Exercise 7

Objectives

The aim of this exercise is to learn the concept of interface. In this exercise students will learn:

- The concept of interface
- How to create a class which implements an interface

A: The Relatable Interface [2 pt]

The following program compares two points by their positions.

```
import java.util.Scanner;

class RelatableApplication{

    public RelatableApplication(){
        Scanner sc = new Scanner(System.in);
        Point p1 = new Point(sc.nextInt(), sc.nextInt());
        Point p2 = new Point(sc.nextInt(), sc.nextInt());

        System.out.print("(" + p1.getX() + ", " + p1.getY() + ") ");
        if ( p1.isSmallerThan(p2) ){
            System.out.print("is smaller than ");
        } else {
            System.out.print("is not smaller than ");
        }
        System.out.println("(" + p2.getX() + ", " + p2.getY() + ") ");
    }

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

The comparison of points is based on the following criteria

- The one with the smaller x-coordinate value is smaller.
- If the x-coordinate values are the same, the one with the smaller y-coordinate value is smaller.

Your task is to implement a Point class that implements the following interfaces

```
public interface Relatable{
    public boolean isSmallerThan(Relatable other);
}
```

The behavior of the program can be seen in the following input/output example.

Sample Input	Sample Output
1 2 3 4	(1, 2) is smaller than (3, 4)
5 4 3 2	(5, 4) is not smaller than (3, 2)
5 7 5 8	(5, 7) is smaller than (5, 8)
1 2 1 2	(1, 2) is not smaller than (1, 2)

Submission Files	Types
Point.java	Java Class
Relatable.java	Java Interface

B: Sorting Relatable Objects I [3 pt]

The following SortingPointApplication is a program that sorts objects of the Point class, which implements the Relatable interface created in Problem A, in ascending order of coordinates.

```
import java.util.Scanner;

class SortingPointApplication{

    public SortingPointApplication(){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        Point[] p = new Point[n];

        for ( int i = 0; i < n; i++ ) {
            p[i] = new Point(sc.nextInt(), sc.nextInt());
        }

        SortingMachine machine = new SortingMachine();
        machine.sort(p);

        for ( int i = 0; i < p.length; i++ ) p[i].print();
    }

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

Your task is to create the SortingMachine class. Check the behavior of the program with the following input/output examples.

Sample Input	Sample Output
8 7 5 3 5 1 2 4 8 4 1 11 9 6 6 2 1	(1, 2) (2, 1) (3, 5) (4, 1) (4, 8) (6, 6) (7, 5) (11, 9)

Submission Files	Types
SortingMachine.java	Java Class
SortingPointApplication.java	Java Class

C: Sorting Relatable Objects II [4 pt]

The following SortingRectangleApplication is a program that sorts objects of the Rectangle class, which implements the Relatable interface created in Problem A, in ascending order of their area. The program outputs the areas of given rectangles in ascending order.

```
import java.util.Scanner;

class SortingRectangleApplication{

    public SortingRectangleApplication(){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        Rectangle[] r = new Rectangle[n];

        for ( int i = 0; i < n; i++ ) {
            Point p1 = new Point(sc.nextInt(), sc.nextInt());
            Point p3 = new Point(sc.nextInt(), sc.nextInt());
            r[i] = new Rectangle(p1, p3);
        }

        SortingMachine machine = new SortingMachine();
        machine.sort(r);

        for ( int i = 0; i < r.length; i++ )
            System.out.println(r[i].getArea());
    }

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

Your task is to create the Rectangle class. An object of a rectangle should be constructed by specifying two points on the diagonal line. **Note that you do not need to modify SortingMachine.** Check the behavior of the program with the following input/output examples.

Sample Input	Sample Output
8 0 0 3 2 0 0 4 6 0 0 8 2 0 0 6 4 0 0 1 1 0 0 12 8 0 0 4 9 0 0 22 1	1 6 16 22 24 24 36 96

Submission Files	Types
SortingRectangleApplication.java	Java Class
Rectangle.java	Java Class

Summary

In this exercise, we learned how to create a class that implements a specific interface. An interface is a kind of contract that guarantees that an object always has the specified methods. We learned how this works through the SortingMachine, which **can sort any object that implements Relatable**. In fact, we confirmed that we could sort both points and rectangles without any modifications to SortingMachine.