Worksheet W2: Comparison, Inheritance, and Polymorphism

Total points: 10

Out: January 28 (Tuesday evening)

Due: January 30 (Thursday end of day [2359 CDT according to D2L])

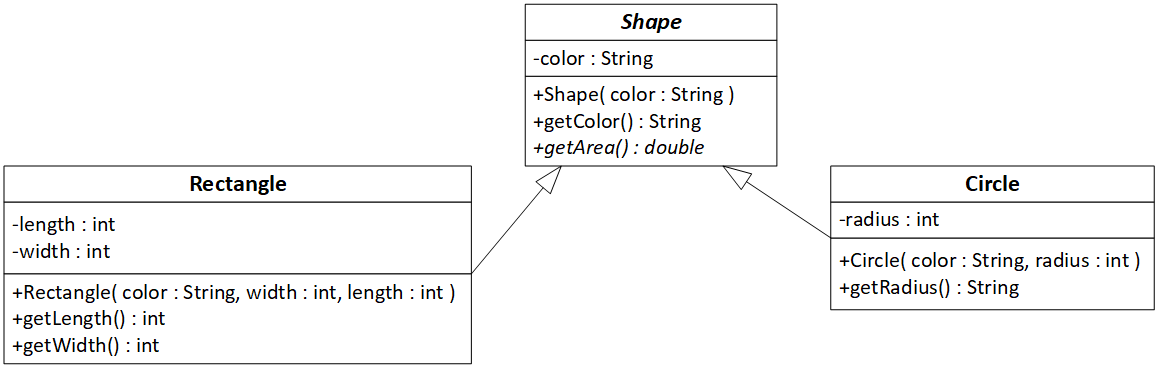
*No late submissions will be accepted*

## What to submit?

Upload exactly one file to the designated D2L folder, which may be a zip file. Type the methods on this sheet,[[1]](#footnote-1) or upload a single source code file and this file with Exercise 4 answered, zipped together. *The solution submitted must be typed.* The worksheet is available online and is open now.

## The UML of the classes for this worksheet

Consider the following UML that represents an **abstract class Shape** and two subclasses, called **Circle** and **Rectangle**. Note that the method **getArea()** is an abstract method (*you can see this because it is italicized*).



# Exercise 1: getArea() and toString() methods (2 pts, 0.5 pt each)

1. Write a concrete getArea() method for the Circle class. Notice that it returns a double. When calculating the area of a circle, use the Java API constant Math.PI rather than writing your own value of π.
2. @Override
3. public double getArea(){
4. return Math.PI\*Math.pow(radius, 2);
5. }
6. Write a concrete getArea() method for the Rectangle class. Notice that it returns a double.
7. @Override
8. public double getArea(){
9. return width\*length;
10. }
11. Write a toString() method for the Circle class. It should return a String of the form Circle, color = *color*, radius = *radius*. For instance, a blue circle of radius 7 should yield the String **Circle, color = blue, radius = 7**
12. @Override
13. public String toString(){
14. return "Circle, color = " + getColor() + ", radius = " + radius;
15. }
16. Write a toString() method for the Rectangle class. It should return a String of the form Rectangle, length = *length*, width = *width*. For instance, a 2x4 green rectangle should yield a String

**Rectangle, color = green, length = 2, width = 4**

@Override

public String toString(){

  return "Rectangle, color = " + getColor() + ", length = " + length + ", width = " + width;

}

# Exercise 2: compareTo() method (2 pts)

1. Write a compareTo method for the Shape class that compares the areas of two Shape objects. Think about whether or not that method has to be abstract. Choose wisely.
2. public String compareTo(Shape other) {
3. double result = Double.compare(this.getArea(), other.getArea());
4. if (result == 0){
5. return "Both shapes have the same area";
6. }
7. else if (result < 0){
8. return "This shape has a smaller area than the other shape";
9. }
10. else {
11. return "This shape has a larger area than the other shape";
12. }
13. }

# Exercise 3: ShapeTest class (2 pts)

1. Write a class ShapeTest that has a single main method and a method findLargestShape().

* **public static void main( String[] args )**
  + declares an array, called myShapes, that can hold up to 4 shapes
  + Adds 4 shapes of your choice to the myShapes array
    - At least one must be a Circle, and at least one must be a Rectangle.
  + calls a static method named findLargestShape that takes the myShapes array as a parameter and returns the largest shape. (See below for details of the method).
  + prints the returned shape using the toString() methods above.
* **public static Shape findLargestShape( Shape[] myShapes )** 
  + Take the myShapes array as a parameter, and use a loop to find the shape in the array with the largest area. Your method should return that Shape. If two or more shapes tie for largest area, you may return any one of those shapes.

1. public class shapeTest{
2. public static void main(String[] args) {
3. Shape[] myShapes = new Shape[4];
4. myShapes[0] = new Circle("red", 5);
5. myShapes[1] = new Rectangle("blue", 4, 5);
6. myShapes[2] = new Circle("green", 3);
7. myShapes[3] = new Rectangle("yellow", 6, 7);
8. System.out.println(findLargestShape(myShapes).toString());
9. }
10. public static Shape findLargestShape( Shape[] myShapes ){
11. Shape largestShape = myShapes[0];
12. for (int i = 0; i < myShapes.length; i++) {
13. if (myShapes[i].getArea() > largestShape.getArea()) {
14. largestShape = myShapes[i];
15. }
16. }
17. return largestShape;
18. }
19. }

# Exercise 4 (4 pts)

1. *(0.5 points each)* For each row in the below table, say whether the statement(s) is/are correct, generate(s) compilation error, or generate(s) run time error. If the statement causes on error, explain the cause of the error.

*Note: You will see a question like this on at least one midterm and probably on the final as well!*

|  |  |  |
| --- | --- | --- |
| Statement | Correct,  Compilation error, or  Runtime error? | Cause of error (if any) |
| Shape s1 = new Shape("red"); | Correct | NONE |
| Shape s2 = new Circle("red",10);  s2.getArea(); | CORRECT | **NONE** |
| Circle c1 = new Circle("red",10);  c1.getArea(); | CORRECT | **NONE** |
| Shape s3 = new Rectangle("blue",10,20);  s3.getWidth(); | **Runtime** | getWidth() is not in Shape class |
| Shape s4 = new Rectangle("blue",10,20);  ((Rectangle)s4).getWidth(); | CORRECT | **NONE** |
| Circle c2 = new Circle("red",10);  ((Rectangle)c2).getWidth(); | Runtime | Casted to the wrong class |
| Rectangle r1 = new Rectangle("blue",10,20);  r1.getWidth(); | CORRECT | **NONE** |
| Shape s5 = new Circle("red",10);  ((Rectangle)s5).getWidth(); | Runtime | Casted to the wrong class |

1. I recommend that you use Notepad or some sort of programming editor to type the methods so that Word doesn’t mess up your capitalization. [↑](#footnote-ref-1)