Write two different representations for the Rectangle ADT in the starter code below, including abstraction functions and a rep invariant for each representation.

There are many ways valid to represent a rectangle. We will provide 2 samples, with 2 implementations each:

Write your class specification below

```java
/**
 * A Rectangle represents an immutable 2D rectangle with
 * the top-left corner p, width w, and height h.
 * We can denote a Rectangle as a triple (p, w, h).
 * All rectangles are rotated the same way. That is, the top
 * edge of the Rectangle is parallel to the x-axis.
 */

public class Rectangle {

    // Your fields for your representation, abstraction function, and rep invariant go below

    // Abstraction Function:
    // AF(this) = a rectangle with
    // top-left corner at (this.x, this.y) and
    // a width of this.width and
    // a height of this.height

    // Rep Invariant:
    // width > 0 and
    // height > 0

    private final double x;
    private final double y;
    private final double width;
    private final double height;

    } // end class Rectangle
```
/**
 * Uses the same class specification as above
 */
public class Rectangle {
    Your fields for your representation, abstraction function, and rep invariant go below

    // Abstraction Function:
    // AF(this) = a rectangle with top-left corner
    // at (this.x1, this.y1) and
    // width of this.x2 - this.x1 and
    // height of this.y1 - this.y2

    // Rep Invariant:
    // x1 < x2 and
    // y1 > y2

    private final double x1;
    private final double y1;
    private final double x2;
    private final double y2;
}

Write two different representations for the Rectangle ADT in the starter code below, including abstraction functions and a rep invariant for each representation.

Here is another valid way to represent a rectangle. There are many more valid ways to do this, but we’ve provided this other sample for you:

Write your class specification below

```java
/**
 * A Rectangle represents a mutable 2D rectangle with 4 corners. We can denote a Rectangle as an ordered list of points [p1, p2, p3, p4], where each point is a corner of the rectangle. The first point is the bottom-left corner, and the rest are assigned going clockwise.
 */
public class Rectangle {
    // Your fields for your representation, abstraction function, and rep invariant go below

    // Abstraction Function:
    // AF(this) = a rectangle with
    // p1 at (this.x1, this.y1)
    // p2 at (this.x2, this.y2)
    // p3 at (this.x3, this.y3)
    // p4 at (this.x4, this.y4)

    // Rep Invariant:
    // sqrt((x1 - x3)^2 + (y1 - y3)^2) ==
    // sqrt((x2 - x4)^2 + (y2 - y4)^2)

    private double x1, y1;
    private double x2, y2;
    private double x3, y3;
    private double x4, y4;
}
```
/**
 * Uses the same class specification as above
 */

public class Rectangle {
    // Your fields for your representation, abstraction function, and rep invariant go below

    // Abstraction Function:
    // AF(this) = a rectangle with
    //     p1 at (this.p.x, this.p.y)
    //     p2 at (this.p.x, this.p.y + this.height)
    //     p3 at (this.p.x + this.width, this.p.y + this.height)
    //     p4 at (this.p.x + this.width, this.p.y)

    // Rep Invariant:
    //     p != null and
    //     height > 0 and
    //     width > 0

    private Point p;
    private double height;
    private double width;
}