Assignment Inheritance CSIS-1410

Learning Objectives:

- Declare a subclass that derives from a superclass
- Demonstrate polymorphic behavior
- Declare a variable of the superclass type and assign it an instance of the subclass type
  - Access the public members of the superclass type
  - Notice how the overridden versions of the subclass type are called
  - Notice how the subclass specific members are inaccessible
- Create an array of superclass type and use a foreach loop to iterate through it

Description:

Write a program to demonstrate the use of inheritance and polymorphism.
You will create **4 classes: Rectangle, Square, IsoscelesRightTriangle, and Circle**.
In addition you will create a class called **InheritanceApp**. This class includes the main method. Here we test the four other classes and we demonstrate the polymorphic behavior.

Declare the classes as described below:

ad Retangle:

- Rectangle has 2 private final fields of type int: **length** and **width**
- It has (exactly) one **parameterized constructor** that initializes both fields
- It provides a **getter** (get accessor method) for each of the fields (no setter)
- It overrides the **toString** method so that it produces a result of the form
  `Rectangle(`*length*`x`*width*`)` `e.g. Rectangle(5x4)`

ad Square:

- Square **extends Rectangle**
- No fields are declared in class Square
- It has a **parameterized constructor** with (only) one parameter
  The parameter is used to initialize both fields of Rectangle
- It has a method called getSide to expose the side-length of the square
- Override the **toString** method so that it will return a String of the following form:
  `Square(`*side*`)` `e.g. Square(4)`

ad IsoscelesRightTriangle:

- IsoscelesRightTriangle has 1 private final field of type **int** that is called **leg**
- It has one **parameterized constructor** that initializes the field
- It has a public method called **hypotenuse** that returns a value of type double
- It provides a **getter** for the field but no setter
- The **toString** method should return a String of the following form:
  `IsoscelesRightTriangle(`*leg*`)` `e.g. IsoscelesRightTriangle(5)`

ad Circle:

- Circle has 1 private final field of type **int** that is called **radius**
- It has one **parameterized constructor** that initializes the field
- It has two public **methods: diameter** and **circumference**
- It provides a **getter** for the field but no setter
- The **toString** method should return a String of the following form:
  `Circle(`*radius*`)` `e.g. Circle(3)`

ad InheritanceApp:

This class include the main method.

- This assignment does NOT accept user input
- Create instances of Rectangle, Square, IsoscelesRightTriangle, and Circle and

assign them to variables of the corresponding type.
Call the variables myRectangle, mySquare, myIsoscelesRightTriangle, and myCircle.
The rectangle has length 5 and width 4
The Square has a side length of 4
The IsoscelesRightTriangle has a leg size of 5
The Circle has a radius of 4
Notice: All arguments passed to the constructors in the main method are hard-coded. However, these values are independent of the implementations of Rectangle, Square, IsoscelesRightTriangle, and Circle.

For each of the 4 instances do the following

- Print the object (i.e. the toString method is called)
- call all the object specific methods and print the results in separate lines
- print a new line to make the output easier to read

Make your output look like the output provided

- To structure the output print the string "rectangle2: " and underline it with dashes

Output:

```
Rectangle(5x4)
```

- Create a variable of type Rectangle. Call it rectangle2 and assign it mySquare (the Square instance that you just created above).
- Print rectangle2 - Notice the Square output even though rectangle2 is a variable of type Rectangle
- In separate lines print the values returned by the methods getLength and getWidth

Try to call getSide. What happens?

- To structure the output print the string "Rectangle Array: " and underline it with dashes
- Create an array or Rectangles and call it rectangles
- Use an array initializer to initialize it with rectangle2, mySquare, and myRectnagle.
- use a for-each loop to do the following:
  - print the Rectangle
  - In separate lines print the values returned by the methods

Notice the polymorphic behavior in toString

Make the output look like the output provided on the right

```
Length: 5
Width: 4

Square(4)
Side: 4

IsoscelesRightTriangle(5)
Leg: 5
Hypotenuse: 7.1

Circle(4)
Diameter: 8
Circumference: 25.1
Radius: 4

rectangle2:
-----------
Square(4)
Length: 4
Width: 4

Rectangle Array:
----------------
Square(4)
Length: 4
Width: 4

Square(4)
Length: 4
Width: 4

Rectangle(5x4)
Length: 5
Width: 4
```

Turning in:

Make sure to **include a block comment with your name, course and assignment number on top of each source code file** (.java file).

Zip up all your source code files and submit the zip file via Canvas.