1. *Perimeter and Area of Rectangle (area = length * breadth, perimeter = 2 * (length + breadth))*

   - Appeared in: Slip No. 1, 11, 23, 25, 29

2. *Program to accept 5 numbers using command line arguments, sort and display them*

   - Appeared in: Slip No. 6, 9, 15, 28

3. *Employee Class (id, name, deptname, salary) with constructor, object count, and static member/method*

   - Appeared in: Slip No. 2, 7, 18

4. *Marker Interface - Class Product (product_id, product_name, product_cost, product_quantity)*

   - Appeared in: Slip No. 2, 3, 17

5. *CricketPlayer Class with batting average and sorting on basis of average*

   - Appeared in: Slip No. 7, 12, 15

6. *Mouse Events - MOUSE_MOVED and MOUSE_CLICK, display position in a TextField*

   - Appeared in: Slip No. 4, 25, 29

7. *Program to find the cube of a given number using function interface*

   - Appeared in: Slip No. 15, 22, 24, 26, 30

8. *Abstract Class "Order" with subclasses Purchase Order and Sales Order*

   - Appeared in: Slip No. 1, 9, 16

9. SYMarks

-Appeared in: Slip No. 8,11

---

1. **Perimeter and Area of Rectangle**:

```
class Rectangle {
   public static void main(String[] args) {
      int length = 5, breadth = 4;
      int area = length * breadth;
      int perimeter = 2 * (length + breadth);
      System.out.println("Area: " + area);
      System.out.println("Perimeter: " + perimeter);
   }
}
```

---

2. 2. **Accept 5 numbers using command line arguments, sort and display**:

```
import java.util.Arrays;

class SortNumbers {
    public static void main(String[] args) {
        int[] numbers = new int[args.length];
        for (int i = 0; i < args.length; i++) {
            numbers[i] = Integer.parseInt(args[i]);
        }
        Arrays.sort(numbers);
        System.out.println("Sorted numbers: " + Arrays.toString(numbers));
    }
}
```

---

3. **Employee Class with constructor, object count, static member/method**:

```
class Employee {
    static int count = 0;
    int id;
    String name, dept;
    double salary;

    Employee(int id, String name, String dept, double salary) {
        this.id = id;
        this.name = name;
        this.dept = dept;
        this.salary = salary;
        count++;
        System.out.println("Employee Created: " + name + ", Total: " + count);
    }
```

```
    public static void main(String[] args) {
        Employee e1 = new Employee(1, "John", "HR", 50000);
        Employee e2 = new Employee(2, "Jane", "IT", 60000);
    }
}
```

---

4. **Marker Interface - Class Product**:

```
interface Marker {}

class Product implements Marker {
    int id;
    String name;
    double cost;
    int quantity;
    static int count = 0;

    Product(int id, String name, double cost, int quantity) {
        this.id = id;
        this.name = name;
        this.cost = cost;
        this.quantity = quantity;
        count++;
    }

    void display() {
        System.out.println("Product: " + name + ", Cost: " + cost);
    }

    public static void main(String[] args) {
        Product p1 = new Product(1, "Phone", 30000, 10);
        Product p2 = new Product(2, "Laptop", 50000, 5);
```

```
        p1.display();
        p2.display();
        System.out.println("Total Products: " + count);
    }
}
```

---

5. **CricketPlayer Class with batting average and sorting**:

```
class CricketPlayer {
    String name;
    int innings, notOut, totalRuns;
    double batAvg;

    CricketPlayer(String name, int innings, int notOut, int totalRuns) {
        this.name = name;
        this.innings = innings;
        this.notOut = notOut;
        this.totalRuns = totalRuns;
        this.batAvg = (double) totalRuns / (innings - notOut);
    }

    void display() {
        System.out.println(name + " - Batting Avg: " + batAvg);
    }

    public static void main(String[] args) {
        CricketPlayer[] players = {
            new CricketPlayer("Player1", 10, 2, 800),
            new CricketPlayer("Player2", 15, 5, 1200)
        };
        for (CricketPlayer player : players) {
            player.display();
```

```
        }
    }
}
```

---

6. **Mouse Events - MOUSE_MOVED and MOUSE_CLICK**:

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

class MouseEvents extends Frame implements MouseListener, MouseMotionListener {
    TextField tf;

    MouseEvents() {
        tf = new TextField();
        tf.setBounds(50, 50, 200, 30);
        add(tf);
        addMouseListener(this);
        addMouseMotionListener(this);
        setSize(300, 300);
        setLayout(null);
        setVisible(true);
    }

    public void mouseClicked(MouseEvent e) {
        tf.setText("Clicked at " + e.getX() + ", " + e.getY());
    }

    public void mouseMoved(MouseEvent e) {
        tf.setText("Moved to " + e.getX() + ", " + e.getY());
    }

    public void mousePressed(MouseEvent e) {}
```

```
    public void mouseReleased(MouseEvent e) {}
    public void mouseEntered(MouseEvent e) {}
    public void mouseExited(MouseEvent e) {}
    public void mouseDragged(MouseEvent e) {}

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

---

7. **Cube of a number using function interface**:

```
import java.util.function.Function;

class Cube {
    public static void main(String[] args) {
        Function<Integer, Integer> cube = x -> x * x * x;
        int number = 3; // example input
        System.out.println("Cube: " + cube.apply(number));
    }
}
```

8. **Abstract Class "Order" with Purchase and Sales Order subclasses**:

```
abstract class Order {
    int id;
    String description;

    abstract void accept();
    abstract void display();
}

class PurchaseOrder extends Order {
```

```
    String customerName;

    PurchaseOrder(int id, String description, String customerName) {
        this.id = id;
        this.description = description;
        this.customerName = customerName;
    }

    void accept() {
        System.out.println("Purchase Order Accepted");
    }

    void display() {
        System.out.println("Purchase Order - ID: " + id + ", Customer: " + customerName);
    }
}

class SalesOrder extends Order {
    String vendorName;

    SalesOrder(int id, String description, String vendorName) {
        this.id = id;
        this.description = description;
        this.vendorName = vendorName;
    }

    void accept() {
        System.out.println("Sales Order Accepted");
    }

    void display() {
```

```
        System.out.println("Sales Order - ID: " + id + ", Vendor: " + vendorName);
    }

    public static void main(String[] args) {
        PurchaseOrder po = new PurchaseOrder(1, "Purchase Order 1", "Customer A");
        SalesOrder so = new SalesOrder(2, "Sales Order 1", "Vendor B");
        po.accept();
        po.display();
        so.accept();
        so.display();
    }
}
```

---

9. SYMarks

```
// SYMarks class
class SYMarks {
    public int ComputerTotal, MathsTotal, ElectronicsTotal;

    public SYMarks(int computerTotal, int mathsTotal, int electronicsTotal) {
        this.ComputerTotal = computerTotal;
        this.MathsTotal = mathsTotal;
        this.ElectronicsTotal = electronicsTotal;
    }

    public void displaySYMarks() {
        System.out.println("SY Marks - Computer: " + ComputerTotal + ", Maths: " + MathsTotal
+ ", Electronics: " + ElectronicsTotal);
    }
}

// TYMarks class
class TYMarks {
```

```
    public int Theory, Practicals;

    public TYMarks(int theory, int practicals) {
        this.Theory = theory;
        this.Practicals = practicals;
    }

    public void displayTYMarks() {
        System.out.println("TY Marks - Theory: " + Theory + ", Practicals: " + Practicals);
    }
}

// Student class
class Student {
    int rollNumber;
    String name;
    SYMarks syMarks;
    TYMarks tyMarks;

    public Student(int rollNumber, String name, SYMarks syMarks, TYMarks tyMarks) {
        this.rollNumber = rollNumber;
        this.name = name;
        this.syMarks = syMarks;
        this.tyMarks = tyMarks;
    }

    public char calculateGrade() {
        int totalComputerMarks = syMarks.ComputerTotal + tyMarks.Theory;
        int percentage = totalComputerMarks / 2;

        if (percentage >= 70) return 'A';
```

```
        if (percentage >= 60) return 'B';
        if (percentage >= 50) return 'C';
        if (percentage >= 40) return 'P';  // Pass class
        return 'F';  // Fail
    }

    public void displayResult() {
        System.out.println("Roll No: " + rollNumber);
        System.out.println("Name: " + name);
        syMarks.displaySYMarks();
        tyMarks.displayTYMarks();
        char grade = calculateGrade();
        System.out.println("Grade: " + grade);
    }

    public static void main(String[] args) {
        SYMarks sy = new SYMarks(75, 85, 65);  // SY marks
        TYMarks ty = new TYMarks(68, 80);  // TY marks
        Student student = new Student(101, "John Doe", sy, ty);

        student.displayResult();
    }
}
```