Project 2

ShapesGUI

Samuel Scalf

CMSC 335 · Dr. Osama Morad

University of Maryland Global Campus

Table of Contents

User’s Guide....................3
System Requirements....................3
Compiling the Program....................3
Windows 10....................3
Linux....................3
Using the Program....................4
Documentation....................4
Testing....................4
Test Case 1: Invalid Input....................5
Test Case 2: 2D Shape – Valid Input....................5
Test Case 3: Change to 3D Tab....................6
Test Case 4: Change to 2D Tab....................6
Lessons Learned....................7
Appendix A: UML Class Diagram....................8

User's Guide

System Requirements

Space: Less than 1 MB

Software: Java Development Kit (JDK) version 8 update 261

Compiling the Program

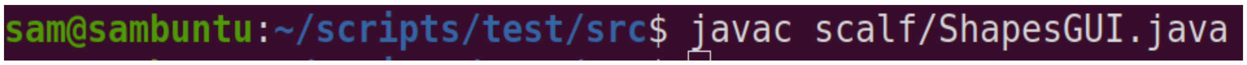
Before compiling the program, the downloaded zip file needs to be uncompressed. The means of uncompressing, or extracting, a zip file is not covered here. Open the directory containing the extracted files, if not already there. Open the "src" folder. Compilation of the program requires the JDK version 8. The procedures required for this are not covered in this guide, but can be found on Oracle's website.

Windows 10

Click File > Open Command Prompt > Open Command Prompt. Type "javac .\scalf\ShapesGUI.java" without the quotes and press the "Enter" key.

Linux

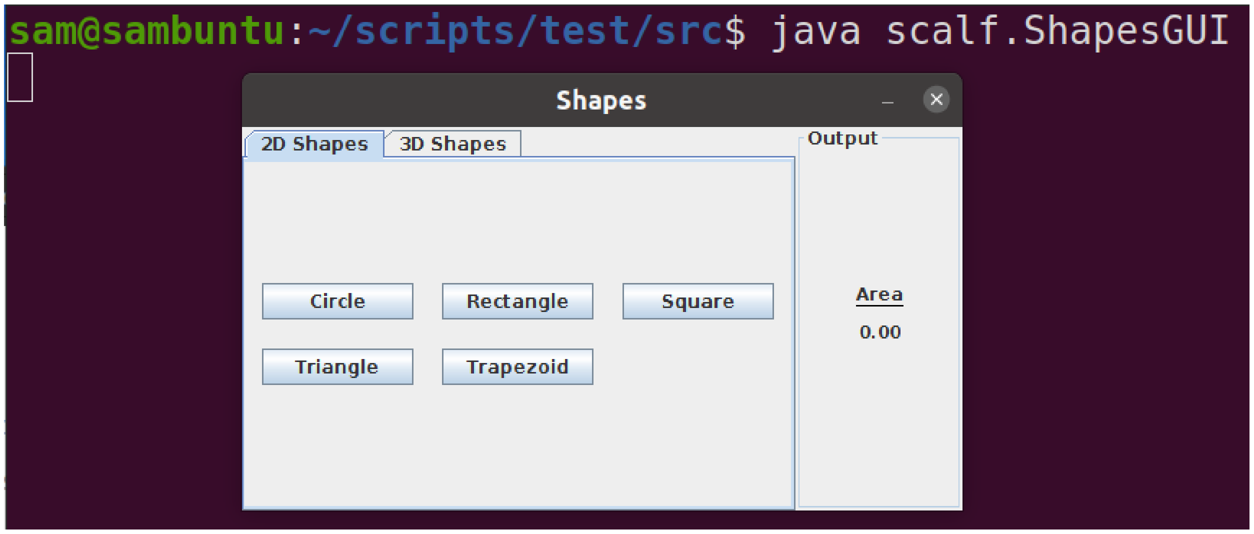
Open a terminal in the location of the files. Type "javac ./scalf/ShapesGUI.java" without the quotes and press the "Enter" key.



```
sam@sambuntu:~/scripts/test/src$ javac scalf/ShapesGUI.java
```

Using the Program

To run the program, from the command prompt or terminal (see Compilation steps), type "java scalf.ShapesGUI" without the quotes and press the "Enter" key. The program is command line-driven, so all user input will be in the command prompt or terminal. To respond to questions, type your answer and press the "Enter" key.



```
sam@sambuntu:~/scripts/test/src$ java scalf.ShapesGUI
```

Documentation

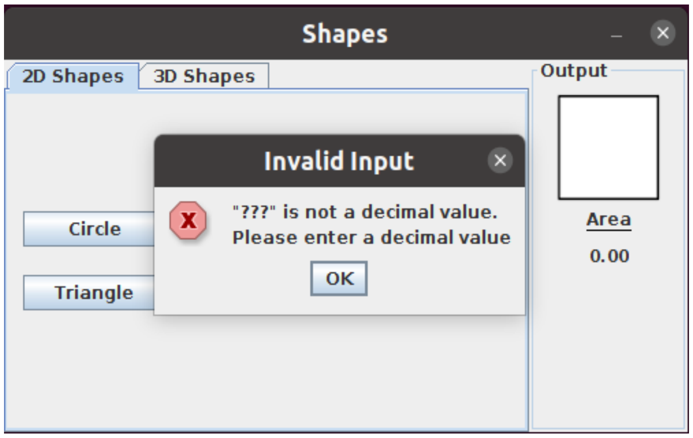
Additional information about the packages and classes can be found by opening "index.html" from the downloaded "doc" folder in a web browser. The website was created using javadoc, so it has the same format as the Oracle Java 8 API.

Testing

All buttons for shapes call the same changeIcon(String) method, and all Shapes call the same getInput(String) method. As such, I only have one test for a 2D object and one for a 3D object. The final test is to ensure the JFrame elements are all updated properly when selecting a new tab.

Test Case 1: Invalid Input

This test is for the getInput(String) method, which has a quick, internal exceptional-handling try-catch block. When a user enters a value that cannot be converted to a double, a JOptionPane should appear, explaining the exception. After acknowledging the message, the user is again presented with the input dialog. In this test, I clicked on Square, as indicated by the square that now appears in the Output panel, and entered '???'.

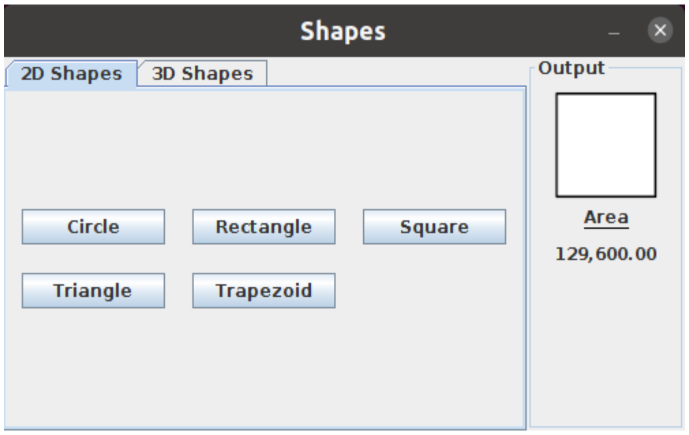


"???" is not a decimal value.
Please enter a decimal value

0.00

Test Case 2: 2D Shape – Valid Input

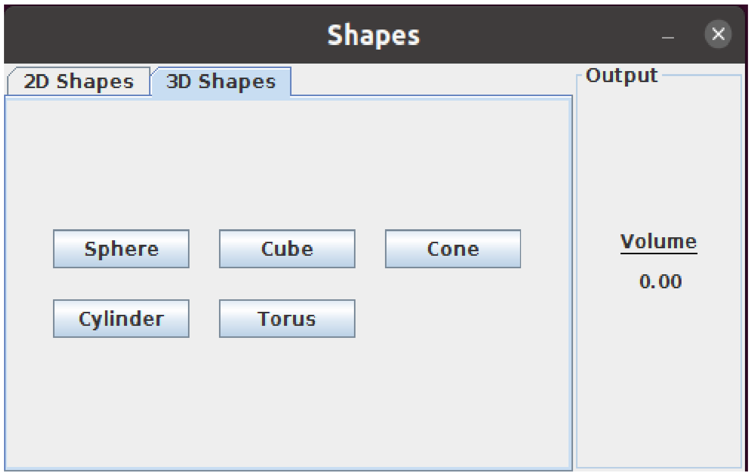
Regardless if the user had originally provided invalid input (see Test Case 1) or not, when the user provides valid input, the program should calculate the area of the two-dimensional shape rounded to two decimal places. This value can be seen in the Output Panel, below the image of the Shape. In this test, I entered 360 for the length of the side of a square.



129,600.00

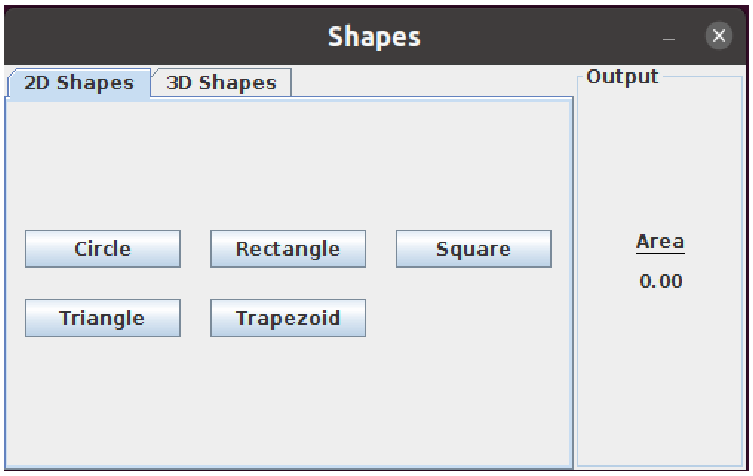
Test Case 3: Change to 3D Tab

When the tab changes, the information in the Output panel should be reset and the measurement type updated to match the selected tab. In this case, the Square should disappear, "Area" should change to "Volume," and the amount should be "0.00."



Test Case 4: Change to 2D Tab

Since there is only one Listener used for the tab changing, different actions are performed depending on the selected tab. When the tab changes, the information in the Output panel should be reset and the measurement type updated to match the selected tab. In this case, any image should disappear, "Volume" should change to "Area," and the amount should be "0.00."



Lessons Learned

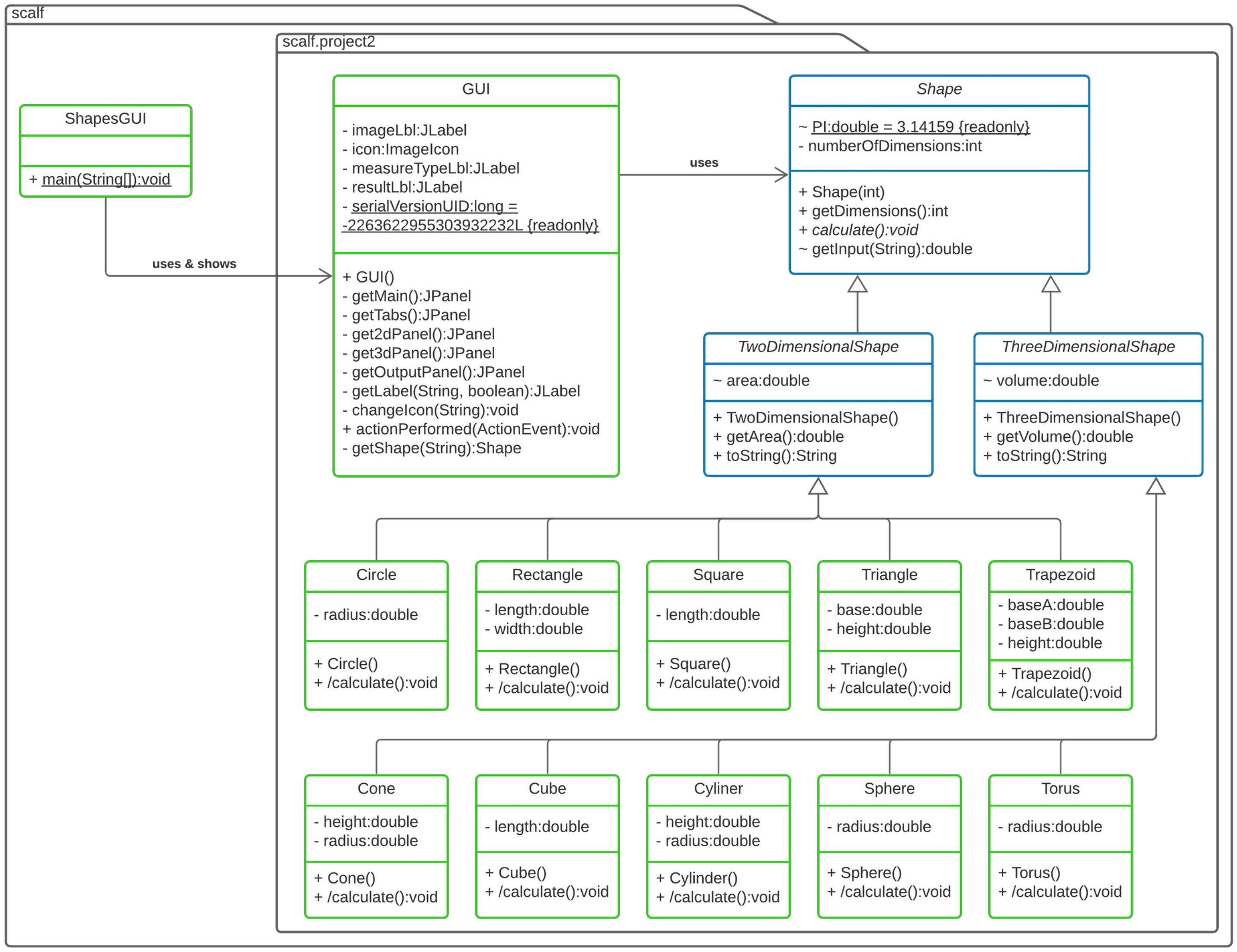
After reading the project instructions, I began working on programming my graphical user interface (GUI). I have used Javax Swing before, so I wasn’t unfamiliar with the process. Still, I wanted to push myself to try something new. This project lent itself perfectly to learning about JTabbedPanes. The requirement to have classes for two-dimensional and three-dimensional shapes gave me the idea to have a tab for each. Each tab would have JButtons for each available Shape in their respective classes.

I also needed my output panel to be updated based on the selected tab: two-dimensional shapes don’t have volume, after all. The problem was I had no idea what type of Listener I needed. A quick lookup in the Java 8 API told me I could add a ChangeListener. Since ChangeListener only has one method, I considered using a lambda expression. I decided to stick with an anonymous class, because I didn’t feel the need to separate the method for just the one operation. Once I figured that out, updating the output panel was quite simple.

Finally, the project instructions suggested displaying images for the shapes. I had done something similar before, but the new challenge here was ensuring I had good images for use. I didn’t want things moving around a lot, so I ensured all images had the same width. I set them all to 90 pixels, which looked small in GIMP, but larger in the program.

All in all, I feel I was able to meet the requirements of the project while learning some new aspects of Javax Swing. I really enjoyed this project. In fact, I think I enjoyed this one more than any of the previous projects I’ve done. I look forward to future challenges and learning opportunities.

Appendix A: UML Class Diagram


scalf
scalf.project2
ShapesGUI
+ main(String[]):void
uses & shows
GUI
- imageLbl:JLabel
- icon:ImageIcon
- measureTypeLbl:JLabel
- resultLbl:JLabel
- serialVersionUID:long = -2263622955303932232L {readonly}
+ GUI()
- getMain():JPanel
- getTabs():JPanel
- get2dPanel():JPanel
- get3dPanel():JPanel
- getOutputPanel():JPanel
- getLabel(String, boolean):JLabel
- changeIcon(String):void
+ actionPerformed(ActionEvent):void
- getShape(String):Shape
uses
Shape
~ PI:double = 3.14159 {readonly}
- numberOfDimensions:int
+ Shape(int)
+ getDimensions():int
+ calculate():void
~ getInput(String):double
TwoDimensionalShape
~ area:double
+ TwoDimensionalShape()
+ getArea():double
+ toString():String
ThreeDimensionalShape
~ volume:double
+ ThreeDimensionalShape()
+ getVolume():double
+ toString():String
Circle
- radius:double
+ Circle()
+ /calculate():void
Rectangle
- length:double
- width:double
+ Rectangle()
+ /calculate():void
Square
- length:double
+ Square()
+ /calculate():void
Triangle
- base:double
- height:double
+ Triangle()
+ /calculate():void
Trapezoid
- baseA:double
- baseB:double
- height:double
+ Trapezoid()
+ /calculate():void
Cone
- height:double
- radius:double
+ Cone()
+ /calculate():void
Cube
- length:double
+ Cube()
+ /calculate():void
Cyliner
- height:double
- radius:double
+ Cylinder()
+ /calculate():void
Sphere
- radius:double
+ Sphere()
+ /calculate():void
Torus
- radius:double
+ Torus()
+ /calculate():void