Project 2 Documentation

Jade Pearl

CMSC 335

Professor Andrew Seely

**UML Diagram**

My UML diagram for this project is below; the image may only be clear if zoomed in since the image used is very wide. I used MermaidChart (a chart maker) to create my diagram.

A diagram of a diagram

Description automatically generated with medium confidence

**User Guide**

To set up the application, open the Project2folder in Visual Studio Code. Make sure all .java files from this program are in the same directory ./Project2/. To run the program, hover over the Run tab in the top left of VSCode and click the start debugging command or click the play button icon in the top right of the window. It is next to the file tabs:

A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated

This should also automatically compile the code for you. You may get a pop-up that mentions a “failed build”. If so, make sure to click the continue button.

A screenshot of a computer screen

Description automatically generated

If using a Linux terminal such as WSL, open the terminal in VSCode using the dropdown menu in the terminal tab at the bottom of the VSCode window.

A screenshot of a computer program

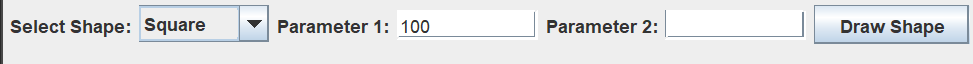
Description automatically generated

Click which terminal you wish to use. The example on my computer is Ubuntu (WSL). Go to the proper directory with the cd command if not in the directory with the java files already. Compile all .java files with the command javac \*.java and then run the Project2 with the command java Project2 command. Follow the program’s directions or force exit with CTRL+Z or Command+Z if on Mac. Please note that the GUI itself may be smaller for some reason than when run through the Linux terminal:

A screenshot of a computer

Description automatically generated

The output GUI when using Visual Studio Code’s built-in Run feature is on the left and running the program using the WSL terminal is on the right.

When running the program, please keep in mind the parameters you must enter for each shape:  
  
Circle: Parameter 1 is the radius. Parameter 2 is not used.

Square: Parameter 1 is the side length. Parameter 2 is not used.

Rectangle: Parameter 1 is the height. Parameter 2 is the width.

Triangle: Parameter 1 is the base length. Parameter 2 is the height of the triangle.

Sphere: Parameter 1 is the radius. Parameter 2 is not used.

Cube: Parameter 1 is the side length. Parameter 2 is not used.

Cylinder: Parameter 1 is the radius. Parameter 2 is the height.

Cone: Parameter 1 is the radius. Parameter 2 is the height.

Torus: Parameter 1 is the major radius. Parameter 2 is the minor radius.

**Test Data/Plan**

My test plan is as follows in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| Test # | Description | Screenshot | PASS/FAIL Flag |
| 1 | Properly draws a Circle |  | PASS |
| 2 | Properly draws a Square |  | PASS |
| 3 | Properly draws a triangle |  | PASS |
| 4 | Properly draws a rectangle |  | PASS |
| 5 | Properly draws a sphere |  | PASS; the sphere seems a little large. Parameter 2 is ignored if entered |
| 6 | Properly draws a cube |  | PASS. Parameter 2 is ignored if entered |
| 7 | Properly draws a cone |  | PASS |
| 8 | Properly draws a cylinder |  | PASS. Seems a little large however |
| 9 | Properly draws a Torus |  | PASS. |
| 10 | Input validation: making sure the user enters the proper amount of parameters depending on shape |  | Possibly about a 90% pass. It allows the user to enter a 0 in parameters which may yield unwanted results. |

**Lessons Learned**

This project introduced completely new concepts to me. In the past, I have done projects that “drew” shapes to the console using asterisks or other symbols and I have also used GUI via JavaFX. For this project, I had to learn how to draw shapes within a GUI panel or frame. JavaFX was not working well for me in this project, so I ended up researching Swing and how I could use different features of Swing based on tutorials provided to the class from the Oracle website in order to draw the shapes needed for this project. I learned how to use ActionListener, and other aspects in Swing like Graphics, Color, Polygon, Ellipses2D, and GradientPaint to draw both 2D and 3D shapes alike. Instead of showing the 3D shapes as an image, I instead learned how to draw the 3D shapes in a 2D fashion but still keep its 3D properties. One way to help visualize 3D effect was to make the 3D shapes gradient or shade a different color in different areas. This project certainly was challenging but very informative on how to develop complex graphics using GUI (specifically Swing).

**Resources**

*Lesson: Getting Started with Swing (The JavaTM Tutorials > Creating a GUI With Swing)*. (n.d.). Docs.oracle.com. Retrieved November 16, 2023, from <https://docs.oracle.com/javase/tutorial/uiswing/start/index.html>

*Lesson: Using Swing Components (The JavaTM Tutorials > Creating a GUI With JFC/Swing)*. (n.d.). Docs.oracle.com. <https://docs.oracle.com/javase/tutorial/uiswing/components/index.html>

*Lesson: Writing Event Listeners (The JavaTM Tutorials > Creating a GUI With Swing)*. (n.d.). Docs.oracle.com. <https://docs.oracle.com/javase/tutorial/uiswing/events/index.html>

*Trail: 2D Graphics (The JavaTM Tutorials)*. (n.d.). Docs.oracle.com. https://docs.oracle.com/javase/tutorial/2d/index.html