CS 2420
Fall semester section 01
Ricardo Gonzalez Mendez

Chapter 4 Review

Written:

4.1: it can be used anywhere the base class is expected. The members that become public are the ones that aren't declared as private or protected and therefor could be used by anyone.

4.4: autoboxing is when it automatically converts the primitive value to the object counterpart, while unboxing converts objects into primitive value.

4.8: to make an abstract method is like to make a check for yourself so when you call the method in child class you must redefine it so it reminds you to override. Abstract class cannot be initialized but can have child classes.

4.9: an interface is just like a blueprint for a class. In a interface you can define functionality but you can not implement it, while with an abstract you can have methods that can be used in the child or overridden. It can have abstract methods and variables.

4.11: before we used basic rules of inertances by having a basic generic class that could be reused easily. To implement generics now you do all the parameters in a generic way with the bound or object to be able to reuses it.

4.17: a. false b. true c. true d. true e. true f. true g. false h. true i. true j. true k. true l. false m. false n. true o. false p. false q. true r. false s. true t. false u. true v. true

Programming:

4.30: compare in circle

```java
public class Circle extends Shape implements Comparable<Circle>
{
    public Circle(double rad, double posx, double posy)
    {
        super(posx, posy);
        if(rad < 0) {
            throw new IllegalArgumentException ("cant be negative");
        }
        radius = rad;
    }

    public double area( )
    {
        return Math.PI * radius * radius;
    }

    public double perimeter( )
    {
        return 2 * Math.PI * radius;
    }

    public String toString( )
    {
        return "Circle: " + radius;
    }

    @Override
    public int compareTo(Circle other){
        if(this.radius == other.radius) {
            return 0;
        }
        else if(this.radius > other.radius) {
            return 1;
        }
        else if(this.radius < other.radius) {
            return -1;
        }
        return 0;
    }

    double radius;
}
```

4.33:

```java
public class Square extends Shape implements Comparable<Square> {

    double side;

     public Square(double side, double posx, double posy) {
         super(posx, posy);
        if(side < 0 ) {
            throw new IllegalArgumentException ("cant be negative");
        }
        this.side = side;
    }

    public double area( )
    {
        return side * side;
    }

    public double perimeter( )
    {
        return 2 * ( side + side );
    }

    public String toString( )
    {
    return "Square: " + side;
    }

    public double getSide( )
    {
        return side;
    }


    @Override
    public int compareTo(Square other){
        if(this.side == other.side) {
            return 0;
        }
        else if(this.side > other.side) {
            return 1;
        }
        else if(this.side < other.side) {
            return -1;
        }
        return 0;
    }
```

4.27:

I throw that exception in all of my shapes to make sure it wasn't less then zero/ negative.

4.47: add the pos x and y to know the position I did it in this class so my other shapes would have to implement it. Then I just added the distance formula.

```java
public abstract class Shape
{
    double posx;
    double posy;
    public abstract double area( );
    public abstract double perimeter( );

    public Shape(double posx, double posy) {
        this.posx = posx;
        this.posy = posy;
    }

    public double semiperimeter( )
    {
        return perimeter( ) / 2;
    }

    public static double distance(Shape sh1, Shape sh2) {

        return Math.sqrt(Math.pow(sh2.posx - sh1.posx, 2) + Math.pow(sh2.posy - sh1.posy, 2));
    }
//    public static void stretchAll(WhatType[] arr, factor) {
//      for(WhatType s: arr) {
//          s.stretch(factor);
//      }
//    }
}
```

```java
    public static void main(String[] args) {
            Circle c1 = new Circle(4, 2, 1);
            Circle c2 = new Circle(4, 2, 1);

            Square s1 = new Square(3, 32, 54);
            Square s2 = new Square(5, 3, 5);

            System.out.println(c1.compareTo(c2));

            System.out.println(distance(c1, c2));

            System.out.println(s1.compareTo(s2));

            System.out.println(distance(s1, s2));

        }

}
```

```
Console
<terminated> Shape [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (Oct 6, 2021, 9:55:09 PM – 9:5
0
0.0
-1
56.938563381947034
```

what I tested in the main and the answers I got. All of the stuff I implement worked.