Polygon Fun

In this problem you will implement four static methods in the `PolygonFun` class. To complete these methods you have been given the `Coordinate` class which is used to represent the ordered pair (x, y) from a Cartesian Coordinate system. The `getX()` method return the x value and the `getY()` method returns the y value of the Cartesian Coordinate.

The first method is the `int whichQuadrant(Coordinate upperLeft, Coordinate lowerRight)` return the quadrant that contains the greatest area of the rectangle given the upper left corner and the lower right corner of the rectangle. The parameter `upperLeft` represents the x, y coordinate of the upper left corner of the rectangle. Similarly, the parameter `lowerRight` represents the x, y coordinate of the lower right left corner of the rectangle.

Note: Use the standard definitions for quadrants as denoted in Figure1.0, quadrant definition.

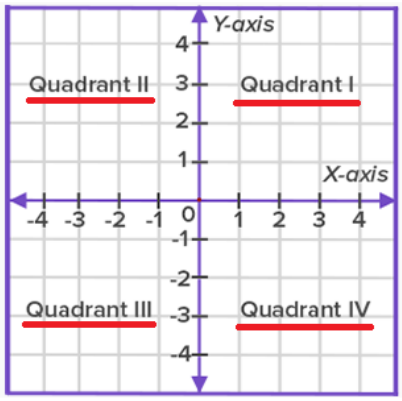


Figure 1.0
quadrant definition

The following code shows the results of the `whichQuadrant` method.

The following code	Returns
`PolygonFun.whichQuadrant( new Coordinate(-8, 10), new Coordinate(15, -6))`	1
`PolygonFun.whichQuadrant( new Coordinate(-4, 3), new Coordinate(1, -10)));`	3

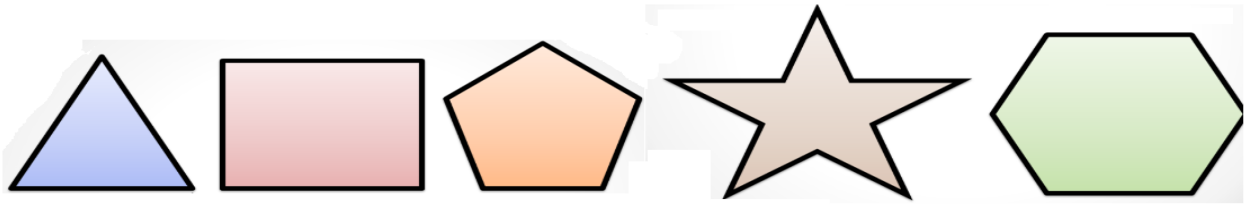
Saw the following problem and decided to make is a programming problem ☺

Given two regular polygons with a total of 17 interior angles and 53 diagonals, how many sides does each polygon have?

The second method is `String getNumberOfSides(int numAngles, int numDiagonals)`. This method finds the two regular polygons with a total of `numAngles` internal angles and a total of `numDiagonals` diagonals. And returns a string of the form "num1-num2" where num1 is number of sides of the regular polygon with the fewer number of sides and num2 is the number of sides of the regular polygon with the larger number of sides.

Recall: A polygon does not have any curved surface. A polygon should have at least three sides. Each side of the line segment must intersect with another line segment only at its endpoint.

The following images are examples of polygons. The star is NOT convex, all others are convex



Recall that the number of diagonals in a polygon = n(n-3)/2, where n is the number of polygon sides.

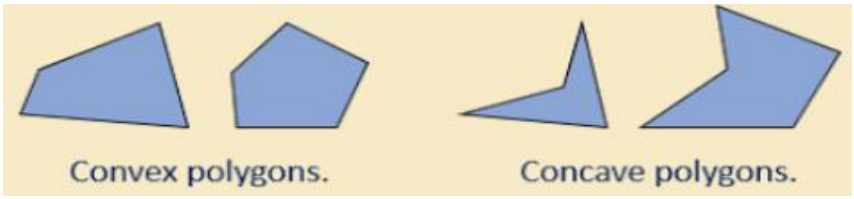
The following code shows the results of the `getNumberOfSides` method.

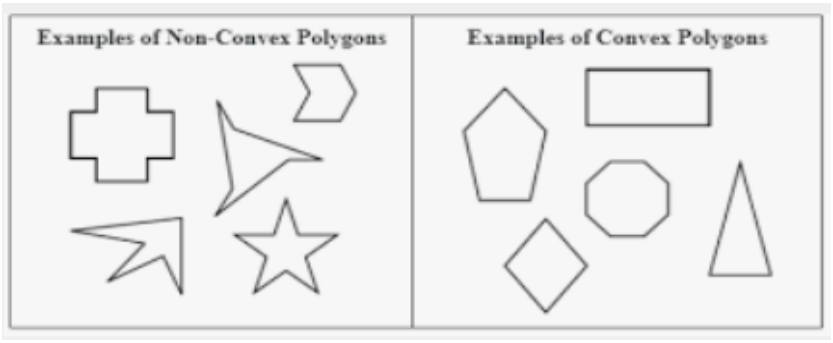
The following code	Returns
`PolygonFun.getNumberOfSides(17, 53);`	`"6-11"`
`PolygonFun.getNumberOfSides(34, 34);`	`"7-8"`

The third method is the `boolean isConvex(Coordinate[] vertices)` which returns true if the Polygon determined by the vertices is a convex polygon.
According to:

- Math Open Reference, a polygon is said to be convex if all of its interior angles are less than 180 degrees.
- byjus.com, a polygon is called a convex polygon when no line segments between the points, goes inside. The boundaries of the convex polygon do not go inside and all the vertices are pointed outside away from the center. The interior angles of a convex polygon are less than 180°. Regularly, a polygon is firmly convex, if each line segment with two nonadjacent vertices of the polygon is strictly internal to the polygon but on its endpoints.

The following images demonstrate examples of both convex and concave (or non-convex) polygons.





You might be able to solve this without the following formula, but I used it (sum of interior angles of a polygon with n sides is: (n-2) * 180° or (n-2) * Pi radians).

In solving this problem you may assume

- `vertices.length ==` number of vertices of the polygon

In addition, `vertices` are given in clockwise order, `vertices[m]` and `vertices[m+1]` are adjacent. You may assume no three adjacent vertices are collinear.

The following code shows the results of the `isConvex` method.

The following code	Returns

`PolygonFun.isConvex(new Coordinate[] {`
`  new Coordinate(0., 0.),`
`  new Coordinate(2., 2.),`
`  new Coordinate(4., 2.),`
`  new Coordinate(2., 0.) } );`
	`true`

`PolygonFun.isConvex(new Coordinate[] {`
`  new Coordinate(3., -20.),`
`  new Coordinate(2., -11.),`
`  new Coordinate(1., 5.),`
`  new Coordinate(5., 9.),`
`  new Coordinate(10., 13.),`
`  new Coordinate(15., 7.),`
`  new Coordinate(23., -3.),`
`  new Coordinate(18., -19.),`
`  new Coordinate(9., -20.) } );`
	`true`

`PolygonFun.isConvex(new Coordinate[] {`
`  new Coordinate(0., 0.),`
`  new Coordinate(1., 12.),`
`  new Coordinate(5., 5.),`
`  new Coordinate(10., 9.),`
`  new Coordinate(11., 1.) } );`
	`false`

`PolygonFun.isConvex(new Coordinate[] {`
`  new Coordinate(0., 0.)`
`  new Coordinate(1., 12.),`
`  new Coordinate(5., 5.),`
`  new Coordinate(10., 9.),`
`  new Coordinate(11., 1.) } );`
	`false`

`PolygonFun.isConvex(new Coordinate[] {`
`  new Coordinate(1., 5.),`
`  new Coordinate(5., 9.),`
`  new Coordinate(10., 13.),`
`  new Coordinate(15., 7.),`
`  new Coordinate(23., -3.),`
`  new Coordinate(18., -19.),`
`  new Coordinate(9., -20.),`
`  new Coordinate(3., -20.),`
`  new Coordinate(2., -1.) } );`
	`false`

`PolygonFun.isConvex(new Coordinate[] {`
`  new Coordinate(23., -3.),`
`  new Coordinate(18., -19.),`
`  new Coordinate(9., -20.),`
`  new Coordinate(3., -20.),`
`  new Coordinate(2., -1.),`
`  new Coordinate(1., 5.),`
`  new Coordinate(5., 9.),`
`  new Coordinate(10., 13.),`
`  new Coordinate(15., 7.) } );`
	`false`

The fourth and final method is the `int getNumConvex( List<Coordinate> v1,`

```
                                        List<Coordinate> v2,
                                        List<Coordinate> v3,
                                        List<Coordinate> v4,
                                        List<Coordinate> v5)
```

which returns the number of possible convex polygons that can be formed by picking one vertex from `v1`, one vertex from `v2`, one vertex from `v3`, one vertex from `v4`, and one vertex from `v5`. You may **NOT** assume the vertices selected from each list form a polygon. That is, you must test if the 5 vertices form a polygon and you must test if the 5 vertices form a convex polygon.

The following code shows the results of the `getNumConvex` method.

The following code	Returns

`List<Coordinate> v1 = new ArrayList<Coordinate>();`
`v1.add( new Coordinate(0., 0.) );`
`v1.add( new Coordinate(0., 1.) );`
 	
`List<Coordinate> v2 = new ArrayList<Coordinate>();`
`v2.add( new Coordinate(2., 2.) );`
`v2.add( new Coordinate(3., 2.) );`
 	
`List<Coordinate> v3 = new ArrayList<Coordinate>();`
`v3.add( new Coordinate(4., 1.) );`
`v3.add( new Coordinate(1., 1.) );`
 	
`List<Coordinate> v4 = new ArrayList<Coordinate>();`
`v4.add( new Coordinate(3., -3.) );`
 	
`List<Coordinate> v5 = new ArrayList<Coordinate>();`
`v5.add( new Coordinate(1., -2.) );`	

`PolygonFun.getNumConvex(v1, v2, v3, v4, v5)`	4

The four convex polygons that can be formed from the given list have the following vertices:

1. (0., 0.), (2., 2.), (4. , 1.), (3., -3.), and (1., -2.)
2. (0., 0.), (3., 2.), (4. , 1.), (3., -3.), and (1., -2.)
3. (0., 1.), (2., 2.), (4. , 1.), (3., -3.), and (1., -2.)
4. (0., 1.), (3., 2.), (4. , 1.), (3., -3.), and (1., -2.)