CSD 3464 – Assignment 07

Overview:

The following question is an **adaption** to *Absolute Java* (6th Ed.)'s Chapter 08 Programming Project Q6 (pg. 520). Please follow the instructions included in **this** document and implement the following Java files:

- ⇒ RoomTester.java *(Contains a `main()` method –* **Provided for you!***)*
- ⇒ Shape.java (Abstract Class - **Provided for you!**)
- ⇒ Paelallogram.java
- ⇒ Renctangle.java
- ⇒ Square.java
- ⇒ Ellipse.java (**Provided for you!**)
- ⇒ Circle.java (**Provided for you!**)
- ⇒ Room.java (**Provided for you!**)

The above classes/files should both be inside a package called *q3*.

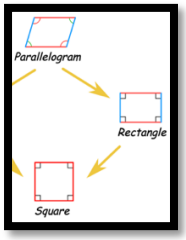
Problem Overview:

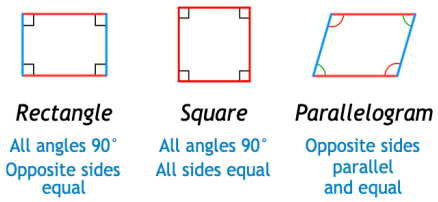
Your friend works as a carpenter and has just been asked to lay carpet in a house with various `Shaped Rooms` including: `Parallelogram`, `Rectangle`, `Square`, `Ellipse`, and `Circle`.

In addition, to calculating the amount of carpet your friend needs to purchase, your friend also needs to know how much trim they require as well. Wooden trim runs along the walls of each room.

Geometry Review:

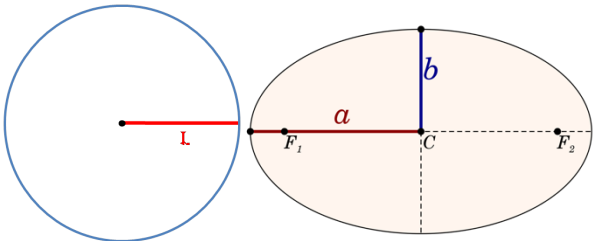
As a quick geometry review, a `Square` is a specific type `Rectangle` whose `length` and `width` are equal. Both a `Square` and `Rectangle` only have `angles` of 90°. A `Parallelogram` unlike a `Square` or `Rectangle` does not require the degrees of all `angles` to be equal, only its opposite `angles`. Additionally, a `Parallelogram` requires only its opposite sides be of the same length. All quadrilaterals have `angles` that sum to 360°.





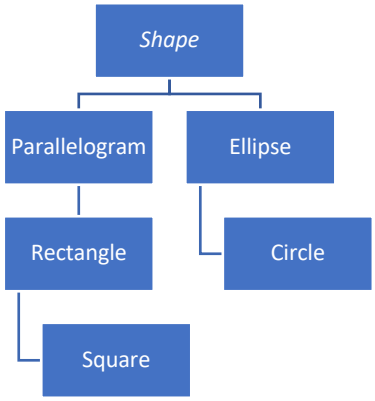
The remaining `Room` shapes we must handle are `Ellipse` and `Circle`. A `Circle` is a special case `Ellipse` much the same way a `Square` is a special case `Rectangle`. A

`Circle` has a `radius` and a `centre` point, while an `Ellipse` has a `major` and `minor` axis (as shown below).



Class Hierarchy:

We will be defining the below class hierarchy in this assignment. Shape will be an `abstract` class. *Shape*, *Ellipse*, and *Circle* are completed for you.

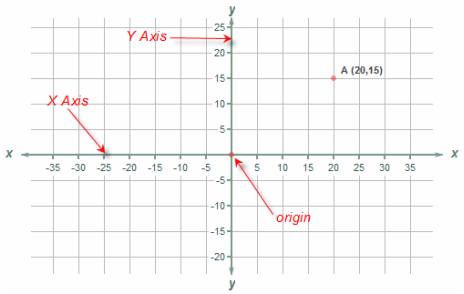


Instructions:

Step 01 – Review Shape.java

Start by reviewing the `public abstract class Shape`. Recall, an `abstract` class can have fields and methods like other classes we have seen throughout this course. Additionally, `abstract` classes are allowed to have `abstract` methods stubs. `Abstract` methods do not have an implementation but instead force derived classes that extend the class to implement them. We will see this in action when completing this assignment.

This `Shape` class has the `protected` properties `xCoordinate` and `yCoordinate` that contains the `double` values representing the `Shape`'s centre location on a 2D plane.



In terms of constructors the class also has a single constructor that takes two paraments, `double x` and `double y`, which are used to set the centre `xCoordinate` and `yCoordinate` fields.

The `Shape` class also contains `public` methods called `setCentreX` and `setCentreY` that take parameters `double x` and `double y` and update the appropriate `Shape` fields. Additionally, the class has two `public abstract` methods, `calcArea` and `calcPerimeter`, that return `double` values.

`Shape` also has a `toString` method that returns a `String` in the format:

- "Centre Coordinate: (**x** , **y**)"
    - **x** corresponds to the value of `xCoordinate` field
    - **y** corresponds to the value of `yCoordinate` field

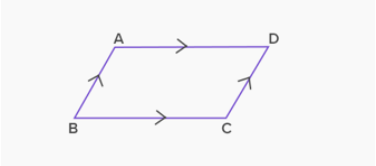
Lastly, `Shape` contains an `equals` method that returns a `boolean` value indicating if two `Shapes`, `Shape shp1` and `Shape shp2`, are equal.

- To accomplish we simply `return shp1.equals(shp2)`
    - **NOTE**: You will be defining an `equals` method for each type of `Shape` so that this code will work properly

Step 02 – Parallelogram.java

The second step of this assignment is to implement a `Parallelogram` class that `extends` the `abstract` class `Shape`. As the abstract `Shape` class has two `abstract` methods, we will have to add these methods to our `Parallelogram` class.

A `Parallelogram` has four sides, where opposite sides are of the same length. Therefore, we need two `protected double` fields `side1Length` and `side2Length` to store these values. A `Parallelogram` by definition also has opposite sides that are parallel meaning that this class requires two additional `protected float` fields, `angleAC` and `angleBD`, to store these opposite angles.



The `Parallelogram` class also should have a constructor that takes the six arguments: `double x, double y, double side1Length, double side2Length, double angleAC,` and `double angleBD` and sets the appropriate field values.

- NOTE: `super()` can be used to call `Shape`'s constructor to set the x and y centre coordinates

The `Parallelogram` class should also have a copy constructor and a corresponding `clone()` method.

Recall, that since the `Parallelogram` class `extends` the `abstract` class `Shape,` we must implement the functions to `calcArea` and `calcPerimeter`.

- `calcArea` should return the area of this `Parallelogram`
- `calcPerimeter` should return the perimeter of this `Parallelogram`

Area = (`side1Length` x `side2Length`) x `sin(radiansBD)`
Perimeter = Sum of all sides

- **Hint:** Use the `Math.sin()` method in area calculation
  - **IMPORTANT:** `Math.sin()` takes a single parameter value in radians, as we are storing angles in degrees use `Math.toRadians()` to convert `angleBD` to radians, `radiansBD`, **before** using it as an argument in `Math.sin()`

`Parallelogram` also has a `toString` method that returns a `String` in the format:

- "Shape: Parallelogram
  Centre Coordinate: (**x** , **y**)
  Angle AC: **angleAC**
  Angle BD: **angleBD**
  Side 1 Length: **side1Length**
  Side 2 Length: **side2Length**"

`Parallelogram` must also have an `equals` method that returns true if and only if two `Parallelogram`'s have the same centre coordinates, the same angles, and the same side lengths.

Next, add a `public` setter and getter for `side1Length` and `side2Length`. Also, add a `public setAngles` method that takes two `double` values representing angles and sets `angleAC` and `angleBD` accordingly.

- By definition, `angleAC` and `angleBD` should sum to 180°, but I do not expect you to handle this.

Lastly, the class requires a `public` getter for `angleAC` and `angleBD`.

Step 03 – Rectangle.java

The third step of this assignment is to implement a `Rectangle` class that `extends Parallelogram.` This class should not add any additional fields.

Your `Rectangle` class should a four argument constructor that takes the below parameters and sets the appropriate fields.

- `double xCoordinate`
- `double yCoordinate`
- `double length`
- `double width`

Additionally, in this constructor need to set both `angleAC` and `angleBD` to `90` as all angles of a rectangle are right angles (i.e. 90°).

- **NOTE:** To set `xCoordinate`, `yCordinate`, `angleAC`, `angleBD`, `length`, and `width` you can use a call to `super()` to access `Parallelogram`'s constructor

The `Rectangle` class should also have a copy constructor and a corresponding `clone()` method.

`Rectangle` also has a `toString` method that returns a `String` in the format:

- "Shape: Rectangle
  Centre Coordinate: (**x** , **y**)
  Angle: **angleAC**
  Length: **side1Length**
  Width: **side2Length**"

In terms of setters this class requires the addition of two `private` setters to set the `length` and `width` fields.

- `setLength()` should update `side1Length`
- `setWidth()` should update `side2Length`

Additionally, two `public` getters are required to access the `length` and `width` field values.

- `getLength()` should return the value of `side1Length`
- `getWidth()` should return the value of `side2Length`

Step 04 – Square.java

The third step of this assignment is to implement a `Square` class that `extends Rectangle.` This derived class introduces no additional fields.

Your `Rectangle` class should a constructor that takes the below three arguments and sets the appropriate fields

- `double xCoordinate`

- `double yCoordinate`
- `double length`

NOTE: Both `this.side1Length` and `this.side2Length` must be set to `length`. The reason for this is because a square has all four sides of the same length. Also, `this.angleAC` and `this.angleBD` need to set to `90` as all angles in a square a are right angles (i.e. 90°).

- **HINT**: To accomplish the above simply use `super()` to call `Rectangle`'s constructor. Just be aware when using this constructor you need to provide `sideLength` for both the `length` and `width` parameters as you are creating a square.

The `Rectangle` class should also have a copy constructor and a corresponding `clone()` method.

`Square` also has a `toString` method that returns a `String` in the format:

- "Shape: Square
  Centre Coordinate: (**x** , **y**)
  Angle: **angleAC**
  Side Length: **side1Length**"

This class should also include a `public` setter, `setSideLength(double length)`, that updates **both** `this.side1Length` and `this.side2Length` to the provided `length` value.

Additionally, a `public getLength()` method should be included to return the side length of the `Square`.

- **NOTE:** You may return the value of either `this.side1Length` or `this.side2Length` as both should be the same length.

Step 05 – Review Room.java

A `Room` class has been developed in full for you. Every `Room` has a `String name` (i.e. Main Floor – Living Room) and a `Shape shape`. The `Room` class' constructors and methods are completed for you. The three methods I want to draw your attention to are:

1. `getArea()` returns the `Room`'s area as a `double`
2. `getPerimeter()` returns the `Room`'s perimeter as a `double`
3. Copy constructor – Relies on `shape.clone()` to call the appropriate `clone()` method based on the type of `Shape` object (i.e. `Rectangle, Ellipse, etc.`).

The above methods use polymorphism and late-binding to call the correct methods based on the type of `Shape` object.

Step 06 – Execute RoomTester.java

Run the `main` method inside the `RoomTester` class. If you match the below output you have properly implemented the `clone()`, `getArea()`, and `getPerimeter()` methods described in steps 2-4. You are now completed the assignment!

```
Please enter the cost of carpert per square meter: $3
Please enter the cost of wooden trim per meter: $1

*******ROOMS********
Room: Cellar
Centre Coordinate: (0.00 , 0.00)
Radius: 5.00
Area: 78.539816 sqaured meters
Perimeter: 31.415927 meters
**********
Room: Oval Office
Centre Coordinate: (3.00 , 2.00)
Major Axis: 10.000000
Minor Axis: 5.000000
Area: 157.079633 sqaured meters
Perimeter: 48.442241 meters
**********
Room: Master Bedroom
Shape: Parallelogram
Centre Coordinate: (5.00 , 5.00)
Angle AC: 60.000000
AngleBD: 120.000000
Side 1 Length: 10.000000
Side 2 Length: 3.000000
Area: 25.980762 sqaured meters
Perimeter: 26.000000 meters
**********
Room: Main Floor - Kitchen
Shape: Rectangle
Centre Coordinate: (7.00 , 7.00)
Angle: 90.000000
Length: 10.000000
Width: 20.000000
Area: 200.000000 sqaured meters
Perimeter: 60.000000 meters
**********
Room: Bathroom
Shape: Square
Centre Coordinate: (10.00 , 10.00)
Angle: 90.000000
Side Length: 6.000000
Area: 36.000000 sqaured meters
Perimeter: 24.000000 meters
```

*******TOTALS*******
Carpet Required: 497.600211 squared meters
Trim Required: 189.858168 meters

*******COST*********
Carpet Cost: $1492.80
Trim Cost: $189.86
Total Cost: $1682.66