# EECS 2311

# Venn Project

# Group #17

# Testing Document

# Description of test cases run and implemented

We have implemented various test cases in order to test and fool-proof our system against potential failures. Since this is an interactive GUI application, we would not be testing out the application through Junit individual test cases. Our test cases have been run manually by accessing each GUI component and programmatically using them.
The following test cases have been implemented

**Upper bound on the number of text box words**
Our first testing criteria was that we should not allow the number of words to exceed the 25 character limit, put in place for visibility purposes. Any sentence under 25 characters was allowed for use within the application. Conversely, any sentence exceeding 25 characters was rejected for use with an appropriate error message.

**ScreenShot button capability**
Testing of the Screenshot button was thoroughly undertaken so that the user could take a real-time snapshot of the Venn Diagram while still in use. Appropriate tests were carried out in order to ensure the snapshot of the system has been saved in the same directory location as the JAR file.

**Screen Resizability**
Another feature tested was the ability of our application to be adaptive under various screen sizes and resolutions. Tests were carried out to make sure the application content and visuals are preserved even after going through various resizing in different environments.

**Text Box Anchors**
A neat functionality added to our application is the ability to "anchor" text boxes neatly close to the circles. This feature can be turned on or off at the discretion of the user. Use cases have been tested in order to make sure that the Anchor button works as per the requirements of the client.

**Runnable JAR File Testing**
We have also thoroughly and extensively tested running our application as a JAR file in various OS platforms, such as Windows, Linux, and MacOS. Detailed simple descriptions have been provided in the User Manual for running the JAR file as required.

**Color, Text Box, and Title modifications**
Finally, the core features such as carrying out modifications to the application were tested out and performed exactly as desired for a user. Boundary cases were tested with expected functionality. We expect more rigorous testing in our future releases.

# How test cases were derived

Derivation of the above test cases was done through critical thinking and analysis of what could go wrong during the running time of the application. Careful consideration and logical expectations went into what the core functionality of the app is and what critical features we are providing to our client

# Why we have sufficient testing

Since we have already tested the execution of our JAR file in multiple operating system platforms, as well as tested the resizability of our application on various resolutions, and general functionality of the app, the test cases described and considered above would be sufficient for testing our Venn Application, since we require a few critical components to run as smoothly as possible for the user.

# Testing Coverage Metrics

Our Venn Diagram Application has been thoroughly and excessively tested around critical use cases and boundary conditions. We have made use of the metrics of code coverage to get an idea of how much percentage of our release code is being executed while the application is still in use.

For our code coverage purposes, we have used the EclEmma extension provided by Eclipse. As shown in the figure below, we have calculated and achieved a testing coverage of nearly **98%**, which is typically considered a good high coverage for large systems.

Some of the code which did not execute throughout the runtime of the application is included as part of our implementation for debugging purposes. Although our application is not as large as typical industry projects, our GUI Venn Application satisfies our expected testing coverage.

Navigate  Search  Project  Run  Window  Help

Main.java ✕

```java
752        });
753
754        Text right = new Text("right");
755        right.setFont(new Font(12));
756        right.setStroke(Color.BLACK);
757        right.setTextAlignment(TextAlignment.CENTER);
758        //right.getText().length()*5/2 : text half length
759        right.layoutXProperty().bind(pane.widthProperty().divide(5.0/3.0).subtract(right.getText().length()*5/2));
760        right.layoutYProperty().bind(pane.heightProperty().divide(2.0).subtract(circleL.radiusProperty()).subtract(5));
761        right.setOnMouseClicked(new EventHandler<MouseEvent>() {
762            @Override
763            public void handle(MouseEvent event) {
764                if (event.getButton() == MouseButton.PRIMARY) {
765                    TextInputDialog dialog = new TextInputDialog("Enter right circle title");
766                    dialog.setTitle("Change text");
767                    dialog.setHeaderText("Enter to change text");
768                    dialog.setContentText("Please enter some text:");
769                    String result = dialog.showAndWait().get();
770                    right.setText(result);
771                    right.layoutXProperty().bind(pane.widthProperty().divide(5.0/3.0).subtract(right.getText().length()*5/2));
772                }
773            }
774        });
775
776        //Right circle color picker label
777        Text cpR = new Text("Right circle color : ");
778        cpR.setFont(new Font(12));
779        cpR.setStroke(Color.BLACK);
780        cpR.layoutXProperty().bind(cpL.layoutXProperty().subtract(100));
```

Problems  @ Javadoc  Declaration  Console  Coverage ✕

Main (Feb 23, 2020 6:30:26 PM)

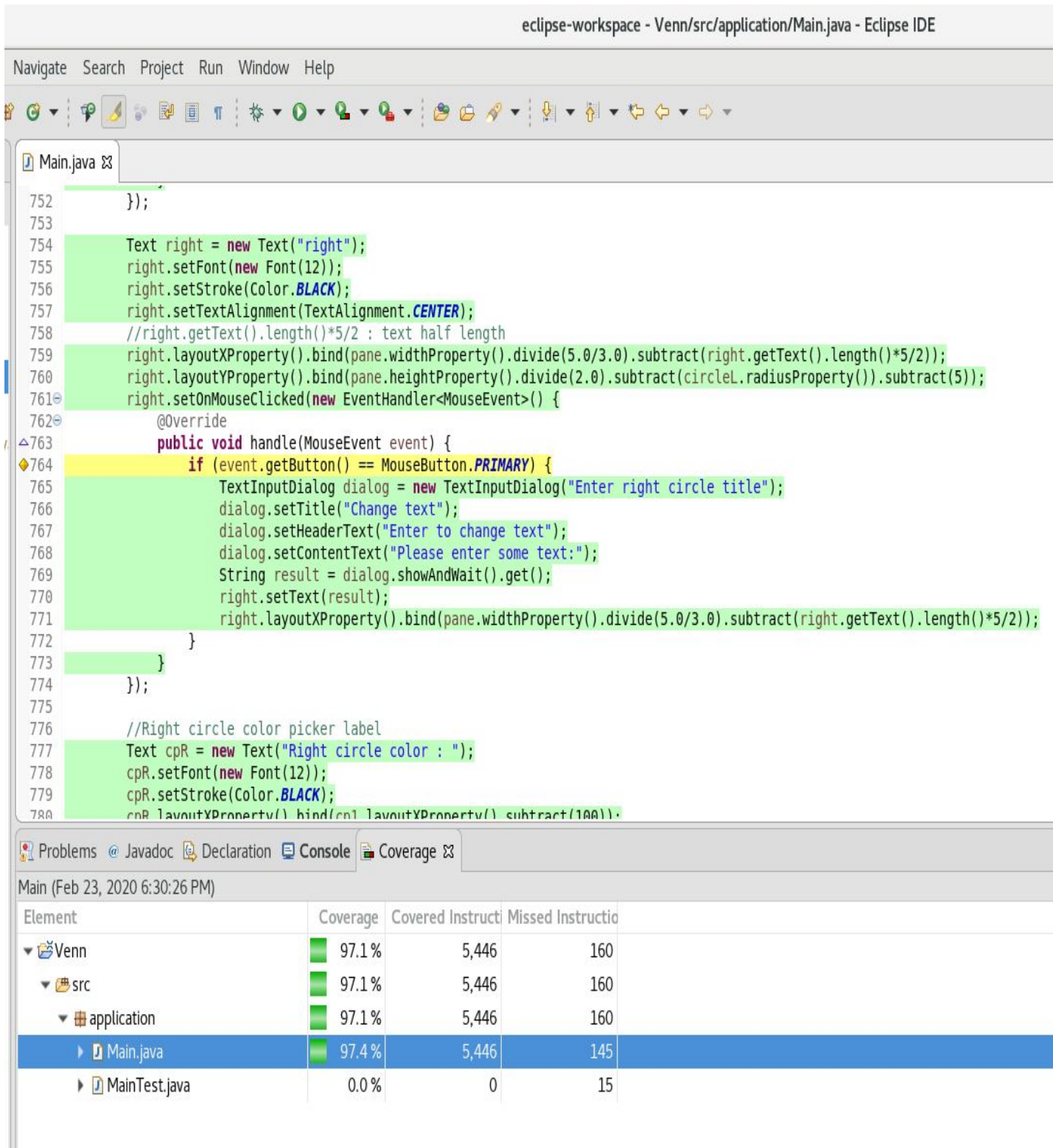| Element | Coverage | Covered Instructi | Missed Instructio |
|---|---|---|---|
| ▼ Venn | 97.1 % | 5,446 | 160 |
| ▼ src | 97.1 % | 5,446 | 160 |
| ▼ application | 97.1 % | 5,446 | 160 |
| ▶ Main.java | 97.4 % | 5,446 | 145 |
| ▶ MainTest.java | 0.0 % | 0 | 15 |

**Fig: Code Coverage of 97.4% has been achieved for our Venn GUI Application.**