Problem set: Writing and testing a `Circle` class

AP Computer Science A

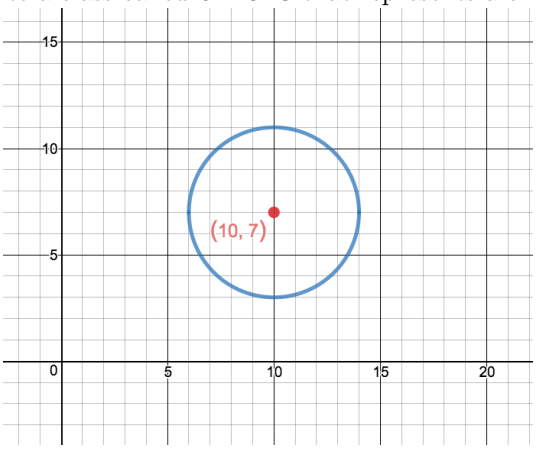
November 20, 2017

This problem set is based on the `circle-point` project under the folder "November Resources" on Schoology. Before you get started, please download the files `Circle.java` and `Canvas.java` and add them to your `circle-point` folder. They should show up as new classes in your BlueJ project.

The assignment is to complete the Circle class. When done, compress the entire folder and submit it to the Schoology assignment. First, you should do extensive testing to make sure that all the methods work correctly.

The `Circle` class

Write a class called `Circle` that represents a circle with a certain radius and center:



For example, this circle has a center at $(10, 7)$ and a radius of 4.

Instance variables

Both instance variables should be `private`.

- A `Point` object for the center.
- An `int` variable for the radius.

Constructors

All constructors should set the color to `"black"` You can reset the color using the `setColor()` method.

All constructors should also call the `draw()` method, so that the circle will be visible when it is created.

- First constructor: Takes no parameters. Creates a "default circle" with a center at $(0, 0)$ and a radius of 10.
- Second constructor: Takes three `int` parameters: the x and y of the center, and the radius.
- Third constructor: Takes two parameters: A `Point` object for the center, and an `int` for the radius.
- **Bonus challenge** (not required): Try to write these in such a way that the first and second constructors both work by *calling* the third constructor. You can call a constructor just like you call a method. If you just write them all separately that's okay too.

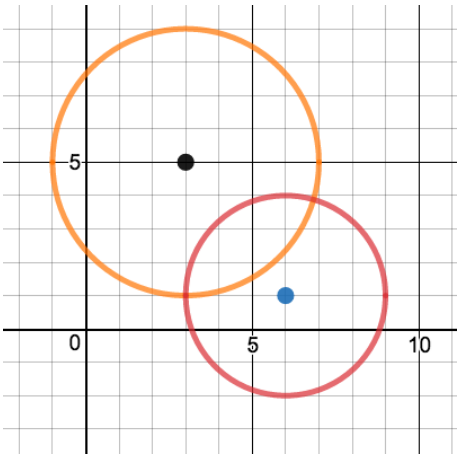
Accessor methods

You need π for a few of these. Whenever you need π, you can use the built-in value `Math.PI` which is a public constant from the `Math` class.

`Math.PI` is a `double` so of course some of these "should" return `double` values. However, in this problem we'll just use `int` return types. So just cast all answers to `int` before returning.

- `public int getRadius()` Returns the radius.
- `public Point getCenter()` Returns the center as a `Point` object.
- `public int perimeter()` Returns the perimeter, casted to `int`.
- `public int area()` Returns the area, casted to `int`.
- `public Point top()` Returns a `Point` object for the top of the circle.
- `public Point bottom()` Returns a `Point` object for the bottom of the circle.
- `public Point left()` Returns a `Point` object for the left-most point on the edge of the circle.
- `public Point right()` Returns a `Point` object for the right-most point on the edge of the circle.
- `public boolean overlaps(Circle other)` Returns `true` if this circle overlaps with `other`, and `false` if not. If the circles intersect at a single point on the edge, that counts as overlapping.

  Two circles overlap if the distance between the centers is less than or equal to the sum of the radii. For example, the two circles below overlap:



  Because:

$$\text{Distance between centers} = 5$$

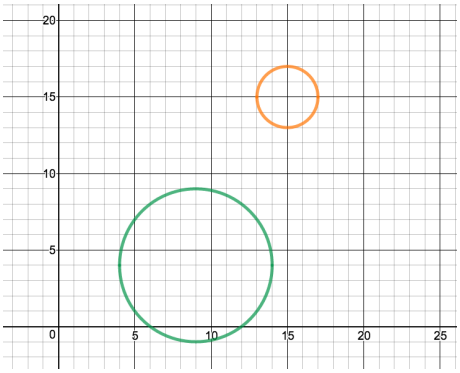
  And:

$$r_1 + r_2 = 3 + 4 = 7$$

  And $5 \leq 7$.

  You should use the `distance()` method in the `Point` class as part of this method.

- `public boolean verticalShadow(Circle other)` Should return `true` if `other` "casts a vertical shadow" on this circle, and `false` if not. What I mean is that in the diagram below,

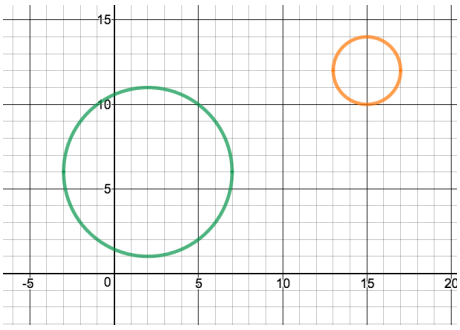


  Using those two circles, the method would return `true` because looking from straight above or straight below, the two circles seem to overlap even though they are actually far apart.

  This method should use some of the methods above that return `Point` objects.

- `public boolean horizontalShadow(Circle other)` Should return `true` if `other` "casts a horizontal shadow" on this circle, and `false` if not. Same idea as the last one, but looking horizontally.

  For example, for these two circles the method should return `true`:



  This method should use some of the methods above that return `Point` objects.

- `public String toString()` Returns a `String` of the form,

  `"Center: (4, 5). Radius: 10"`

  Or whatever the numbers are for the center and radius. This method should call `toString()` in `Point`.

Modifier methods

At the start of each of these methods, you should call the private `erase()` method to erase the old drawing from the Canvas. At the very end, call `draw()` to draw the new Circle.

- `public void move(int dx, int dy)` Moves the circle by `(dx, dy)`. This method should call the `move()` method in `Point`.
- `public void grow(int scale)` Multiplies the radius by `scale` to make the circle bigger.
- `public void shrink(scale)` Divides the radius by `scale`. EXCEPT if this would make the radius 0; in that case, do nothing. That way the radius can never be set to 0.
- `public void setColor(String color)` Re-sets the color of the circle to the parameter. You can use common color names like "blue" and "red" etc.
- `public void reCenter(Circle other)` Moves the center of this circle to the center of `other`. Note that `other` doesn't move; this circle moves.