Week 3 worksheet: Inheritance & Polymorphism

Total points: 12 (graded out of 10)

Out: 2024 September 16 (Monday)

Due: 2024 September 18 (Wednesday end of day [2359 CDT according to D2L])

*No late submissions will be accepted*

## What to submit?

Upload exactly one file to the “Worksheet W3” D2L folder. Type the answers on that sheet. I recommend that you use some sort of programming editor or Notepad to type the methods so that Word doesn’t mess up your capitalization. *The solution submitted must be typed.* The worksheet is available online and is open now.

## UML Diagram for Exercises 1-3

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*).

|  |
| --- |
| **Circle** |
| -radius: int |
| +Circle(color: String, radius: int)  +getRadius(): int |

|  |
| --- |
| ***Shape*** |
| -color: String |
| +Shape(color: String)  +getColor(): String  *+getArea(): double* |

|  |
| --- |
| **Rectangle** |
| -length: int  -width: int |
| +Rectangle(color: String, width: int, length: int)  +getLength(): int  +getWidth(): int |

# Exercise 1

The Shape class has a public abstract double getArea() method. This means that you need to write a concrete getArea() method for Circle and Rectangle classes.

Part 1 *(1 point)*: Write a getArea() method for the Circle class. Note: When calculating the area of a circle, use the Java API constant Math.PI, rather than writing your own value of π.

public double getArea() {

double a = Math.PI \* (Math.pow(getRadius(), 2));

return a;

}

Part 2 *(1 point)*: Write a getArea() method for the Rectangle class

public double getArea() {

double a = getLength() \* getWidth();

return a;

}

# Exercise 2

Write public toString() methods for Circle and Rectangle meeting the specifications below. Remember that a toString() method doesn’t print anything itself, but just returns a String that describes the object.

Part 1 *(1 point)*: Write the 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 a String of

**Circle, color = blue, radius = 7**

public String toString() {

String o;

o = "Circle, " + "color = " + getColor() + ", radius = " + getRadius();

return o;

}

Part 2 *(1 point)*: Write the 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 of

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

public String toString() {

String o;

o = "Rectangle, " + "color = " + getColor() + ", length = " + getLength() + " width = " + getWidth();

return o;

}

Part 3 *(1 point)*: Does it make sense to write a toString() method for the abstract Shape class? Circle your answer, and explain why you said that.

No, the abstract Shape method, toString() would not be used. As an abstract class it is used as a base class to extend other classes. Therefore, it would not need to return anything to a string. If we wanted to get color we could use either the Circle or Rectangle classes.

# Exercise 3

*(3 points)*

On the next page, write a class ShapeTest that has a single main method and a method findLargestShape(). The main method does the following:

* declares an array, called myShapes, that can hold up to 100 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.

Write a public, static method called findLargestShape in the ShapeTest class that takes the myShapes array as a parameter, and uses 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.

***Your findLargestShapes( Shape[] myShapes ) method must not assume that the array parameter passed has four elements.***

public static Shape findLargestShape(Shape[] s) {

Shape largeShape = s[0];

for (Shape i : s) {

if (i != null && (i.getArea() > largeShape.getArea())) {

largeShape = i;

}

}

return largeShape;

}

*write exercise 3 answers here*

# Exercise 4

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

|  |  |  |
| --- | --- | --- |
| Statement | Correct,  Compilation error, or  Runtime error? | Cause of error (if any) |
| Shape s1 = new Shape("red"); | Compilation | Cannot instantiate the type Shape |
| Shape s2 = new Circle("red",10);  s2.getArea(); | Correct |  |
| Circle c1 = new Circle("red",10);  c1.getArea(); | Correct |  |
| Shape s3 = new Rectangle("blue",10,20);  s3.getWidth(); | Compilation | The method getWidth() is undefined for the type Shape |
| Shape s4 = new Rectangle("blue",10,20);  ((Rectangle)s4).getWidth(); | Correct |  |
| Circle c2 = new Circle("red",10);  ((Rectangle)c2).getWidth(); | Compilation | Cannot cast from Circle to Rectangle |
| Rectangle r1 = new Rectangle("blue",10,20);  r1.getWidth(); | Correct |  |
| Shape s5 = new Circle("red",10);  ((Rectangle)s5).getWidth(); | ClassCastException | class Circle cannot be cast to class Rectangle |