EECS 2311
Winter 2020

Venn Project

Group #17
Testing Document - Final Submission

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. This interactive GUI application was tested both manually and through individual JUnit 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.
Example test case:

```
testCharacters() {
        assertTrue(textBox.getText().length() <= 25)
}
```

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 chosen directory location according to the user's choice.

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 and operating systems.

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.

Colors, Text Boxes, and Title modifications
The fundamental modification features such as carrying out modifications to the circle, background, text boxes, and button colors/titles were tested out and performed exactly as desired for a user. Boundary cases were tested with expected functionality.

Import/Export/Compare functionality
One core feature of our Venn application was ability to save and export our diagrams as metadata in a simple text file, which can later be used to import and compare with any future Venn diagrams. We have extensively tested, saved, compared, and reloaded the state of our diagram multiple times to provide complete functionality for the user.

Sliding Menu functionality
A neat aesthetic feature in our Venn application is the addition of a sliding menu, which has most of the important buttons included. We have tested the menu through the use of dragging the cursor to the left of the screen which will prompt the menu to appear.

Undo/Redo functionality
Another core feature of our Venn application is the ability to undo and redo our actions. This is a handy feature that will help the user to fix any unwanted actions or even redo actions accidentally undone. This feature was thoroughly and extensively tested through manual operation and events such as changing colors, titles, backgrounds to adding/deleting/dragging text boxes were undone and re-done to verify correctness.

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 for our final product release.

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 in terms of our final released application.

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. Code coverage provides us with a rough idea of the predictability of our application.
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 **97.5%** in our Main(**VennBase**) application, which is typically considered a good high coverage for large systems tested using code coverage within the software engineering industry.
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.

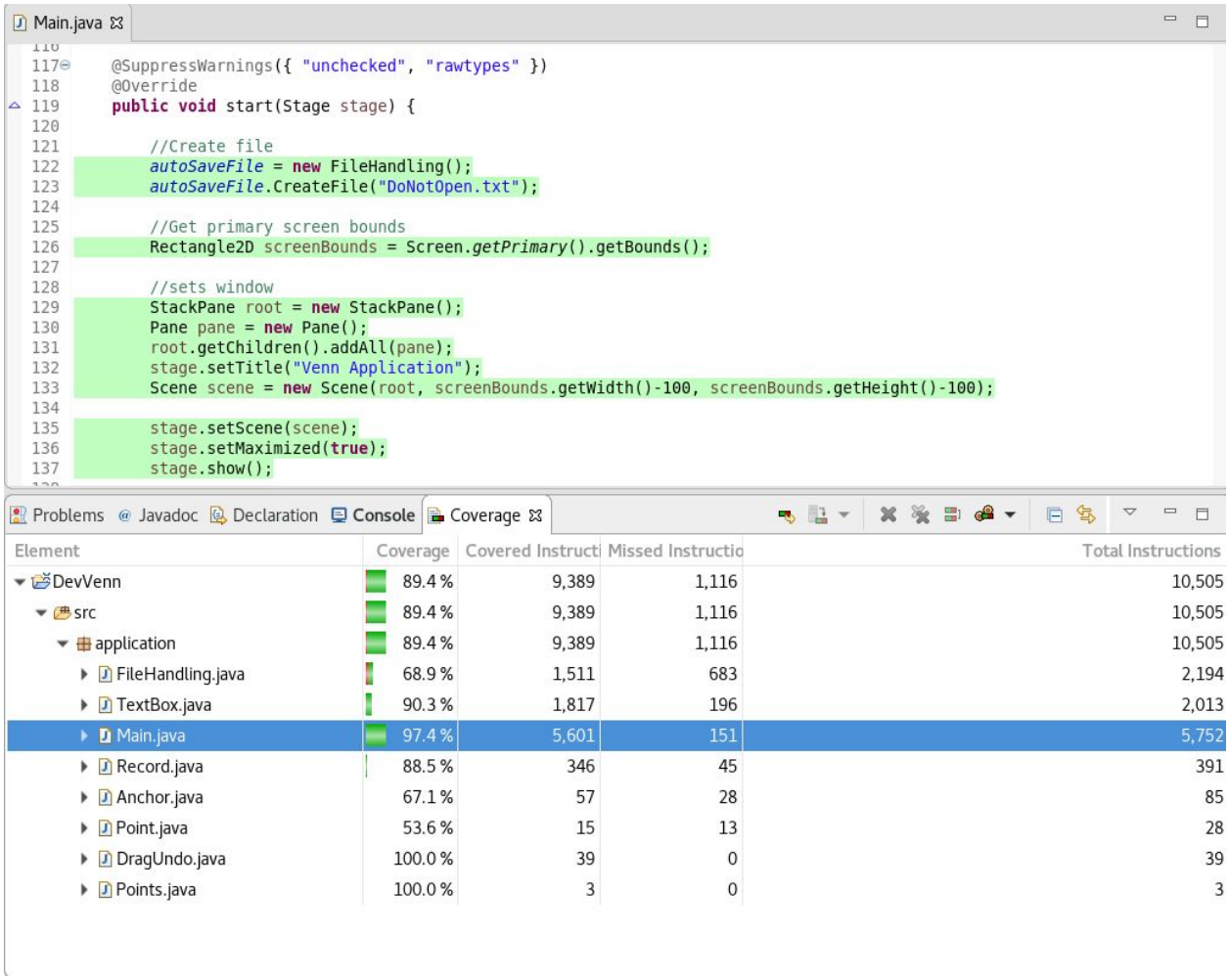

Fig: Code Coverage of 97.4% has been achieved for our Venn GUI Application.