John Kucera

Prof. Didier Vergamini

CMSC 335 Object-Oriented and Concurrent Programming

15 November 2020

Week 4: Project 2 Documentation & Solution Description

(Lessons Learned on page 44)

**Class Hierarchy for Shapes + GUI**

**Assumptions + Design decisions:**

* The core menu feature in my program is the combo box that allows the user to select a shape. The frame has a CardLayout, which allows the user to switch between the shape menus (cards). Since each child class of Shape requires their own card, they have each been given their own createMenu() method that fills in the panel with GUI widgets (buttons, text fields, labels). There, each class also has their own listeners and drawing methods. Although this causes code duplication, I could not find a convenient way to share common listeners/variables/methods among the classes due to scope issues (for this setup in particular). More on my thought process concerning this in Lessons Learned.
* **Negative integer** or **zero** values are NOT ALLOWED for dimension input. In this program, every shape must have positive dimensions. Example: Inputting a height of 0 or -6 for a Cylinder will cause a JOptionPane to be displayed that notifies the user to try again.
* **Only Integer input (not Double) is allowed for this program.** This is to account for the 2d-shape drawing using java.awt.Graphics, whose draw methods only allow integer input. The output properties, however, are still unrounded Double to ensure accurate information.
* I removed numberOfDimensions variable as it was not required for this project. It had no substantial use here.
* White space (before and after) is allowed for all input.
* Triangles with non-existent area or 0 area are not allowed. Example: Triangle with sides of 2, 4, and 6 has an area of 0. Triangle with sides of 1, 3, and 50 cannot exist. When this occurs, the user is notified to try again with proper dimensions.
* A Spherical Cap’s Height cannot be more than the Sphere’s Radius, as that would technically turn the Cap into a Hemisphere. This is not allowed in this program. Example: Inputting a Sphere Radius of 5 and a Spherical Cap Height of 6 will result in notifying the user to try again with proper dimensions.
* **The 2D graphics drawings ARE displayed to scale**, equating the input as PIXELS. While I tried to center and create fitting frames for each shape, it is possible for input to be so large that it does not fit the user’s monitor.
* **The 3D graphics are NOT displayed to scale.** If I try changing the sizes of the images according to user input, the image quality gets ruined. All .jpg files that get loaded are all set to 250x250 pixels to maintain consistency, no matter what the input is.
* Drawing the triangle given 3 side lengths is complicated, causing the center-point of each triangle (in relation to the frame) to differ depending on which side is longer. I could not find a way to center the triangle drawings, but I at least got them all to fit in the frames.

**UML Class diagram:** See Fig. 1.

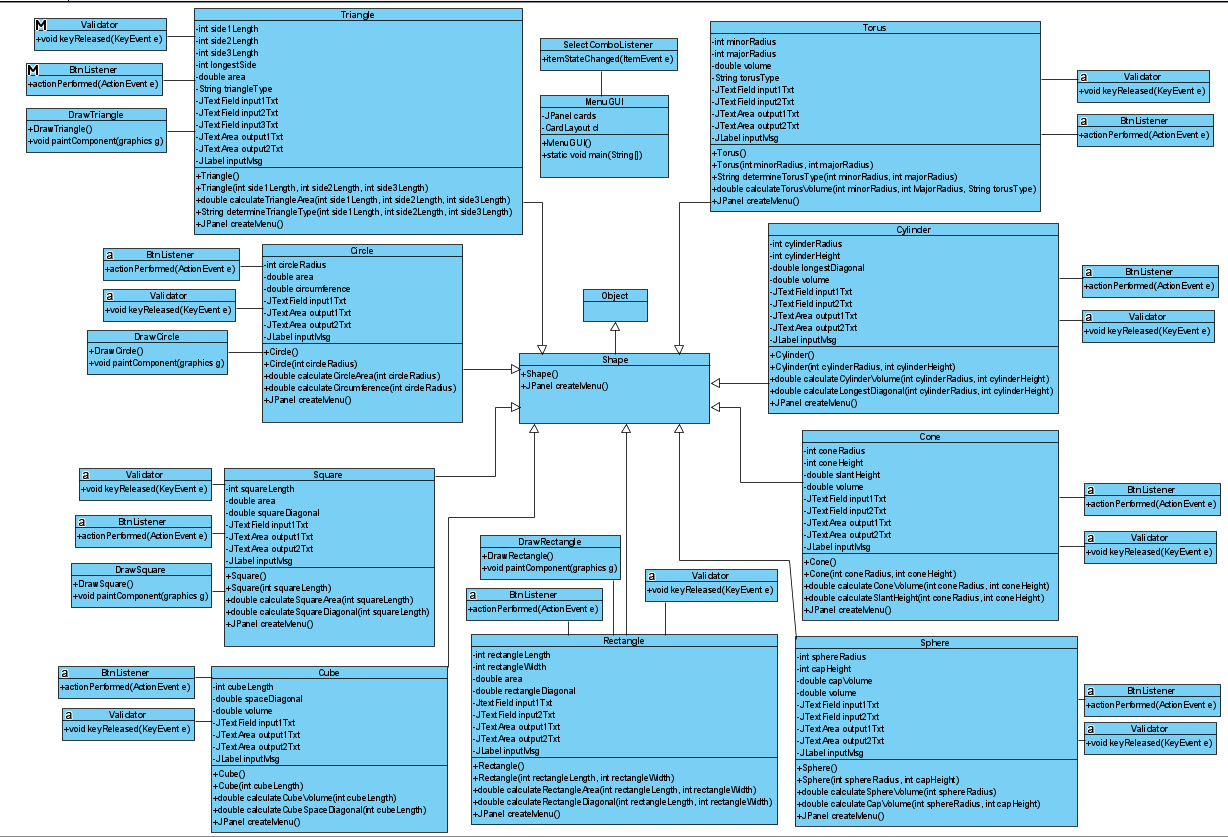


Figure 1: Project 2 UML Class diagram. (Kucera, 2020)

**User’s Guide: How to set up and run this application**

1. With a software tool that can manage .zip and .rar files such as WinRAR, unzip my submitted zip file **JohnKuceraProject2.zip**. You can do this easily by right-clicking **JohnKuceraProject2.zip** and clicking **Extract Files**, then click **OK** (See Fig. 2). This gives you a readable folder with the application files inside (See Fig. 3).

Figure 2: Unzipping a .zip file. (Kucera, 2020)

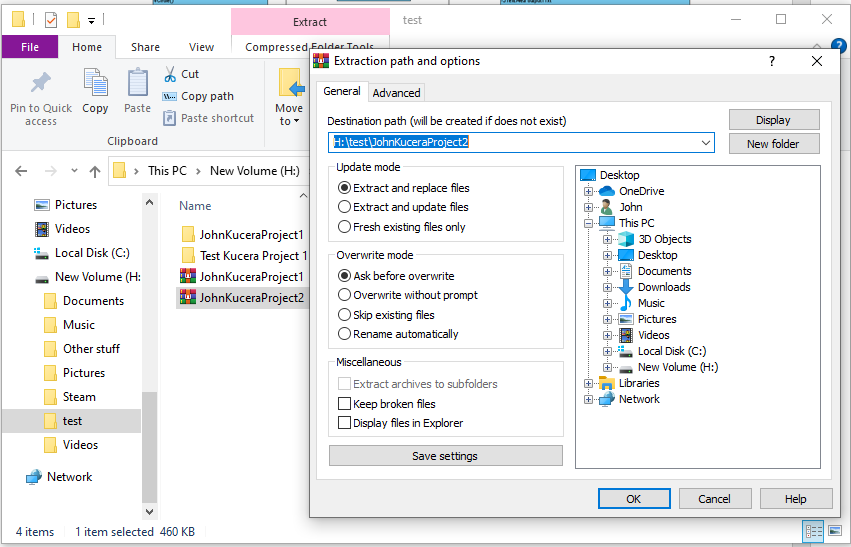
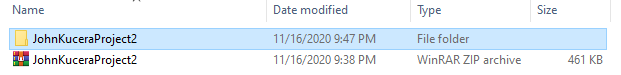


Figure 3: .zip file has been unzipped. (Kucera, 2020)



1. Open your IDE and create a new project (any IDE will work). Select Java Application (See Fig. 4 for example in Netbeans IDE).

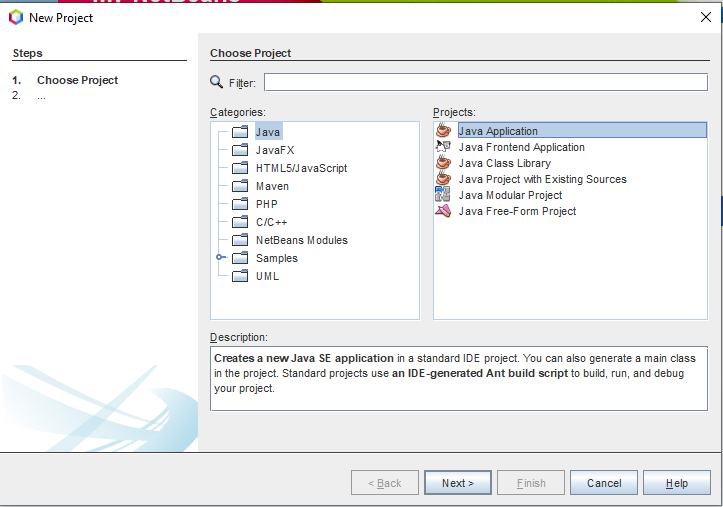
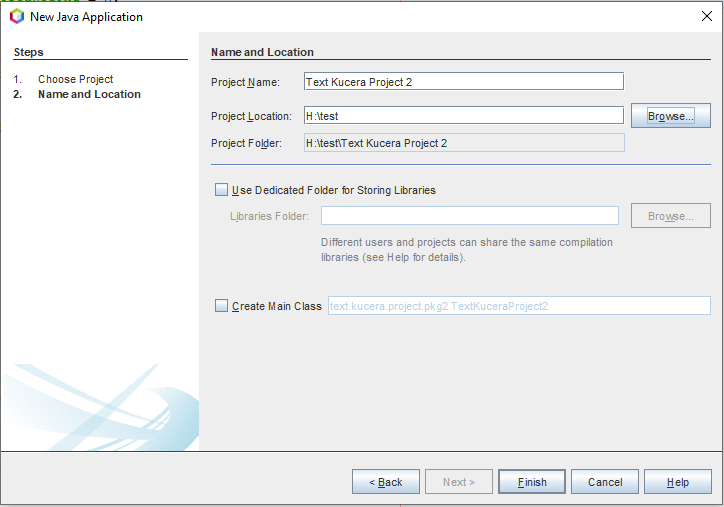


Figure 4: Creating Java Application Project in Netbeans. (Kucera, 2020)

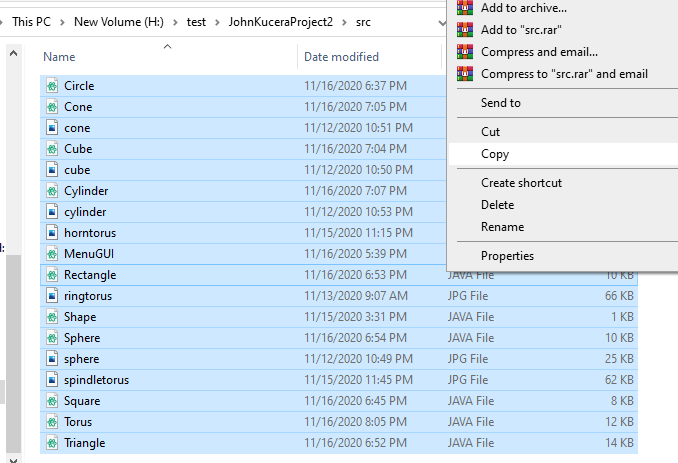
1. (See Fig. 5) Name the project “**Test Kucera Project 2**”. Identify the project location (which is where the application files will be saved). **DO NOT allow the IDE to automatically create main class**.

Figure 5: Creating Test Kucera Project 2. (Kucera, 2020)



1. In your File Explorer, go to the unzipped **JohnKuceraProject2 > src** folder and copy the 18 .java and .jpg source files: **Circle.java, Cone.java, Cube.java, Cylinder.java, MenuGUI.java, Rectangle.java, Shape.java, Sphere.java, Square.java, Torus.java, Triangle.java, cone.jpg, cube.jpg, cylinder.jpg, horntorus.jpg, ringtorus.jpg, sphere.jpg, spindletorus.jpg** (See Fig. 6).

Figure 6: Copying Application Source Files. (Kucera, 2020)



1. Paste them into the **src** folder in YOUR project folder: **Test Kucera Project 2 > src** (See Fig. 7). They will appear in your IDE under the new project’s Source Packages (See Fig. 8).

Figure 7: Pasting Application Source Files in Test Project. (Kucera, 2020)

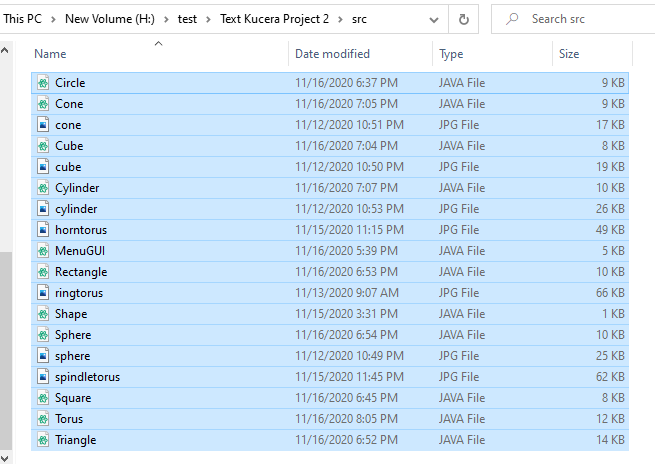
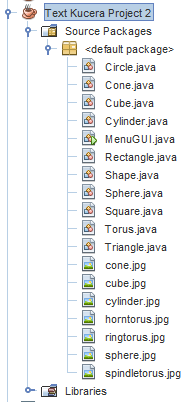
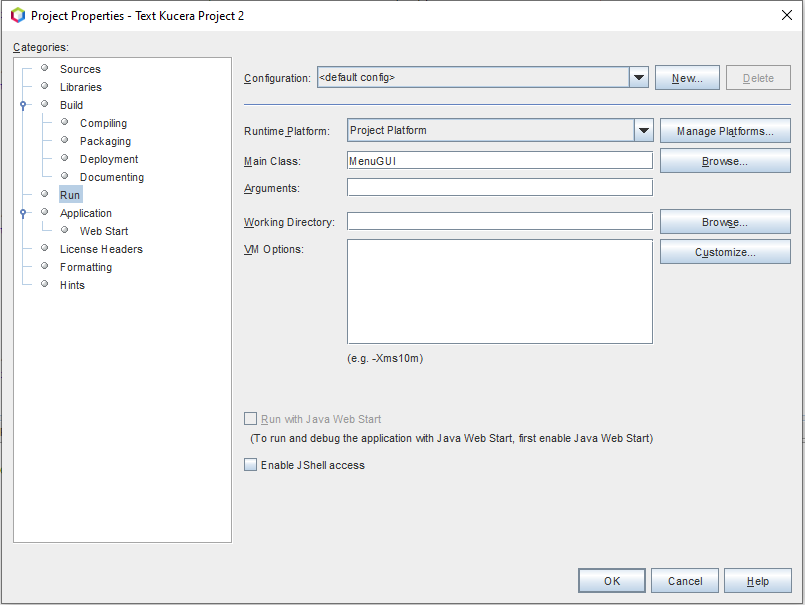


Figure 8: Pasted Source Files appear in IDE. (Kucera, 2020)



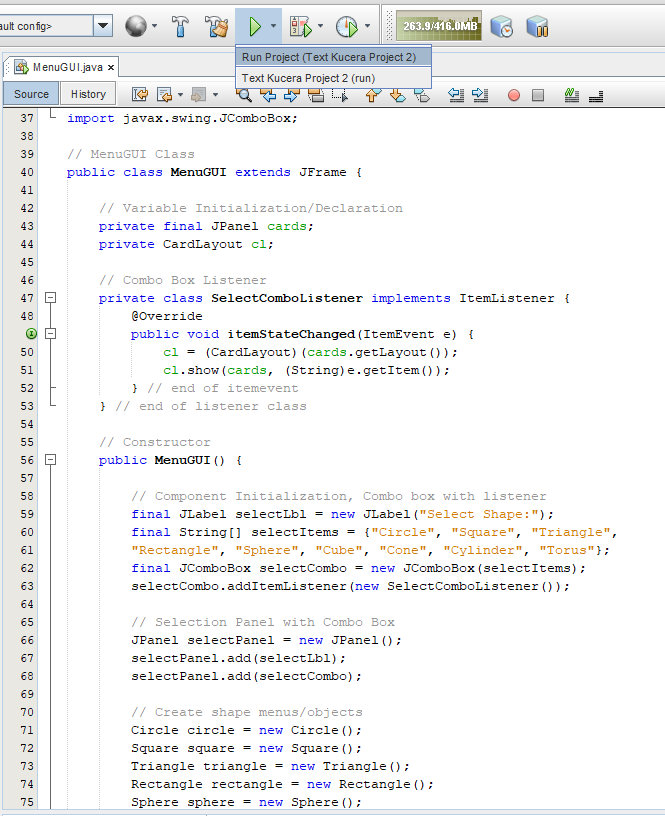
1. In the IDE, open the **Project Properties** of **Test Kucera Project 2**. You can do this by right-clicking **Test Kucera Project 2** and clicking **Properties**. In the “Run” options, change the Main Class to “**MenuGUI**”, since the Main Method for this application is in **MenuGUI.java** (See Fig. 9).

Figure 9: Project Properties > Run, type in Main Class. (Kucera, 2020)



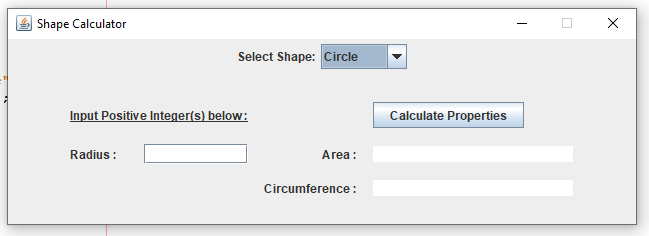
1. You can click on each source file to open them and view the Java code in each. To run the application, make sure **MenuGUI.java** is open and click **Run Project** (See Fig. 10).

Figure 10: Open MenuGUI.java, click Run Project. (Kucera, 2020)



1. The program will be displayed in the form of a GUI JFrame. You can interact with this interface, such as typing input into the text fields and clicking the button. (See Fig. 11).

Figure 11: Run project, menu GUI displayed. (Kucera, 2020)



1. User input is allowed in the GUI as the application runs. Try testing the application by performing the Test Cases shown below.

**Test Cases (Summary)**

Test cases are numbered and are organized according to the shape output they test.

|  |  |
| --- | --- |
| **Aspect Tested** | **Test Case #** |
| Program ends upon clicking Close button | 1 |
| Invalid input for dimensions | 2 to 8 |
| Circle | 9, 10 |
| Square | 11, 12 |
| Triangle | 13 to 17 |
| Rectangle | 18, 19 |
| Sphere | 20 to 22 |
| Cube | 23, 24 |
| Cone | 25, 26 |
| Cylinder | 27, 28 |
| Torus | 29 to 31 |
| Results: **31 out of 31 Test Cases PASSED. The program is successful.** | |

**Test Cases**

1. **Aspect Tested:** Program exits upon clicking the close button

**Input:** Click the close button in the top-right corner of the JFrame

**Expected Output:** Program exits

**Actual Output**: See Fig. 12.

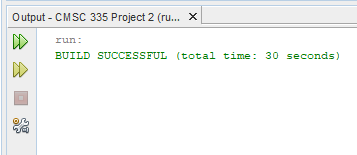


Figure 12: Test Case 1 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Menu items change according to the Shape chosen in Combo box

**Input:** In the combo box, switch from Circle to Square

**Expected Output:** Menu Panel switches from Circle items to Square items

**Actual Output**: See Fig. 13.

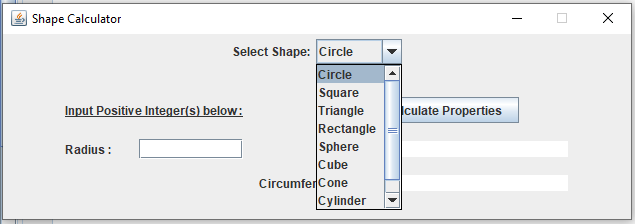
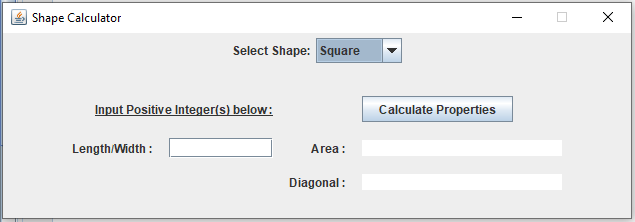


Figure 13: Test Case 2 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input 0 as a dimension (not allowed)

**Input:** Circle Radius: 0, click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Invalid”, Dialog displayed that reads “Please enter a positive integer(s).”

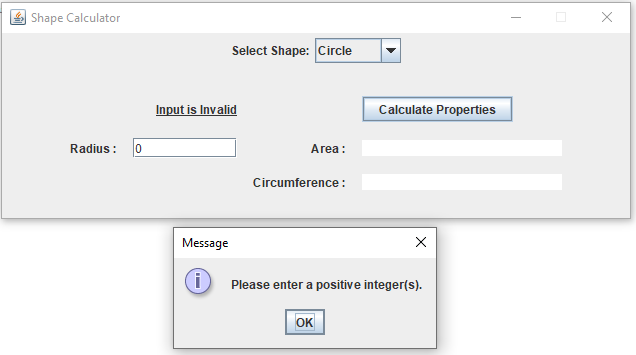
**Actual Output**: See Fig. 14. 

Figure 14: Test Case 3 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input negative integer as a dimension (not allowed)

**Input:** Triangle: Side 1 Length: 45. Side 2 Length: -30. Side 3 Length: 50.

Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Invalid”, Dialog displayed that reads “Please enter a positive integer(s).”

**Actual Output**: See Fig. 15.

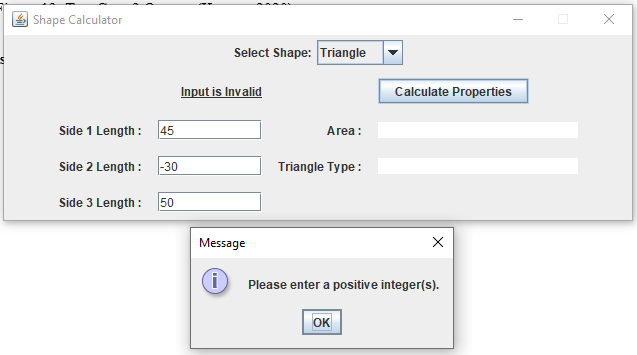


Figure 15: Test Case 4 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input double as a dimension (not allowed)

**Input:** Rectangle: Length: 45. Width: 43.1. Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Invalid”, Dialog displayed that reads “Please enter a positive integer(s).”

**Actual Output**: See Fig. 16.

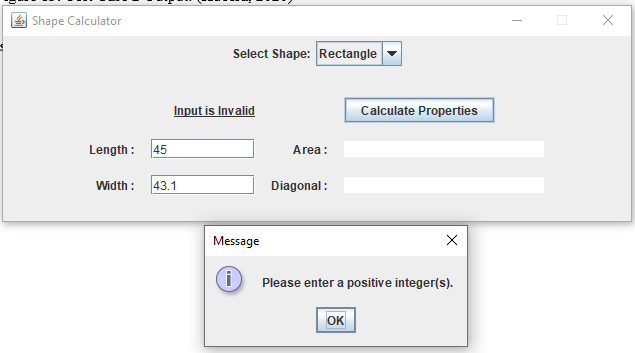


Figure 16: Test Case 5 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** No input for a dimension (not allowed)

**Input:** Cylinder: Radius: [empty]. Height: 56. Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input Positive Integer(s) below:”, Dialog displayed that reads “Please enter a positive integer(s).”

**Actual Output**: See Fig. 17.

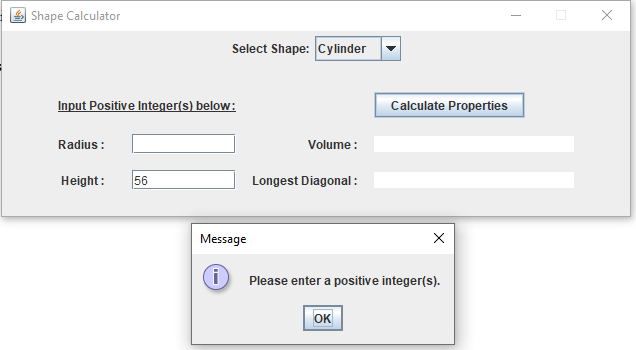


Figure 17: Test Case 6 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input character for a dimension (not allowed)

**Input:** Sphere: Radius: 7. Spherical Cap Height: g. Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Invalid”, Dialog displayed that reads “Please enter a positive integer(s).”

**Actual Output**: See Fig. 18.

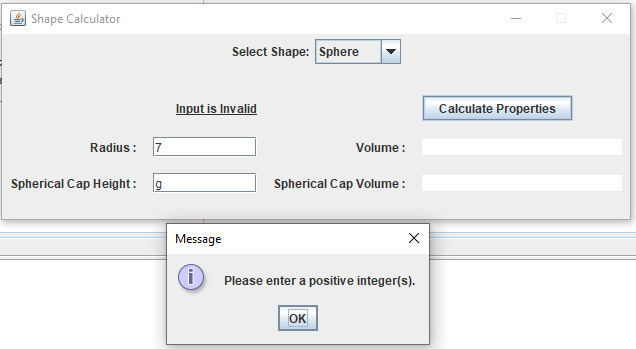


Figure 18: Test Case 7 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input symbol for a dimension (not allowed)

**Input:** Torus: Minor Radius: &. Major Radius: 2. Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Invalid”, Dialog displayed that reads “Please enter a positive integer(s).”

**Actual Output**: See Fig. 19.

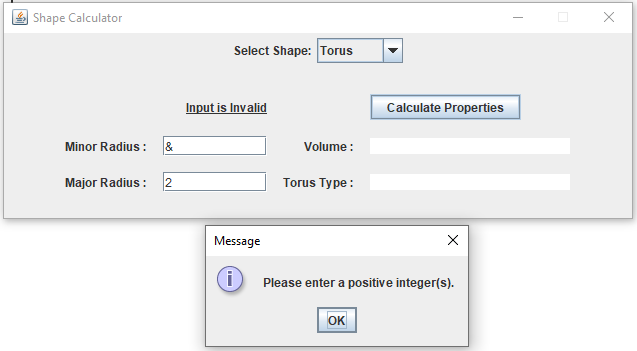


Figure 19: Test Case 8 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for a small Circle

**Input:** Circle: Radius: 80. Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Area: 20106.192982974677. Circumference: 502.6548245743669.

Frame displayed containing accurately drawn Circle with a radius of 80 pixels.

**Actual Output**: See Fig. 20.

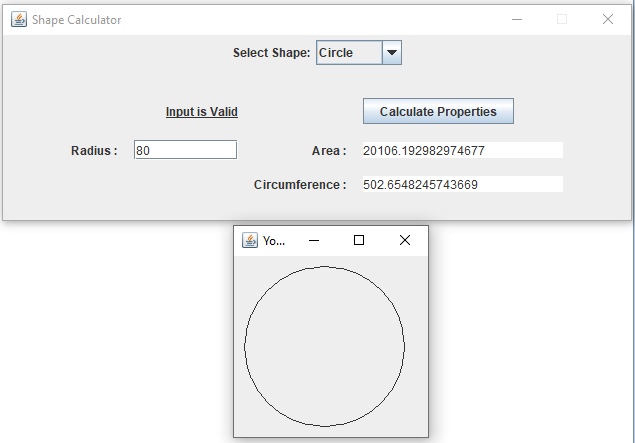


Figure 20: Test Case 9 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for a big Circle

**Input:** Circle: Radius: 288. Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Area: 260576.2610593518. Circumference: 1809.5573684677208.

Frame displayed containing accurately drawn Circle with a radius of 288 pixels.

**Actual Output**: See Fig. 21.

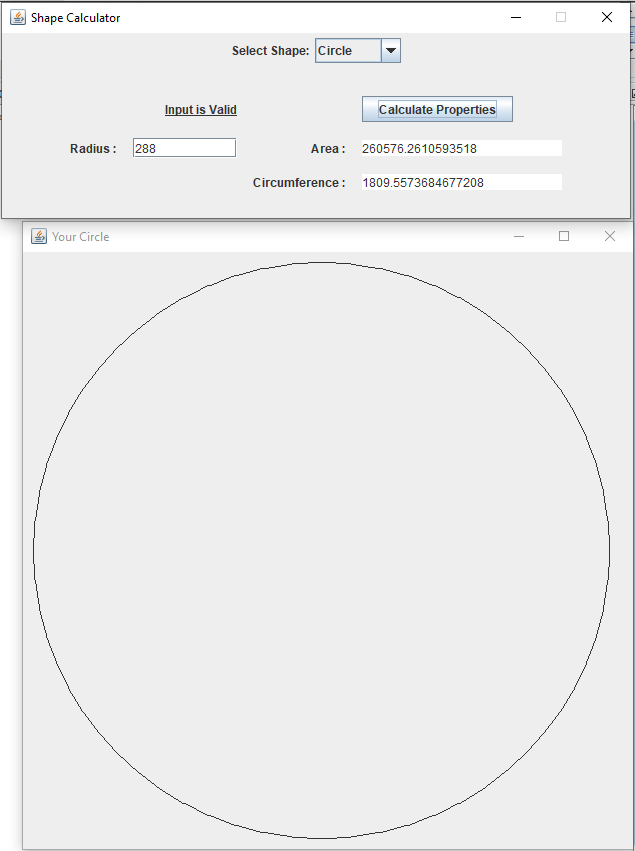


Figure 21: Test Case 10 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for a small Square

**Input:** Square: Length: 122. Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Area: 14884.0. Diagonal: 172.5340546095176.

Frame displayed containing accurately drawn Square with a length of 122 pixels.

**Actual Output**: See Fig. 22.

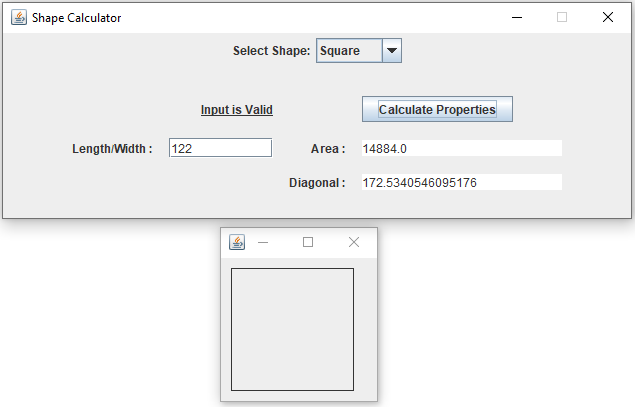


Figure 22: Test Case 11 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for a big Square

**Input:** Square: Length: 371. Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Area: 137641.0. Diagonal: 524.6732316404183.

Frame displayed containing accurately drawn Square with a length of 371 pixels.

**Actual Output**: See Fig. 23.

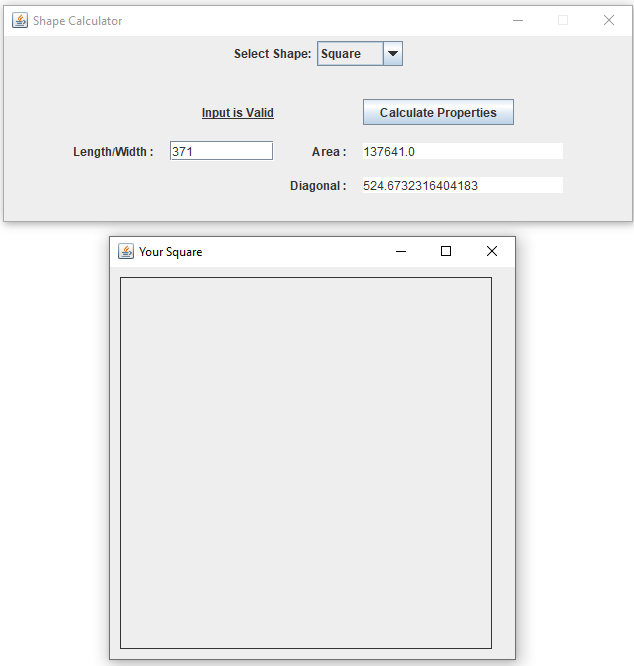


Figure 23: Test Case 12 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input dimensions for a triangle whose area cannot exist (not allowed)

**Input:** Triangle: Side 1 Length: 6. Side 2 Length: 2000. Side 3 Length: 80.

Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Dialog displayed that reads “This Triangle cannot exist with the dimensions provided. Please try again.”

**Actual Output**: See Fig. 24.

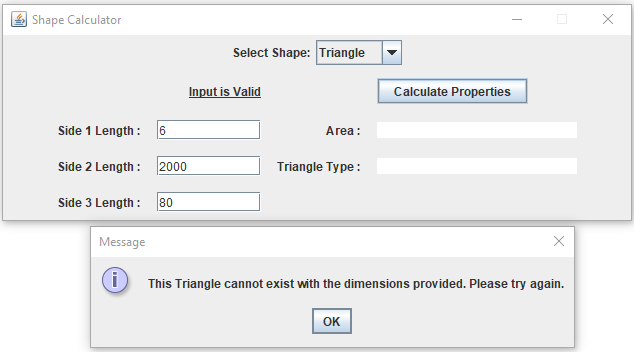


Figure 24: Test Case 13 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input dimensions for a triangle whose area is 0 (not allowed)

**Input:** Triangle: Side 1 Length: 60. Side 2 Length: 70. Side 3 Length: 130.

Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Dialog displayed that reads “This Triangle cannot exist with the dimensions provided. Please try again.”

**Actual Output**: See Fig. 25.

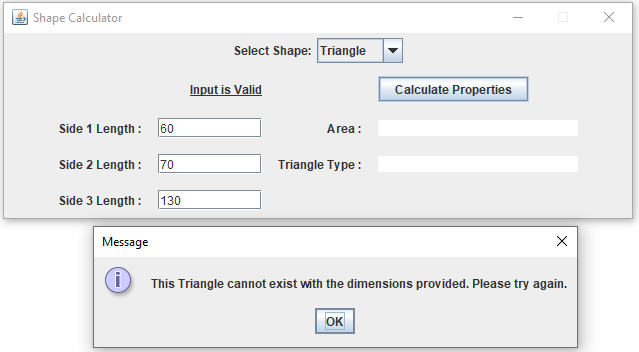


Figure 25: Test Case 14 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for an equilateral triangle

**Input:** Triangle: Side 1 Length: 546. Side 2 Length: 546. Side 3 Length: 546.

Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Area: 129088.01463730086. Triangle Type: Equilateral.

Frame displayed containing accurately drawn Triangle with sides of 546 pixels each.

**Actual Output**: See Fig. 26.

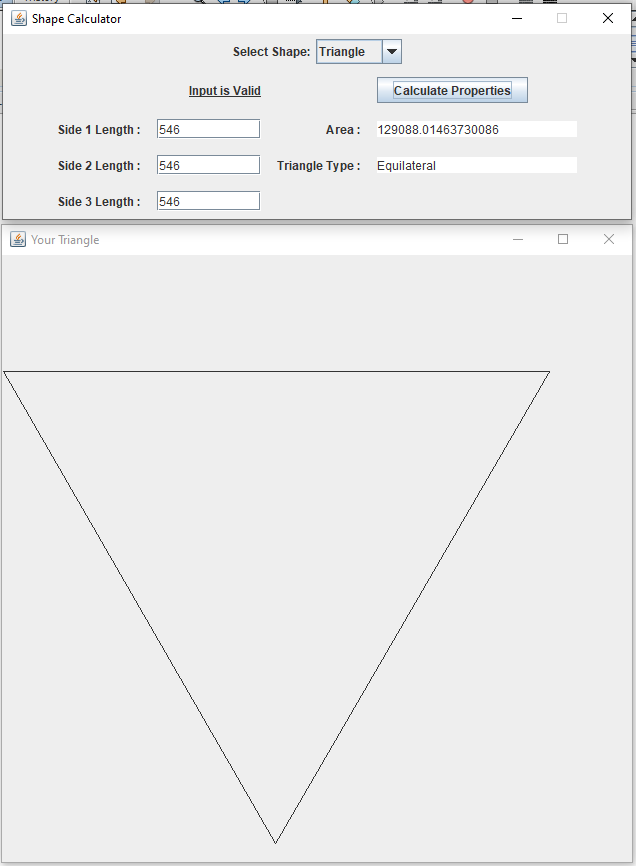


Figure 26: Test Case 15 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for an isosceles triangle

**Input:** Triangle: Side 1 Length: 49. Side 2 Length: 96. Side 3 Length: 49.

Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Area: 472.74517448621305. Triangle Type: Isosceles.

Frame displayed containing accurately drawn Triangle with sides of 49, 96, 49 pixels.

**Actual Output**: See Fig. 27.

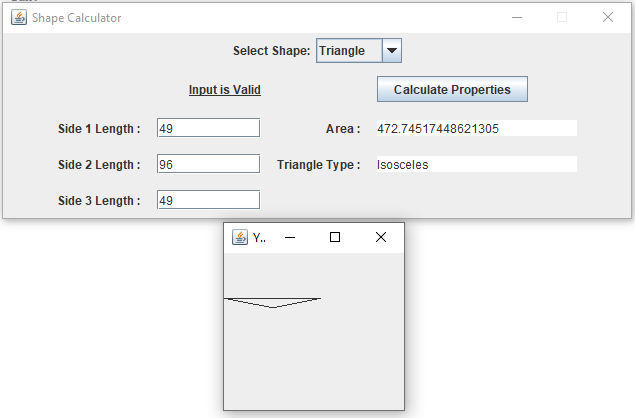


Figure 27: Test Case 16 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for a scalene triangle

**Input:** Triangle: Side 1 Length: 333. Side 2 Length: 91. Side 3 Length: 267.

Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Area: 9288.76026106283. Triangle Type: Scalene.

Frame displayed containing accurately drawn Triangle with sides of 333, 91, 267 pixels.

**Actual Output**: See Fig. 28.

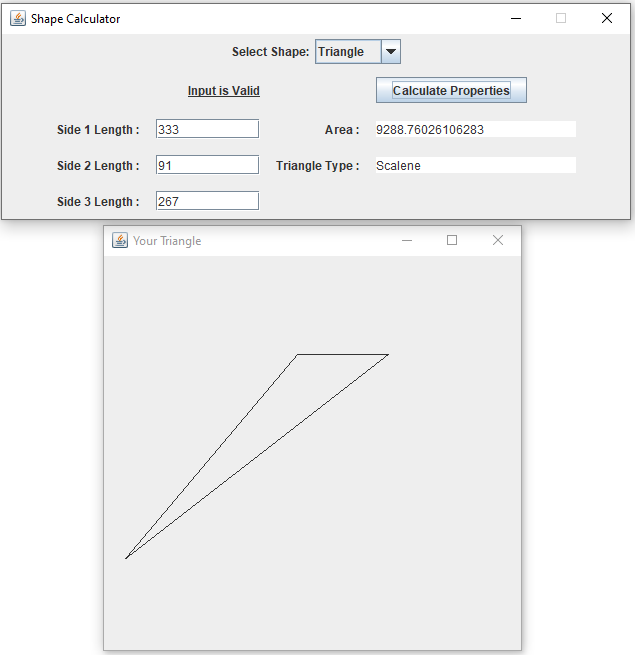


Figure 28: Test Case 17 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for a small rectangle

**Input:** Rectangle: Length: 8. Width: 14.

Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Area: 112.0. Diagonal: 16.1245154965971.

Frame displayed containing accurately drawn Rectangle with sides of 8x14 pixels.

**Actual Output**: See Fig. 29.

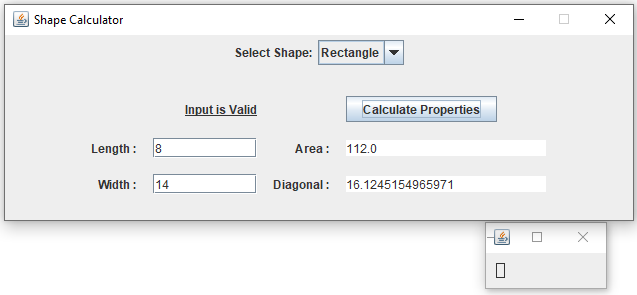


Figure 29: Test Case 18 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for a big rectangle

**Input:** Rectangle: Length: 661. Width: 455.

Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Area: 300755.0. Diagonal: 802.4624601811602.

Frame displayed containing accurately drawn Rectangle with sides of 661x455 pixels.

**Actual Output**: See Fig. 30.

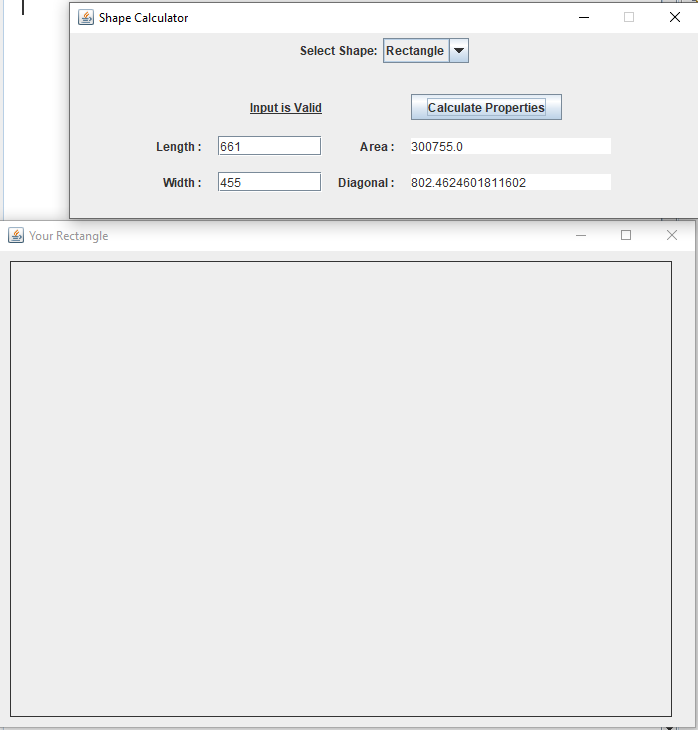


Figure 30: Test Case 19 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input dimensions for a sphere whose spherical cap height is greater than its radius (not allowed)

**Input:** Sphere: Radius: 45. Spherical Cap Height: 47.

Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Dialog displayed that reads “The Spherical Cap cannot have a Height greater than or equal to the Sphere’s Radius. Please try again.”

**Actual Output**: See Fig. 31.

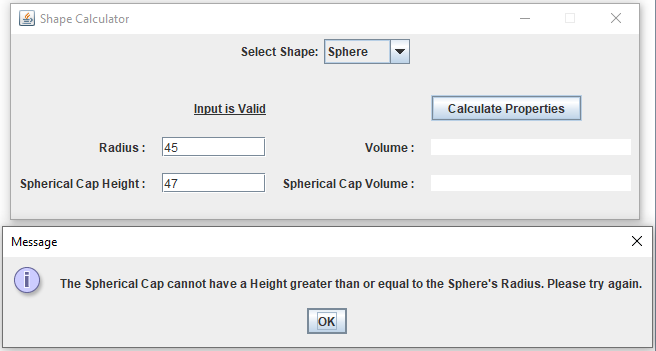


Figure 31: Test Case 20 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for a small sphere

**Input:** Sphere: Radius: 45. Spherical Cap Height: 13.

Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Volume: 381703.5074111598. Spherical Cap Volume: 21591.11911057145.

Frame displayed containing sphere.jpg.

**Actual Output**: See Fig. 32.

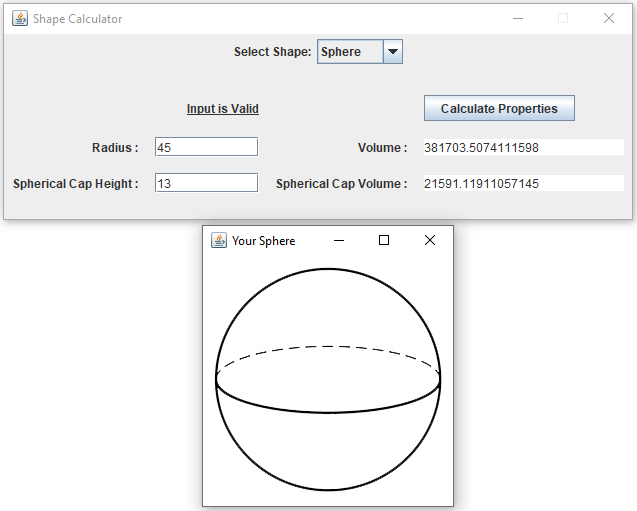


Figure 32: Test Case 21 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for a big sphere

**Input:** Sphere: Radius: 1445. Spherical Cap Height: 619.

Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Volume: 1.2638401574319454E10. Spherical Cap Volume: 1.491027389556571E9.

Frame displayed containing sphere.jpg.

**Actual Output**: See Fig. 33.

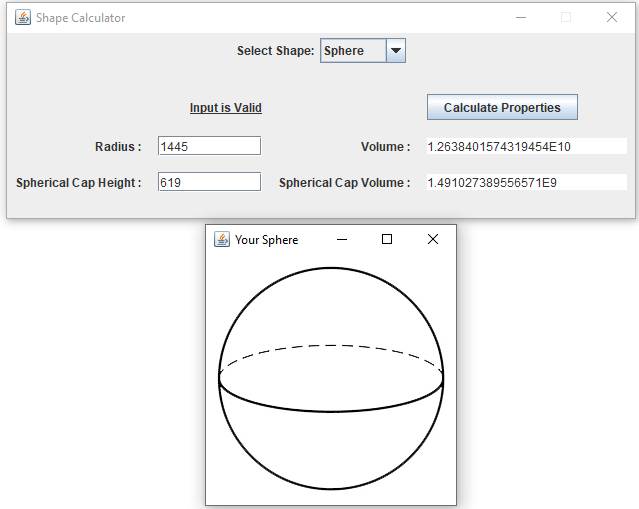


Figure 33: Test Case 22 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for a small cube

**Input:** Cube: Length/Width/Height: 7. Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Volume: 343.0. Space Diagonal: 12.12435565298214.

Frame displayed containing cube.jpg.

**Actual Output**: See Fig. 34.

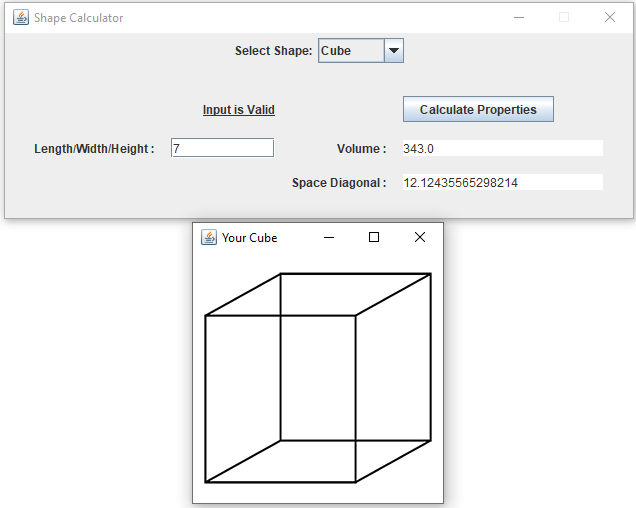


Figure 34: Test Case 23 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for a big cube

**Input:** Cube: Length/Width/Height: 770. Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Volume: 4.56533E8. Space Diagonal: 1333.6791218280355.

Frame displayed containing cube.jpg.

**Actual Output**: See Fig. 35.

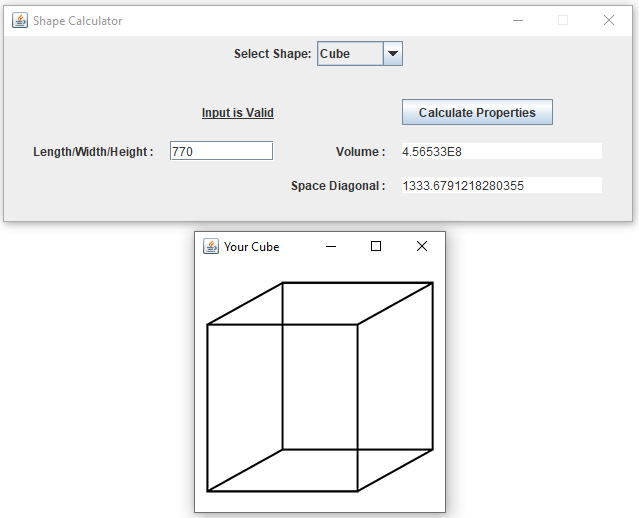


Figure 35: Test Case 24 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for a small cone

**Input:** Cone: Radius: 10. Height: 14. Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Volume: 1466.076571675237. Slant Height: 17.204650534085253.

Frame displayed containing cone.jpg.

**Actual Output**: See Fig. 36.

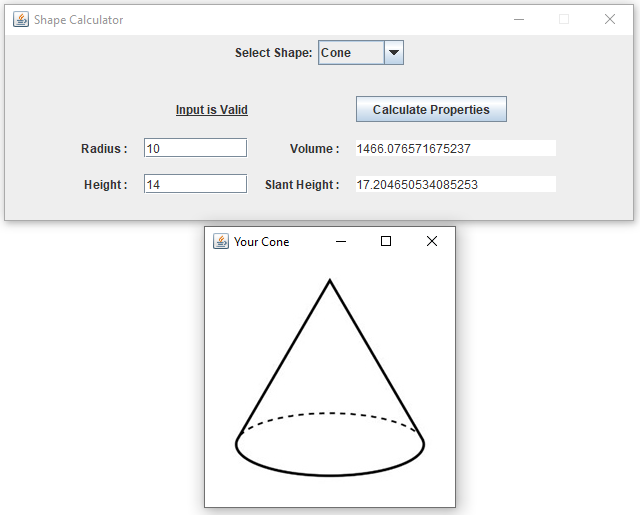


Figure 36: Test Case 25 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for a big cone

**Input:** Cone: Radius: 91. Height: 34. Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Volume: 294842.65932960686. Slant Height: 97.14422267947796.

Frame displayed containing cone.jpg.

**Actual Output**: See Fig. 37.

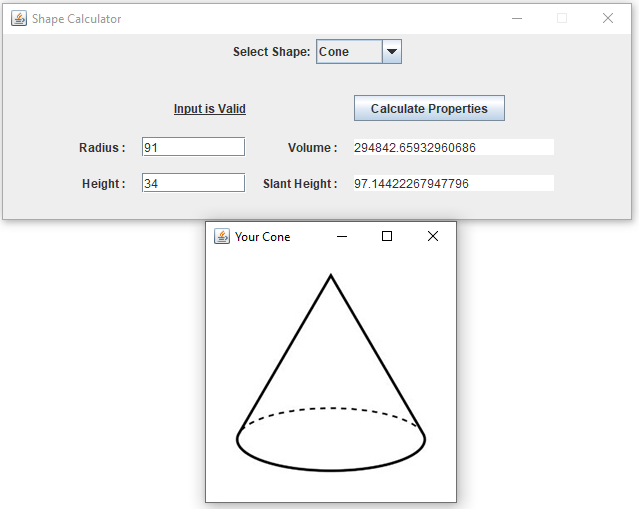


Figure 37: Test Case 26 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for a small cylinder

**Input:** Cylinder: Radius: 22. Height: 8. Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Volume: 12164.24675469968. Longest Diagonal: 44.721359549995796.

Frame displayed containing cylinder.jpg.

**Actual Output**: See Fig. 38.

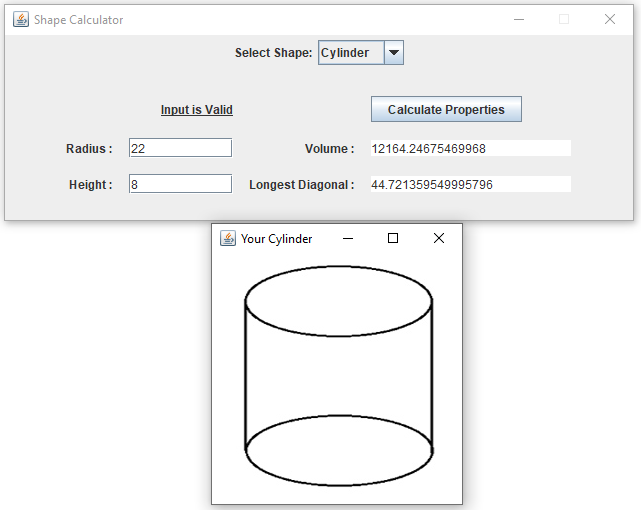


Figure 38: Test Case 27 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for a big cylinder

**Input:** Cylinder: Radius: 344. Height: 791. Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Volume: 2.9406493502986455E8. Longest Diagonal: 1048.3439321138842.

Frame displayed containing cylinder.jpg.

**Actual Output**: See Fig. 39.

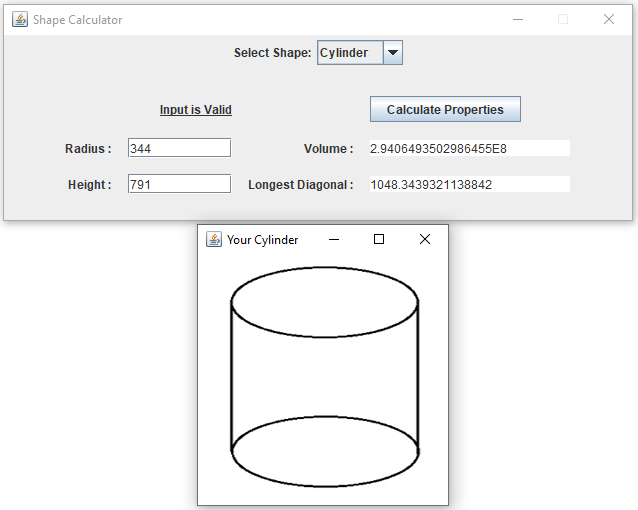


Figure 39: Test Case 28 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for a ring torus

**Input:** Torus: Minor Radius: 38. Major Radius: 233. Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Volume: 6641296.279910632. Torus Type: Ring.

Frame displayed containing ringtorus.jpg.

**Actual Output**: See Fig. 40.

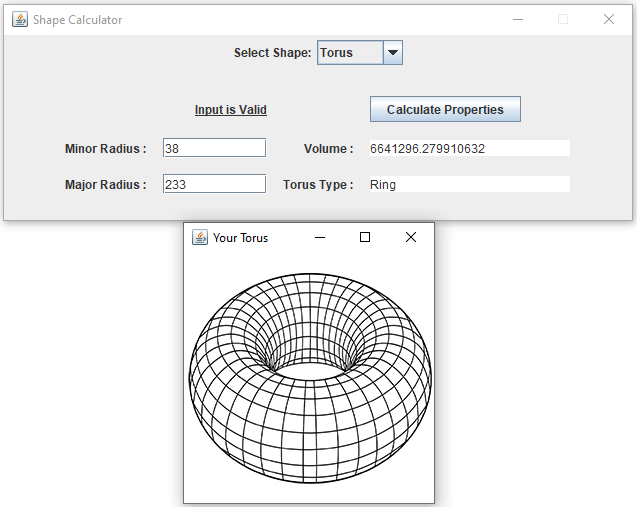


Figure 40: Test Case 29 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for a horn torus

**Input:** Torus: Minor Radius: 423. Major Radius: 423.

Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Volume: 1.4940008452166097E9. Torus Type: Horn.

Frame displayed containing horntorus.jpg.

**Actual Output**: See Fig. 41.

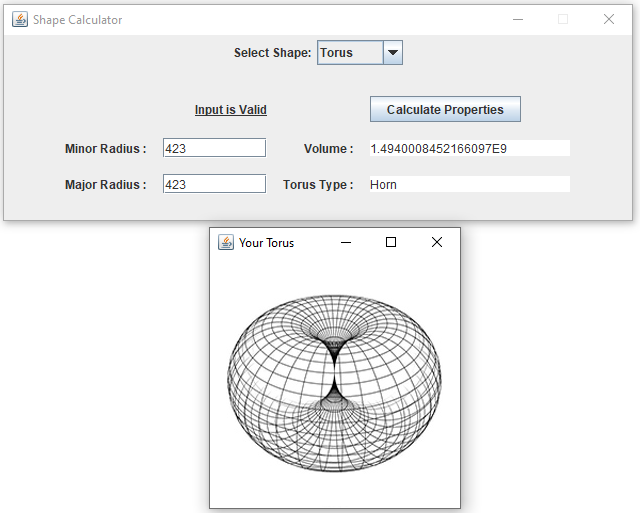


Figure 41: Test Case 30 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

1. **Aspect Tested:** Input valid dimensions for a spindle torus

**Input:** Torus: Minor Radius: 102. Major Radius: 67.

Click “Calculate Properties” Button

**Expected Output:** Input Label reads “Input is Valid”.

Volume: Cannot be calculated for Spindle. Torus Type: Spindle.

Frame displayed containing spindletorus.jpg.

**Actual Output**: See Fig. 42.

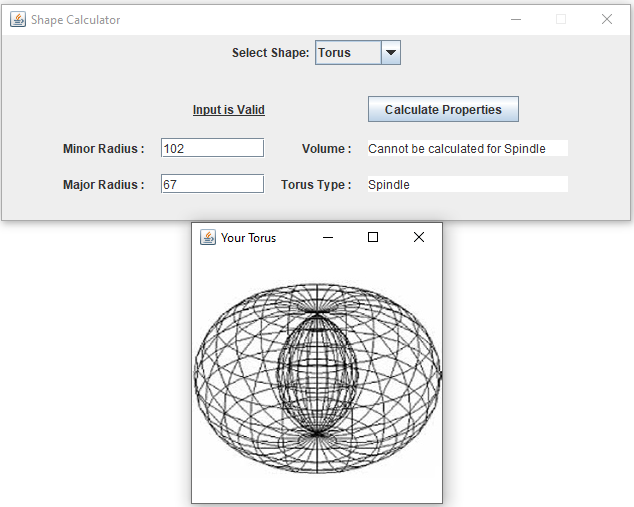


Figure 42: Test Case 31 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

**Lessons learned**

Soon after reading the instructions for this project, I knew I wanted to try using a Combo Box in the GUI (This was my first time). I read up on CardLayout to get the combo box to work (each combo box option has a corresponding “card” that is displayed) and how different panels can be used as the different cards. In this case, the cards were the shape menus. My initial plan was to have a single method in the main class that would act as a template for creating each card. It took a couple days to figure this out and I was able to get the menus to display, but soon realize there were too many scope problems. In my design, the main class had no proper way of accessing the Shape child classes (Circle, Square, etc.) and using their methods with the user input. The whole plan dismantled. So, the first lesson I learned was: plan the scope design ahead of time.

Another failure here was trying to use the single method for each card. I also tried having one listener to listen/handle the input from each card (KeyAdapter and ActionListener). All the variables, having the same names to avoid code duplication, collided with each other and ruined each other’s assignments. The lesson to learn here was: sharing too many elements among objects is difficult to handle and keep track of.

After facing this obstacle, I switched the plan entirely and decided to have each shape class create their own unique menu panel. This also meant giving each class their own listener inner classes. The main class would use each shape’s createMenu() method and turn those into cards. The user input finally had access to the calculation methods since they were in their same respective classes. While I really wanted to avoid this heaviness in code duplication (for example, putting the exact same listener inner classes in each shape class), I was running out of time and this point and did not have much of a choice. With more time, I would have tried to find a more efficient design.

I am ultimately unsatisfied with the final product of the code itself simply because there ended up being way too much code duplication. Perhaps with a combo box it’s difficult to avoid that. As for the GUI itself, however, it works perfectly, and I am glad I had this learning experience with the combo box and stuck to it. I am also happy with the way the GUI looks. The biggest lesson I took from this is: I need to better plan the scope of all variables/methods before getting to work on the initial design. A single scope issue could potentially ruin an entire design. On that note, I want to spend more of my time learning about scope solutions and designs to avoid this in the future.

References

Kucera, 2020. .zip file has been unzipped.

Kucera, 2020. Creating Java Application Project in Netbeans.

Kucera, 2020. Creating Test Kucera Project 2.

Kucera, 2020. Copying Application Source Files.

Kucera, 2020. Open MenuGUI.java, click Run Project.

Kucera, 2020. Pasted Source Files appear in IDE.

Kucera, 2020. Pasting Application Source Files in Test Project.

Kucera, 2020. Project 2 UML Class Diagram.

Kucera, 2020. Project Properties > Run, type in Main Class.

Kucera, 2020. Run project, menu GUI displayed.

Kucera, 2020. Test Case 1 Output.

Kucera, 2020. Test Case 2 Output.

Kucera, 2020. Test Case 3 Output.

Kucera, 2020. Test Case 4 Output.

Kucera, 2020. Test Case 5 Output.

Kucera, 2020. Test Case 6 Output.

Kucera, 2020. Test Case 7 Output.

Kucera, 2020. Test Case 8 Output.

Kucera, 2020. Test Case 9 Output.

Kucera, 2020. Test Case 10 Output.

Kucera, 2020. Test Case 11 Output.

Kucera, 2020. Test Case 12 Output.

Kucera, 2020. Test Case 13 Output.

Kucera, 2020. Test Case 14 Output.

Kucera, 2020. Test Case 15 Output.

Kucera, 2020. Test Case 16 Output.

Kucera, 2020. Test Case 17 Output.

Kucera, 2020. Test Case 18 Output.

Kucera, 2020. Test Case 19 Output.

Kucera, 2020. Test Case 20 Output.

Kucera, 2020. Test Case 21 Output.

Kucera, 2020. Test Case 22 Output.

Kucera, 2020. Test Case 23 Output.

Kucera, 2020. Test Case 24 Output.

Kucera, 2020. Test Case 25 Output.

Kucera, 2020. Test Case 26 Output.

Kucera, 2020. Test Case 27 Output.

Kucera, 2020. Test Case 28 Output.

Kucera, 2020. Test Case 29 Output.

Kucera, 2020. Test Case 30 Output.

Kucera, 2020. Test Case 31 Output.

Kucera, 2020. Unzipping a .zip file.