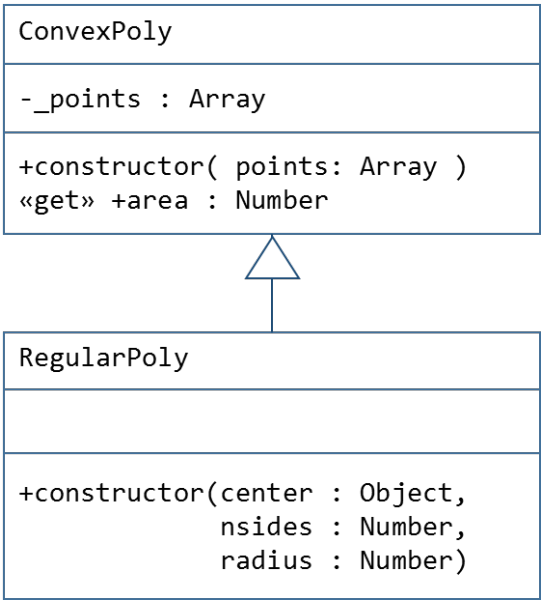
Objectives

- Practice defining classes and setting up prototypal inheritance.

Instructions

Write a JavaScript program that implements the following classes with the given inheritance relationship. Follow your class definitions with several examples that create instances of ConvexPoly and RegularPoly. Invoke the `area` getter method on each object.



ConvexPoly

- ConvexPoly encapsulates the concept of a convex polygon, which is a polygon with all internal angles having a value less than 180°.
- The ConvexPoly constructor takes an Array of points, where each point is an Object of the form `{x:…, y:…}` with `x` and `y` properties as Numbers. The constructor saves the `points` Array to the `_points` property.
- `area` is a 'getter' method that computes and returns the area of the ConvexPoly. Refer to Hints, below for details on one way to compute the area of a convex polygon.

RegularPoly

- RegularPoly encapsulates the concept of a regular polygon, which is a specialization of convex polygon. For a regular polygon all internal angles are of equal magnitude and all sides are of equivalent length. Examples of a regular polygon are a square (4 sides), pentagon (5 sides), hexagon (6 sides) and octagon (8 sides).
- The RegularPoly constructor takes a center point as an Object of the form `{x:…, y:…}`, the number of sides (nsides), and the radius of the circle that inscribes the polygon (radius). The constructor should generate an Array of all vertices that make up the regular polygon using these parameters and pass it to the superclass constructor. Each vertex may be computed by looping an angle variable (theta) from 0 to 2π in increments of (2π/nsides). Recall that each angle value, the fixed radius, and the center point may be transformed to (x, y) coordinates using the following formulas:

$$x = radius * \cos(theta) + center.x$$

$$y = radius * \sin(theta) + center.y$$

- The area getter should be inherited from ConvexPoly.

Hints

You may use the following helper function to implement the `area` getter method of ConvexPoly.

Heron's Formula computes the area of a triangle given its three side lengths. The helper function below expands this to start with the three vertices and computes the three side lengths.

```
// Private helper function implementing Heron's formula to compute triangle area
// given the three triangle vertices. Each point is an object of the form {x:…, y:…}
// area = sqrt(s*(s-a)*(s-b)*(s-c))
// a = length of side 1
// b = length of side 2
// c = length of side 3
// s = semiperimeter = (a+b+c)/2
let _heron = function(p1, p2, p3) {
  let a = Math.sqrt((p2.x-p1.x)*(p2.x-p1.x) + (p2.y-p1.y)*(p2.y-p1.y));
  let b = Math.sqrt((p3.x-p2.x)*(p3.x-p2.x) + (p3.y-p2.y)*(p3.y-p2.y));
  let c = Math.sqrt((p3.x-p1.x)*(p3.x-p1.x) + (p3.y-p1.y)*(p3.y-p1.y));
  let s = (a+b+c)/2;

  return Math.sqrt(s*(s-a)*(s-b)*(s-c));
}
```

The area of a convex polygon may be computed by dividing it into non-overlapping component triangles that tile the entire convex polygon and summing all component triangle areas. The series of triangles covering the entire internal area of a convex polygon may be generated by selecting one vertex from the convex polygon and iterating over the remaining adjacent vertex pairs, in order, excluding the original vertex. The original vertex combined with each vertex pair forms a triplet that defines a component triangle. Taken together, this series of triangles covers the entire convex polygon. The area of each component triangle may be computed with the `_heron()` helper function. Summing the areas of all component triangles produces the total area of the convex polygon.

Testing

To test my program I created two squares using each of the two classes and printed their areas. Following is the code and the resulting output.

```
let s1 = new RegularPoly( {x:50, y:50}, 4, 50*Math.sqrt(2));
console.log(`Square1 area: ${s1.area}`);

let s2 = new ConvexPoly( [{x:0, y:0},{x:0, y:100},{x:100, y:100},{x:100, y:0}] );
console.log(`Square2 area: ${s2.area}`);
```

```
Square1 area: 10000
Square2 area: 10000
```

Finishing Up

- You MUST enter header comments into your JavaScript file including (1) File name, (2) Your name, (3) Description and or purpose of the assignment
- You MUST comment your code, explaining what your code does in each section.
- Submit your single JavaScript file using Canvas under the appropriate assignment name.