Langara
THE COLLEGE OF HIGHER LEARNING.

Department of Computing Science & Information Systems
CPSC 1181
Lab#4
Oct 6, 2022

Objectives:

practice Inheritance and polymorphism
learn abstract class
create class hierarchy

Preparation:

Study class notes and example of sections Inheritance and polymorphism
Optional: Study chapter 9 of your text book

Due date:

Due Date: 11:00 PM on Wednesday, Oct 12, 2022

Where to upload:

Zip your files into yourstudentID.zip where yourstudentID is your student number, and upload it to dropbox lab4 in D2L.

No tutorial this week

What do to:

Important Notes:

- You should use full power of Polymorphism and inheritance in this lab assignment. Your marker will mark you based on the quality of your design. A working program is a must, but quality of your code and using polymorphism is another essential part of this lab assignment.
- You should not use keyword instanceof in your code for this lab assignment.

This lab assignment consists of one abstract class(GeometricShape), four subclasses extending from abstract class (Rectangle, Square, Oval, and Circle), and class GeometricShapeTester with a main method.

A Rectangle is defined by a location (xTopLeft,yTopLeft) and dimension (w , h).

A Square is defined by a location xTopLeft,yTopLeft) and dimension (size).

An Oval is defined by its center (x,y) and the horizontal and vertical radius (a,b).

A Circle is defined by its center (x,y) and radius (r).

1. The abstract class GeometricShape is give as shown below:

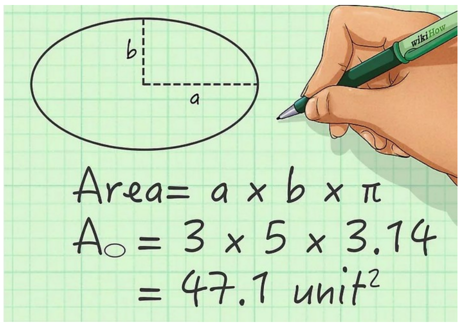
```
abstract class GeometricShape{
      private int x;
      private int y;
```

```
	public GeometricShape(int x,int y){
		this.x = x;
		this.y = y;
	}
	public int getX(){
		return x;
	}
	public int getY(){
		return y;
	}
	abstract public double getArea();
	abstract public String toString();
}
```

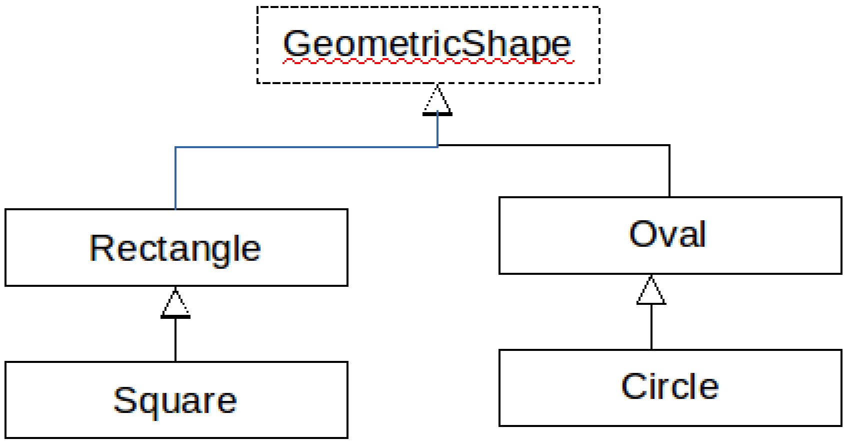
The class contains two private instance fields (x,y) that either represents the top-left coordinates of the shape for Rectangle and Square classes, or center of the shapes for Oval and Circle classes.
GeometricShape contains a constructor, two accessors methods, and two abstract methods. The method area() returns the area of the shape, and toString() method returns the string representation of the object. (Refer to the sample output of the program for the required format).

2. class Rectangle is a subclass of GeometricShape, and class Square is a subclass of Rectangle class. Rectangle has a width and a height. Those values should be positive integers. Add appropriate accessors methods to the class Rectangle.
Square is a subclass of Rectangle and should only contain a new constructor and toString() methods as the Rectangle version of all other methods will be sufficient. Square and Rectangle should have an appropriate constructor for initializing instance data.
In Rectangle, be sure to implement the abstract methods of the parent class.
A rectangle and a square are defined by their upper left coordinates, width and height. A square is a rectangle with equal width and height.

3. class Oval is a subclass of GeometricShape, and class Circle is a subclass of class Oval. Oval is defined by its center, a horizontal radius (a), and a vertical radius (b). Those values should be positive integers. Add appropriate accessors methods to the class Oval.
Circle is a subclass of Oval and it should only contain a new constructor and toString() method as the Oval version of other methods will be sufficient. Circle and Oval should have an appropriate constructor for initializing instance data.
In Oval, be sure to implement the abstract methods of the parent class.
Note that a circle is an oval with equal horizontal and vertical radius (a=b).
The formula to find area of an oval is shown in the figure that follows:



source: https://www.wikihow.com/Calculate-the-Area-of-an-Ellipse



The UML diagram of the class hierarchy is shown below:

4. Download GeometricShapeTester.java first. The file contains all classes required for this lab assignment. Note that although all classes have package access level except class GeometricShapeTester which is public, all instance fields of the classes regardless of access level must be declared as private. The class contains a static main method to run the program, and a constructor, that creates an array of GeometicShape objects, and adds twenty random shapes with random parameters to the array.
5. Develop body of the four classes Rectangle, Square, Oval, and Circle.
6. Develop body of the following method of class GeometricShapeTester:
// calculates and returns the average of the area of the collection of the shapes of the array
public double findAverage( ){ }

// Returns reference of the shape with maximum area.
public GeometricShape getMax(){ }

// Sort the shapes in the array in ascending order based on their area value

public void sort(){ }
Note: Do not use interface or Java built in utilities to sort the objects. We have not covered it yet. You should sort it using selection sort that is being covered in CPSC1150 to learn more about ArrayList , otherwise your solution will not be marked.

7. Uncomment the codes in the body of the main method to run the test cases. The format of the output of the test cases should be the same as shown in sample run of the program. The sample output of the test cases is shown below. Be sure that the program you are developing produces similar output. Study the format of the output and design your code based on the sample provided.
Note: You can use Math.round(..) method to round the values to the nearest integer for display purpose.

```
Test case 1:
average: 2223

Test case 2:
Square: [x:49, y:32, length:91, area:8281.0]

Test case 3:
list of the shapes:
Circle: [x:42, y:30, radius:9, area:254.0]
Square: [x:21, y:40, length:32, area:1024.0]
Square: [x:49, y:32, length:91, area:8281.0]
Square: [x:33, y:3, length:46, area:2116.0]
Square: [x:26, y:39, length:36, area:1296.0]
Circle: [x:8, y:19, radius:19, area:1134.0]
Rectangle [x:48, y:17, width:73, height:26, area:1898.0]
Oval: [x:14, y:18, h_radius:27, v_radius:56, area:4750.0]
Square: [x:22, y:31, length:57, area:3249.0]
Rectangle [x:24, y:38, width:11, height:10, area:110.0]
Oval: [x:25, y:47, h_radius:10, v_radius:43, area:1351.0]
Rectangle [x:27, y:1, width:56, height:40, area:2240.0]
Square: [x:29, y:48, length:25, area:625.0]
Circle: [x:14, y:37, radius:26, area:2124.0]
Rectangle [x:48, y:15, width:68, height:1, area:68.0]
Oval: [x:48, y:45, h_radius:1, v_radius:49, area:154.0]
Circle: [x:28, y:30, radius:31, area:3019.0]
Rectangle [x:22, y:14, width:99, height:80, area:7920.0]
Circle: [x:4, y:45, radius:30, area:2827.0]
Square: [x:39, y:47, length:4, area:16.0]

Test case 4:
shapes sorted in ascending order based on their area:
Square: [x:39, y:47, length:4, area:16.0]
Rectangle [x:48, y:15, width:68, height:1, area:68.0]
Rectangle [x:24, y:38, width:11, height:10, area:110.0]
Oval: [x:48, y:45, h_radius:1, v_radius:49, area:154.0]
Circle: [x:42, y:30, radius:9, area:254.0]
Square: [x:29, y:48, length:25, area:625.0]
Square: [x:21, y:40, length:32, area:1024.0]
Circle: [x:8, y:19, radius:19, area:1134.0]
Square: [x:26, y:39, length:36, area:1296.0]
Oval: [x:25, y:47, h_radius:10, v_radius:43, area:1351.0]
Rectangle [x:48, y:17, width:73, height:26, area:1898.0]
Square: [x:33, y:3, length:46, area:2116.0]
Circle: [x:14, y:37, radius:26, area:2124.0]
Rectangle [x:27, y:1, width:56, height:40, area:2240.0]
Circle: [x:4, y:45, radius:30, area:2827.0]
```

```
Circle: [x:28, y:30, radius:31, area:3019.0]
Square: [x:22, y:31, length:57, area:3249.0]
Oval: [x:14, y:18, h_radius:27, v_radius:56, area:4750.0]
Rectangle [x:22, y:14, width:99, height:80, area:7920.0]
Square: [x:49, y:32, length:91, area:8281.0]
```

Bonus: 10

Refer to Note page and listen to the Junit recorded lecture and study the notes. Then use Junit and develop automated test cases for Point3D_ver1.zip. Refer to course note page for the point3d.zip file.
You do not need to develop javadoc comments for your test cases.

What to submit

1. Comment all your classes and methods using the javadoc notation except for bonus part.
2. Capture and include the sample output of your program.
3. If you are using IDEs like Eclipse, then be sure that your program is complied and run in command window, otherwise your lab assignment may not be marked.
4. Comments about your assignment if needed these comments are not the comments documenting your code but rather something you need to convey to us about your assignment
5. Zip your assignment into yourStudentId.zip file and submit it to D2l Dropbox: lab4

TOTAL MARK: 40

Practice

You do not need to upload anything for this part. Do it just to practice.
Consider the following syntactically correct three Java classes:

File Slim.java:

```
public abstract class Slim{
    abstract public String getA();
    abstract public String getB();
}
```

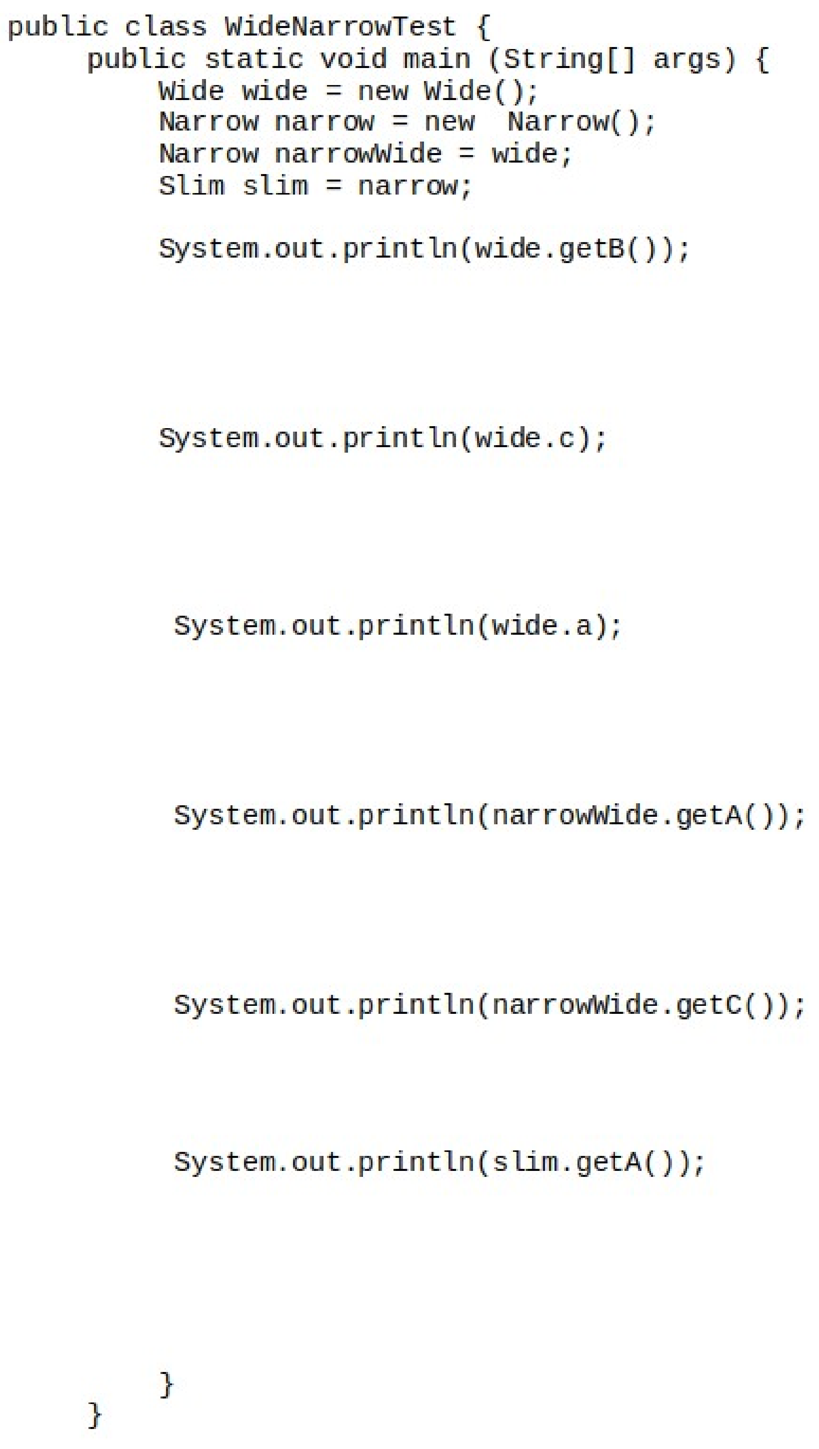
File Narrow.java

```
public class Narrow extends Slim {
    protected String a;
    private String b;
    public Narrow(){
        a = "NarrowA_";
        b = "NarrowB_" ;
    }
    public String getA(){
        return a;
    }
    public String getB(){
        return b;
    }
}
```

File Wide.java

```
class Wide extends Narrow {
    private String c;
    public Wide(){
        super();
        c = "Wide_";
    }
    public String getA(){
        return c+a;
    }
    public String getB(){
        return super.getB() + c;
    }
    public String getC(){
        return c;
    }
}
```

Write the output produced by each of the following statements on the space provided below each statement without running the program. If you think the statement produces an error message, write the word “error” and explain the error.
Note: The three classes (Slim,Wide, Narrow, and WideNarrowTest) belong to the same package.



```
public class WideNarrowTest {
     public static void main (String[] args) {
          Wide wide = new Wide();
          Narrow narrow = new  Narrow();
          Narrow narrowWide = wide;
          Slim slim = narrow;

          System.out.println(wide.getB());




          System.out.println(wide.c);




           System.out.println(wide.a);




           System.out.println(narrowWide.getA());




           System.out.println(narrowWide.getC());



           System.out.println(slim.getA());




          }
     }
```