

Testing Document

Group 2 EECS2311

February 23, 2020

Marco Carusoni

Arjun Kaura

Joseph Spagnuolo

Jeffrey Walker

Supervised by: Dr Vassilios Tzerpos



Various sections such as Left Pane, Right Pane and Toolbar are labeled in red. These phrases will be used in this document to refer to these elements shown in the image.

Test Cases:

Testing Drag and Drop

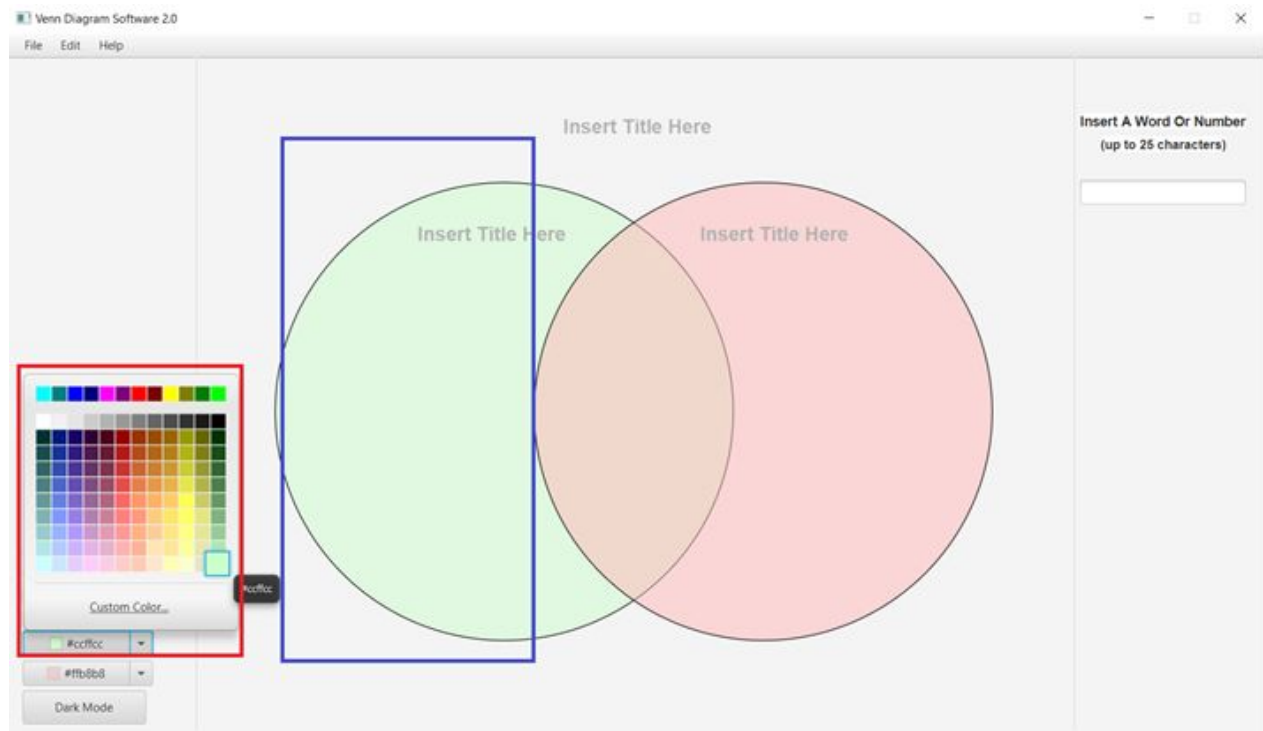


One other feature that was tested is adding text elements onto the venn diagram. There is an invisible grid over the two venn circles and when text is dragged onto the grid it snaps to the cell on that grid which the mouse is hovering over. It was tested to see what would happen when one text object is dragged and dropped over another than is already on the grid. Another important thing we tested is the situation in which the user begins to drag but doesn't drop on the grid. In this situation the text element is awaiting to be placed at whichever cell the mouse hovers over and no other text element can be dragged while another is still selected. The moment the mouse hovers over the Venn diagram it will snap to the invisible grid.

The limit of the number of text objects on right pane

We decide on the maximum number of text objects the program will allow to be made and placed on the Right Pane. Junit testing can be used to check whether, at the end of continuously adding text objects, the program will have stopped adding any more text objects. We will assert that the number of text objects are less than or equal to the maximum number of allowed text objects, which we declared as a constant.

Changing the colour of Venn Diagram



Colour buttons on the bottom left pane are used to change the venn diagram circle to a specified colour by the user. This test would require each individual circle to change to its specified colour, and not the entire venn diagram. This Junit test can check the state of each individual circle object. If the circle's variable that represented a colour code is equal to the specified colour code then the Junit test would pass. It would be best to make each individual circle a unique colour code to distinguish that both are correct, this would also increase test coverage.

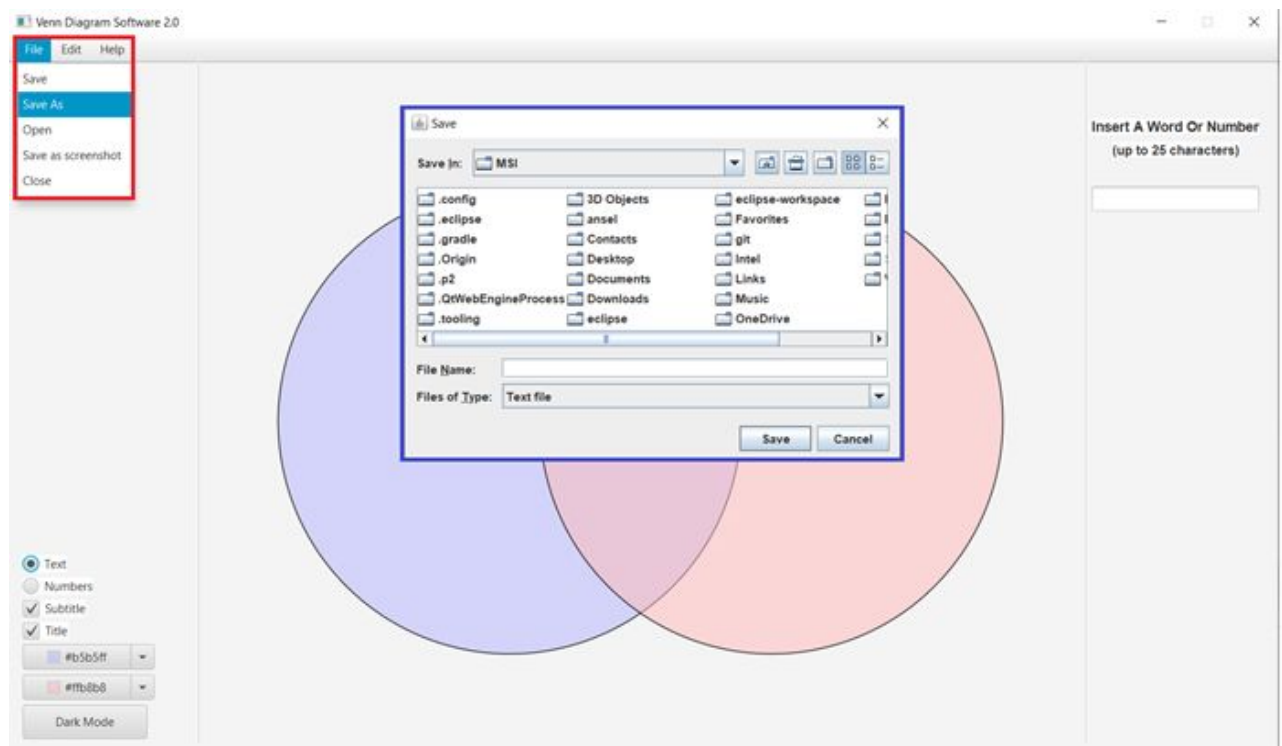
Functionality of Various Toggle Buttons/Check Boxes

Toggle buttons on the Left Pane are used to switch from a text based venn diagram to a numbers based and vice versa. We test to see if, when the Numbers option is selected, that the user can no longer enter text in the text field on the Right Pane. We use Junit testing to check if no mouse input is recognized on the right pane text field while Numbers option is selected, as the user should not be able to write and create text elements, and so mouse input should not be taken.

We also ensure that three new fields appear on the venn diagrams, one for each section of venn diagram where the user can enter the numerical values.

As for the check boxes, they make the title and subtitles appear when the respective box is checked. We also test to see if this operation happens correctly.

Saving and Opening File



The user can save the venn diagram as well as open an existing one. When saving a file the data that is saved is the text elements the user creates to place in their venn diagram. So we ensure that when the user saves, the text data is saved in a .txt file, named whatever the user named their save as, and that the txt file contains a list of all the text elements made earlier.

Then, when the user opens a venn diagram, they select that text file. We ensure that the result is a venn diagram which has all the text elements saved previously.

How these test cases were derived:

Testing Drag and Drop:

The limit of the number of text objects on right pane:

It is important to account for the situation when the user has many text elements and whether they can all fit on the Right Pane.

Changing the colour of Venn Diagram:

The application should allow users to change each individual circle to a specific colour. It is also important that when testing, each colour is unique to check that both circles are being accounted for.

Functionality of Various Toggle Buttons/Check Boxes:

We need to test whether these buttons do their intended purposes. Such as pressing the subtitles checkbox -- we need to ensure that the subtitles appear when it is checked, and are removed when it is unchecked.

Saving and Opening File:

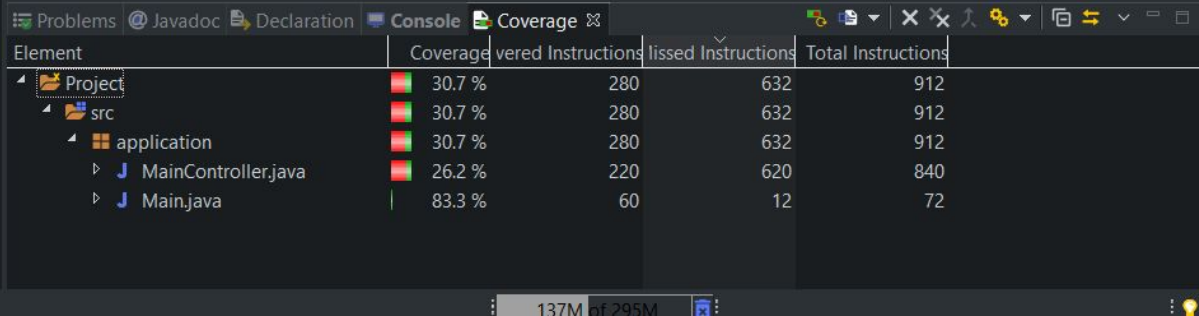
The application should be able to save the exact state of where the user left the program. If the user would like to start where they left off, they can open a saved file to load their previous state.

Why these test cases are sufficient:

We believe these test cases are sufficient because all major regions of the application are tested. This includes the toolbar, the left pane, and the right pane. All functions of the application were tested. We tested the situations where there could be overflow of data, user drags and drop in certain ways/places, and other situations that might cause errors. So we did not simply test whether certain features worked when they are used in the way expected, but we tested the scenarios where users try those features but attempt them in unexpected ways.

Test Coverage:

The test coverage is 30.7% for the overall project.



The screenshot shows the 'Coverage' tab in an IDE. It displays a tree view of the project structure with columns for 'Element', 'Coverage', 'Covered Instructions', 'Missed Instructions', and 'Total Instructions'. The 'Project' node has a 30.7% coverage. The 'src' directory also has 30.7% coverage. The 'application' sub-project has 30.7% coverage. The 'MainController.java' file has 26.2% coverage, and the 'Main.java' file has 83.3% coverage. The status bar at the bottom shows '137M' and '295M'.

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
Project	30.7 %	280	632	912
src	30.7 %	280	632	912
application	30.7 %	280	632	912
MainController.java	26.2 %	220	620	840
Main.java	83.3 %	60	12	72