John Kucera

Prof. Didier Vergamini

CMSC 335 Object-Oriented and Concurrent Programming

22 November 2020

Week 4: Project 2 **(REWORK)** Documentation & Solution Description

**Class Hierarchy for Shapes + GUI**

**Assumptions + Design decisions:**

* **The GUI works exactly the same as my first design, but with a lot of improvements in the code. TwoDimensionalShape and ThreeDimensionalShape have been included this time for a more structured class hierarchy and consistency.**
* The major flaw in my previous design was using a CardLayout which forced me to create variables for all shape menus since none of them could easily be shared. This time, I designed the program without CardLayout and this allowed me to have a single panel for the I/O menu. All shapes share this one panel, and the labels would be edited according to the shape selected with the combo box. This drastically decreases the amount of code to the point where there is nearly no code duplication.
* **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.
* 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.

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

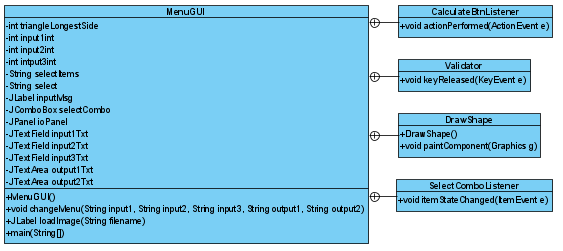
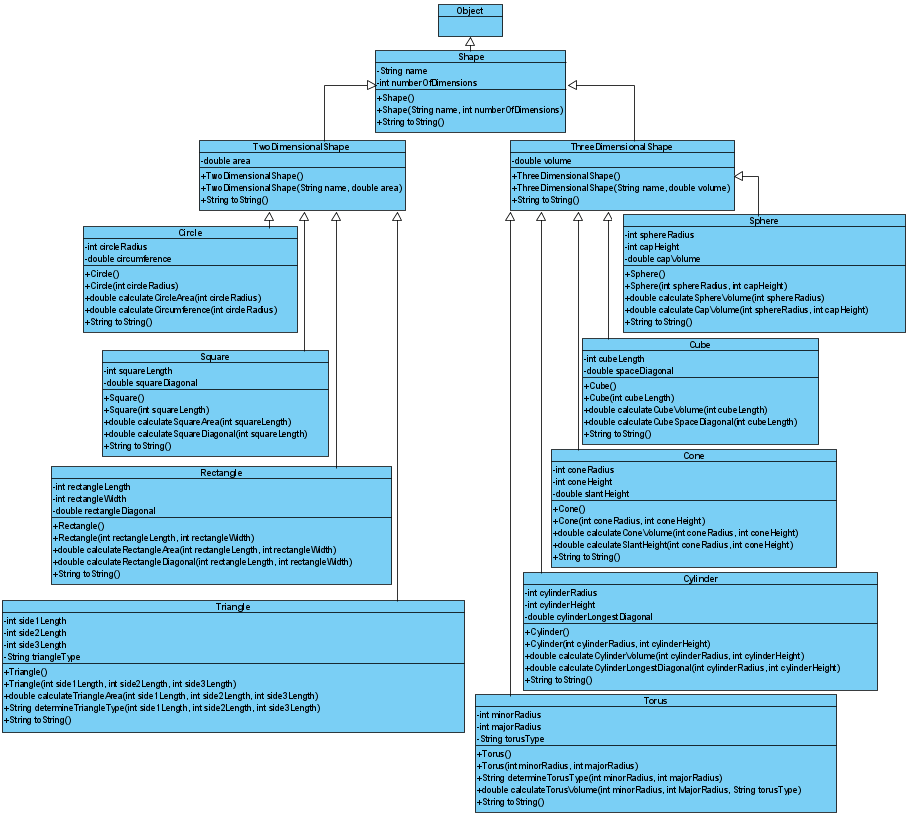


Figure 1: Project 2 Rework UML Class diagram. (Kucera, 2020)

**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-Rework.zip**. You can do this easily by right-clicking **JohnKuceraProject2-Rework.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)

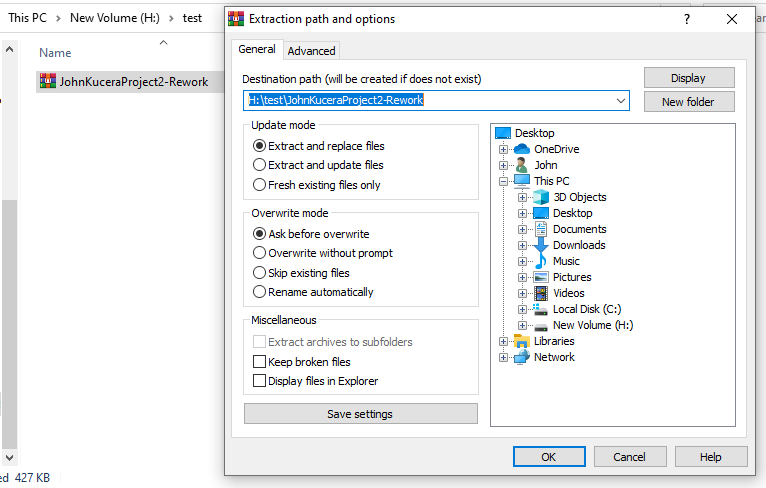
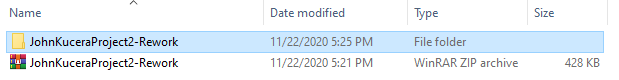


Figure 3: .zip file has been unzipped. (Kucera, 2020)



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

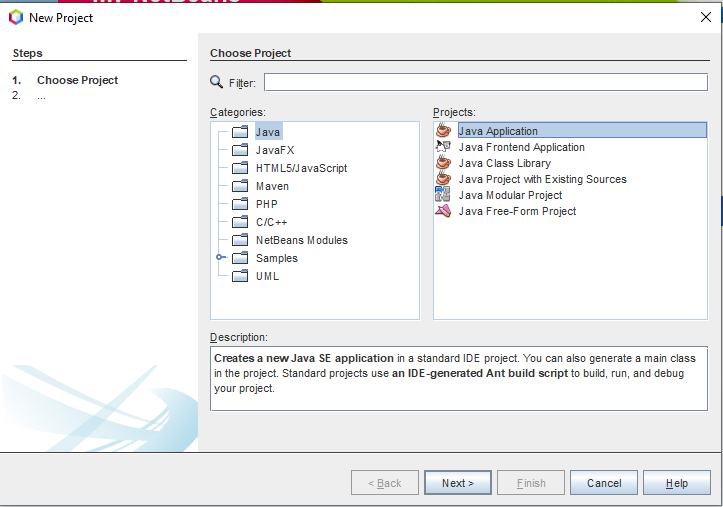
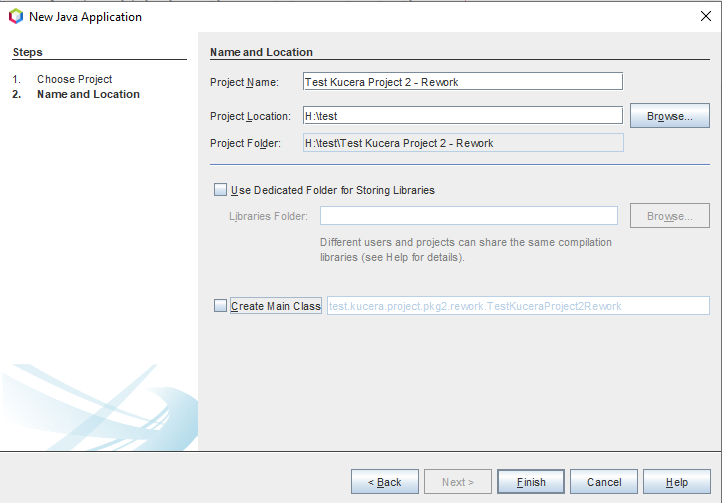


Figure 4: Creating Java Application Project in Netbeans. (Kucera, 2020)

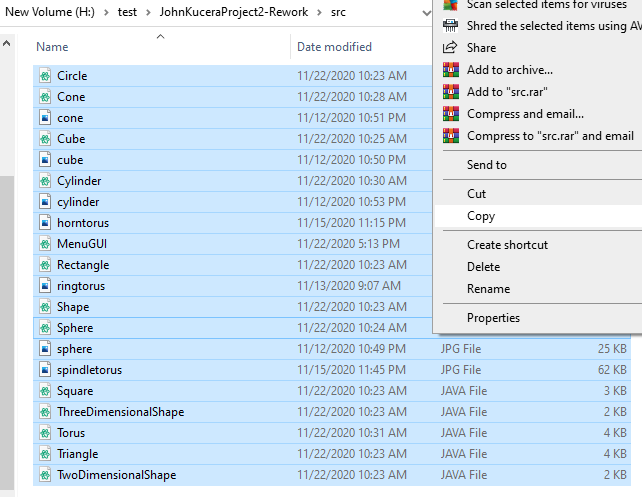
1. (See Fig. 5) Name the project “**Test Kucera Project 2 - Rework**”. Identify the project location (which is where the application files will be saved). **DO NOT allow the IDE to automatically create main class**.

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



1. In your File Explorer, go to the unzipped **JohnKuceraProject2-Rework > src** folder and copy the 20 .java and .jpg source files: **Circle.java, Cone.java, Cube.java, Cylinder.java, MenuGUI.java, Rectangle.java, Shape.java, Sphere.java, Square.java, ThreeDimensionalShape.java, Torus.java, Triangle.java, TwoDimensionalShape.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 – Rework > 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)

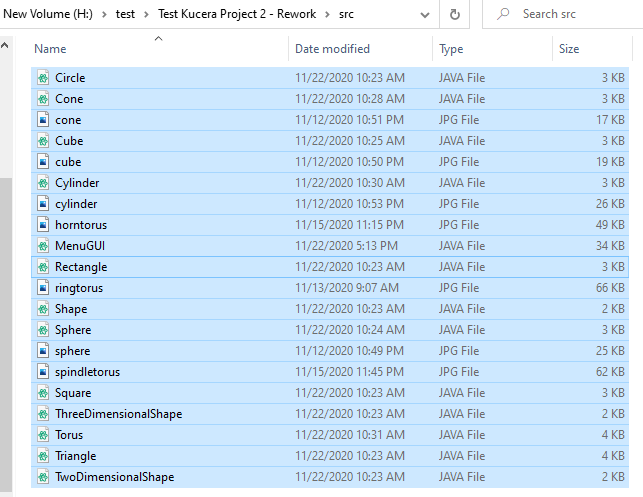
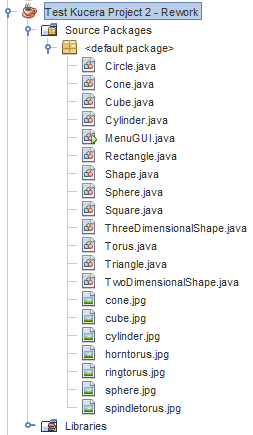
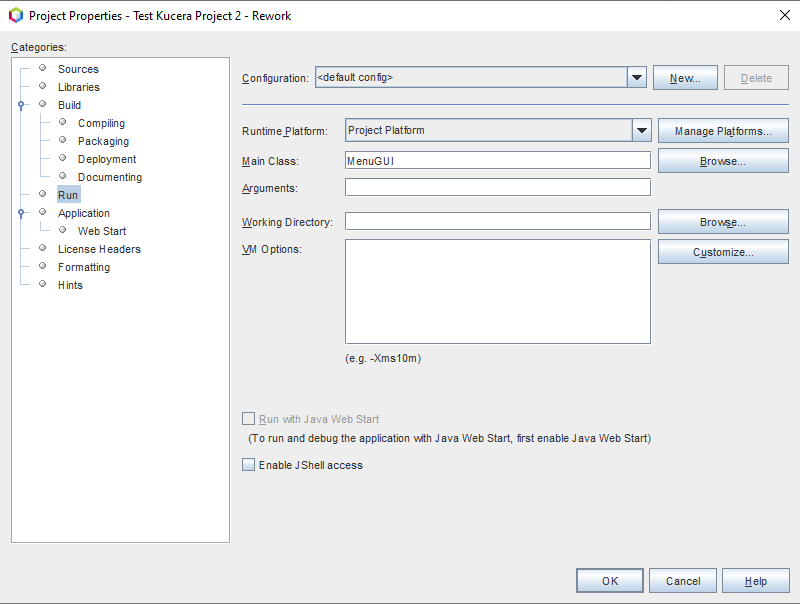


Figure 8: Pasted Source Files appear in IDE. (Kucera, 2020)



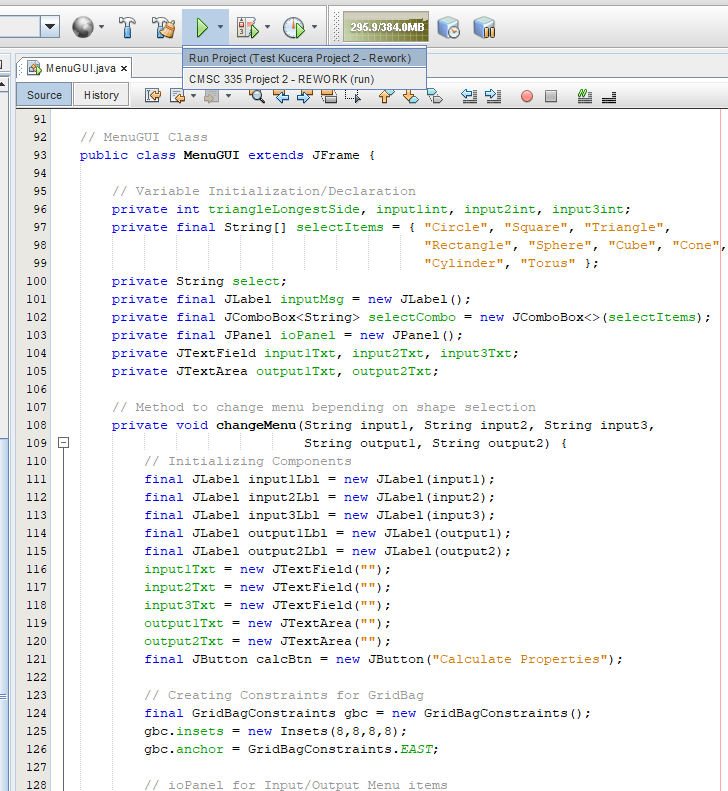
1. In the IDE, open the **Project Properties** of **Test Kucera Project 2 - Rework**. You can do this by right-clicking **Test Kucera Project 2 - Rework** and clicking **Properties**. In the “Run” options, change the Main Class to “**MenuGUI**”, since the Main Method for this application is in **MenuGUI.java**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

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



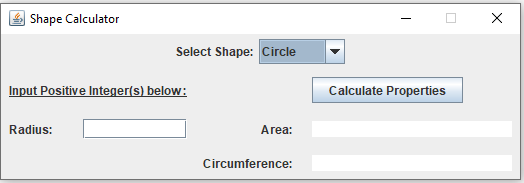
1. You can click on each source file to open them and view the Java code in each. To run the application, make sure **MenuGUI.java** is open and click **Run Project** (See Fig. 10).

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. **NOTE: Although the test cases are similar to the previous Project 2 Submission, these are new screenshots from this reworked program. The reworked GUI is slightly smaller than my previous design.**

**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 |
| Combo Box switches between menus | 2 |
| Error checking/warnings | 3 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.

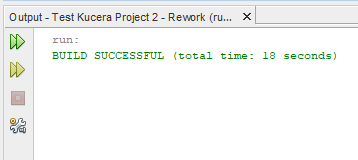


Figure 12: Test Case 1 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

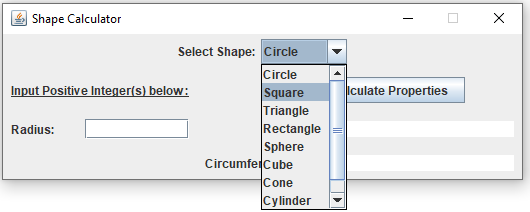
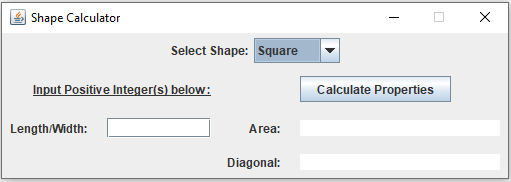
1. **Aspect Tested:** Menu items change according to the Shape chosen in Combo box

**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).”

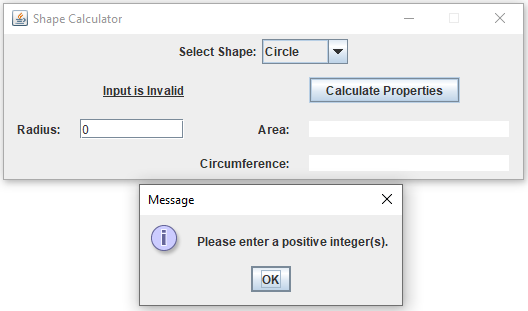
**Actual Output**: See Fig. 14. 

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.

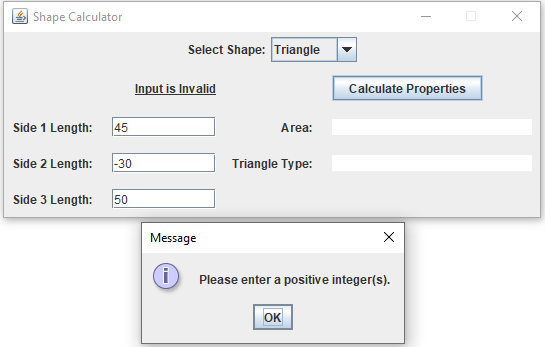


Figure 15: Test Case 4 Output. (Kucera, 2020)

**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.

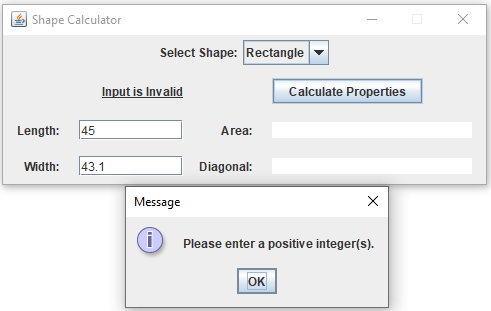


Figure 16: Test Case 5 Output. (Kucera, 2020)

**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.

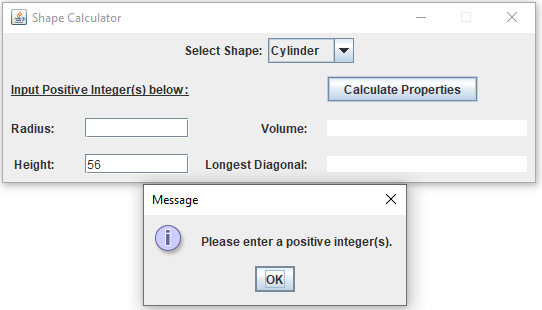


Figure 17: Test Case 6 Output. (Kucera, 2020)

**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.

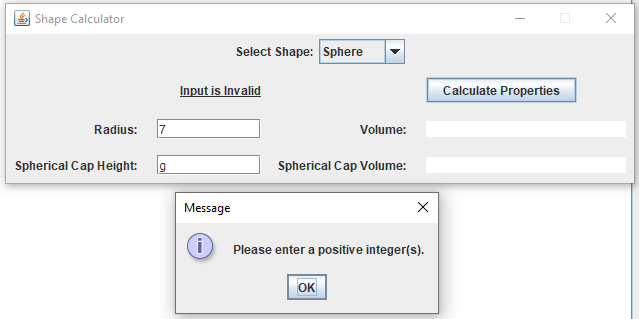


Figure 18: Test Case 7 Output. (Kucera, 2020)

**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.

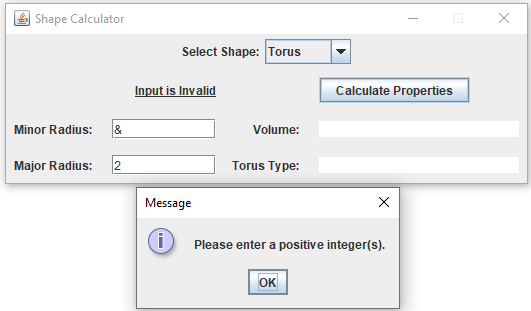


Figure 19: Test Case 8 Output. (Kucera, 2020)

**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.

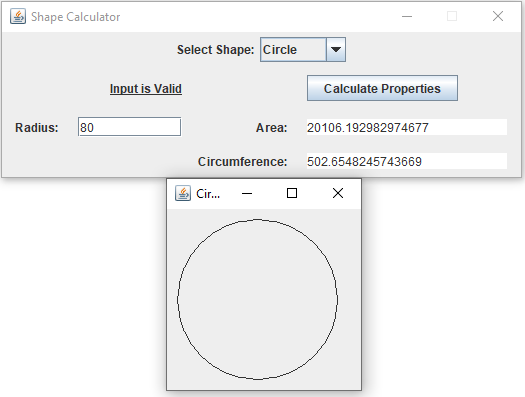


Figure 20: Test Case 9 Output. (Kucera, 2020)

**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.

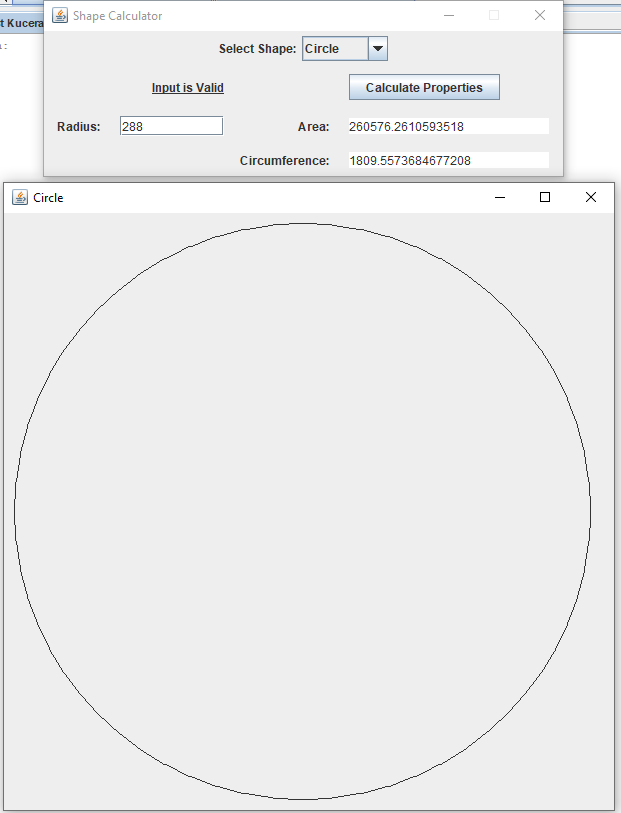


Figure 21: Test Case 10 Output. (Kucera, 2020)

**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.

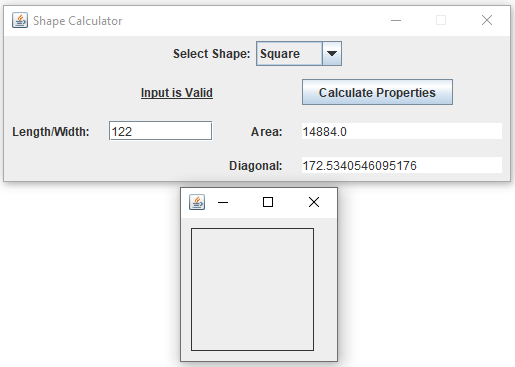


Figure 22: Test Case 11 Output. (Kucera, 2020)

**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.

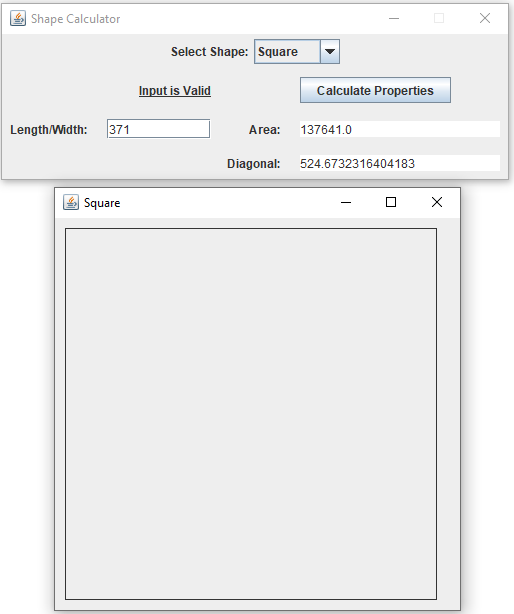


Figure 23: Test Case 12 Output. (Kucera, 2020)

**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.

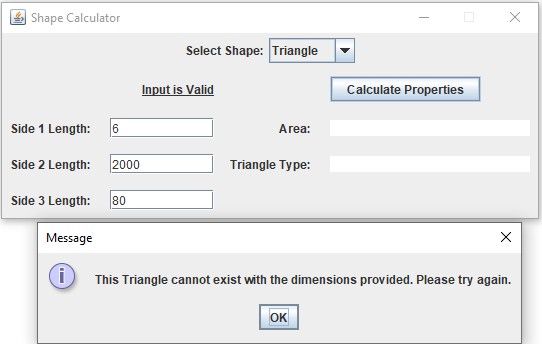


Figure 24: Test Case 13 Output. (Kucera, 2020)

**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.

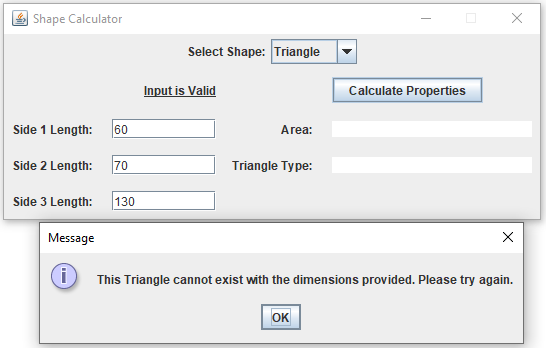


Figure 25: Test Case 14 Output. (Kucera, 2020)

**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.

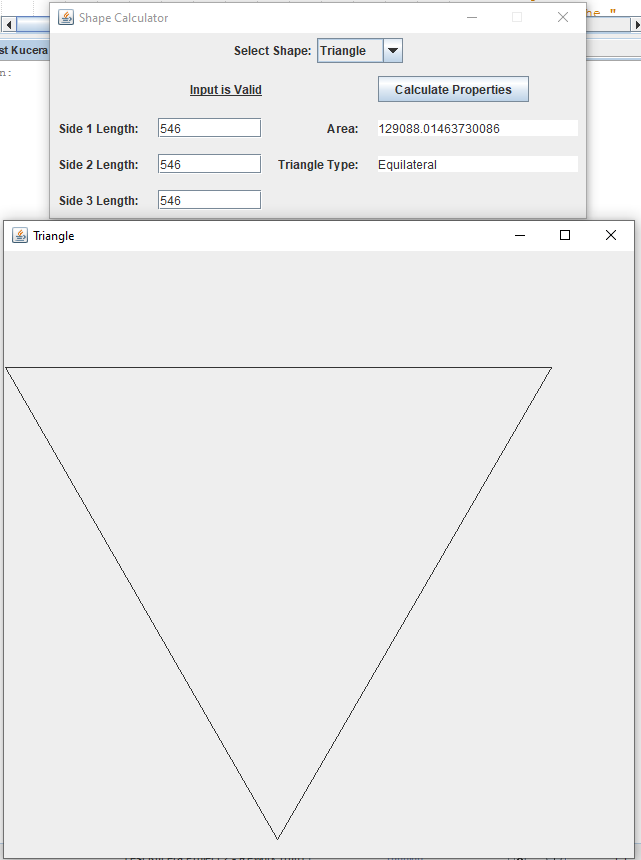


Figure 26: Test Case 15 Output. (Kucera, 2020)

**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.

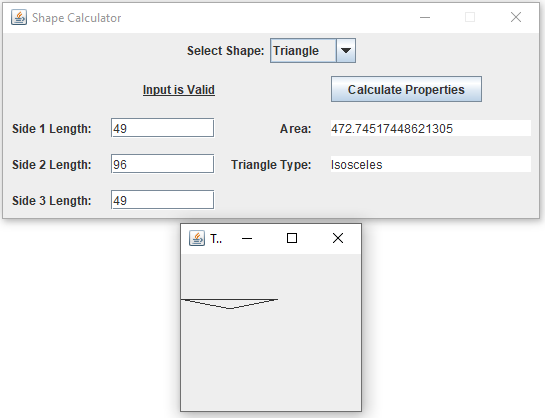


Figure 27: Test Case 16 Output. (Kucera, 2020)

**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.

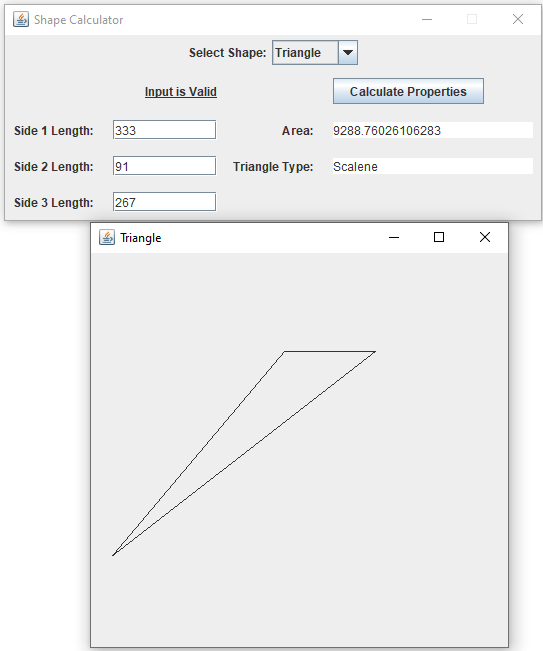


Figure 28: Test Case 17 Output. (Kucera, 2020)

**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.

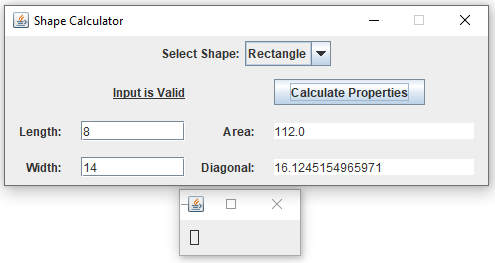


Figure 29: Test Case 18 Output. (Kucera, 2020)

**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: 294145.0. Diagonal: 796.8349891916142.

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

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

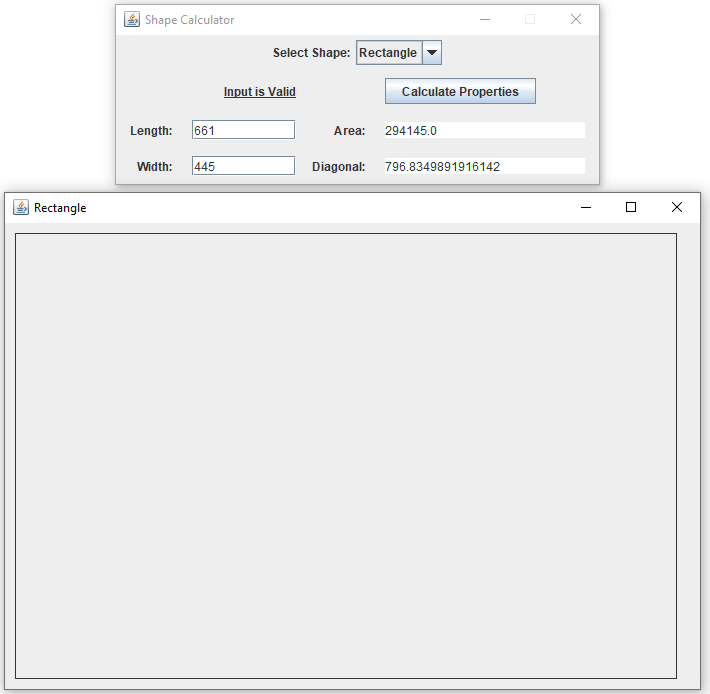


Figure 30: Test Case 19 Output. (Kucera, 2020)

**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.

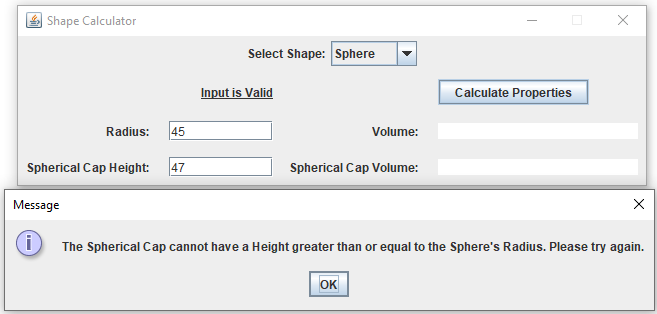


Figure 31: Test Case 20 Output. (Kucera, 2020)

**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.

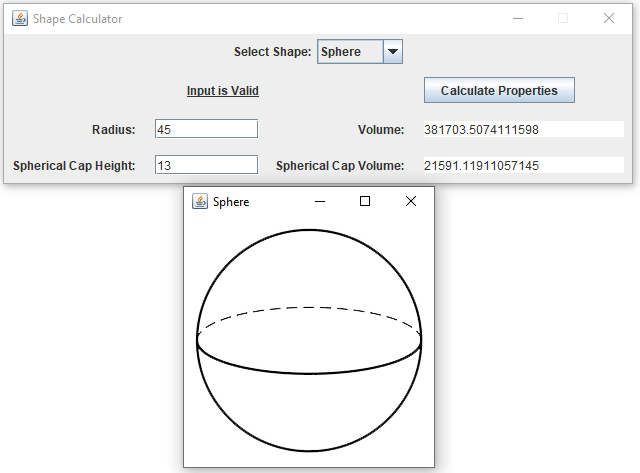


Figure 32: Test Case 21 Output. (Kucera, 2020)

**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.

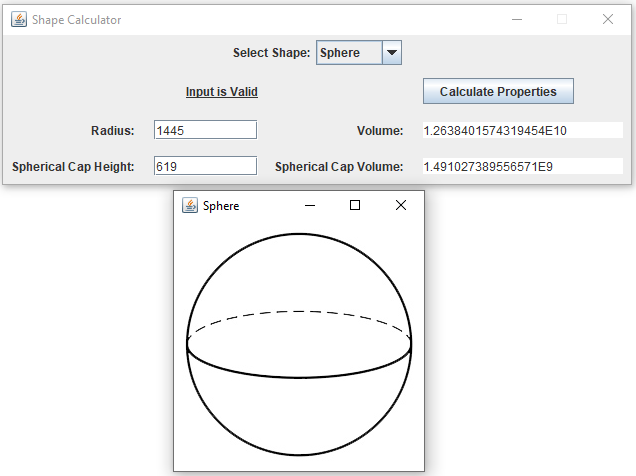


Figure 33: Test Case 22 Output. (Kucera, 2020)

**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.

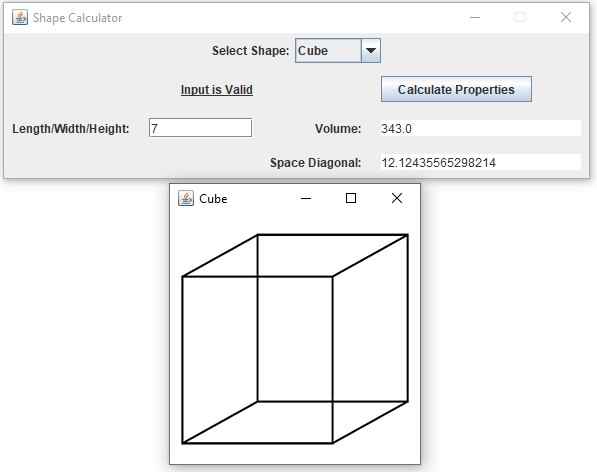


Figure 34: Test Case 23 Output. (Kucera, 2020)

**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.

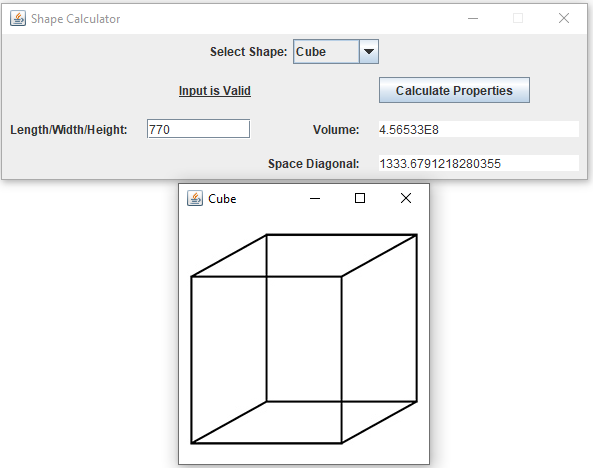


Figure 35: Test Case 24 Output. (Kucera, 2020)

**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.

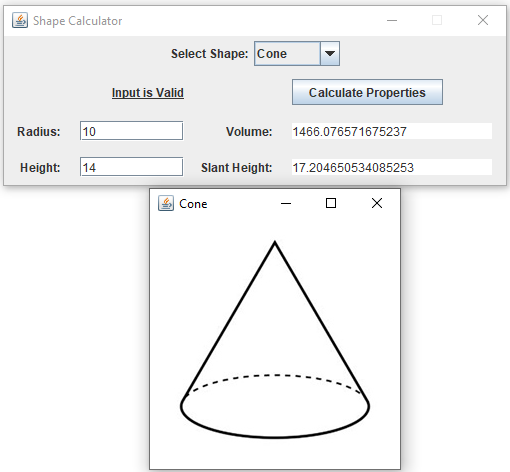


Figure 36: Test Case 25 Output. (Kucera, 2020)

**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.

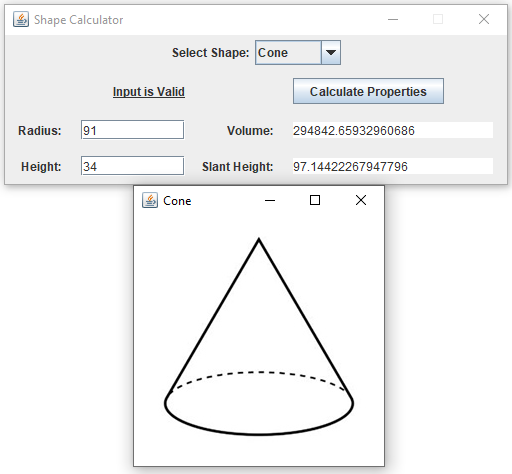


Figure 37: Test Case 26 Output. (Kucera, 2020)

**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.

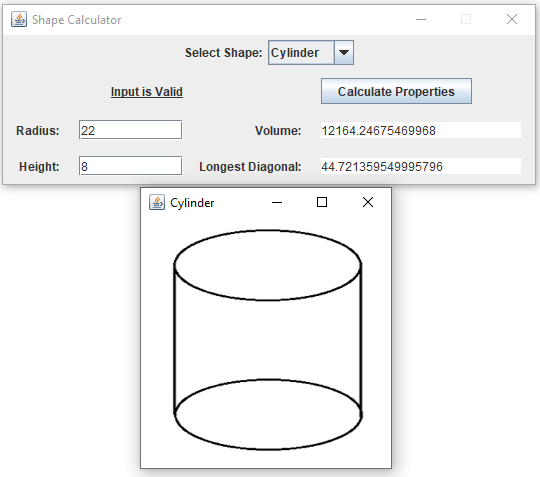


Figure 38: Test Case 27 Output. (Kucera, 2020)

**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.

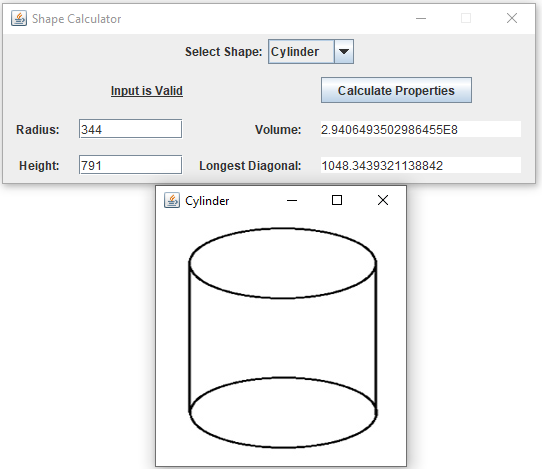


Figure 39: Test Case 28 Output. (Kucera, 2020)

**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.

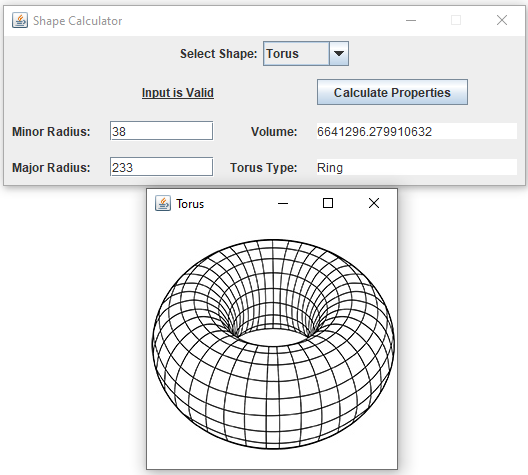


Figure 40: Test Case 29 Output. (Kucera, 2020)

**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.

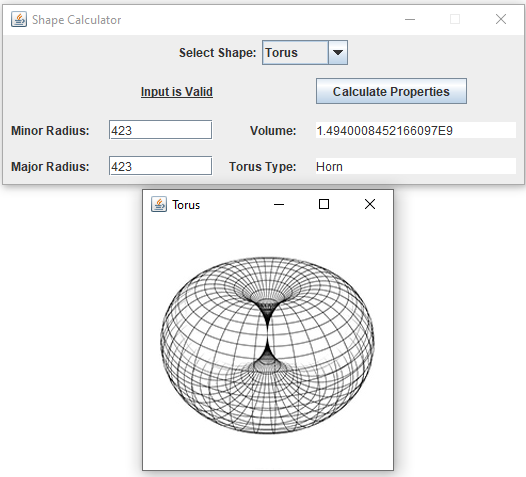


Figure 41: Test Case 30 Output. (Kucera, 2020)

**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.

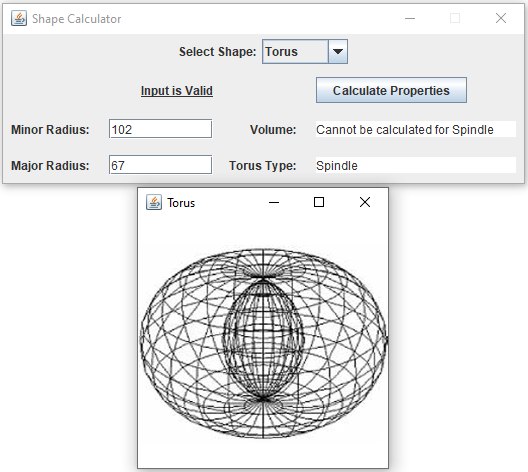


Figure 42: Test Case 31 Output. (Kucera, 2020)

**Pass or Fail?**: PASS

**Lessons learned**

Last time, I learned to think about the scope of my variables and methods BEFORE going through with a design. Running into scope issues caused me to run out of time and come up with a half-baked solution with way too much code duplication. I kept that in mind for this rework and made one decision that made this design a lot better: not using CardLayout. CardLayout, a layout that allows you to store JPanels that can be switched between (with a combo box or a button), is very inefficient for using with so much options like Shapes. This is because every shape has at least 5 attributes (dimensions) that cannot be shared with the others if CardLayout is used. In addition to the attributes, Shapes will have a visual and listeners for their respective menus. With CardLayout, I had to create all of those over and over again for each shape.

Without CardLayout in this rework, I went back to my very first design idea of having a single changeMenu() method that would alter the JLabels in the GUI based on the shape. This means there is a single JPanel that all shapes can share, drastically reducing the amount of code needed. Additionally, all shapes can share the same listeners and Graphics method in this one JPanel. I was able to factor out all GUI information from the shape classes, and now they look very similar to how they did in Project 1.

As for the class hierarchy, I added ThreeDimensionalShape and TwoDimensionalShape back in (as well as numberOfDimensions for some consistency). They contain area and volume to be inherited by the child shapes. I did this just to have a finer structure to the class hierarchy – in my previous design, I only had Shape class and the 9 shapes as children, which felt very lackluster and was hardly a substantial hierarchy.

Some things I wish I still designed better on this rework is dealing some of the variables that need to be shared among the listener methods. It seems too unconventional to try and make a listener method that has additional parameters, so I ended up with a handful of variables that had to be declared outside of the methods (such as the combo box, the text fields, and the text areas). Not that it is a problem; keeping these variables outside of methods was completely necessary for this design. However, I am left with the feeling that I still could have designed a better method to hold more of those variables.

Overall, I am much more satisfied with this rework than I was with my previous design. Code duplication has been greatly reduced and the class hierarchy is more substantial and consistent. The GUI works perfectly with all successful test case results.

References

Kucera, 2020. .zip file has been unzipped.

Kucera, 2020. Creating Java Application Project in Netbeans.

Kucera, 2020. Creating Test Kucera Project 2 - Rework.

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 Rework 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.