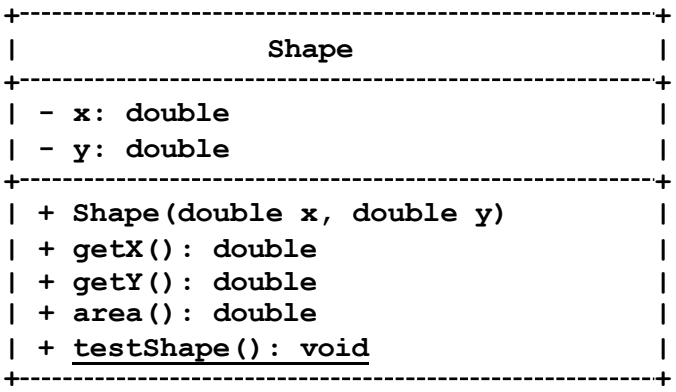
Lab 6 Requirements

Create a new Eclipse workspace named "OOP" on the desktop of your computer. Create a Java Project named “Lab5”. Download the sample code lab6-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 "Lab6_1234567890.zip" (replace 1234567890 with your student ID number). Upload the ZIP file onto AutoLab.

question 1 (Mandatory)

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



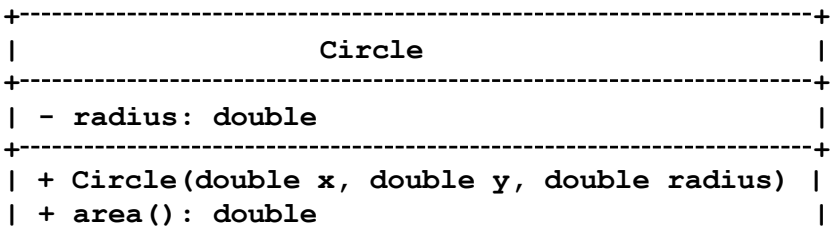
where the **x** and **y** instance variables store the position of the central point of the shape. The **area** method computes as result the area of the shape: unfortunately an unknown shape has an unknown area (we only know how to compute the area of specific shapes, like triangles, for example) so the **area** method just prints a message “**An unknown shape has an unknown area!**” and returns a meaningless result such as **-1.0** (because the **area** method must return something which is of type **double**). 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 (Mandatory)

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



```
| + testCircle(): void                          |
+-----------------------------------------------+
```

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

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

question 3 (Mandatory)

Add a new classes **Rectangle**. **Rectangle** derives from **Shape. Rectangle** has the following UML specification:

```
+-------------------------------------------------------------+
|                  Rectangle                                  |
+-------------------------------------------------------------+
| - width: double                                             |
| - length: double                                            |
+-------------------------------------------------------------+
| + Rectangle(double x, double y, double width, double length) |
| + getLength(): double                                       |
| + getWidth(): double                                        |
| + area(): double                                            |
| + equals(Object otherObject): boolean                       |
| + testRectangle(): void                                     |
+-------------------------------------------------------------+
```

Change the **main** method of the **Start** class to run the unit tests of the new **Rectangle** class. The **equals** method in the class **Rectangle** will check if an object is of same class type with same length and width. Use the **instanceof** operator method to determine the type of each shape.

It will return **true** when two objects have same type and same length and width, **false** for others.

question 4 (Mandatory)

We now want to be able to manipulate many shapes together in our software, not just one shape at a time. So add a **ManyShapes** class to your program with the following UML diagram:

```
+--------------------------+
|         ManyShapes       |
+--------------------------|
| - allShapes: ArrayList<Shape>|
+--------------------------+
| + ManyShapes()           |
| + addShape(Shape s): void |
| + listAllShapes(): void  |
| + testManyShapes (): void |
+--------------------------+
```

The **allShapes** instance variable is of type **ArrayList** which is a class provided to you by Java that you need to import into your program using: `import java.util.ArrayList;`

You can then define the **allShapes** instance variable like this: `private ArrayList allShapes;`

In the **ManyShapes** constructor you need to create a new **ArrayList** object and store it in the instance variable **allShapes**, like this: `this.allShapes = new ArrayList();`

(if you forget to do this then the instance variable **allShapes** will point at nothing and you will get an error when you run your program and you try to call a method of the nonexistent arraylist object).

An **ArrayList** object works both like an array and like a list, in which you can store as many other objects of type **Object** as you want:

- you can add a new object **o** to the arraylist by using the **add(o)** method of the arraylist;
- you can get the number of elements in the arraylist by using the **size()** method of the arraylist;
- you can access a specific element of the arraylist at index **i** by using the **get(i)** method of the arraylist (element indexes start at zero in an arraylist).

The **addShape** method takes a shape as argument and adds it to the arraylist.

Add a new method **toString** to the **Shape** class that overrides the **toString** method which is inherited by the **Shape** class from the **Object** class. This method should then return the string "**Shape area is XXX**", where **XXX** is the result of the **area** method from the same class **Shape**. Then override the **toString** method in every subclass of **Shape** to return the right string for the subclass in a similar way.
Modify the **listAllShapes** method to tell us both the area and the type for each shape in the arraylist. For example, if the arraylist currently contains a **Rectange** object of area 20 and a **Circle with area 50.2** object then the **listAllShapes** method should print:

```
Rectangle area is 20.0
Circle area is 50.2
```

Here is the code of the **testManyShapes** method:

```
public static void main(String[] args) {
        System.out.println("testShape");
        Shape.testShape();
        System.out.println("\ntestCircle");
        Circle.testCircle();
        System.out.println("\ntestRectangle");
        Rectangle.testRectangle();
        System.out.println("\ntestManyShapes");
        ManyShapes.testManyShapes();
    }
```

Test Cases:

question 1

```
public static void testShape() {
        Shape s = new Shape(1.2, 3.4);
        System.out.println(s.getX() == 1.2);
        System.out.println(s.getY() == 3.4);
        System.out.println(s.area() == -1.0); // Will print an error message too.
}
```

question 2

```
public static void testCircle() {
        Circle c = new Circle(1.2, 3.4, 4.0);
       // getX and getY are inherited from Shape.
        // area comes 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);
}
```

question 3

```
 public static void testRectangle() {
            Rectangle r = new Rectangle(1.2, 3.4, 4.0, 5.0);
            Rectangle r1 = new Rectangle(10, 11, 4.0, 5.0);
            Rectangle r2 = new Rectangle(10, 11, 8.0, 5.0);
            Circle c = new Circle(1.2, 3.4, 5.0);
            // getX and getY are inherited from Shape.
            // area comes from Rectangle itself.
            System.out.println(r.getX() == 1.2);
            System.out.println(r.getY() == 3.4);
            System.out.println(r.area() == 20.0);
            System.out.println(r.equals(r1));
            System.out.println(!r1.equals(r2));
            System.out.println(!r1.equals(c));
      }
```

question 4

```
public static void testShape() {
        Shape s = new Shape(1.2, 3.4);
        System.out.println(s.getX() == 1.2);
        System.out.println(s.getY() == 3.4);
        System.out.println(s.area() == -1.0); // Will print an error message too.
        System.out.println(s.toString().equals("Shape area is -1.0"));
 }
```

```
 public static void testRectangle() {
            Rectangle r = new Rectangle(1.2, 3.4, 4.0, 5.0);
            Rectangle r1 = new Rectangle(10, 11, 4.0, 5.0);
            Rectangle r2 = new Rectangle(10, 11, 8.0, 5.0);
            Circle c = new Circle(1.2, 3.4, 5.0);
            // getX and getY are inherited from Shape.
            // area comes from Rectangle itself.
            System.out.println(r.getX() == 1.2);
            System.out.println(r.getY() == 3.4);
            System.out.println(r.area() == 20.0);
            System.out.println(r.equals(r1));
            System.out.println(!r1.equals(r2));
            System.out.println(!r1.equals(c));
            System.out.println(r.toString().equals("Rectangle area is " + 4.0 * 5.0));
}
```

```
 public static void main(String[] args) {
            System.out.println("testShape");
            Shape.testShape();
            System.out.println("\ntestCircle");
            Circle.testCircle();
            System.out.println("\ntestRectangle");
            Rectangle.testRectangle();
            System.out.println("\ntestManyShapes");
            ManyShapes.testManyShapes();

}
```