.author = anotherBook.author;
    }

    public void display() {
        System.out.println("Title: " + title + ", Author: " + author);
    }
}

public class Library {  // ✅ Ensure this matches the filename (Library.java)
    public static void main(String[] args) {
        // Default constructor
        Book b1 = new Book();
        b1.display();

        // Overloaded constructor
        Book b2 = new Book("Book of Optics", "Ibn al-Haytham");
        b2.display();

        // Copy constructor
        Book b3 = new Book(b2);
        b3.display();
    }
}
```

//1**Write a program to create a class and implement the concepts of Method Overloading.**

```
class OperOver {
   // Method to add two integers
   public int add(int a, int b) {
      return a + b;
   }

   // Method to add three integers
   public int add(int a, int b, int c) {
      return a + b + c;
   }
}

public class Pr1b {
   public static void main(String[] args) {
      OperOver obj = new OperOver();

      // Method Overloading Examples
      int sum1 = obj.add(5, 10);
      int sum2 = obj.add(5, 10, 15);

      System.out.println("Sum of two integers: " + sum1);
      System.out.println("Sum of three integers: " + sum2);
   }
}
```

//1Write a program to create a class and implement the concepts of Static methods.

```
class DemoStaticMethods {
   // Static method to add two numbers
   public static int add(int a, int b) {
      return a + b;
   }

   // Static method to subtract two numbers
   public static int subtract(int a, int b) {
      return a - b;
   }
}

public class Pr1c {
   public static void main(String[] args) {
      // Calling static methods directly on the class
```

```
        int sum = DemoStaticMethods.add(8, 4);
        int difference = DemoStaticMethods.subtract(7, 6);

        System.out.println("Sum: " + sum);
        System.out.println("Difference: " + difference);
    }
}
```

//2 Write a program to implement the concepts of Inheritance and Method Overriding.

```
class A {

    void show() {

        System.out.println("Base Class");

    }

}


class B extends A {

    // Overriding Method of Base Class

    void show() {

        System.out.println("Derived Class");

    }

}


class Pr2a {

    public static void main(String args[]) {

        B s = new B();

        s.show();

    }

}
```

//2bwrite a program to implement the concept of abstract classes and method

```
// Abstract class
abstract class Shape {
    // Abstract method for calculating area
    public abstract double area();
}

// Concrete subclass - Circle
class Circle extends Shape {
    private double radius;

    // Constructor to initialize radius
    public Circle(double radius) {
        this.radius = radius;
    }

    @Override
    public double area() {
        return Math.PI * radius * radius;
    }
}

public class Pr2b {
    public static void main(String[] args) {
        // Create a Circle object

        Circle circle = new Circle(10.0);

        // Calculate and display the area of the circle
        System.out.println("Circle Area: " + circle.area());
    }
}
```

//2cWrite a Java program to implement the concept of **interfaces**.

```
// Define an interface
interface Shape {
    // Abstract methods (implicitly public and abstract)
    double area();
    double perimeter();
}

// Implement the interface in a class
class Circle implements Shape {
    private double radius;

    // Constructor
    public Circle(double radius) {
        this.radius = radius;
    }

    @Override
    public double area() {
        return Math.PI * radius * radius;
    }

    @Override
    public double perimeter() {
        return 2 * Math.PI * radius;
    }
}

// Main class
public class Pr2c {
    public static void main(String[] args) {
        // Create a Circle object
        Circle circle = new Circle(10.0);

        // Calculate and display the area and perimeter
        System.out.println("Circle Area: " + circle.area());
        System.out.println("Circle Perimeter: " + circle.perimeter());
    }
}
```

//3Write a Java program to demonstrate the concept of **user-defined exceptions**.

```
// Program to handle built-in exceptions (ArithmeticException)
public class Pr3a {
    public static void main(String[] args) {
        try {
            int result = divide(10, 0); // This will cause division by zero
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.err.println("Error: Division by zero.");
        }
    }

    public static int divide(int a, int b) {
        return a / b; // Division by zero will throw ArithmeticException
    }
}
```

//3Write a program to define user-defined exceptions and raise them as per the requirements.

```
// Define a custom exception class
class CustomException extends Exception {
    public CustomException(String message) {
        super(message); // Passes the message to the Exception class
    }
}

public class Pr3b {
    public static void main(String[] args) {
        try {
            int age = -20; // Negative age value

            // Check if age is negative
            if (age < 0) {
                throw new CustomException("Age cannot be negative.");
            }

            System.out.println("Age: " + age);
        } catch (CustomException e) {
            System.err.println("Error: " + e.getMessage());
        }
    }
}
```

//4Write a Java application to demonstrate multiple bouncing balls of different colors using threads

```
import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class BouncingBalls extends JPanel implements Runnable {
    public static final int WIDTH = 800;
    public static final int HEIGHT = 600;
    private static final int NUM_BALLS = 5;

    private List<Ball> balls;

    public BouncingBalls() {
        balls = new ArrayList<>();
        Random random = new Random();
        for (int i = 0; i < NUM_BALLS; i++) {
            int x = random.nextInt(WIDTH);
            int y = random.nextInt(HEIGHT);
            int xSpeed = random.nextInt(5) + 1;
            int ySpeed = random.nextInt(5) + 1;
            Color color = new Color(random.nextInt(256), random.nextInt(256),
random.nextInt(256));
            balls.add(new Ball(x, y, xSpeed, ySpeed, color));
        }
    }

    @Override
    public void run() {
        while (true) {
            for (Ball ball : balls) {
                ball.move();
            }
            repaint();
            try {
                Thread.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        for (Ball ball : balls) {
            ball.draw(g);
        }
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("5 Colours Bouncing Balls");
        BouncingBalls bouncingBalls = new BouncingBalls();
        frame.add(bouncingBalls);
        frame.setSize(WIDTH, HEIGHT);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);

        Thread thread = new Thread(bouncingBalls);
        thread.start();
    }
}
```

```
class Ball {
    private int x, y, xSpeed, ySpeed;
    private Color color;
    private static final int SIZE = 20;

    public Ball(int x, int y, int xSpeed, int ySpeed, Color color) {
        this.x = x;
        this.y = y;
        this.xSpeed = xSpeed;
        this.ySpeed = ySpeed;
        this.color = color;
    }

    public void move() {
        x += xSpeed;
        y += ySpeed;
        if (x < 0 || x > BouncingBalls.WIDTH - SIZE) {
            xSpeed = -xSpeed;
        }
        if (y < 0 || y > BouncingBalls.HEIGHT - SIZE) {
            ySpeed = -ySpeed;
        }
    }

    public void draw(Graphics g) {
        g.setColor(color);
        g.fillOval(x, y, SIZE, SIZE);
    }
}
```

//7.aflow chart

```
package layouts;  // Lowercase package name

import java.awt.FlowLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.SwingUtilities;

public class FlowLayoutExample {
    public static void main(String[] args) {
        // Ensure GUI is created on the Event Dispatch Thread (EDT)
        SwingUtilities.invokeLater(FlowLayoutExample::createAndShowGUI);
    }

    private static void createAndShowGUI() {
        JFrame frame = new JFrame("Flow Layout Example");
        // Set FlowLayout with left alignment and a 10px horizontal & vertical gap
        frame.setLayout(new FlowLayout(FlowLayout.LEFT, 10, 10));

        // Add buttons dynamically
        for (int i = 1; i <= 5; i++) {
            frame.add(new JButton("Button " + i));
        }

        frame.setSize(500, 500);
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

//7b grid layout

```
package layouts;

import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JFrame;

public class GridLayoutExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Flow Layout Example");

        GridLayout layout = new GridLayout(3, 4);
        frame.setLayout(layout);

        for(int i = 0; i < 12; i++) {
            JButton button = new JButton("My Button " + i);
            frame.add(button);
        }

        frame.setSize(500, 500);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

//7.c border layout

```
package Layouts;

import java.awt.BorderLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
```

```
public class BorderLayoutExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Flow Layout Example");

        BorderLayout layout = new BorderLayout();
        frame.setLayout(layout);

        JButton nButton = new JButton("North");
        JButton sButton = new JButton("South");
        JButton wButton = new JButton("West");
        JButton eButton = new JButton("East");
        JButton cButton = new JButton("Center");

        frame.add(nButton, BorderLayout.NORTH);
        frame.add(sButton, BorderLayout.SOUTH);
        frame.add(wButton, BorderLayout.WEST);
        frame.add(eButton, BorderLayout.EAST);
        frame.add(cButton, BorderLayout.CENTER);

        frame.setSize(500, 500);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

//8.a Action event

```
package EventHandling;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

public class ActionEventExample {

    public static void main(String[] args) {
        JFrame frame = new JFrame("Action Event Example");

        JButton button = new JButton("My Button");
        frame.add(button);

        ButtonActionListener listener = new ButtonActionListener(frame);
        button.addActionListener(listener);

        frame.setSize(500, 500);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}

class ButtonActionListener implements ActionListener {

    private JFrame parentFrame;

    public ButtonActionListener(JFrame frame) {
        this.parentFrame = frame;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(parentFrame, e.getActionCommand() + " clicked");
    }
}
```

//8.b or 10 mouse event

```
package EventHandling;

import java.awt.Font;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class MouseEventExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Mouse Event Example");

        JLabel label = new JLabel("Hello", JLabel.CENTER);
        label.setFont(new Font("Arial", Font.ITALIC, 20));
        frame.add(label);

        label.addMouseListener(new MouseListener() {
            @Override
            public void mouseExited(MouseEvent e) {
                System.out.println(e.getX() + ", " + e.getY());
            }

            @Override
            public void mouseEntered(MouseEvent e) {
                System.out.println(e.getX() + ", " + e.getY());
            }

            @Override
            public void mouseReleased(MouseEvent e) {}

            @Override
            public void mousePressed(MouseEvent e) {}

            @Override
            public void mouseClicked(MouseEvent e) {
                System.out.println(e.getButton());
            }
        });

        frame.setSize(500, 500);
        frame.setExtendedState(JFrame.MAXIMIZED_BOTH);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

//8.c or 9 key event

```
package EventHandling;

import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

public class KeyEventExample {

    public static void main(String[] args) {
        JFrame frame = new JFrame("Key Event Example");

        frame.addKeyListener(new KeyAdapter() {
            @Override
            public void keyTyped(KeyEvent e) {
                JOptionPane.showMessageDialog(frame, e.getKeyChar());
            }
        });

        frame.setExtendedState(JFrame.MAXIMIZED_BOTH);
        frame.setSize(450, 600);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```