Lab 8 Requirements

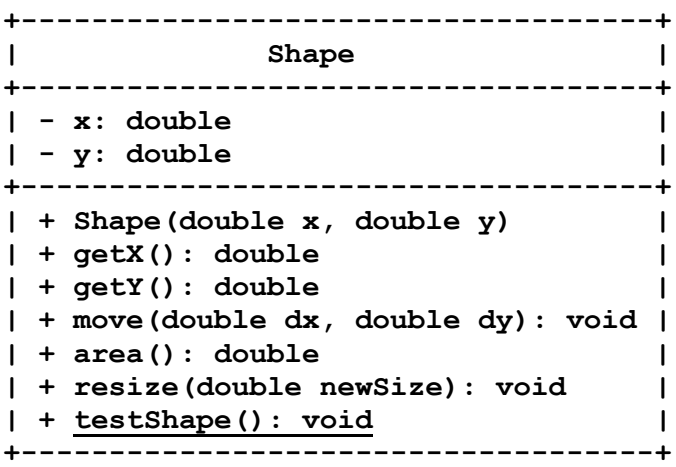
Create a new Eclipse workspace named "OOP" on the desktop of your computer. Create a Java Project named “Lab8”. Download the sample code lab8-src-for-student.zip from iSpace and unzip it. Next to import the sample code (right click the src -> Import… -> File System -> locate to the unzip folder -> import all the questions to your project) . More details, refer to file *How to Use Eclipse and submit to Autolab for student.pdf* on iSpace. Note: you must use the sample code, so AutoLab can work correctly.

Answer all questions below. Note: a question may need answers of the previous question(s), just copy paste the previous code to your new package. At the end of the lab, create a ZIP archive of all the src. Refer to previous labs requirements if you forgot how to compress files.

Extract your zip file to confirm that your zip file contains folders. Rename your src.zip to "Lab8_1234567890.zip" (replace 1234567890 with your student ID number). Upload the ZIP file onto AutoLab.

question 1 (Reviewed)

Create a **Shape** class with the following UML specification:



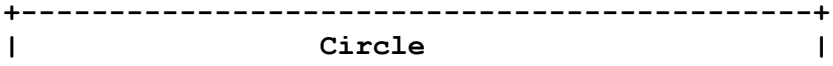
where the **x** and **y** instance variables store the position of the central point of the shape. The **move** method moves the central point of the shape by the amounts **dx** and **dy**. The **resize** method is used to change the size of a shape. The **testShape** method is static.

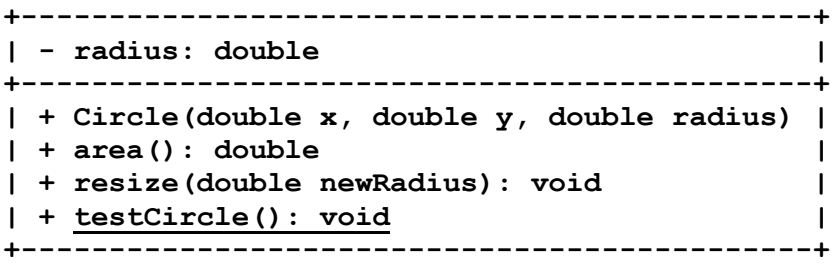
Add the following code to your program to test the **Shape** class:

```
public class Start {
	public static void main(String[] args) {
		Shape.testShape();
	}
}
```

question 2 (Mandatary)

Add a **Circle** class that derives from the **Shape** class and has the following UML specification:





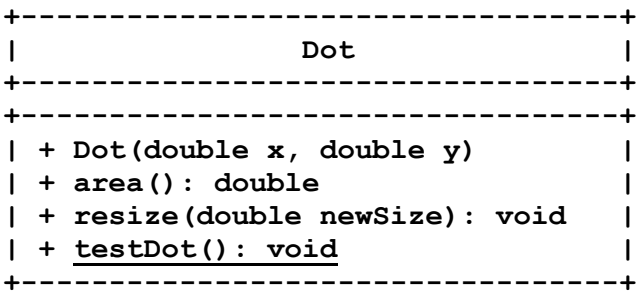
Use **`Math.PI`** to compute the area of a circle.

Resizing a circle changes its radius to be **`newRadius`**.

Do not forget to change the **`main`** method of the **`Start`** class to run the unit tests of the new **`Circle`** class.

question 3 (Mandatary)

Add a **`Dot`** class that derives from the **`Shape`** class and has the following UML specification:



It is not possible to resize a dot (a dot has no size). Therefore the **`resize`** method of the **`Dot`** class must throw an exception. The type of the exception object must be **`Exception`** and the message must be "**`Cannot resize a dot!`**"

Make sure you properly test the **`resize`** method using **`try-catch`**.

What is the problem with the **`resize`** method of the **`Shape`** class? How do you solve this problem?

Do you need to change the **`resize`** method of the **`Circle`** class then?

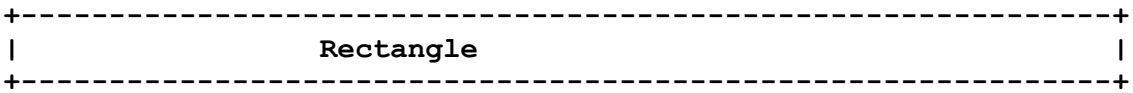
question 4 (Mandatary)

Add a **`CannotResizeException`** class that derives from the **`Exception`** class. Modify the **`resize`** and **`testDot`** methods of the **`Dot`** class to use this new class instead of using the class **`Exception`**.

Modify the **`Shape`** class accordingly.

question 5 (Mandatary)

Add two new classes **`Rectangle`** and **`Square`**. **`Rectangle`** derives from **`Shape`** and **`Square`** derives from **`Rectangle`**. **`Rectangle`** has the following UML specification:



```
| - width: double                                                    |
| - length: double                                                   |
+--------------------------------------------------------------------+
| + Rectangle(double x, double y, double width, double length)       |
| + area(): double                                                   |
| + resize(double newSize): void                                     |
| + testRectangle(): void                                            |
+--------------------------------------------------------------------+
```

Resizing a rectangle changes its **width** and **length** to both be **newSize**.

The constructor for **Square** takes three arguments: the **x** and **y** positions of the center of the square, and the **size** of the square.

Does the **Square** class need its own **area** and **resize** methods or not?

question 6 (Mandatary)

The **resize** method of the **Rectangle** class is annoying: it only allows you to resize a rectangle into a rectangle that has equal width and length. We want to be able to resize a rectangle into a rectangle that has different width and length. To do this, add to the **Rectangle** class a second new **resize** method that takes two arguments: a new width and a new length.

What is then the problem for the **Square** class?

question 7 (Optional)

To solve the previous problem, add to the **Square** class a new **resize** method that overrides the **resize** method that you just added to the **Rectangle** class.

In the new **resize** method of the **Square** class, if the new **width** and the new **length** are equal then the **resize** method should resize the square; if the new **width** and the new **length** are different then the **resize** method should throw a **CannotResizeException** exception with the message "**Cannot resize a square into a rectangle!**"

Make sure you properly test the **resize** method.

What happens with the **resize** method of the **Rectangle** class?

question 8 (Optional)

Add a **BadRadiusException** class that derives from the **Exception** class. Modify both the constructor and the **resize** method of the **Circle** class to use this new class when the given **radius** is negative. The message of the exception must be "**Radius must be positive!**"

Make sure you properly test both the modified constructor and the **resize** method.

What is then the problem for the **Shape** class?

Test Cases:

question 1

```
public static void testShape() {
}
```

question 2

```
public static void testCircle() {
	Circle c = new Circle(1.2, 3.4, 4.0);
	// getX, getY, and move are inherited from Shape.
	// area and resize come from Circle itself.
	System.out.println(c.getX() == 1.2);
	System.out.println(c.getY() == 3.4);
	System.out.println(c.area() == Math.PI * 16.0);
	// Move the circle.  The area does not change.
	c.move(7.8, 9.0);
	System.out.println(c.getX() == 9.0);
	System.out.println(c.getY() == 12.4);
	System.out.println(c.area() == Math.PI * 16.0);
	// Resize the circle.  The area changes but not the position.
	c.resize(8.0);
	System.out.println(c.getX() == 9.0);
	System.out.println(c.getY() == 12.4);
	System.out.println(c.area() == Math.PI * 64.0);
}
```

question 3

```
public static void testDot() {
	Dot d = new Dot(1.2, 3.4);
	// getX, getY, and move are inherited from Shape.
	// area and resize come from Dot itself.
	System.out.println(d.getX() == 1.2);
	System.out.println(d.getY() == 3.4);
	System.out.println(d.area() == 0.0);
	// Move the dot.  The area does not change.
	d.move(7.8, 9.0);
	System.out.println(d.getX() == 9.0);
	System.out.println(d.getY() == 12.4);
	System.out.println(d.area() == 0.0);
	// Resize the dot.  An exception is thrown, caught, and tested.
	try {
		d.resize(12.3);
	} catch(Exception ex) {
		System.out.println(ex.getMessage() == "Cannot resize a dot!");
		}
	// The area and position do not change.
	System.out.println(d.getX() == 9.0);
	System.out.println(d.getY() == 12.4);
	System.out.println(d.area() == 0.0);
}
```

question 4

```
public static void testDot() {
	Dot d = new Dot(1.2, 3.4);
	// getX, getY, and move are inherited from Shape.
```

```
        // area and resize come from Dot itself.
        System.out.println(d.getX() == 1.2);
        System.out.println(d.getY() == 3.4);
        System.out.println(d.area() == 0.0);
        // Move the dot.  The area does not change.
        d.move(7.8, 9.0);
        System.out.println(d.getX() == 9.0);
        System.out.println(d.getY() == 12.4);
        System.out.println(d.area() == 0.0);
        // Resize the dot.  An exception is thrown, caught, and tested.
        try {
              d.resize(12.3);
        } catch(Exception ex) {
              System.out.println(ex.getMessage() == "Cannot resize a dot!");
              }
        // The area and position do not change.
        System.out.println(d.getX() == 9.0);
        System.out.println(d.getY() == 12.4);
        System.out.println(d.area() == 0.0);

        try {
              d.resize(12.3);
        } catch(CannotResizeException ex) {
              System.out.println(ex.getMessage() == "Cannot resize a dot!");
        }
}
```

question 5

```
public static void testRectangle() {
        Rectangle r = new Rectangle(1.2, 3.4, 4.0, 5.0);
        // getX, getY, and move are inherited from Shape.
        // area and resize come from Rectangle itself.
        System.out.println(r.getX() == 1.2);
        System.out.println(r.getY() == 3.4);
        System.out.println(r.area() == 20.0);
        // Move the rectangle.  The area does not change.
        r.move(7.8, 9.0);
        System.out.println(r.getX() == 9.0);
        System.out.println(r.getY() == 12.4);
        System.out.println(r.area() == 20.0);
        // Resize the rectangle.  The area changes but not the position.
        r.resize(12.0);
        System.out.println(r.getX() == 9.0);
        System.out.println(r.getY() == 12.4);
        System.out.println(r.area() == 144.0);
}

public static void testSquare() {
        Square s = new Square(1.2, 3.4, 5.0);
        // getX, getY, and move are inherited from Shape.
        // area and resize are inherited from Rectangle.
        System.out.println(s.getX() == 1.2);
        System.out.println(s.getY() == 3.4);
        System.out.println(s.area() == 25.0);
        // Move the square.  The area does not change.
        s.move(7.8, 9.0);
        System.out.println(s.getX() == 9.0);
        System.out.println(s.getY() == 12.4);
        System.out.println(s.area() == 25.0);
        // Resize the square.  The area changes but not the position.
        s.resize(12.0);
```

```
        System.out.println(s.getX() == 9.0);
        System.out.println(s.getY() == 12.4);
        System.out.println(s.area() == 144.0);
}
```

question 6

```
public static void testRectangle() {
        Rectangle r = new Rectangle(1.2, 3.4, 4.0, 5.0);
        // getX, getY, and move are inherited from Shape.
        // area and resize come from Rectangle itself.
        System.out.println(r.getX() == 1.2);
        System.out.println(r.getY() == 3.4);
        System.out.println(r.area() == 20.0);
        // Move the rectangle.  The area does not change.
        r.move(7.8, 9.0);
        System.out.println(r.getX() == 9.0);
        System.out.println(r.getY() == 12.4);
        System.out.println(r.area() == 20.0);
        // Resize the rectangle.  The area changes but not the position.
        r.resize(12.0);
        System.out.println(r.getX() == 9.0);
        System.out.println(r.getY() == 12.4);
        System.out.println(r.area() == 144.0);
        // Resize the rectangle again with different width and length.
        // The area changes but not the position.
        r.resize(10.0, 15.0);
        System.out.println(r.getX() == 9.0);
        System.out.println(r.getY() == 12.4);
        System.out.println(r.area() == 150.0);

}

public static void testSquare() {
        Square s = new Square(1.2, 3.4, 5.0);
        // getX, getY, and move are inherited from Shape.
        // area and resize are inherited from Rectangle.
        System.out.println(s.getX() == 1.2);
        System.out.println(s.getY() == 3.4);
        System.out.println(s.area() == 25.0);
        // Move the square.  The area does not change.
        s.move(7.8, 9.0);
        System.out.println(s.getX() == 9.0);
        System.out.println(s.getY() == 12.4);
        System.out.println(s.area() == 25.0);
        // Resize the square.  The area changes but not the position.
        s.resize(12.0);
        System.out.println(s.getX() == 9.0);
        System.out.println(s.getY() == 12.4);
        System.out.println(s.area() == 144.0);

        // Resize the square again with different width and length!
        // Now the square is not square anymore!
        // The area changes but not the position.
        s.resize(10.0, 15.0);
        System.out.println(s.getX() == 9.0);
        System.out.println(s.getY() == 12.4);
        System.out.println(s.area() == 150.0);

}
```

question 7

```
public static void testRectangle() {
	Rectangle r = new Rectangle(1.2, 3.4, 4.0, 5.0);
	// getX, getY, and move are inherited from Shape.
	// area and resize come from Rectangle itself.
	System.out.println(r.getX() == 1.2);
	System.out.println(r.getY() == 3.4);
	System.out.println(r.area() == 20.0);
	// Move the rectangle.  The area does not change.
	r.move(7.8, 9.0);
	System.out.println(r.getX() == 9.0);
	System.out.println(r.getY() == 12.4);
	System.out.println(r.area() == 20.0);
	// Resize the rectangle.  The area changes but not the position.
	r.resize(12.0);
	System.out.println(r.getX() == 9.0);
	System.out.println(r.getY() == 12.4);
	System.out.println(r.area() == 144.0);


	// Resize the rectangle again with different width and length.
	// The area changes but not the position.
	try {
		r.resize(10.0, 15.0);
	} catch(CannotResizeException ex) {
		System.out.println("BUG! This must never happen!");
	}
	System.out.println(r.getX() == 9.0);
	System.out.println(r.getY() == 12.4);
	System.out.println(r.area() == 150.0);
}

public static void testSquare() {
	Square s = new Square(1.2, 3.4, 5.0);
	// getX, getY, and move are inherited from Shape.
	// area and resize are inherited from Rectangle.
	System.out.println(s.getX() == 1.2);
	System.out.println(s.getY() == 3.4);
	System.out.println(s.area() == 25.0);
	// Move the square.  The area does not change.
	s.move(7.8, 9.0);
	System.out.println(s.getX() == 9.0);
	System.out.println(s.getY() == 12.4);
	System.out.println(s.area() == 25.0);
	// Resize the square.  The area changes but not the position.
	s.resize(12.0);
	System.out.println(s.getX() == 9.0);
	System.out.println(s.getY() == 12.4);
	System.out.println(s.area() == 144.0);

	// Resize the square again with different width and length!

	try {
		s.resize(10.0, 15.0);
		System.out.println(s.getX() == 9.0); // Unreachable.
		System.out.println(s.getY() == 12.4);
		System.out.println(s.area() == 150.0);
	} catch(CannotResizeException ex) {
		System.out.println(ex.getMessage() == "Cannot resize a square into a
rectangle!");
	}
	// The area and position do not change.
```

```
        System.out.println(s.getX() == 9.0);
        System.out.println(s.getY() == 12.4);
        System.out.println(s.area() == 144.0);
        // Resize the square again with equal width and length.
        // The area changes but not the position.
        try {
              s.resize(10.0, 10.0);
        } catch(CannotResizeException ex) {
              System.out.println("BUG! This must never happen!");
        }
        System.out.println(s.getX() == 9.0);
        System.out.println(s.getY() == 12.4);
        System.out.println(s.area() == 100.0);
}
```

question 8

```
public static void testCircle() {
        // Create a circle with a positive radius.
        // No exception is thrown until we resize with a
        // negative radius.
        try {
              Circle c = new Circle(1.2, 3.4, 4.0);
              // getX, getY, and move are inherited from Shape.
              // area and resize come from Circle itself.
              System.out.println(c.getX() == 1.2);
              System.out.println(c.getY() == 3.4);
              System.out.println(c.area() == Math.PI * 16.0);
              // Move the circle.  The area does not change.
              c.move(7.8, 9.0);
              System.out.println(c.getX() == 9.0);
              System.out.println(c.getY() == 12.4);
              System.out.println(c.area() == Math.PI * 16.0);
              // Resize the circle.  The area changes but not the position.
              c.resize(8.0);
              System.out.println(c.getX() == 9.0);
              System.out.println(c.getY() == 12.4);
              System.out.println(c.area() == Math.PI * 64.0);
              // Resize the circle with a negative radius.
              // An exception is thrown, caught, and tested.
              c.resize(-12.3);
              System.out.println(c.getX() == 9.0); // Unreachable.
              System.out.println(c.getY() == 12.4);
              System.out.println(c.area() == Math.PI * 64.0);
        } catch(BadRadiusException ex) {
              System.out.println(ex.getMessage() == "Radius must be positive!");
        }
        // Create a circle with a positive radius.
        // Resize the circle with a zero radius.
        // No exception is thrown.
        try {
              Circle c = new Circle(1.2, 3.4, 4.0);
              c.resize(0.0);
                    // The area is now zero, the position does not change.
              System.out.println(c.getX() == 1.2);
              System.out.println(c.getY() == 3.4);
              System.out.println(c.area() == 0.0);
        } catch(BadRadiusException ex) {
              System.out.println("BUG! This must never happen!");
        }
        // Try to create a circle with a negative radius.
        // An exception is thrown, caught, and tested.
```

```
        try {
                Circle c2 = new Circle(1.2, 3.4, -5.6);
        } catch(BadRadiusException ex) {
                System.out.println(ex.getMessage() == "Radius must be positive!");
        }
        // Create a circle with a zero radius.
        // No exception is thrown.
        try {
                Circle c3 = new Circle(1.2, 3.4, 0.0);
                System.out.println(c3.getX() == 1.2);
                System.out.println(c3.getY() == 3.4);
                System.out.println(c3.area() == 0.0);
        } catch(BadRadiusException ex) {
                System.out.println("BUG! This must never happen!");
        }
}

public static void testDot() {
        Dot d = new Dot(1.2, 3.4);
        // getX, getY, and move are inherited from Shape.
        // area and resize come from Dot itself.
        System.out.println(d.getX() == 1.2);
        System.out.println(d.getY() == 3.4);
        System.out.println(d.area() == 0.0);
        // Move the dot.  The area does not change.
        d.move(7.8, 9.0);
        System.out.println(d.getX() == 9.0);
        System.out.println(d.getY() == 12.4);
        System.out.println(d.area() == 0.0);
        // Resize the dot.  An exception is thrown, caught, and tested.
        try {
                d.resize(12.3);
        } catch(Exception ex) {
                System.out.println(ex.getMessage() == "Cannot resize a dot!");
                }
        // The area and position do not change.
        System.out.println(d.getX() == 9.0);
        System.out.println(d.getY() == 12.4);
        System.out.println(d.area() == 0.0);

        try {
                        d.resize(12.3);
                } catch(CannotResizeException ex) {
                        System.out.println(ex.getMessage() == "Cannot resize a dot!");
                }
}

public static void testRectangle() {
        Rectangle r = new Rectangle(1.2, 3.4, 4.0, 5.0);
        // getX, getY, and move are inherited from Shape.
        // area and resize come from Rectangle itself.
        System.out.println(r.getX() == 1.2);
        System.out.println(r.getY() == 3.4);
        System.out.println(r.area() == 20.0);
        // Move the rectangle.  The area does not change.
        r.move(7.8, 9.0);
        System.out.println(r.getX() == 9.0);
        System.out.println(r.getY() == 12.4);
        System.out.println(r.area() == 20.0);
        // Resize the rectangle.  The area changes but not the position.
        r.resize(12.0);
        System.out.println(r.getX() == 9.0);
        System.out.println(r.getY() == 12.4);
```

```
        System.out.println(r.area() == 144.0);
        // Resize the rectangle again with different width and length.
        // The area changes but not the position.

        // Resize the rectangle again with different width and length.
        // The area changes but not the position.
        try {
                r.resize(10.0, 15.0);
        } catch(CannotResizeException ex) {
                System.out.println("BUG! This must never happen!");
        }
        System.out.println(r.getX() == 9.0);
        System.out.println(r.getY() == 12.4);
        System.out.println(r.area() == 150.0);
}

public static void testSquare() {
        Square s = new Square(1.2, 3.4, 5.0);
        // getX, getY, and move are inherited from Shape.
        // area and resize are inherited from Rectangle.
        System.out.println(s.getX() == 1.2);
        System.out.println(s.getY() == 3.4);
        System.out.println(s.area() == 25.0);
        // Move the square.  The area does not change.
        s.move(7.8, 9.0);
        System.out.println(s.getX() == 9.0);
        System.out.println(s.getY() == 12.4);
        System.out.println(s.area() == 25.0);
        // Resize the square.  The area changes but not the position.
        s.resize(12.0);
        System.out.println(s.getX() == 9.0);
        System.out.println(s.getY() == 12.4);
        System.out.println(s.area() == 144.0);

        // Resize the square again with different width and length!
        // Now the square is not square anymore!

        try {
                s.resize(10.0, 15.0);
                System.out.println(s.getX() == 9.0); // Unreachable.
                System.out.println(s.getY() == 12.4);
                System.out.println(s.area() == 150.0);
        } catch(CannotResizeException ex) {
                System.out.println(ex.getMessage() == "Cannot resize a square into a
rectangle!");
        }
        // The area and position do not change.
        System.out.println(s.getX() == 9.0);
        System.out.println(s.getY() == 12.4);
        System.out.println(s.area() == 144.0);
        // Resize the square again with equal width and length.
        // The area changes but not the position.
        try {
                s.resize(10.0, 10.0);
        } catch(CannotResizeException ex) {
                System.out.println("BUG! This must never happen!");
        }
        System.out.println(s.getX() == 9.0);
        System.out.println(s.getY() == 12.4);
        System.out.println(s.area() == 100.0);
}
```