CS: 114 **OPPDA** **Spring, 2025**

Project2 (100 points) **Due 2/12/2025**

Submit your solutions to canvas. Do not send the entire project. All that is needed are the files ending in .java. Each class will be defined in its own separate .java file. All driver code should be put in one .java file. For this project that file is on canvas and is called `Project2_driver.java` Please make sure your name is included at the top of each .java file. Make one zip file that includes all the .java files and submit that one zip file to canvas.

Project Goals

1. Practice with ordering objects in different ways
2. Practice with polymorphism and polymorphic methods.
3. Practice implementing abstract methods, and abstract classes.
4. Practice implementing concrete methods, and concrete classes.
5. Practice using `super`
6. Practice overriding `toString()`
7. Practice with the interfaces `Comparable` and `Comparator`
8. Practice with `compareTo` and `compare`

<u>Problem 1</u>

Create an abstract `Shape` class. It should have two data members of type `double` that represent a position in the x-y plane. These data members should be named `x` and `y`. The `Shape` class should have the appropriate getters and setters for `x` and `y`, as well as a parametric constructor.

The Shape class should contain two abstract methods that compute the shape's area and perimeter. You should name those methods `ComputeArea`, and `ComputePerimeter`. They should both return a `double`.

Next create a `Circle` class and a `Rectangle` class that both extend the `Shape` class. In both of these classes you need to define concrete methods for `ComputeArea`, and `ComputePerimeter`.

Note the following lines that should be in the driver:

```
// Sort by area using the Comparable Interface
Arrays.sort(shapesList);
```

This above line will require the programmer to implement the `Comparable` interface. The `Comparable` interface requires the programmer to implement the abstract method `compareTo` in the classes `Circle`, and `Rectangle`.

Note the following line that will go in the driver:
`Arrays.sort(shapesList, new ShapeXPosComparator());`

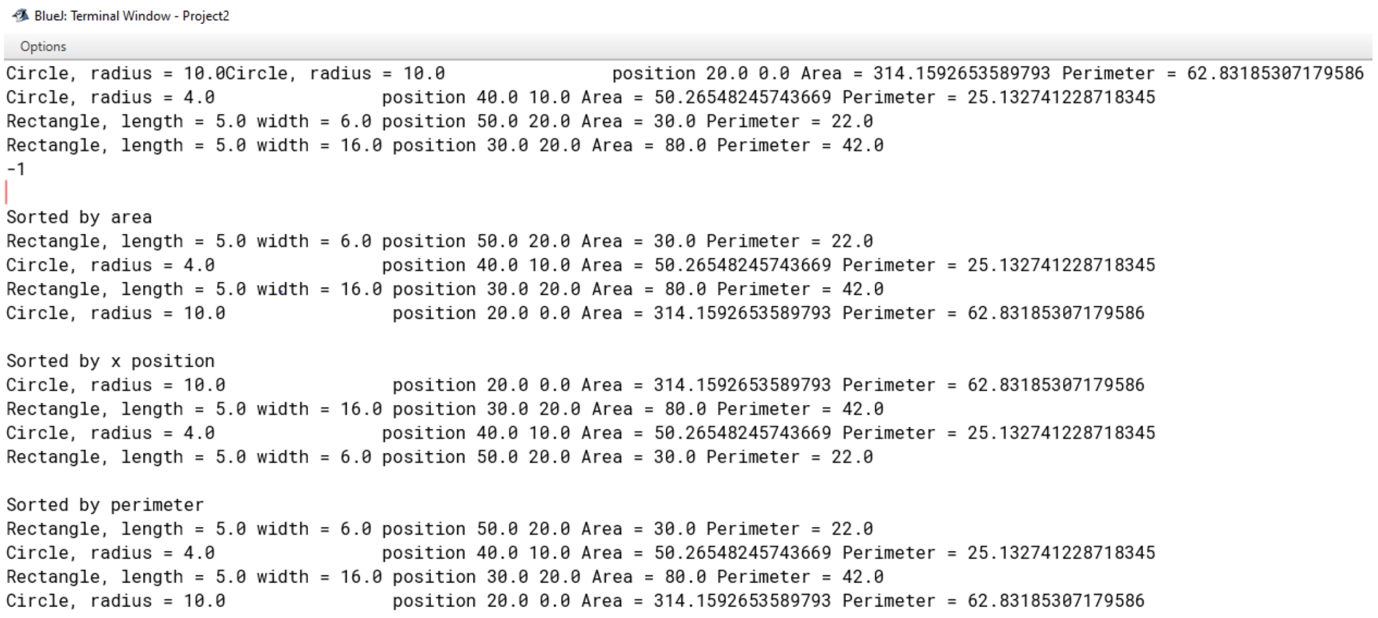
This will require you to implement a class that implements a `Comparator interface` and supplies a concrete implementation for the `compare` method. In the above line of code the class that implements the `Comparator` interface is called `ShapeXPosComparator.`

Note the following line that will go in the driver:
`Arrays.sort(shapesList, new ShapePerimeterComparator());`

This will require you to implement a class that implements a `Comparator interface` and supplies a concrete implementation for the `compare` method. In the above line of code the class that implements the `Comparator` interface is called `ShapePerimeterComparator.`

Be sure that all loops are enhanced for loops.

Output from your P1 of your driver program should look like:



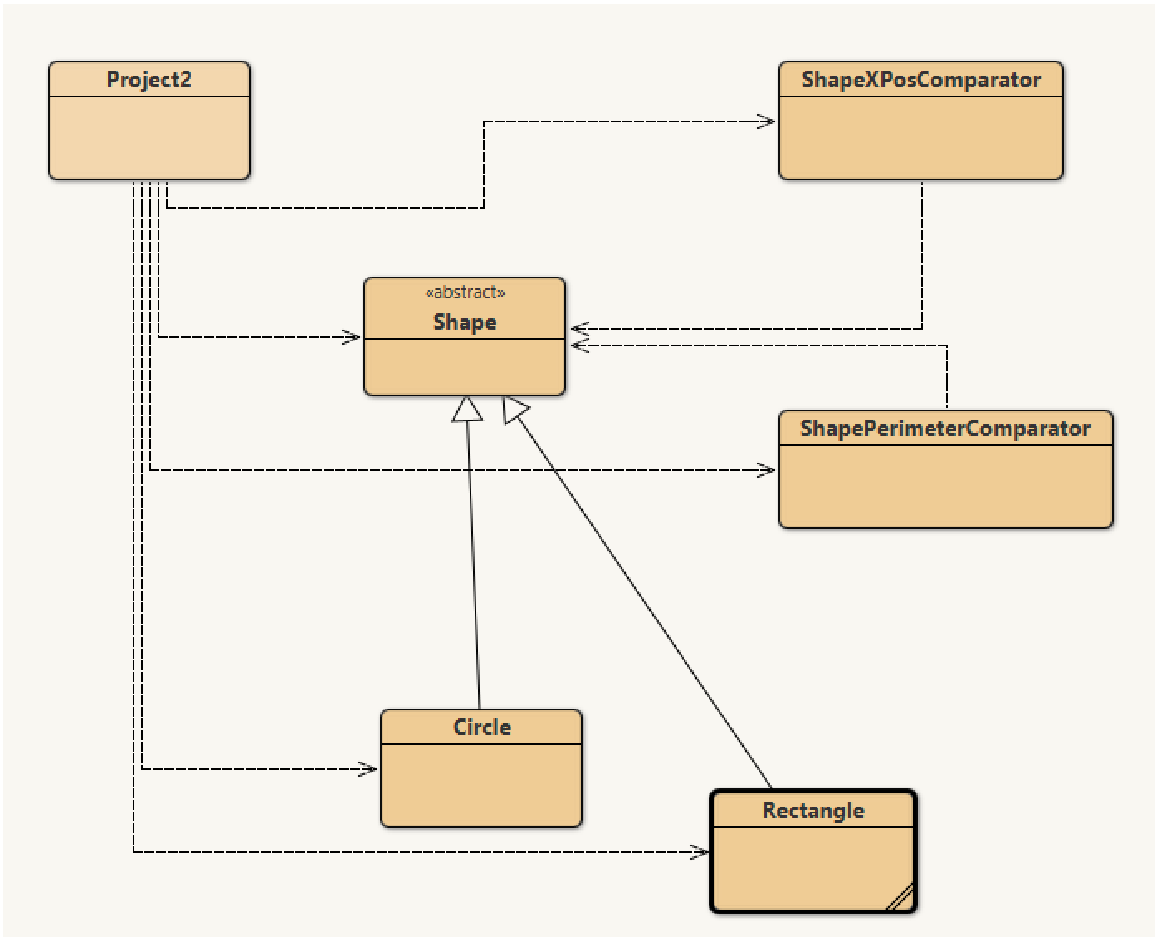
```
Circle, radius = 10.0Circle, radius = 10.0                position 20.0 0.0 Area = 314.1592653589793 Perimeter = 62.83185307179586
Circle, radius = 4.0                position 40.0 10.0 Area = 50.26548245743669 Perimeter = 25.132741228718345
Rectangle, length = 5.0 width = 6.0 position 50.0 20.0 Area = 30.0 Perimeter = 22.0
Rectangle, length = 5.0 width = 16.0 position 30.0 20.0 Area = 80.0 Perimeter = 42.0
-1

Sorted by area
Rectangle, length = 5.0 width = 6.0 position 50.0 20.0 Area = 30.0 Perimeter = 22.0
Circle, radius = 4.0                position 40.0 10.0 Area = 50.26548245743669 Perimeter = 25.132741228718345
Rectangle, length = 5.0 width = 16.0 position 30.0 20.0 Area = 80.0 Perimeter = 42.0
Circle, radius = 10.0                position 20.0 0.0 Area = 314.1592653589793 Perimeter = 62.83185307179586

Sorted by x position
Circle, radius = 10.0                position 20.0 0.0 Area = 314.1592653589793 Perimeter = 62.83185307179586
Rectangle, length = 5.0 width = 16.0 position 30.0 20.0 Area = 80.0 Perimeter = 42.0
Circle, radius = 4.0                position 40.0 10.0 Area = 50.26548245743669 Perimeter = 25.132741228718345
Rectangle, length = 5.0 width = 6.0 position 50.0 20.0 Area = 30.0 Perimeter = 22.0

Sorted by perimeter
Rectangle, length = 5.0 width = 6.0 position 50.0 20.0 Area = 30.0 Perimeter = 22.0
Circle, radius = 4.0                position 40.0 10.0 Area = 50.26548245743669 Perimeter = 25.132741228718345
Rectangle, length = 5.0 width = 16.0 position 30.0 20.0 Area = 80.0 Perimeter = 42.0
Circle, radius = 10.0                position 20.0 0.0 Area = 314.1592653589793 Perimeter = 62.83185307179586
```

The BlueJ class diagram looks like:



Create a class diagram for each class and include the class relationships. You do not have to create a class diagram for the driver.

<u>Problem 2</u>

Import the Java source code from Project1, Problem2. Modify the necessary code to sort the employee `List` by `lastName.` Then use an enhanced for-loop to display this newly ordered `List`. Next sort the `List` by `id`. Display the `List` with this new ordering. Finally, sort the `List` by salary and display the `List` in this ordering using a `ListIterator`, and then display the `List` a second time in reverse order.

Be sure to use Javadoc comments in all the code.

Your zip file should contain two UML diagrams (1 diagram for each problem), and a .java file for each class including the driver.