CSC120 2025S Lab No.11
Rectangles

This lab aims to write a multiple-source-code program, `RecMain`, for testing whether or not two rectangles intersect. The rectangles are placed on the two-dimensional grid. The program receives the coordinates of the upper-left and lower-right corners of each rectangle from the user. Then, the program receives a point from the user and answers whether the point is in the rectangles or, if not, the distance of the point from each rectangle. Here are some execution examples of the program:

```
1 Enter the coordinates of the two corners for rectangle 1: 5 10 20 2
2 Enter the coordinates of the two corners for rectangle 2: 6 10 19 3
3 The rectangles intersect.
4 Enter x and y: 5 5
5 The point is in No.1.
6 The point is not in No.2.
7 Its distance is 1.000000.
```

```
1 Enter the coordinates of the two corners for rectangle 1: 1 5 5 1
2 Enter the coordinates of the two corners for rectangle 2: 3 7 7 3
3 The rectangles intersect.
4 Enter x and y: 4 4
5 The point is in No.1.
6 The point is in No.2.
```

```
1 Enter the two corners of rectangle 1: 1 5 5 1
2 Enter the two corners of rectangle 2: 10 17 16 19
3 The rectangles are disjoint.
4 Enter x and y: 2 3
5 The point is in No.1.
6 The point is not in No.2.
7 The point's distance is 16.124515.
```

The coordinates of the corners of each rectangle are decomposed into two ranges: the range of the x-coordinates and the range of the y-coordinates. For example, a rectangle whose upper-left corner is $(4, 10)$ and its lower-right corner is $(11, -1)$ is represented by the x-range $[4, 11]$ and the y-range $[-1, 10]$.

The main class of the application is given as `RecMain.java` and you should use it. Your task is to two object classes, `Range` and `Rectangle`. The class `Range` is for storing a range and the `Rectangle` uses two `Range` objects as its instance variables to encode a rectangle.

`The class Range` The class `Range` has two `double` instance variables, `low` and `high`. These are the interval's lower and higher ends represented by the object. The class has the following constructors and methods:

```
public Range( double l, double h )
public double getLow()
public double getHigh()
public boolean intersect( Range o )
public boolean isIn( double p )
public double distance( double p )
```

The constructor's role is simple. It stores the values of the parameters in the two instance variables. If the order is reversed between the two formal parameters (that is, `l > h`), the constructor uses `l` for `high` and `h` for `low`. The methods `getLow` and `getHigh` are getters of the two instance variables. The method `intersect` checks whether or not the range represented by the object intersects with another `Range` object, `o`. The method `isIn` returns whether or not the value of `p` is in the range. The method `distance` returns the distance of `p` from the range, where the distance is defined as follows:

- if `p` is in the range, then the distance is 0;
- if `p` is greater than the higher end of the range, return `p` minus the value of the higher end;
- if `p` is smaller than the lower end of the range, return the value of the lower end minus `p`.

The class `Rectangle`

The class `Rectangle` uses a range for x-coordinates and a range for y-coordinates to define a rectangle. The two `Range` objects are the class's instance variables. The class has the following constructor and instance methods:

```
public Rectangle ( double xLow, double yLow, double xHigh, double yHigh )
public Range getXRange()
public Range getYRange()
public boolean intersect( Rectangle o )
public boolean isIn( double x, double y )
public double distance( double x, double y )
```

The constructor receives the coordinates for the rectangle's lower left corner and the upper left corner. The two getters are for retrieving the two ranges. The method `intersect` returns whether the object intersects with another `Rectangle` object, `o`. The method should return `true` if the object intersects with `o` concerning the `Range` instance variable for the x-coordinates and concerning the `Range` instance variable for the y-coordinates. The method `isIn` returns whether the point specified by `x` and `y` is inside the rectangle. The method `distance` returns the distance of the rectangle from the point specified by `x`. The distance is 0 if the point is in the rectangle. Otherwise, the distance is the square root of the sum of squares of coordinate-wise distance values.