

Hong Kong Polytechnic University  
EIE3320 Object-Oriented Design and Programming

Lab 1 report

Student Name:

Wong Keith (19067393D)

Kwong Wai Ki (19030979D)

## 1 Introduction

The objective of this program is to provide a user interface for calculating the area and perimeter of circles, squares and rectangles. The users are required to enter some basic information about these shapes. Canvas components are used to draw those shapes on a rectangle message box based on the input's values.

### Public Class ShapeTester: Create objects of different shapes and Display

```
*****
* Please choose one the followings: *
* Press 'c' - Circle                *
* Press 's' - Square                *
* Press 'r' - Rectangle              *
* Press 'x' - EXIT                  *
*****
```

Figure 1

A menu should be designed for the user to create geometric shapes and input the related information. This is the task of the public class ShapeTester. It is required to have input validation and display the error message if the input is invalid (as figure 1.6). If the user presses 'c', 's', or 'r', an Circle / Square / Rectangle object should be created. The user needs to input the information (radius/length/width) of the needed geometric shape. The perimeter and area will be calculated and shown in the terminal window. Also, the geometric shapes will display on a rectangle message box with a random position. It is expected that the program will not stop until the user presses 'x' as shown in figure 1.4.

Figure 1.1 is the expected output if the user presses 'c'. It is required to input the radius of a circle. The perimeter and area are displayed right after that. A circle is then shown in the rectangle message box named 'BlueJ Shapes Demo'.

Figure 1.2 is the expected output if the user presses 's'. It is required to input the length of a square. The perimeter and area are displayed right after that. A square is then shown in the rectangle message box named 'BlueJ Shapes Demo'.

Figure 1.3 is the expected output if the user presses 'r'. It is required to input the width and length of a rectangle. The perimeter and area are displayed right after that. A rectangle is then shown in the rectangle message box named 'BlueJ Shapes Demo'.

If the user continues to create different objects without closing the rectangle message box named 'BlueJ Shapes Demo', it will display all those created objects at the same time and they may overlap each other (refers to figure 1.5).

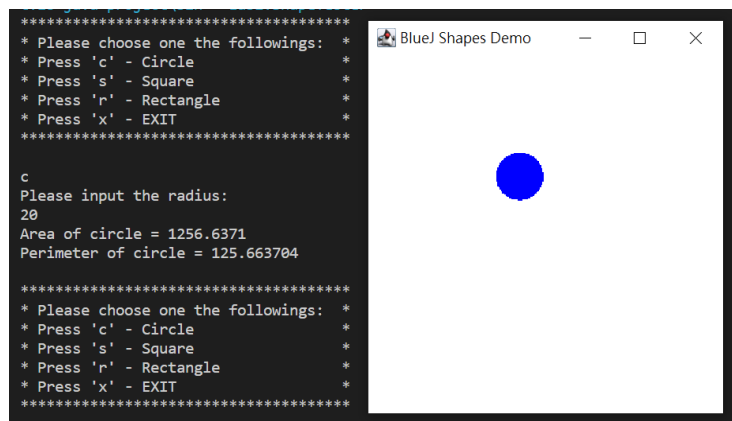


Figure 1.1

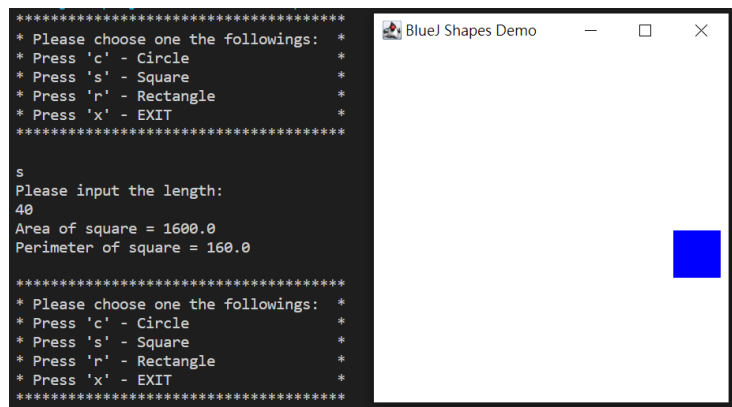


Figure 1.2

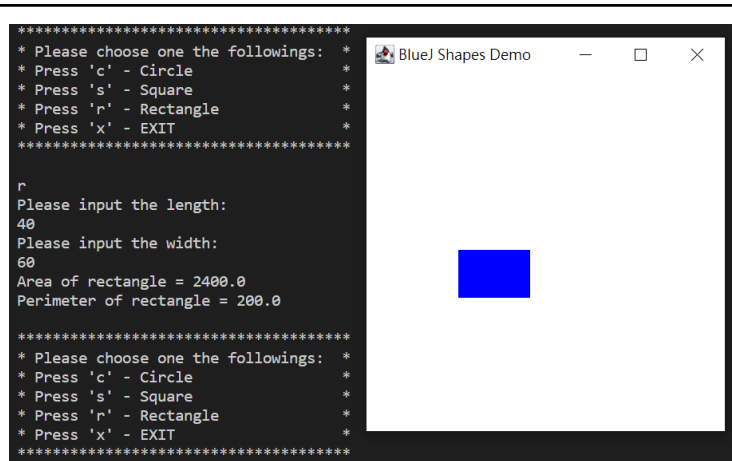


Figure 1.3

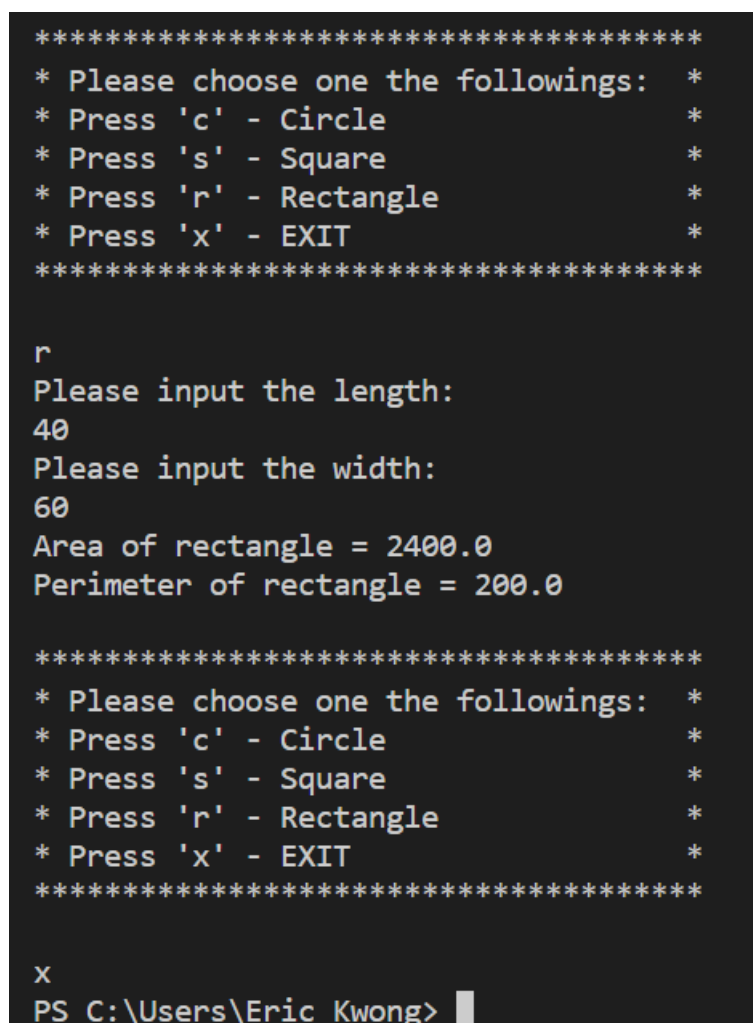


Figure 1.4

```
* Press 'r' - Rectangle
* Press 'x' - EXIT
*****

s
Please input the length:
60
Area of square = 3600.0
Perimeter of square = 240.0

*****
* Please choose one the followings: *
* Press 'c' - Circle                *
* Press 's' - Square                *
* Press 'r' - Rectangle              *
* Press 'x' - EXIT                  *
*****

r
Please input the length:
80
Please input the width:
60
Area of rectangle = 4800.0
Perimeter of rectangle = 280.0
```

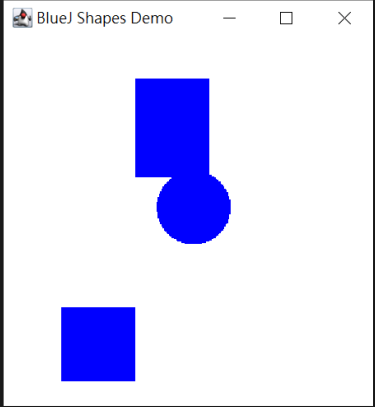


Figure 1.5

```
*****
* Please choose one the followings: *
* Press 'c' - Circle                *
* Press 's' - Square                *
* Press 'r' - Rectangle              *
* Press 'x' - EXIT                  *
*****

xqwe
Invalid command!

*****
* Please choose one the followings: *
```

Figure 1.6

### **Public Class PictureTester**

In this class, it is required to create a Picture object and allow manually adding different shapes. The class Picture() will help to store a collection of Circle, Square, and Rectangle objects. It is able to compute the areas and perimeters of all those objects, invoke the method displayShape() to display the areas and perimeters of all those objects and display the areas and perimeters of all those objects belonging to a particular class name. The testing code has been provided and the expected result is shown in figure 2.

```
Area of square = 4.0
Perimeter of square = 8.0

Area of square = 4.0
Perimeter of square = 8.0

Area of circle = 28.274334
Perimeter of circle = 18.849556

Area of circle = 50.265484
Perimeter of circle = 25.132742

Area of rectangle = 30.0
Perimeter of rectangle = 22.0

Area of rectangle = 56.0
Perimeter of rectangle = 30.0
```

Figure 2

## **2 Methodology**

### **2.1 Work division**

The works of this project are mainly the programming part and report writing. As this project is not complex, Kwong helped code and construct the program within two days with Wong's advice on program optimisation and debugging. Both of us contributed equally to this report.

### **2.2 Schedule**

Date	Work
4/10 - 5/10	Coding
5/10	Debugging and optimisation
14/10 - 16/10	Report writing

### **2.3 Program structure of the program developed**

The program should fulfill all requirements described in the lab 1 manual.

**The description of the classes:**

#### **Class Drawable (public interface Drawable)**

This class is defined as an interface class. There are four classes implemented interface Drawable, including class Shape, Square, Rectangle and Circle. A method draw() is declared.

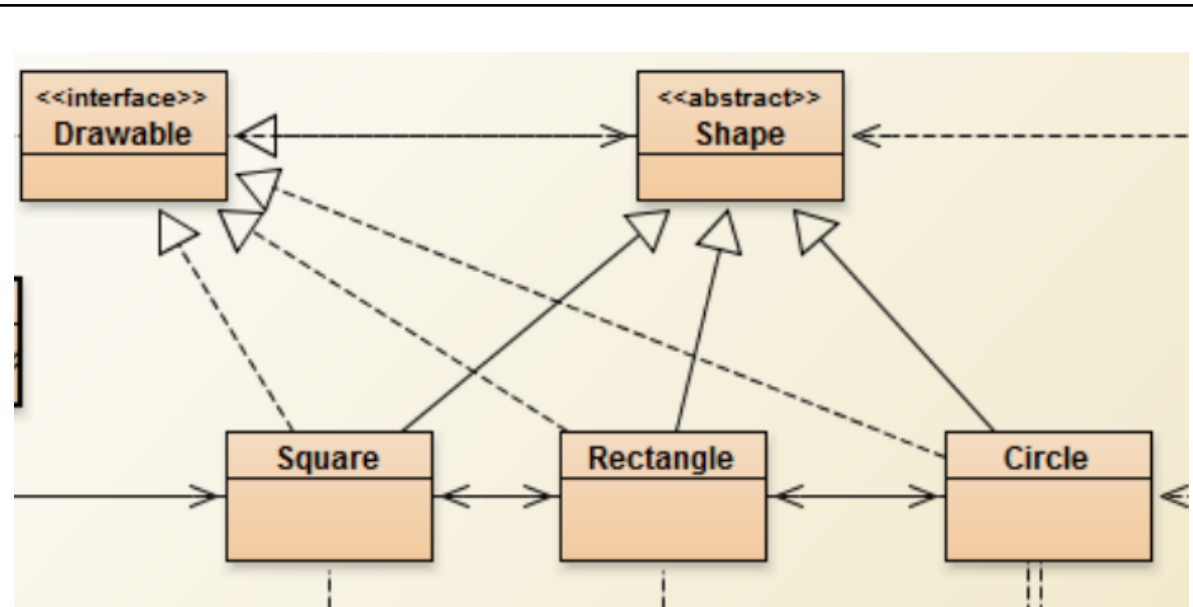


Figure 3

Method	Specification	Description
draw()	void	To plot the user-defined shape in a rectangular message box

### **Class Shape (public abstract class Shape implements Drawable)**

It is an abstract class which has declared four methods which are declared 'abstract' and two protected instance variables: area and perimeter. This class implements the interface Drawable.



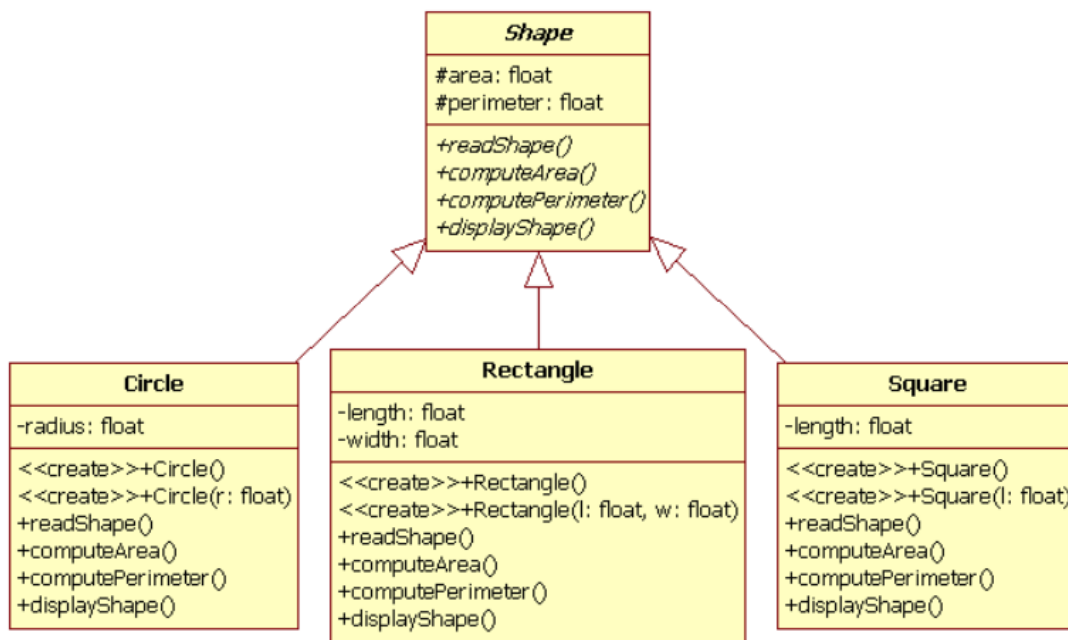


Figure 4

Abstract method	Specification	Description provided in lab manual
readShape()	public abstract void	To read the shape information from users
computeArea()	public abstract void	To compute the shape's area
computePerimeter() ( )	public abstract void	To computer the shape's perimeter
displayShape()	public abstract void	To display the area and perimeter of the shape

Instance variables	Specification	Description
area	protected float	To store the area of the shape
perimeter	protected float	To store the perimeter of the shape

Class Shape implements class Shape and has subclasses Circle, Rectangle, and Square. The abstract class cannot be instantiated, while all its subclass should act as its extension. Some states or behaviours of these subclasses are the same, all the objects (geometric shapes) can inherit from the same abstract parent object. It is required to implement all the abstract methods that are declared in the abstract class as shown in figure 4. The protected variables are accessible in the same package and subclasses.

### **class Circle/ Rectangle/ Square**

These classes are the subclass of the class Shape and implemented class Shape. Therefore, they need to implement all the abstract methods in class Shape and method draw in class Drawable. There are also some unique attributes for those classes, which will be shown in the following table:

#### **Class Circle (public class Circle extends Shape implements Drawable):**

Attribute	Specification	Description
radius	private float	To store the radius of the circle

#### **Class Rectangle (public class Rectangle extends Shape implements Drawable):**

Attribute	Specification	Description
length	private float	To store the length of the rectangle
width	private float	To store the width of the rectangle

#### **Class Square (public class Square extends Shape implements Drawable):**

Attribute	Specification	Description
length	private float	To store the length of the square

The imported packages are java.util.\* and java.awt.geom.Ellipse2D (class Circle) / java.awt.geom.Rectangle2D (class Rectangle and Square). Each of them should declare two constructors, one is for creating objects with specialised attributes (mentioned in the tables above) while the another one is for initialisation.

### The utilisation of the abstract methods in class Shape:

Method	Description
readShape()	<ul style="list-style-type: none"><li>→ Print out the input message(s)</li><li>→ Create a Scanner object and read user inputs by .nextFloat()</li></ul>
computePerimeter()	<ul style="list-style-type: none"><li>→ Calculate the perimeter of the shape and store the value (in floating point number) to the protected float instance variable 'perimeter' in class Shape</li><li>→ May use Math.PI in the calculation</li></ul>
computeArea()	<ul style="list-style-type: none"><li>→ Calculate the area of the shape and store the value (in floating point number) to the protected float instance variable 'area' in class Shape</li><li>→ May use Math.PI or Math.pow in the calculation</li></ul>
displayShape()	<ul style="list-style-type: none"><li>→ Display the information (area and perimeter) of the shape by System.out.println</li></ul>
draw()	<ul style="list-style-type: none"><li>→ Create an object of class Canvas and invoke the method Canvas.getCanvas()</li><li>→ Invoke the method Canvas.draw()</li><li>→ Use the AWT's eclipse/rectangle to draw the circle/rectangle/square on Canvas and set their position randomly by using Math.random()</li></ul>

### class Picture (public class Picture)

The class Picture contains a private ArrayList called shapes (private ArrayList<Shape> shapes) that stores a collection of Circle, Square, and Rectangle objects. There is a constructor Picture() to create the new ArrayList shapes.

Method	Specification	Type of parameter	Description
addShape	void	Shape	To add a new Circle/ Square/ Rectangle objects by .add()
computeShape	void	/	To invoke method computeArea() and computePerimeter() in class Circle/ Square/ Rectangle for all objects stored in ArrayList shapes
listAllShapeTypes	public void	/	To invoke method displayShape() in class Circle/ Square/ Rectangle for all objects stored in ArrayList shapes
listSingleShapeType	public void	String	To invoke method displayShape() in a class Circle/ Square/ Rectangle for all objects of a particular geometric shape (user request) stored in ArrayList shapes

**class PictureTester (public class PictureTester) / ShapeTester (public class ShapeTester)**

These two classes are used to test the program whether the outputs are correct or not. Class ShapeTester should include the codes about the user menu.

In the class ShapeTester, a while loop with the label 'menu' is used to keep asking the user's choice as shown in figure 1. After getting the user input, it first comes to the input-checking process. There will be an error message if the input is invalid. Then, a variable with the type Shape named ShapeObject has been declared, initially assigned a null value. Next, a switch expression is used for creating a Circle / Square / Rectangle object or exiting the program based on the user input. All methods in the corresponding class will be invoked if a new object is successfully created.

In the class PictureTester, a Picture object is created and assigned to the variable p (object reference). Then it attempts to invoke all the methods in the class Picture.

## class Canvas (public class Canvas)

Canvas is a class to allow for simple graphical drawing on a canvas. The code has been provided to us.

## 2 Program Testing

The target of this session is to test the validation of the program, and the session will be separated into two parts, Demo Testing, and Extreme Case Testing.

### Demo Testing

To make sure the program works like the instruction have been told, the session will focus on testing inputs exactly the same as that of the demos shown in the Lab1 menu.

### // PictureTester.java

```
//EIE3320 Lab 1
//Kwong Wai Ki 19030979D
//Wong Keith 19067393D
//Visual Studio Code

package Lab1;
public class PictureTester {
    Run | Debug
    public static void main(String[] args) {
        Picture p = new Picture();
        p.addShape(new Square(Length: 2));
        p.addShape(new Square(Length: 2));
        p.addShape(new Circle(Radius: 3));
        p.addShape(new Circle(Radius: 4));
        p.addShape(new Rectangle(Length: 5, Width: 6));
        p.addShape(new Rectangle(Length: 7, Width: 8));
        p.computeShape();
        p.listAllShapeTypes();
        p.listSingleShapeType(className: "Circle");
    }
}
```

```
Area of square = 4.0
Perimeter of square = 8.0
Area of square = 4.0
Perimeter of square = 8.0
Area of circle = 28.274334
Perimeter of circle = 18.849556
Area of circle = 50.265484
Perimeter of circle = 25.132742
Area of rectangle = 30.0
Perimeter of rectangle = 22.0
Area of rectangle = 56.0
Perimeter of rectangle = 30.0
Area of circle = 28.274334
Perimeter of circle = 18.849556
Area of circle = 50.265484
Perimeter of circle = 25.132742
```

## // ShapeTester.java

```
c
Please input the radius:
50
Area of circle = 7853.9814
Perimeter of circle = 314.15927

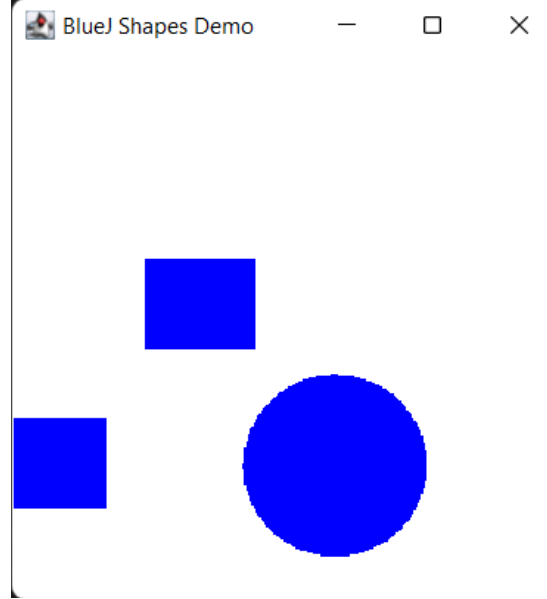
*****
* Please choose one the followings: *
* Press 'c' - Circle                *

s
Please input the length:
50
Area of square = 2500.0
Perimeter of square = 200.0

*****
* Please choose one the followings: *
* Press 'c' - Circle                *
* Press 's' - Square                 *
* Press 'r' - Rectangle              *
* Press 'x' - EXIT                   *
*****

r
Please input the length:
50
Please input the width:
60
Area of rectangle = 3000.0
Perimeter of rectangle = 220.0

*****
* Please choose one the followings: *
* Press 'c' - Circle                *
```



```
*****
* Please choose one the followings: *
* Press 'c' - Circle                *
* Press 's' - Square                 *
* Press 'r' - Rectangle              *
* Press 'x' - EXIT                   *
*****

3
Invalid command!

*****
* Please choose one the followings: *
* Press 'c' - Circle                *
* Press 's' - Square                 *
* Press 'r' - Rectangle              *
* Press 'x' - EXIT                   *
*****
```

(Invalid Command Test)

## Extreme Case Testing

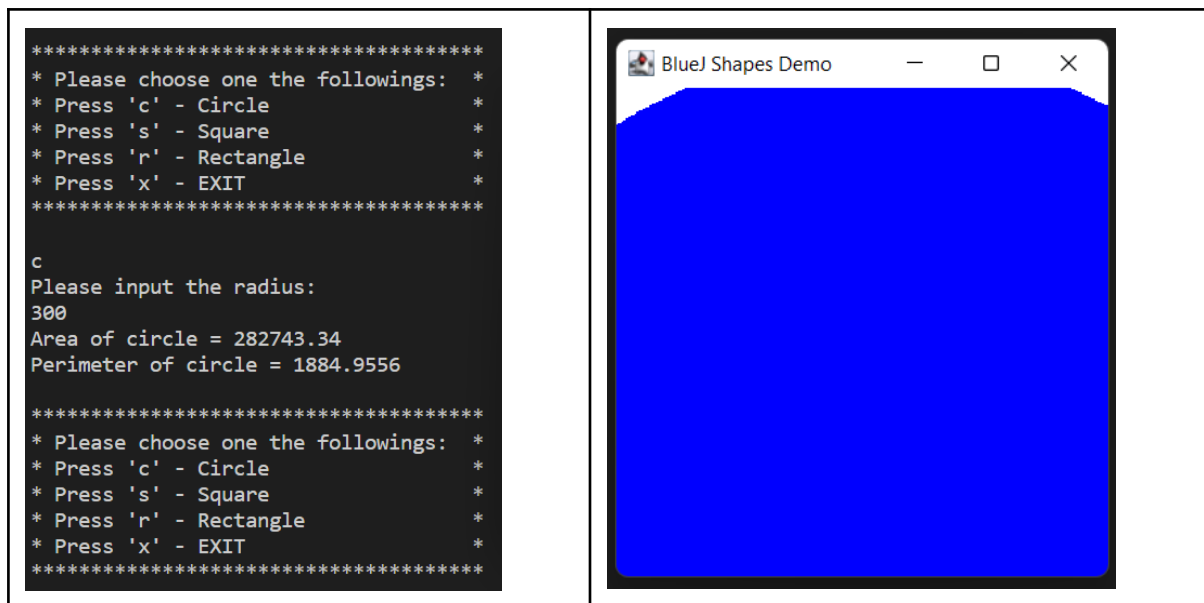
To further test the reliability of the program, some extreme inputs will be included to test the program.

It is the testing of zero input to the program. The result meets our expectations.



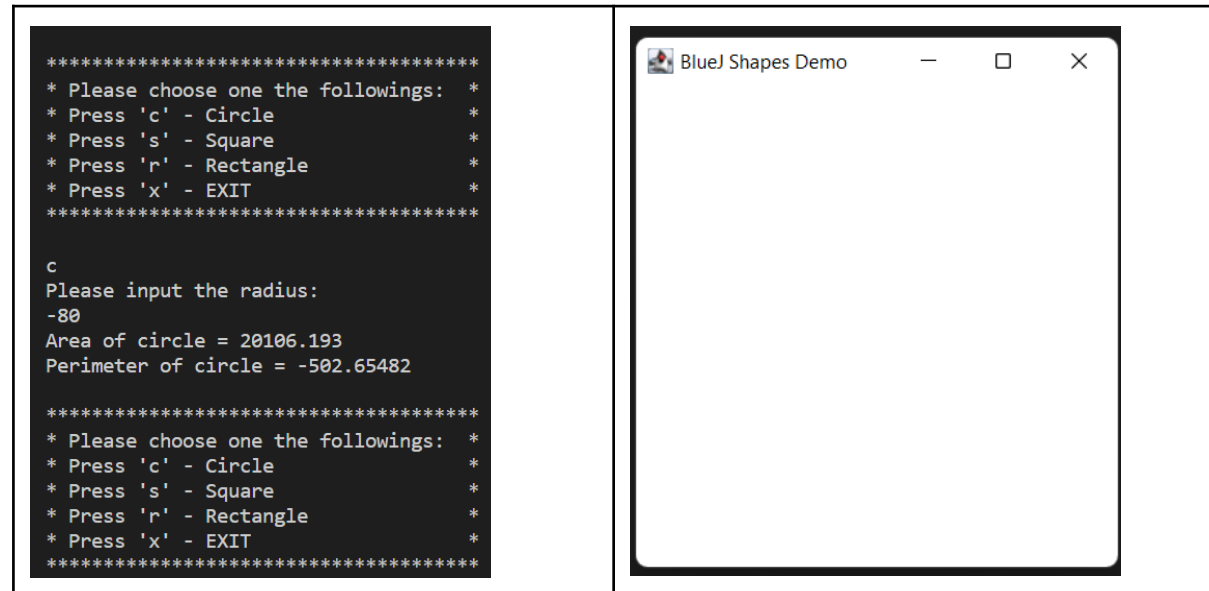
It is the testing of an unusually large input to the program, which is much larger than that of the border defined in canvas.java. And the result meets our expectations.

// ShapeTester.java



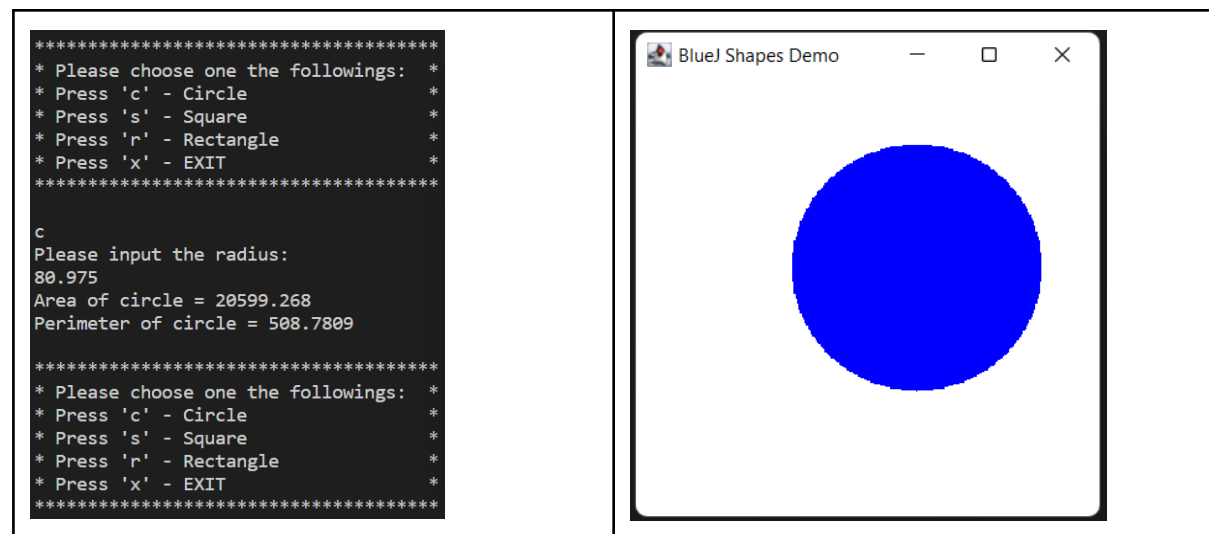
It is the testing of negative input to the program. The result does not meet our expectations, which allows the input to pass the program and output a strange result, the issue will be discussed in the Future Development session.

#### // ShapeTester.java



It is the testing of a decimal input to the program, and the result meets our expectations.

#### // ShapeTester.java



It is the testing of string input to the program. The result does not meet our expectations, which is the program has crashed, and this should not be expected from users when users using the program. The issue will be discussed in the Future Development session.



// ShapeTester.java

```
*****
* Please choose one the followings: *
* Press 'c' - Circle                *
* Press 's' - Square                *
* Press 'r' - Rectangle              *
* Press 'x' - EXIT                  *
*****

c
Please input the radius:
abc
Exception in thread "main" java.util.InputMismatchException
    at java.base/java.util.Scanner.throwFor(Scanner.java:939)
    at java.base/java.util.Scanner.next(Scanner.java:1594)
    at java.base/java.util.Scanner.nextFloat(Scanner.java:2496)
    at Lab1.Circle.readShape(Circle.java:20)
    at Lab1.ShapeTester.main(ShapeTester.java:49)
```

It is the testing of null input to the program. The result does not meet our expectations, which is the program has no response to it, the issue would not affect the users badly but should be avoided. The issue will be discussed in the Future Development session.

// ShapeTester.java

```
*****
* Please choose one the followings: *
* Press 'c' - Circle                *
* Press 's' - Square                *
* Press 'r' - Rectangle              *
* Press 'x' - EXIT                  *
*****

c
Please input the radius:
```

### **3 Conclusion**

In a nutshell, in this lab, we learned the technique of implementing inheritance and polymorphism in our program. This makes the program classes much more organized and provides convenience for further developments, including adding child classes (more kinds of shapes) and inserting this program easily into the others.

Also, the lab has introduced the convenience of working coherently with partners while using java, the workload can be easily divided by classes, and just putting all the files together in the completion.

### **4 Future Development**

As mentioned in the program testing session, the input protection for the shapes should be strengthened, and unexpected inputs (e.g., negative numbers, null values, and characters) should be guarded against entering the class, to prevent unexpected outputs and crashes.

During the development, we find that the problem of not being able to clean the prompt windows annoyed us, if we would like to further develop the program, adding the function to clean the window would be the utmost issue to be done.

Adding an exit option for users who enters the wrong shape section.

Furthermore, as a program aiming to print shapes, only printing circles, squares, and rectangles would not be too useful for most of the users, adding more shapes, or even some rare shapes (like Squire, Lemniscate, Reuleaux triangles) will give more choices for the users.

Modifying (e.g., moving and scaling, and deleting) existing shapes would also be one of the approaches for further developing the program, providing a higher degree of freedom for the users in controlling the printed items.