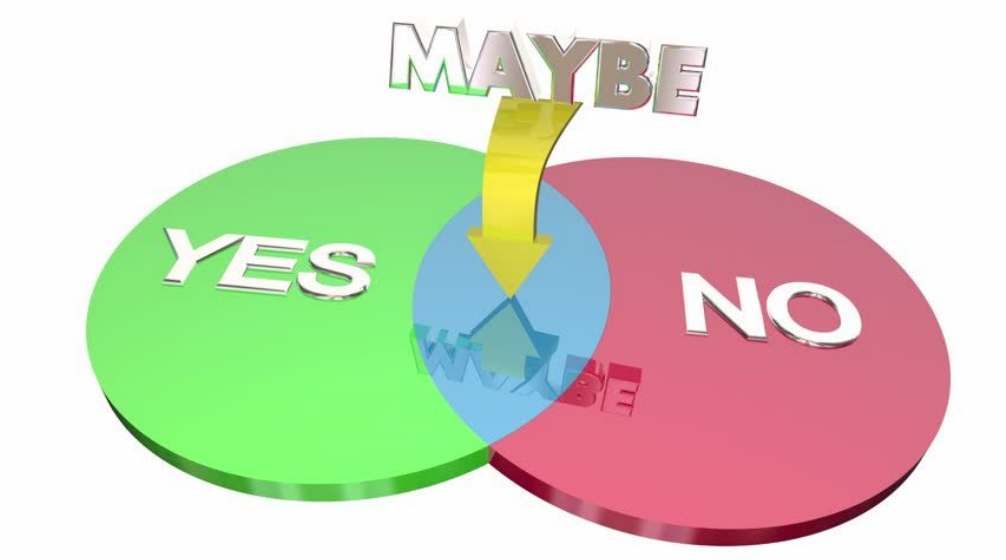
EECS 2311: Venn Diagram Project
Final Submission



Product Name – VennSmarter

Group 15:
Warsame Osman 215551914
Jada Jones 215562044
Parssa Khazra 216147936

Table of Contents

Introduction

This is our venn diagram term project which was assigned to us for EECS 2311. We were given minimum requirements that the UI should be able to do which are:-

- Each element in the Venn diagram may have a longer description that is by default hidden, but the user must be able to display it
- The app must support a mode where the user is asked to arrange a set of tags on the Venn diagram. Once finished, the user can compare their arrangement to a previously hidden correct answer
- Your system must allow the users to select multiple objects at once in order to customize them, e.g. select all objects in the intersection, and increase their font size
- Your system must also implement an Undo / Redo mechanism

Throughout the term we also made releases and documentation to receive feedback from our peers in preparation for a bug-free final release. We used all of the feedback from our peers as well as the professors and teaching assistants. This feedback allowed is to create a more user friendly UI and for that we thank each of you for your input. In this final release we have updated each of our documents to reflect the final release of the UI.

Requirements Document

Client Needs:

- A program that creates a customizable venn diagram
- A program that enables you to create your venn diagram template of choice
- The venn diagram needs to have the option of selecting 2 or 3 regions
- The venn diagram needs to have the option of selecting the intersections of the regions
- The venn diagram needs to be able to insert text into these regions
- The text needs an additional option where we can elaborate/describe the text

Use Cases:

The functionality of our program provides many cases towards the orientation of the Venn diagram.
Press “create new Venn Diagram” button to open the Venn diagram builder form

- Select 3 regions from the drop down menu
  - Select up to three intersection choices for a three region spaced diagram
- Select 2 regions from the drop down menu
  - Select the intersection for a classic Venn diagram, press done

There are several other functionalities of the product that enhance the users experience,such as the following.

- Press “Load a new Venn Diagram” to load the Venn Diagram from a “.txt” file ,This text file contains the code needed to bring back the desired Venn diagram that was priorly created.

- Press “Title” on the menu bar to create a title.

- Press “Circle”.
- Select between three Circle choices,to edit its”Circle color”,”Circle size”and ”Circle drag”
- To move the circle to its designated spot.

- Press the “Label” menu button , select between three label choices.
- Select between three label choices,to edit its”label color”,”label font”,”label color”.

- Press “Save” to save the Venn Diagram in a “.txt” file , the text file will now contain code of all the nodes and widgets that are on the venn Diagram pane.

Acceptance Test Cases:

Two region venn diagram:

Select “2” regions from the drop down menu
- Choose to select no intersection
- No generated venn diagram, please exit application by first exiting the venn builder form window, then the larger window titled “ venn diagram maker”. Both have exit buttons on the top right of their windows.

Select “2” regions from the drop down menu
- select “S1 intersect S2” checkbox
- Generated venn diagram preview, press the “done” button on the bottom right. A venn diagram is generated.

Three region venn diagram:

Select “3” regions from the drop down menu
- Only one or less intersection chosen
- No generated venn diagram, please exit application by first exiting the venn builder form, then the larger window titled “ venn diagram maker”. Both have exit buttons on the top right of their windows

Select “3” regions from the drop down menu
- Select “S1 intersects S3” and “S1 intersects S2”
- Generated venn diagram preview, press the “done” button on the bottom right. A venn diagram is generated.

Select “3” regions from the drop down menu
- Select “S1 intersects S3” and “S2 intersects S3”
- Generated venn diagram preview, press the “done” button on the bottom right. A venn diagram is generated.

Select “3” regions from the drop down menu
- Select “S1 intersects S2” and “S2 intersects S3”

- Generated venn diagram preview, press the “done” button on the bottom right. A venn diagram is generated.

Select “3” regions from the drop down menu

- Select “S1 intersects S2” and “S1 intersects S3”
- Generated venn diagram preview, press the “done” button on the bottom right. A venn diagram is generated.

Design Document

Maintenance Scenarios:

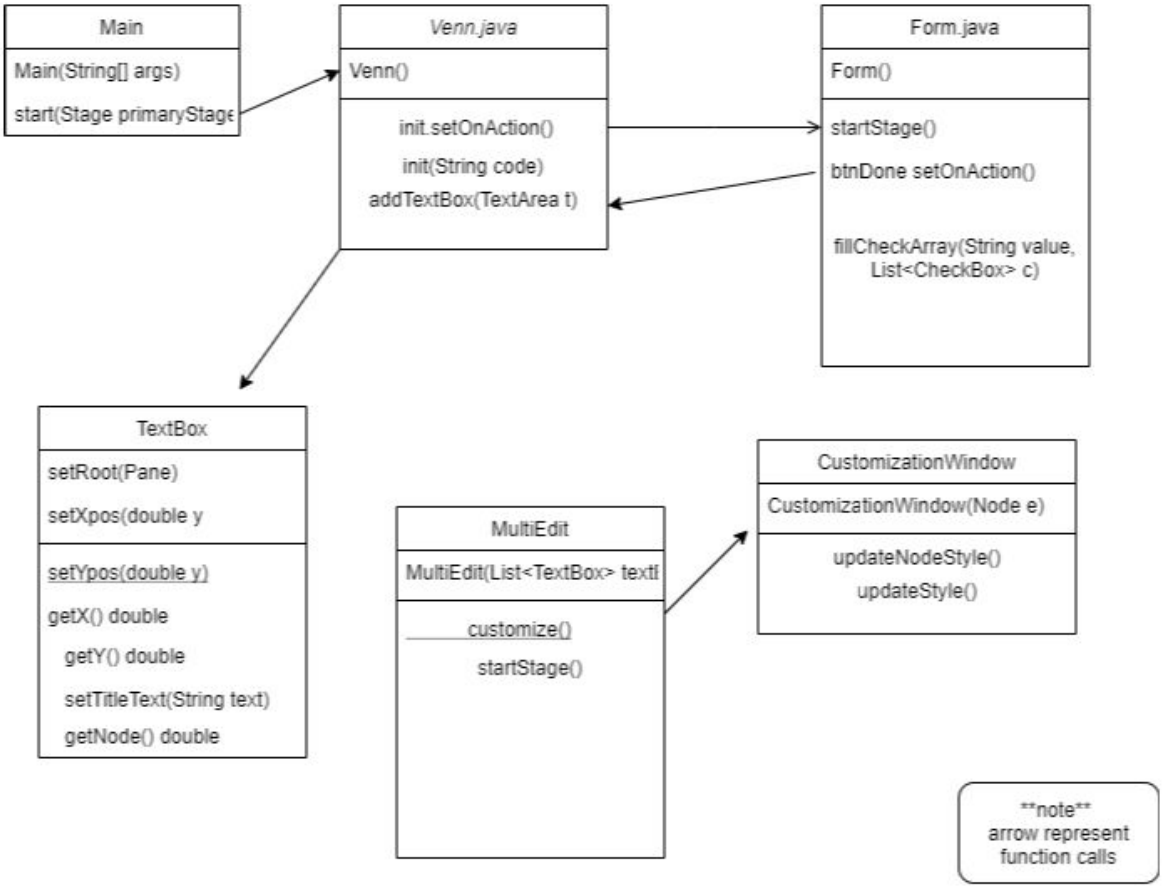
There are several potential changes in our application that we would need to do after this system is released. One of the most crucial changes would be to make the application more responsive. Responsive in terms of when we minimize or maximize our window, how will the application appear. At this moment our main window is kept on fullscreen with no option to minimize because we have been relying on the applications window dimensions to have our widgets to be where we need them.

We can make the application responsive successful if we were to implement two things to our application.The use of more built-in node methods instead of constant values, and the use of a ScrollPane to navigate the window when the window is minimized by the user.Some built-in node methods we we have been using are “getVisualBounds()” which returns the bounds of a screen you previously created.”getMaxX()” and ”getMaxY()”, which gets the maximum x-axis and y-axis component of the screen.”getPrefWidth()” and ”getPrefHeight()” which gets you the preferred x-axis and y-axis of the pane/region of your choosing.

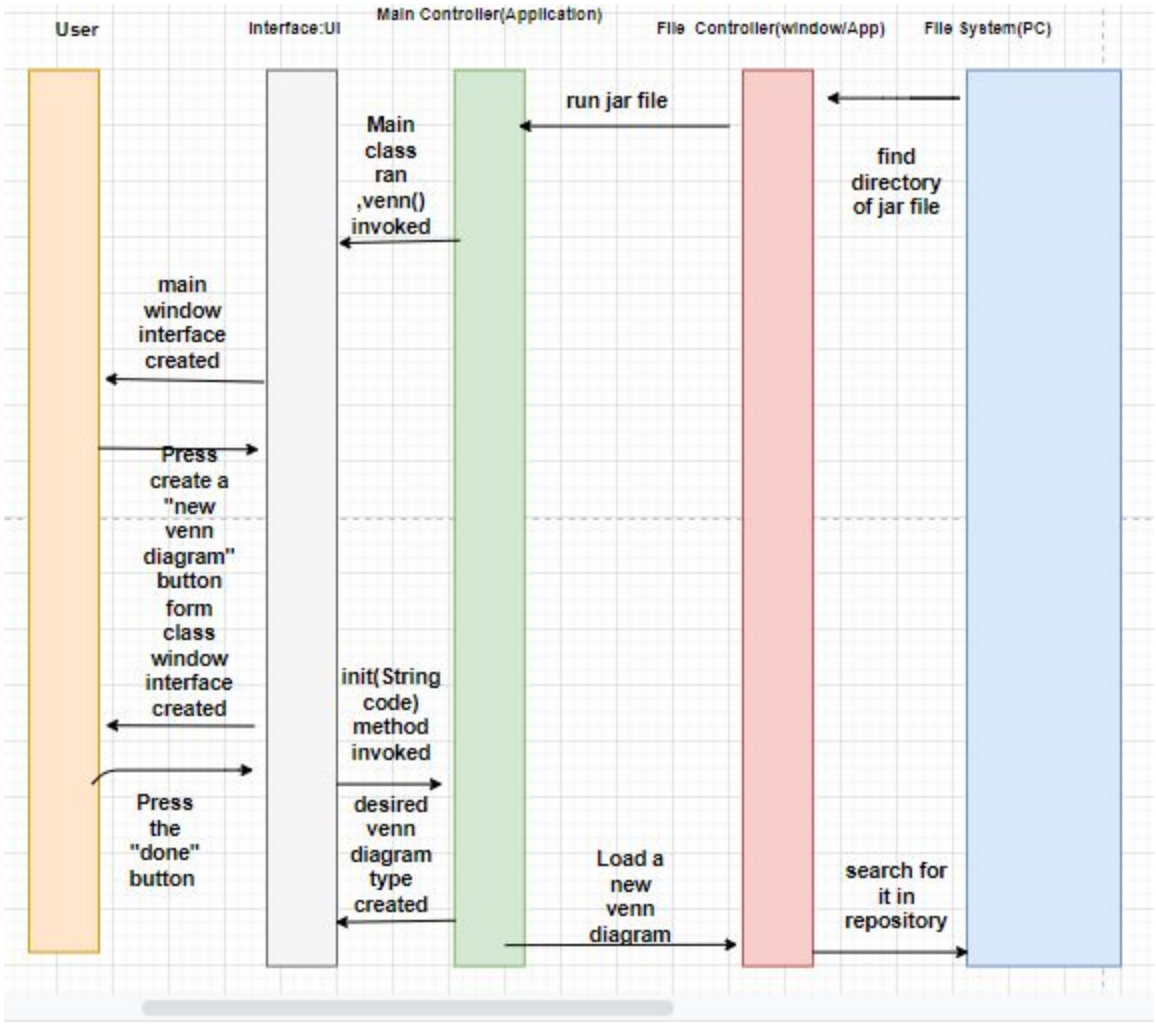
One example of these methods being put to use is how we have a pane in a certain location on the window called “panel”. Through trial and error we found the perfect coordinates being “panel.getPrefWidth() +280 “and “panel.getPrefHeight() + 400”. Then we did panel.setLayoutX(panel.getPrefWidth()+280) and panel.setLayoutY(panel.getPrefHeight()+400). The problem with this solution is that if the window is minimized, our precise trial and error attempts are in vain, and if we wanted the window to be smaller than 280 by 400 there would be major issues. This is why we plan on using more built-in methods like relocate(),which relocates where you want your widget to be in the window. Also we can use Pane instead of BorderPane. BorderPane has methods in its library that help the window be more responsive, such as setTop(),setCenter(),setRight() and setBottom(). We can also add that BorderPane within a created ScrollPane and we can set two ScrollBar widgets on the left and bottom sides to help with navigation.These methods we have discussed are not in the class diagram because we decided to only include functions that call other methods and that are eventListeners in the diagram.

We would've made “labels” and “circles” similar to how we incorporated textboxes in the program. Two additional classes, labels would have its own textarea to write in and stimulate it through a button. Circle can have an unlimited option and we would implement Circle Drag to the class so you can position your generated circles in a position you want. We would have created a load file option. This would have been done by reading the file that was saved priorly line by line and generating an venn diagram pane from that.

Class Diagrams:



Sequence Diagrams:


User
Interface:UI
Main Controller(Application)
File Controller(window/App)
File System(PC)
run jar file
Main class ran ,venn() invoked
find directory of jar file
main window interface created
Press create a "new venn diagram" button form class
window interface created
init(String code) method invoked
Press the "done" button
desired venn diagram type created
Load a new venn diagram
search for it in repository

User manual

In this section we will explain how to use the UI and show how our UI satisfies the requirements specified by the client. We will include screenshots of an example of what to expect if the code is run. You are expected to have the software required to run .jar files such as an up to date JDK or JRE that can run executable .jar files.

This section describes the features of the final prototype for the Venn Diagram Project as well as if they are implemented yet or not.

We satisfy the needs of the clients by:

- Allowing freedom to add/delete text boxes and change names of aspects which allows the user to create a customizable venn diagram.
- Allow the user to have 2 or 3 circles.
- Be able to drag text boxes into different sections of the venn diagram.

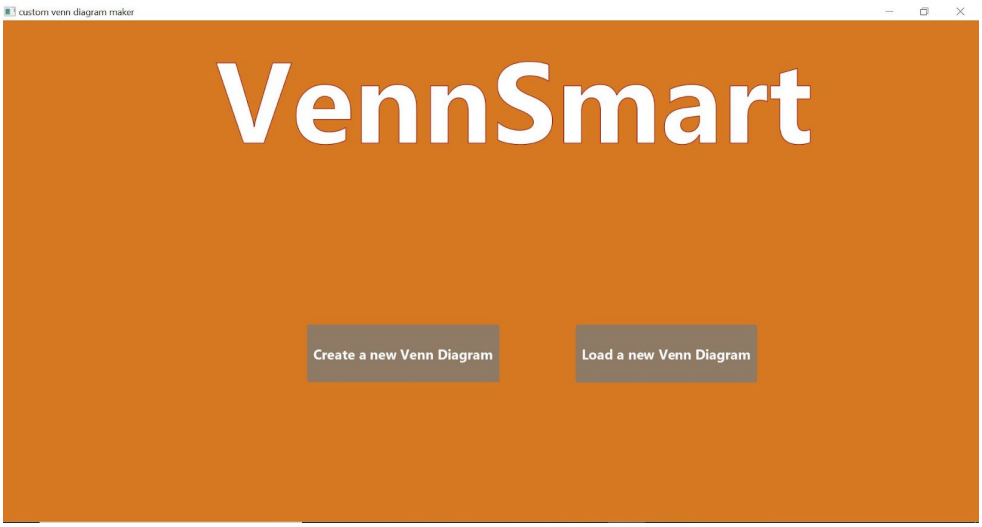
Features of the Venn Diagram Project

Feature	*Description*
Screenshot	Saves pane as an image
Drag and Drop Drag	Drag and drop text boxes, label and circles
Delete Items	Delete text boxes
Change Text	Change names of textboxes, labels
Save Venn Diagram	Save as a file that can be loaded into UI
Load Venn Diagram	Restart UI from last place user left off

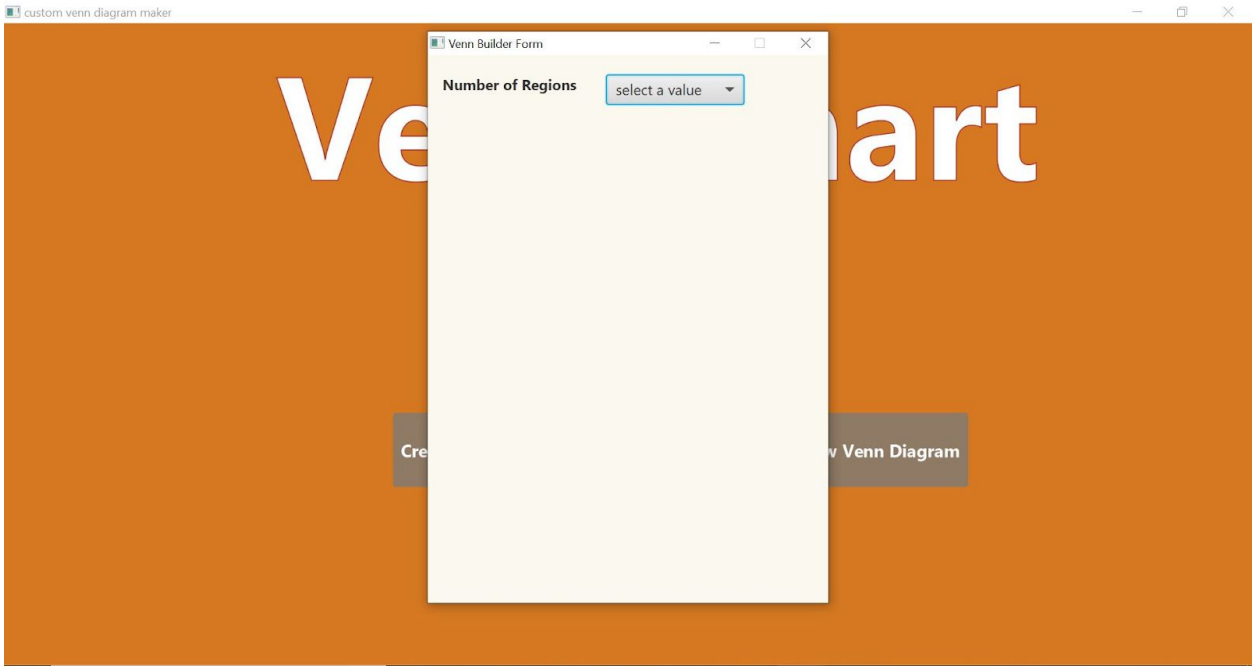
Installation and user instructions:

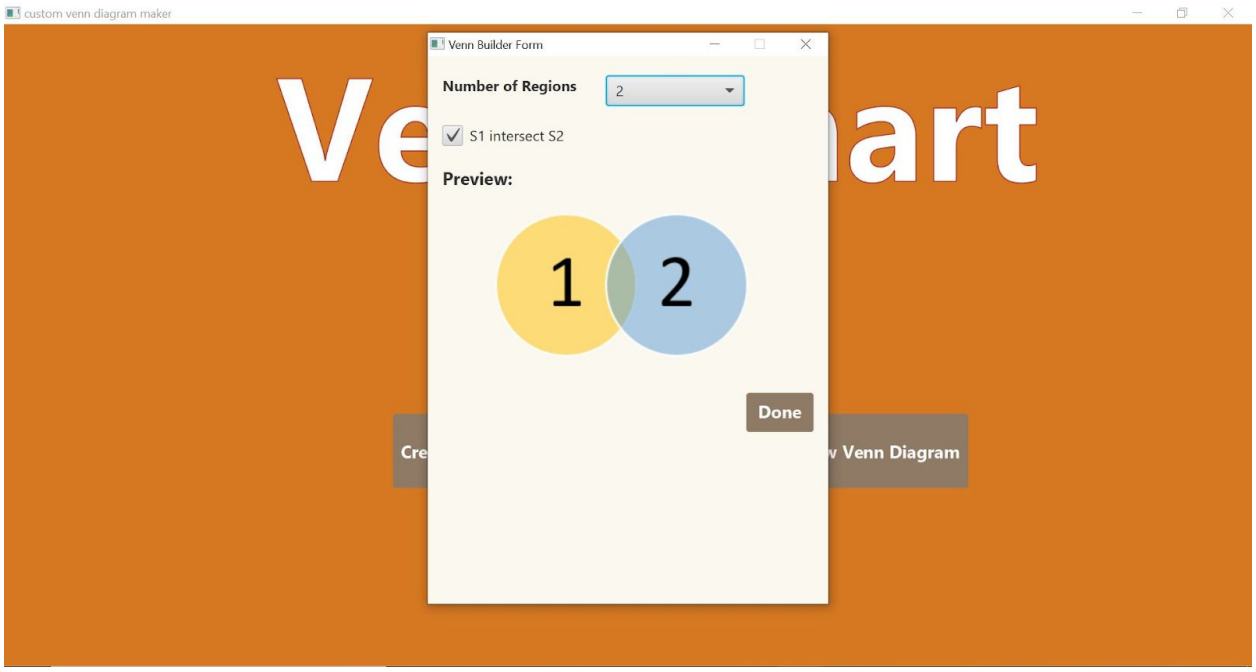
1) Download the executable jar file from the final release section of our groups github.

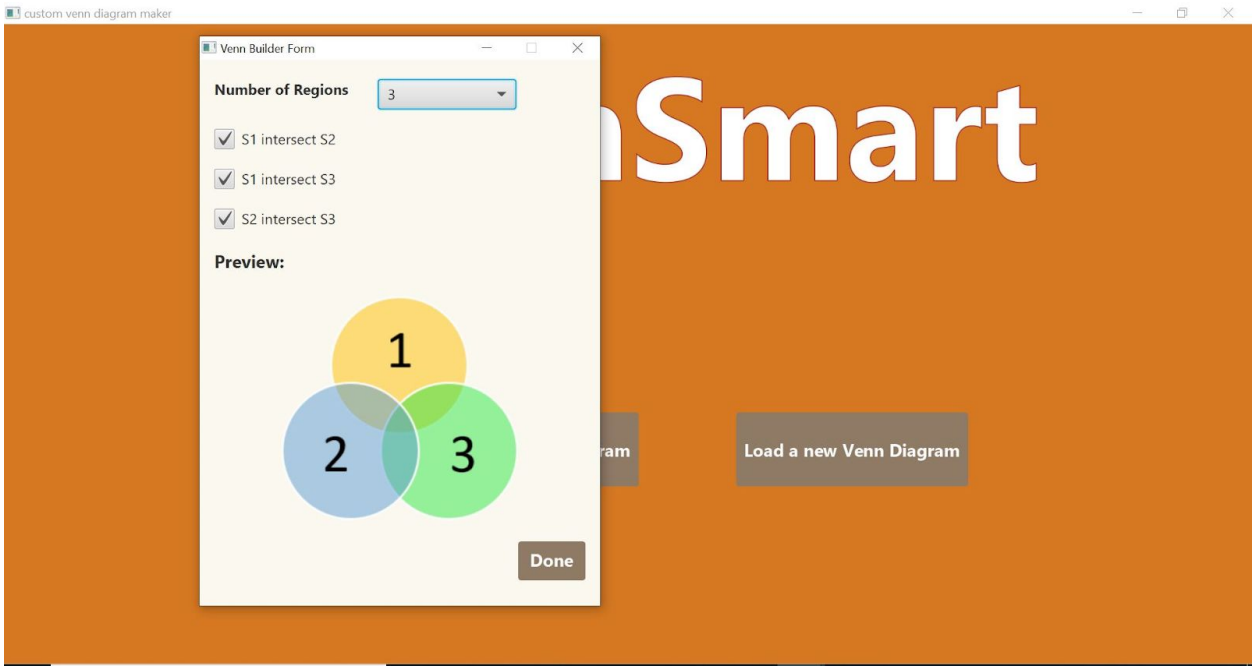
2) Once the program is downloaded, run it through the command line with the following command: `java -jar smart_venn.jar` (make sure you are in the same directory).
3) As a backup plan if the .jar file isn't able to executed on your system you can run it through eclipse or visual studio code you must first download our files from the vF branch of the group github and New -> Import -> General -> Existing projects -> Archive File and then click the .zip folder with the files. You must make sure that JavaFX library is added if not you must add it to the build path. Then you can open the Main.java file found in the Venn package and press the run button.
4) After successfully running the program you will see the following



5) If it's your first time running the program you must select "create a new venn diagram".
6) You will then be prompted to select how many circles you want and how you would like them to be intersected as seen below:


custom venn diagram maker
Venn Builder Form
Number of Regions
select a value
Cre
v Venn Diagram


custom venn diagram maker
Venn Builder Form
Number of Regions
2
S1 intersect S2
Preview:
1
2
Done
Cre
v Venn Diagram



7) After selecting the configuration you would like you would be taken to the Venn Diagram

8) You can then start adding text boxes and customizing each aspect of the pane.

Testing Document

Test Cases Run:

The test cases run similarly based on limited actions within the creation of the venn diagram but really spread apart from how much more options become available. Upon creating the venn diagram the user is met with listed options for themselves. Selecting either 2 or 3 regions will pop open a preview with all checks already selected for the users. From preference the user may decide on deselecting some checks and the preview changes based on the selection. If the user deselects them all, the creation cannot be completed. If the user wishes to close the creation window, he/she may do so and is still capable of reopening the window from the start up. Each version of the venn diagram that can be created works as intended within the test cases.

Then there includes the testing for adding the information to the venn diagram. This includes: inputting the different blocks (which are seperated with an enter key), converting the text blocks into movable blocks and moving the blocks to the venn diagram. Starting off light, inputting short and small numbers of text work quite well when converting them to moving blocks. Dragging them to the venn diagram works as intended. There lies an issue when it comes to going to bigger extremes with the test cases. Longer sentences and such don't fit and are not readable to the user when converted to moving blocks. Converting a mass number of text blocks to moving blocks isn't possible as some are not converted in the end. Lastly, adding too much to the venn diagram proves problematic when space becomes an apparent issue.

Test Cases Derive & Implement:

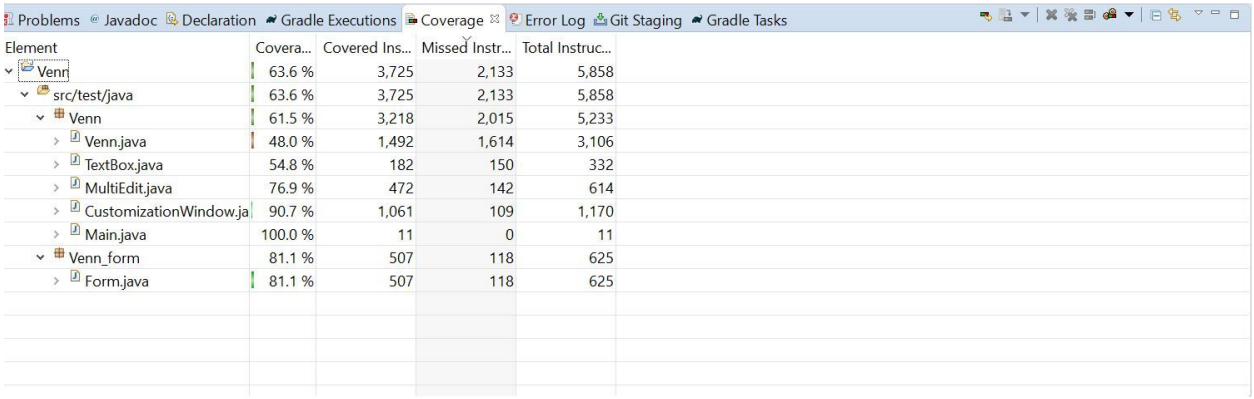
The test cases were derived from expectations on what the client requested. From the previous meetings there were several requests and improvements to what the clients wanted within the program. Such improvements included quality of life additions within the creation menu of the venn diagrams: where the user is still able to decide on the number of colliding circles but the preview is changed to see available options first and further deselecting to preference. Handling the cases of how the text options were added to the venn diagram were also included in the clients request. Upon looking at the client's request there was an additional requirement for handling odd cases. Due to the freedom of adding text options to the venn diagram, there needed to be insurance that it could run well from mass loads as well as little.

Reasonings for Sufficient Testings:

The reasoning that this is sufficient testing for the program is because the client can have a very open ended experience. The program enables the ability to be freely used for a multitude of purposes and is made to be extremely easy to use. The test cases were made to represent that. A major note that is in mind is how easy it would be for a child to use the program without even knowing what it is. We've decided that these test cases are sufficient because it covers most of the features that we wanted to have included in the end product.

Testing Metrics:

We used the coverage function in java to test that we had no unused methods. The test coverage shows that the program runs with all the implementations. What's missing is a method to trace back steps and include the different venn diagram variations in the same instance of the program. Running through each variation will cover the entirety of the code.



Element	Covera...	Covered Ins...	Missed Instr...	Total Instruc...
Venn	63.6 %	3,725	2,133	5,858
src/test/java	63.6 %	3,725	2,133	5,858
Venn	61.5 %	3,218	2,015	5,233
Venn.java	48.0 %	1,492	1,614	3,106
TextBox.java	54.8 %	182	150	332
MultiEdit.java	76.9 %	472	142	614
CustomizationWindow.ja	90.7 %	1,061	109	1,170
Main.java	100.0 %	11	0	11
Venn_form	81.1 %	507	118	625
Form.java	81.1 %	507	118	625

-