2020

VennCreate Testing Document

AUTHOURS:
CHIDALU AGBAKWA (216337784), SHANGRU LI (214488993),
JIHAL PATEL (216376436), ROBERT SUWARY (21544016)

LASSONDE SCHOOL OF ENGINEERING | EECS2311 Group 18

Contents

Table of Figures

1 Introduction

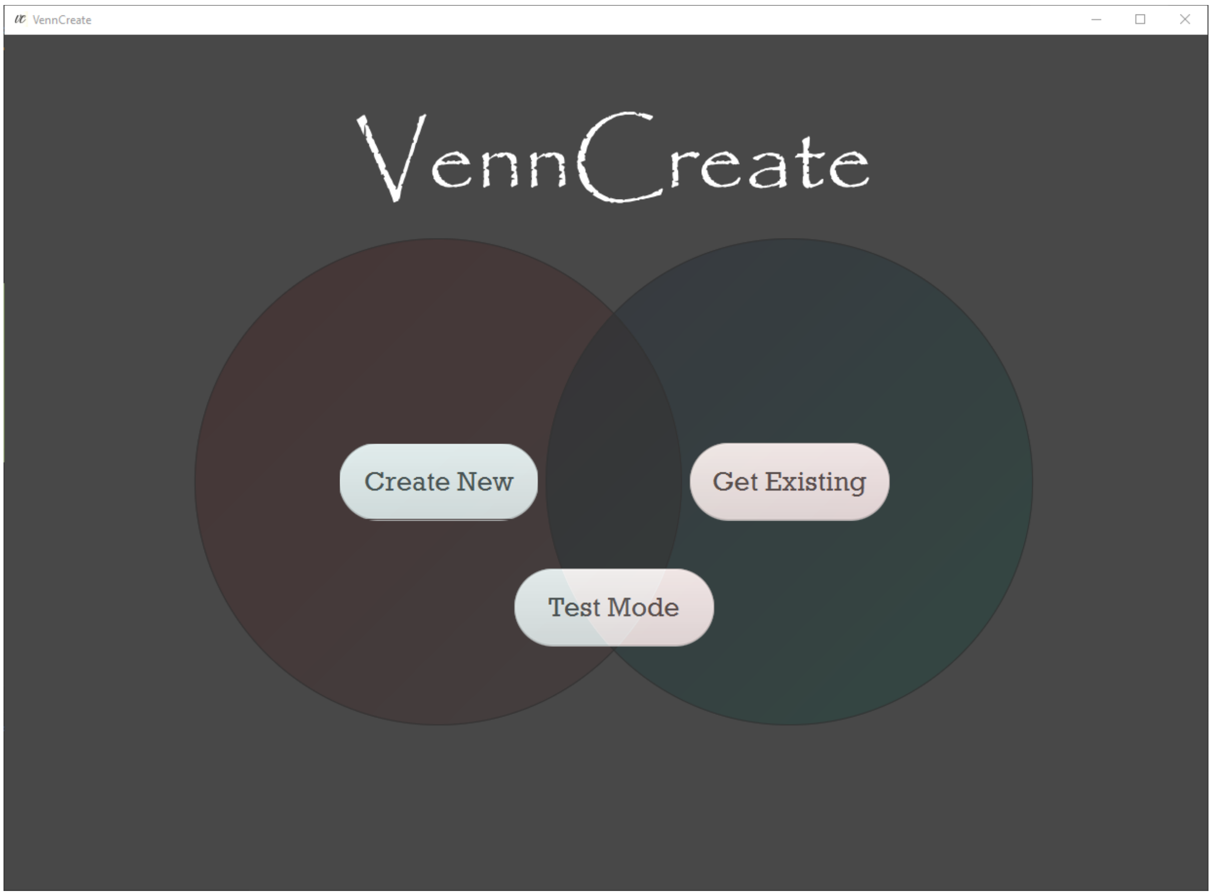


Figure 1 - VennCreate's Home Page

1.1 Testing Strategy

In this test document we will be testing for proper functionality of VennCreate. In order to test our application, we used the TestFX library, and open source library compatible with JavaFX in order to test our code.

1.2 Objectives

We will be testing all aspects of our software in the test cases, including the core functionalities, test mode, undo/redo command, diagram customization etc. Which will be explained in detail in this document.

By passing rigorous series of acceptance tests, we can demonstrate the software's robustness and completeness, delivering quality software to our customer.

2 Test File Structure

Overall, there are in total seven tests file, each test file corresponds to the functionality that it is testing.

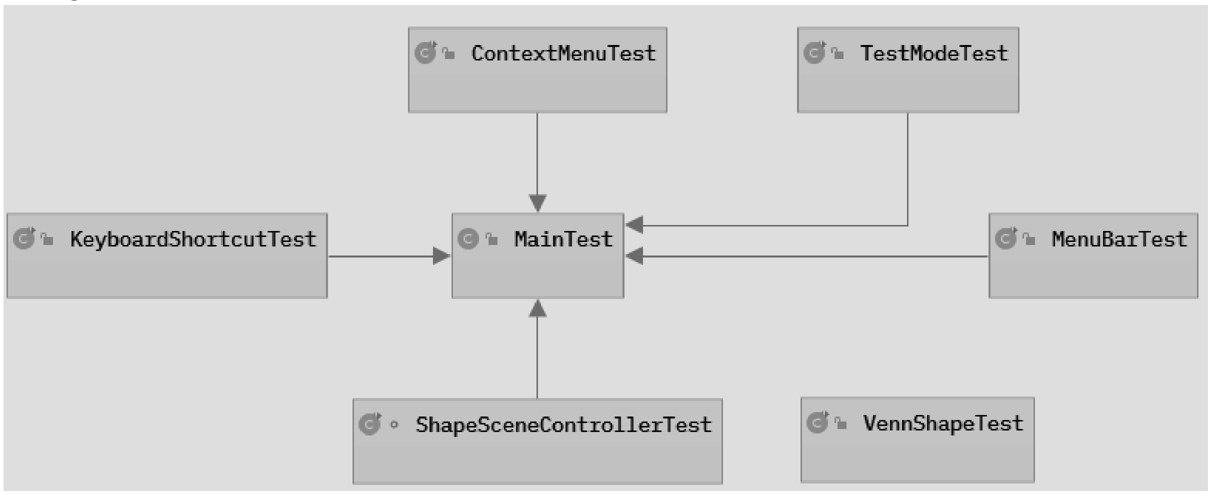
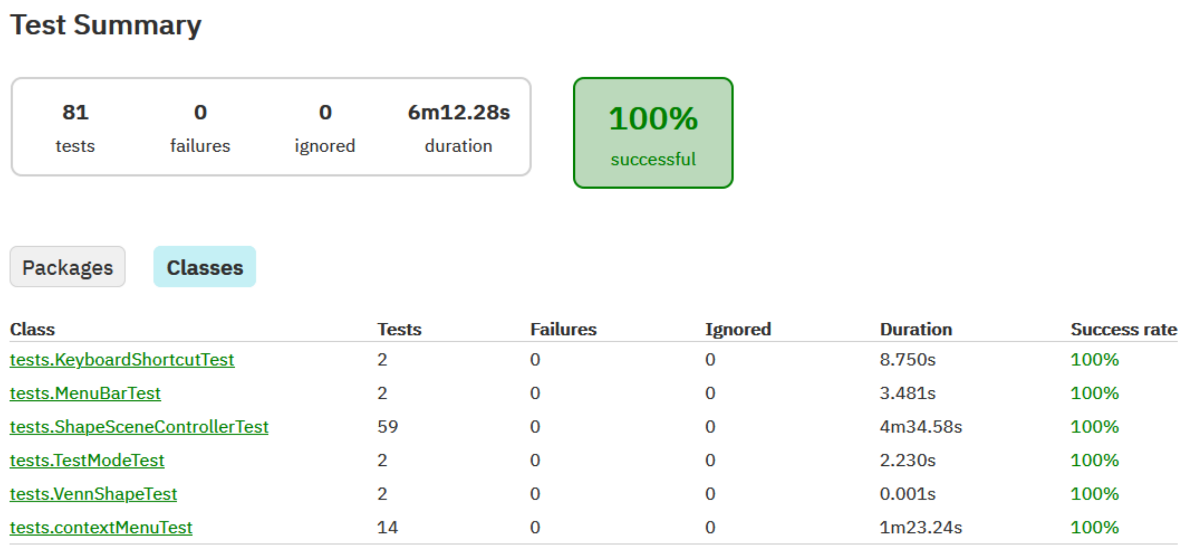


Figure 2 - Relationship between test files

- **MainTest**: the parent class for the front-end acceptance tests, defines functions that would be shared across other test class. For example: method for launching and closing main application, method for finding a specific element and method for entering a specific scene.
- **ShapeSceneControllerTest**: the suite of test cases for shapeScene, the main scene for editing Venn diagrams. This file tests the main features, as well as undo/redo command for the actions in shapeScene. Examples would be: adding a circle and/or a text to the current diagram, changing the colour of the Venn circles.
- **KeyboardShortcutTest**: this file mainly tests the keyboard shortcuts for actions in shapeScene: Ctrl-Z for undo, Ctrl-Shift-Y for redo and so on.
- **ContextMenuTest**: this is where we test the context menus in shapeScene. Context menus includes right click on a diagram text label to remove, edit or adding a longer description for it. And right click on a Venn circle to remove all diagram text labels or toggle hover.
- **MenuBarTest**: tests functionalities of the buttons in menu bar in shapeScene. E.g. create new Venn diagram, add new Venn circle, undo and redo.
- **TestModeTest**: tests the Test Mode of VennCreate, where the users can import a txt file with information that they would like to be tested on and VennCreate will give them a test.
- **VennShapeTest**: Note that all the above test files are front-end tests and inherited from **MainTest**. Whereas this file tests the back-end logic of VennCreate, ensuring the correctness of the whole software.

3 Overall Summary of Testing Results

This is the test summary of all the test files generated by Gradle.



Test Summary

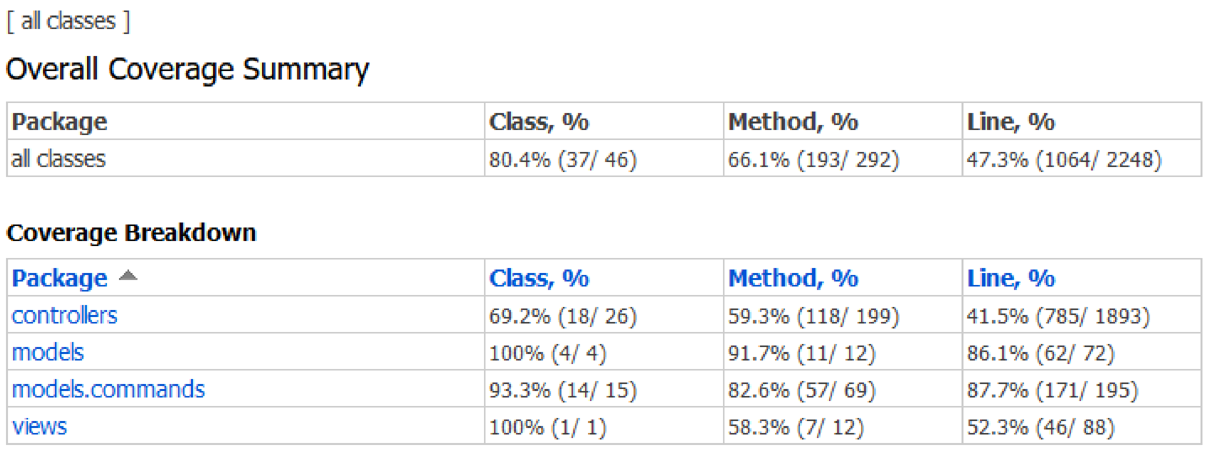
81	0	0	6m12.28s	100%
tests	failures	ignored	duration	successful

Packages Classes

Class	Tests	Failures	Ignored	Duration	Success rate
tests.KeyboardShortcutTest	2	0	0	8.750s	100%
tests.MenuBarTest	2	0	0	3.481s	100%
tests.ShapeSceneControllerTest	59	0	0	4m34.58s	100%
tests.TestModeTest	2	0	0	2.230s	100%
tests.VennShapeTest	2	0	0	0.001s	100%
tests.contextMenuTest	14	0	0	1m23.24s	100%

Figure 3 - Test Results

As illustrated above, a total of 81 tests from 6 files are passed without any failure. Moreover, we also generated coverage report:



[all classes]

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	80.4% (37/ 46)	66.1% (193/ 292)	47.3% (1064/ 2248)

Coverage Breakdown

Package ▲	Class, %	Method, %	Line, %
controllers	69.2% (18/ 26)	59.3% (118/ 199)	41.5% (785/ 1893)
models	100% (4/ 4)	91.7% (11/ 12)	86.1% (62/ 72)
models.commands	93.3% (14/ 15)	82.6% (57/ 69)	87.7% (171/ 195)
views	100% (1/ 1)	58.3% (7/ 12)	52.3% (46/ 88)

Figure 4 - Test Coverage Metrics

We have achieved overall 80% of class coverage, 66% of method coverage. Where the classes in *models* folder are almost fully covered.

With 80% of class coverage and 100% success rate, the test cases can prove the viability and correctness of the actual product.

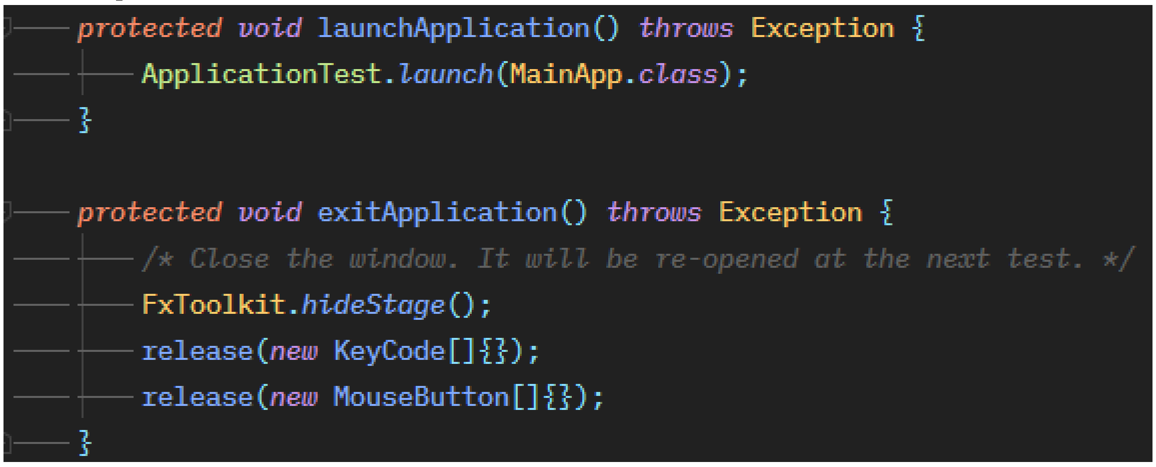
4 Examining Test Files

4.1 Main Test

In order to have a good design and structure for the test classes, we used inheritance for the front-end test files.

MainTest is the parent class for all the test classes, it extends from TestFx's testing framework *ApplicationTest*. The class itself does not contain any test cases, it defines the functions shared across all the child test classes, applying the Single Choice Principle.

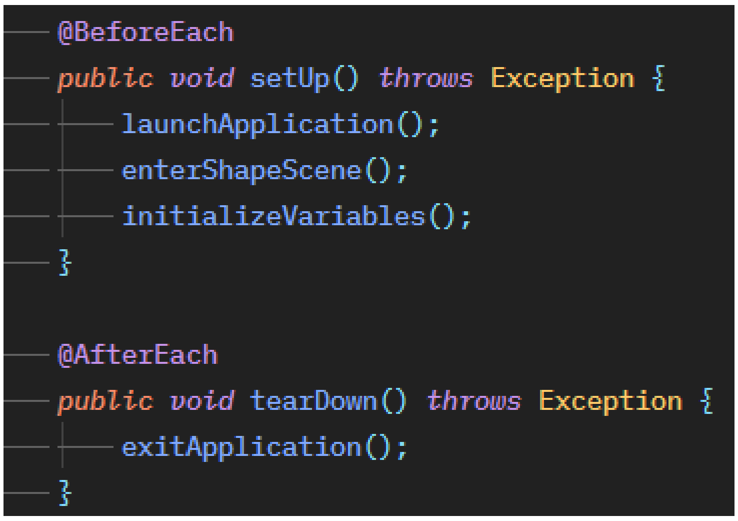
For example, consider these two functions:



```
protected void launchApplication() throws Exception {
    ApplicationTest.launch(MainApp.class);
}

protected void exitApplication() throws Exception {
    /* Close the window. It will be re-opened at the next test. */
    FxToolkit.hideStage();
    release(new KeyCode[]{});
    release(new MouseButton[]{});
}
```

Figure 5 - Main Test Methods#1

launchApplication and *exitApplication* is used in all the child classes' `BeforeEach` and `AfterEach`, in order to launch the application before each test cases and close it after the test case finishes. In **ShapeSceneControllerTest:**



```
@BeforeEach
public void setUp() throws Exception {
    launchApplication();
    enterShapeScene();
    initializeVariables();
}

@AfterEach
public void tearDown() throws Exception {
    exitApplication();
}
```

Figure 6 - Main Test Methods #2

The child class is calling the parent’s function, and this is similar in other test classes. So that we will be able to avoid code duplication, and if we need to change the implementation of any method, we can go to the parent class, obeying the Single Choice Principle.

Document Continues Below…

4.2 ShapeSceneControllerTest

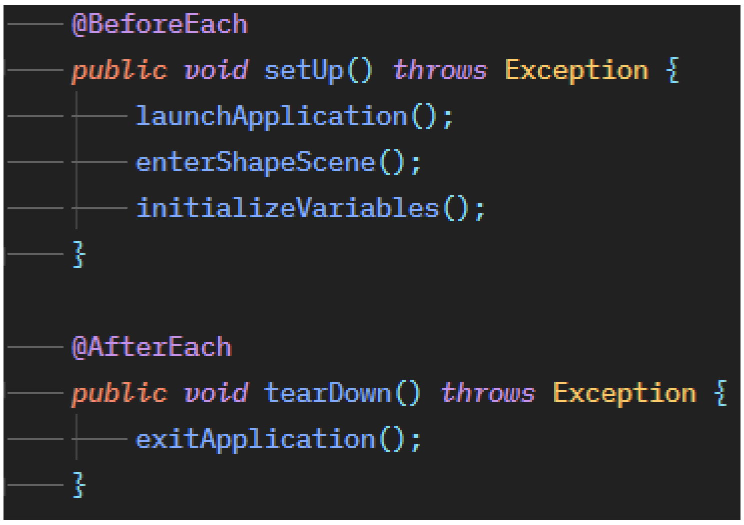
This test file mainly targets the ShapeSceneController class, through TestFX, we will be able to automate the process mouse clicking, typing or other input actions, hence automate the process of front-end testing.

As the shapeScene is the main scene of VennCreate, it is the place where users can cutomize their Venn dirgrams and do most of the operations. Hence, the controller for this scene is the most complex controller among all other controllers.

As a result, we produced 59 test cases to ensure the correctness of this scene. Test cases cover almost all the functionalities in shapeScene, these test cases are demonstrated below.

BeforeEach and AfterEach

Before each test cases in this class starts, we will first launch the main application, enter shapeScene and initialize the necessary variables that will be used in the test cases. And after each test cases, we will exit the application.

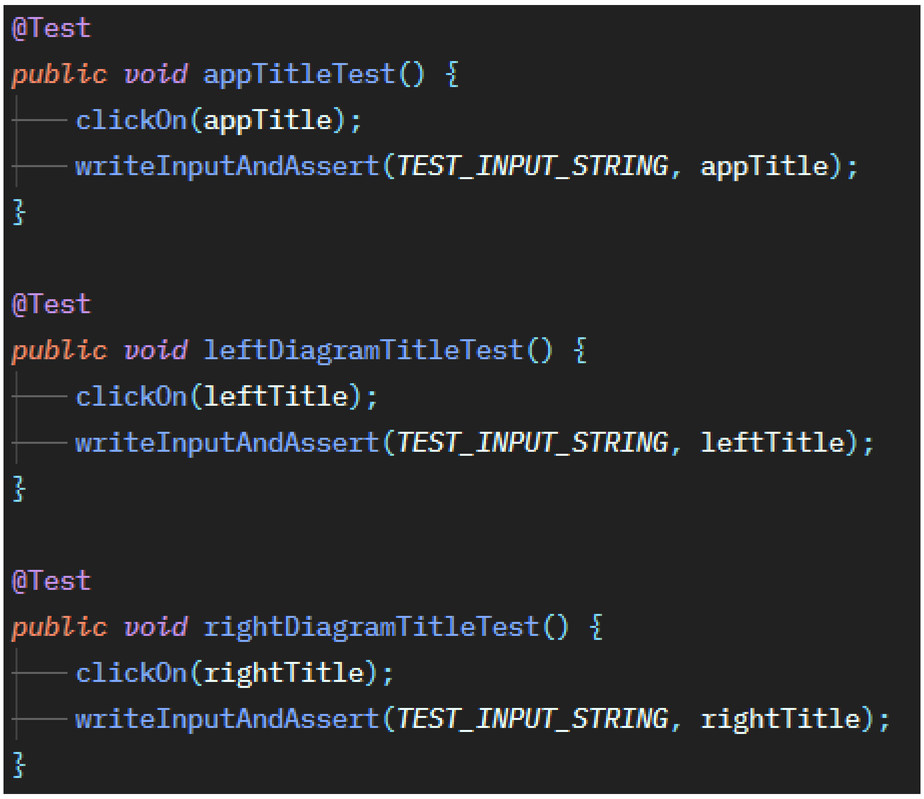


```java
@BeforeEach
public void setUp() throws Exception {
    launchApplication();
    enterShapeScene();
    initializeVariables();
}

@AfterEach
public void tearDown() throws Exception {
    exitApplication();
}
```

Figure 7 - ShapeSceneControllerTest BeforeEach and AfterEach

4.2.1 Tests Concerning Titles

We conducted several tests concerning the input of text into the circle titles. The following is an example of such a test.



```
@Test
public void appTitleTest() {
    clickOn(appTitle);
    writeInputAndAssert(TEST_INPUT_STRING, appTitle);
}

@Test
public void leftDiagramTitleTest() {
    clickOn(leftTitle);
    writeInputAndAssert(TEST_INPUT_STRING, leftTitle);
}

@Test
public void rightDiagramTitleTest() {
    clickOn(rightTitle);
    writeInputAndAssert(TEST_INPUT_STRING, rightTitle);
}
```

Figure 8 - Title Test

Description of Test Case: These test cases mainly targets the titles, including the Venn diagram title, left and right circle title.

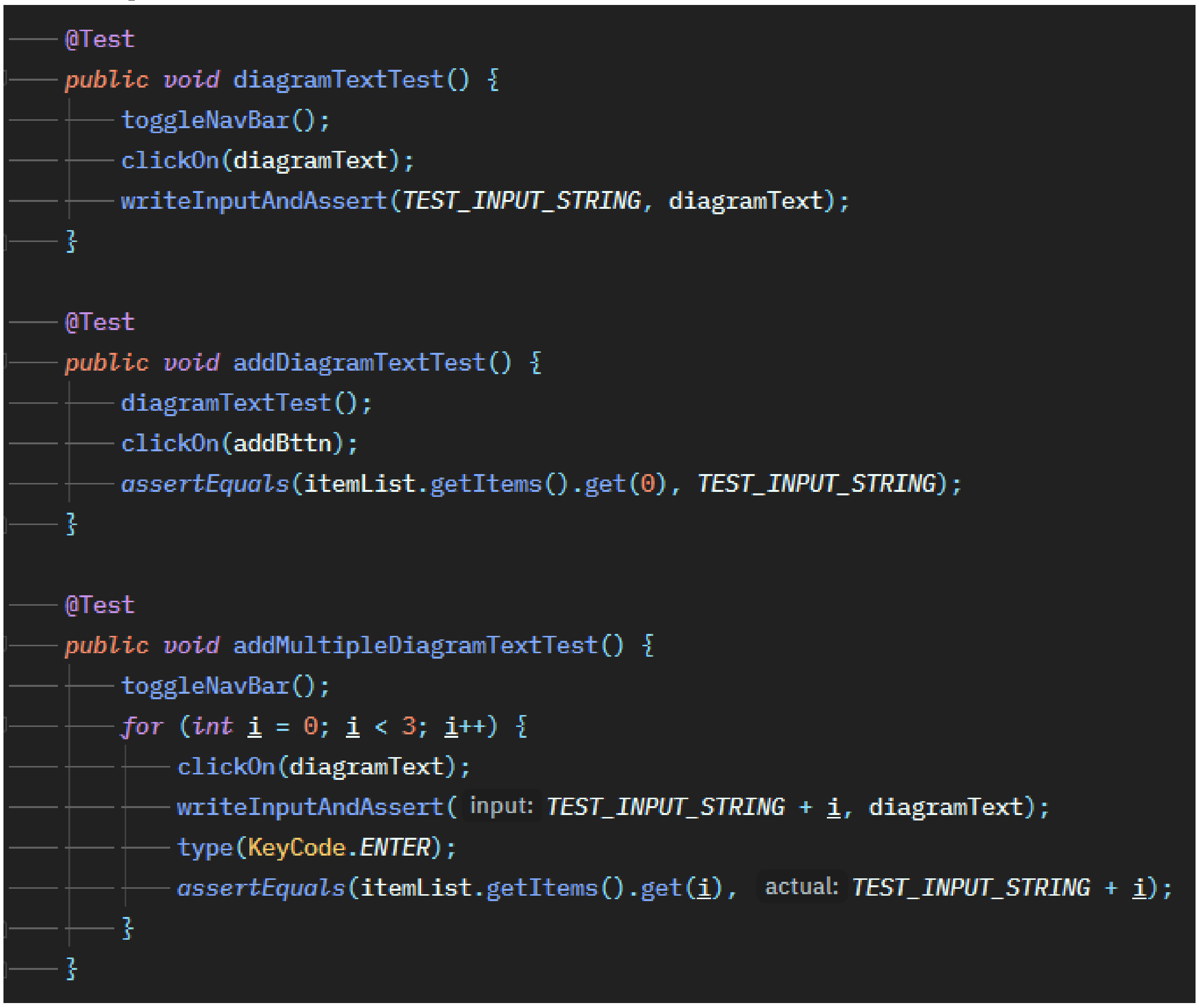
Derivation: These test cases were to ensure no error occurs during the modification of the titles, which is the fundamental feature of this application.

Implementation: These test cases were all implemented using TextFX function *clickOn()*, which will make mouse move directly to the element of our choice and clicks it. Then it will utilize the *writeInputAndAssert()* function defined in the parent class, that writes a sample input to the text filed and assert the content of the text filed.

Test Status: **PASSED**

4.2.2 Tests Concerning Text Field Functionality

We conducted several tests concerning adding draggable text fields into the diagrams. The following is an example of such a test.



```
@Test
public void diagramTextTest() {
    toggleNavBar();
    clickOn(diagramText);
    writeInputAndAssert(TEST_INPUT_STRING, diagramText);
}

@Test
public void addDiagramTextTest() {
    diagramTextTest();
    clickOn(addBttn);
    assertEquals(itemList.getItems().get(0), TEST_INPUT_STRING);
}

@Test
public void addMultipleDiagramTextTest() {
    toggleNavBar();
    for (int i = 0; i < 3; i++) {
        clickOn(diagramText);
        writeInputAndAssert( input: TEST_INPUT_STRING + i, diagramText);
        type(KeyCode.ENTER);
        assertEquals(itemList.getItems().get(i),  actual: TEST_INPUT_STRING + i);
    }
}
```

Figure 9 - Adding Text to Diagram

Description of Test Case: These test cases mainly target adding diagram text.

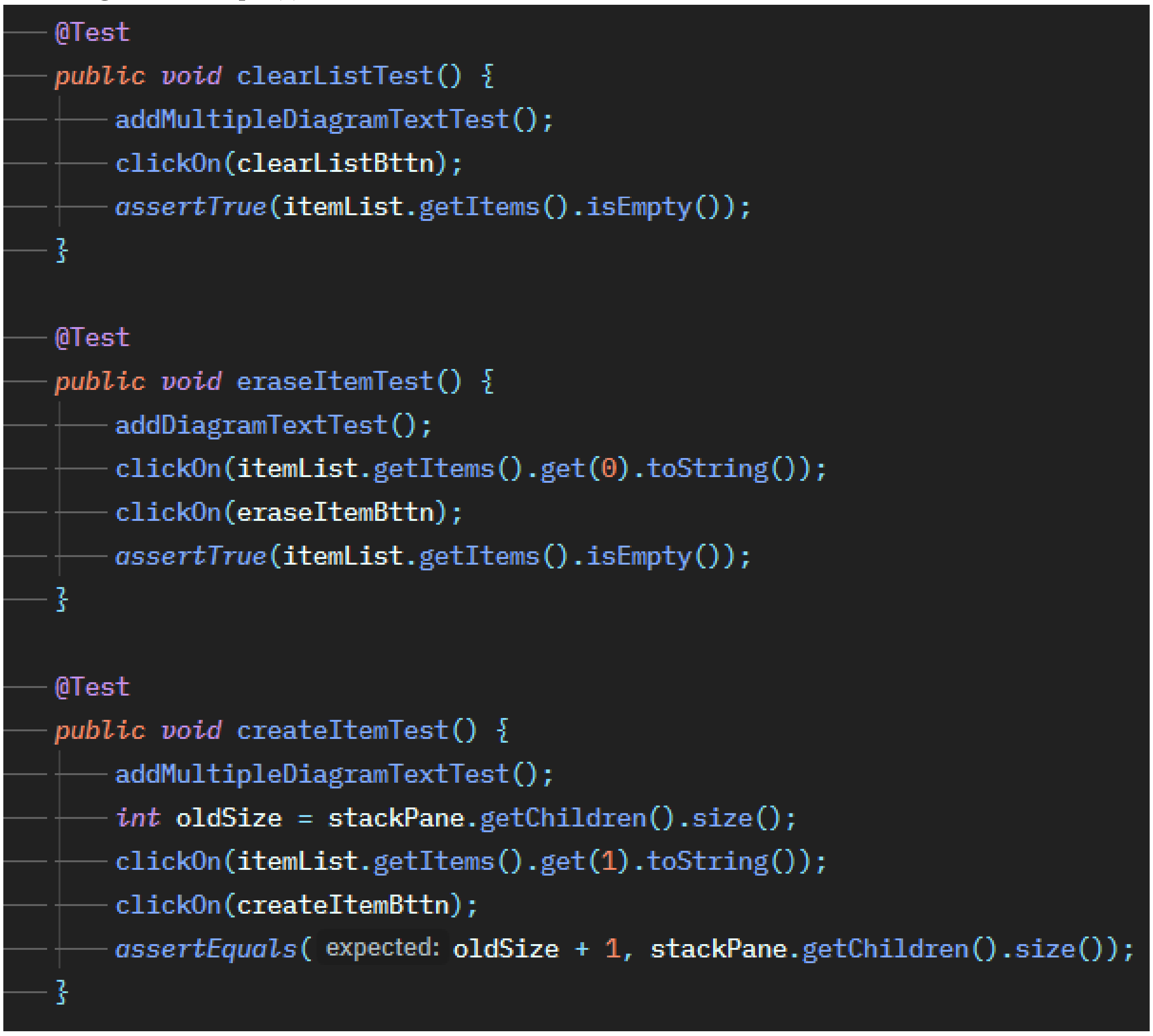
Derivation: These test cases were to ensure no error occurs when user tries to add text to diagrams, which is also one of the fundamental features of this application.

Implementation: These test cases uses the same TestFX functions introduced above, while the option to add diagram text is in the navigation bar, we defined a private helper function to toggle the navigation bar. Test cases first open the navigation bar, then clicks the text field, and input a sample text, then clicks the add button to add diagram text.

Test Status: **PASSED**

4.2.3 Tests Concerning Item List Functionality

We conducted several tests concerning the manipulation of the ListView object in shape scene. The following is an example(s) of such a test.



```
@Test
public void clearListTest() {
    addMultipleDiagramTextTest();
    clickOn(clearListBttn);
    assertTrue(itemList.getItems().isEmpty());
}

@Test
public void eraseItemTest() {
    addDiagramTextTest();
    clickOn(itemList.getItems().get(0).toString());
    clickOn(eraseItemBttn);
    assertTrue(itemList.getItems().isEmpty());
}

@Test
public void createItemTest() {
    addMultipleDiagramTextTest();
    int oldSize = stackPane.getChildren().size();
    clickOn(itemList.getItems().get(1).toString());
    clickOn(createItemBttn);
    assertEquals( expected: oldSize + 1, stackPane.getChildren().size());
}
```

.

Figure 10 - Adding Text to Diagram

Description of Test Case: These test cases mainly target the three buttons controlling the item list that kept track of the items that were added to the diagram.

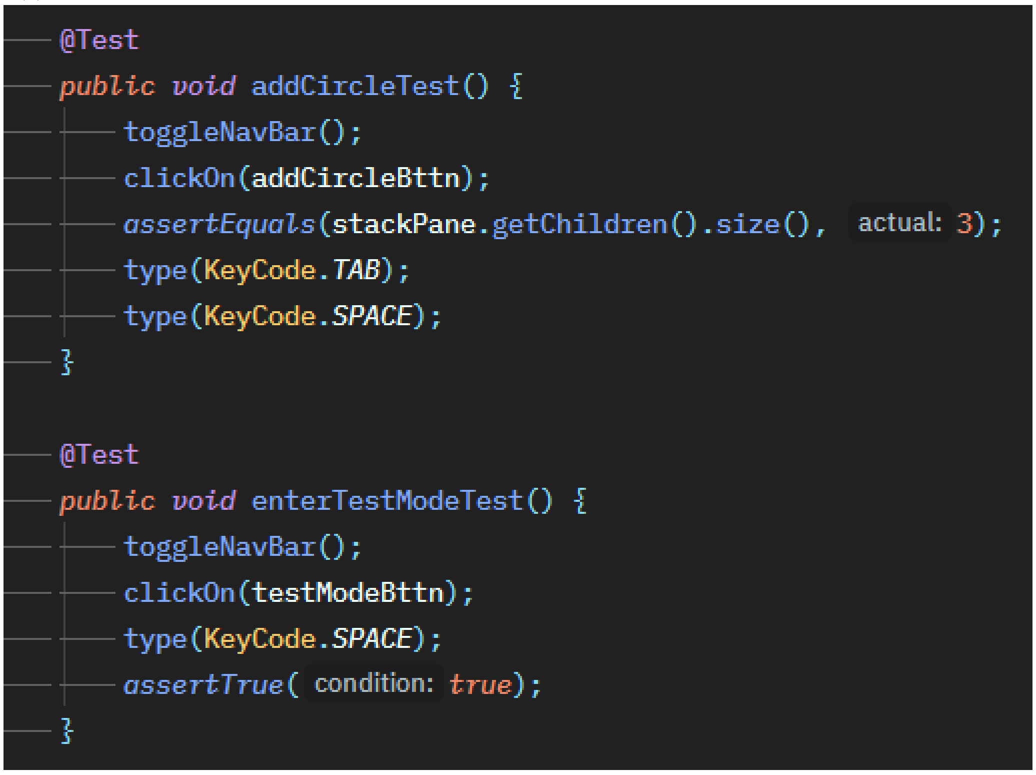
Derivation: These test cases were to ensure the correctness of the buttons controlling the item list.

Implementation: These test cases uses the *addDiagramTextTest()* and *addMultipleDiagramTextTest()* function that were defined above, in order to reduce code duplication. Then it will use TestFX's function to click these three buttons.

Test Status: **PASSED**

4.2.4 Tests Concerning Quick Actions

We conducted several tests concerning the Quick Actions in the navigation bar. The following is an example(s) of such a test.



```
@Test
public void addCircleTest() {
    toggleNavBar();
    clickOn(addCircleBttn);
    assertEquals(stackPane.getChildren().size(),  actual: 3);
    type(KeyCode.TAB);
    type(KeyCode.SPACE);
}

@Test
public void enterTestModeTest() {
    toggleNavBar();
    clickOn(testModeBttn);
    type(KeyCode.SPACE);
    assertTrue( condition: true);
}
```

Figure 11 - Quick Actions Test

Description of Test Case: We will be focusing on Add Circle and Enter Test Mode button under Quick Actions section.

Derivation: These test cases were to ensure the correctness of the two buttons that adds a circle and enters test mode, respectively.

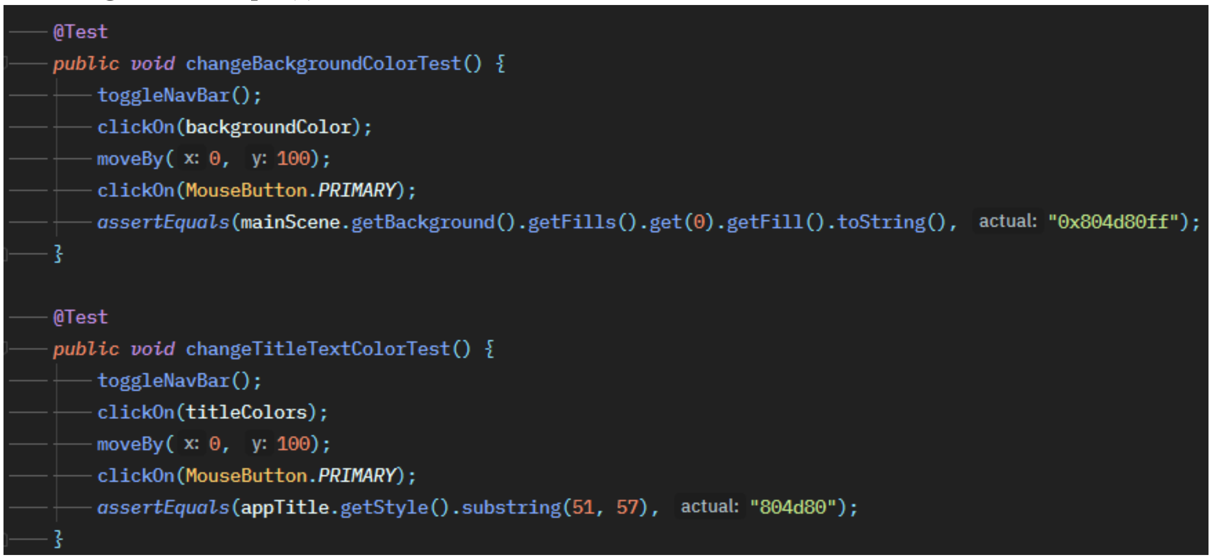
Implementation: These tests also uses TestFX's function to click the two buttons programmatically.

Test Status: PASSED

Document Continues Below…

4.2.5 Tests Concerning Main Scene Appearance

We conducted several tests concerning the customization of the appearance of the main scene. The following is an example(s) of such a test.



```java
@Test
public void changeBackgroundColorTest() {
    toggleNavBar();
    clickOn(backgroundColor);
    moveBy( x: 0,  y: 100);
    clickOn(MouseButton.PRIMARY);
    assertEquals(mainScene.getBackground().getFills().get(0).getFill().toString(),  actual: "0x804d80ff");
}

@Test
public void changeTitleTextColorTest() {
    toggleNavBar();
    clickOn(titleColors);
    moveBy( x: 0,  y: 100);
    clickOn(MouseButton.PRIMARY);
    assertEquals(appTitle.getStyle().substring(51, 57),  actual: "804d80");
}
```

Figure 12 - Changing Main Scene Appearance

Description of Test Case: In these two test cases, we will be focusing on the option to change the background colour of the Venn diagram and changing the title text colour.

Derivation: These test cases were to ensure the correctness of the customization of the Main Scene appearance.

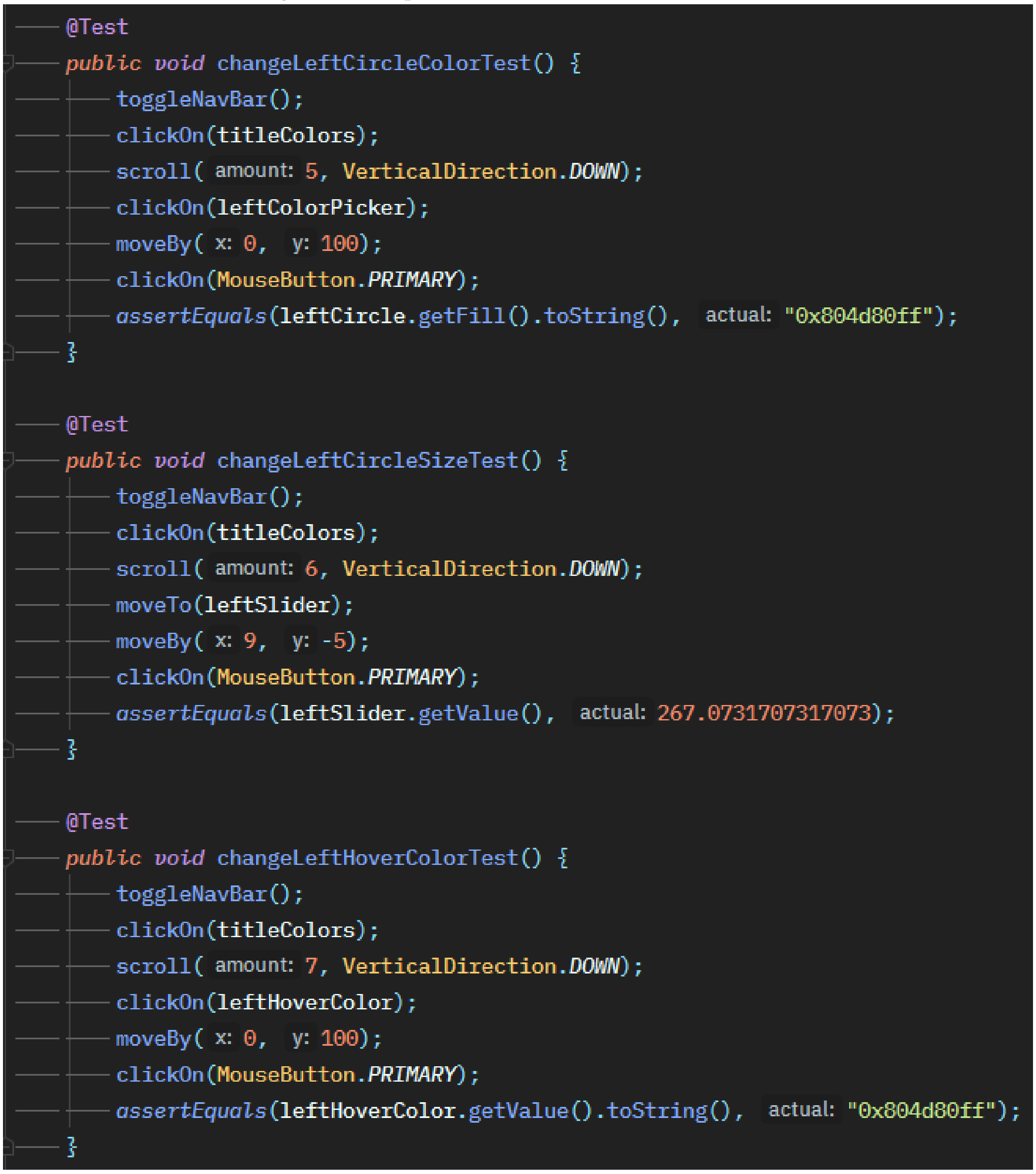
Implementation: These tests also uses TestFX's function to move the mouse to the specified location and then clicks in order to change the colour. Then asserting the value of actual colour to the expected value.

Test Status: PASSED

Document Continues Below…

4.2.6 Tests Concerning Venn Circle Appearance

We conducted several tests concerning the customization of the appearance of the Venn circle, there is the left circle and the right circle, since the test cases are similar, we will only show the test cases for the left circle. The following is an example(s) of such a test.



```
@Test
public void changeLeftCircleColorTest() {
    toggleNavBar();
    clickOn(titleColors);
    scroll( amount: 5, VerticalDirection.DOWN);
    clickOn(leftColorPicker);
    moveBy( x: 0,  y: 100);
    clickOn(MouseButton.PRIMARY);
    assertEquals(leftCircle.getFill().toString(),  actual: "0x804d80ff");
}

@Test
public void changeLeftCircleSizeTest() {
    toggleNavBar();
    clickOn(titleColors);
    scroll( amount: 6, VerticalDirection.DOWN);
    moveTo(leftSlider);
    moveBy( x: 9,  y: -5);
    clickOn(MouseButton.PRIMARY);
    assertEquals(leftSlider.getValue(),  actual: 267.0731707317073);
}

@Test
public void changeLeftHoverColorTest() {
    toggleNavBar();
    clickOn(titleColors);
    scroll( amount: 7, VerticalDirection.DOWN);
    clickOn(leftHoverColor);
    moveBy( x: 0,  y: 100);
    clickOn(MouseButton.PRIMARY);
    assertEquals(leftHoverColor.getValue().toString(),  actual: "0x804d80ff");
}
```

Figure 13 - Changing Left Circle Appearance

Description of Test Case: In these test cases, we will be focusing on changing the appearance of the left Venn circle, including the circle colour, circle size and hover color.

Derivation: These test cases were to ensure the correctness of the customization of the Venn circles' appearance.

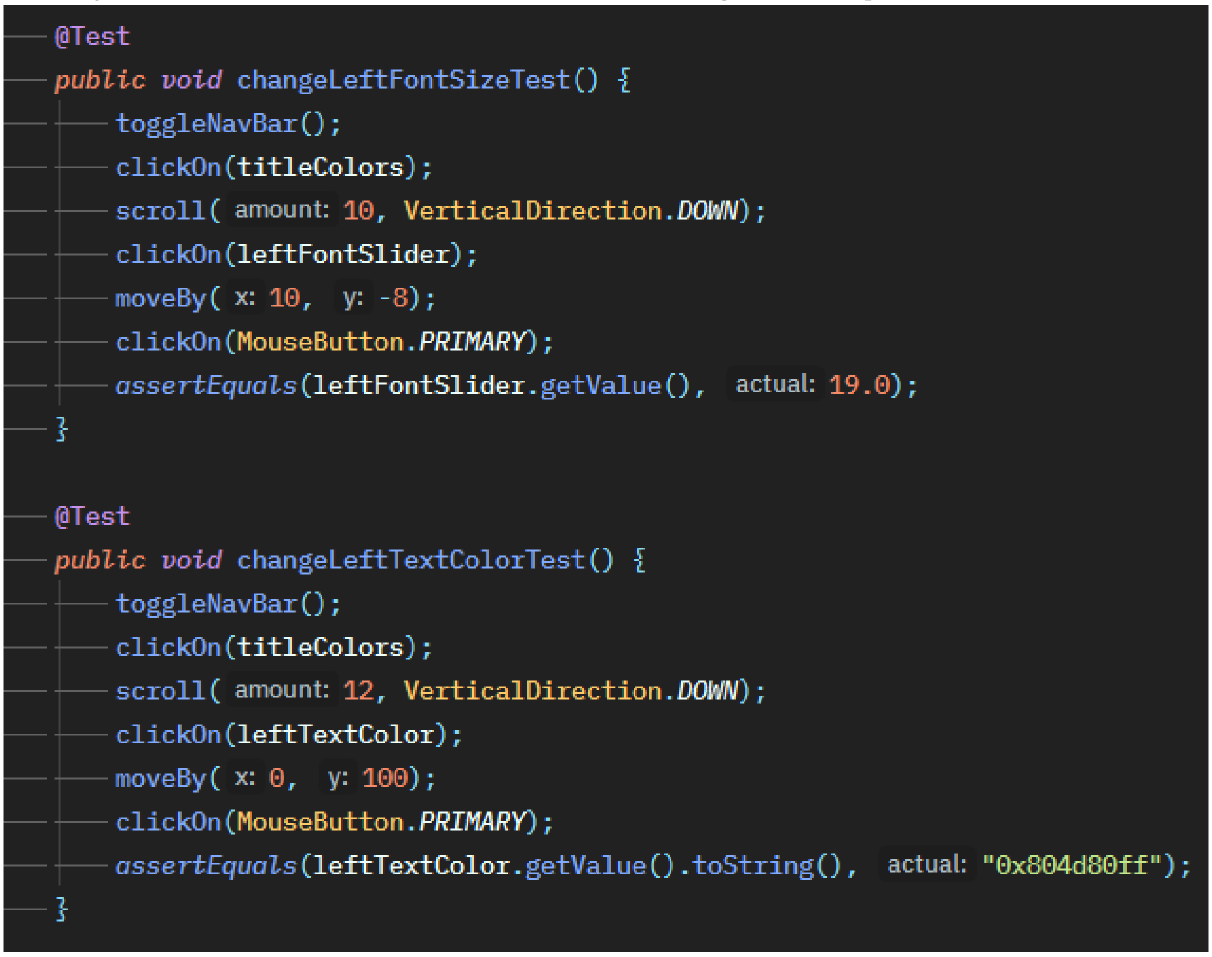
Implementation: These tests also uses TestFX's function to first scroll the option into view, then move the mouse to the specified location and change the options. As well as asserting the actual value to the expected value.

Test Status: **PASSED**

Document Continues Below…

4.2.7 Tests Concerning Text Appearance

We conducted several tests concerning the customization of the appearance of the diagram text in the Venn diagram, note that there is the left circle and the right circle, since the test cases are similar, we will only show the test cases for the left circle. The following is an example(s) of such a test.



```
@Test
public void changeLeftFontSizeTest() {
    toggleNavBar();
    clickOn(titleColors);
    scroll( amount: 10, VerticalDirection.DOWN);
    clickOn(leftFontSlider);
    moveBy( x: 10,  y: -8);
    clickOn(MouseButton.PRIMARY);
    assertEquals(leftFontSlider.getValue(),  actual: 19.0);
}

@Test
public void changeLeftTextColorTest() {
    toggleNavBar();
    clickOn(titleColors);
    scroll( amount: 12, VerticalDirection.DOWN);
    clickOn(leftTextColor);
    moveBy( x: 0,  y: 100);
    clickOn(MouseButton.PRIMARY);
    assertEquals(leftTextColor.getValue().toString(),  actual: "0x804d80ff");
}
```

Figure 14 - Changing Left Text Appearance

Description of Test Case: In these two test cases, we will be focusing on the option to change the font size and colour of the diagram text in the left Venn circle.

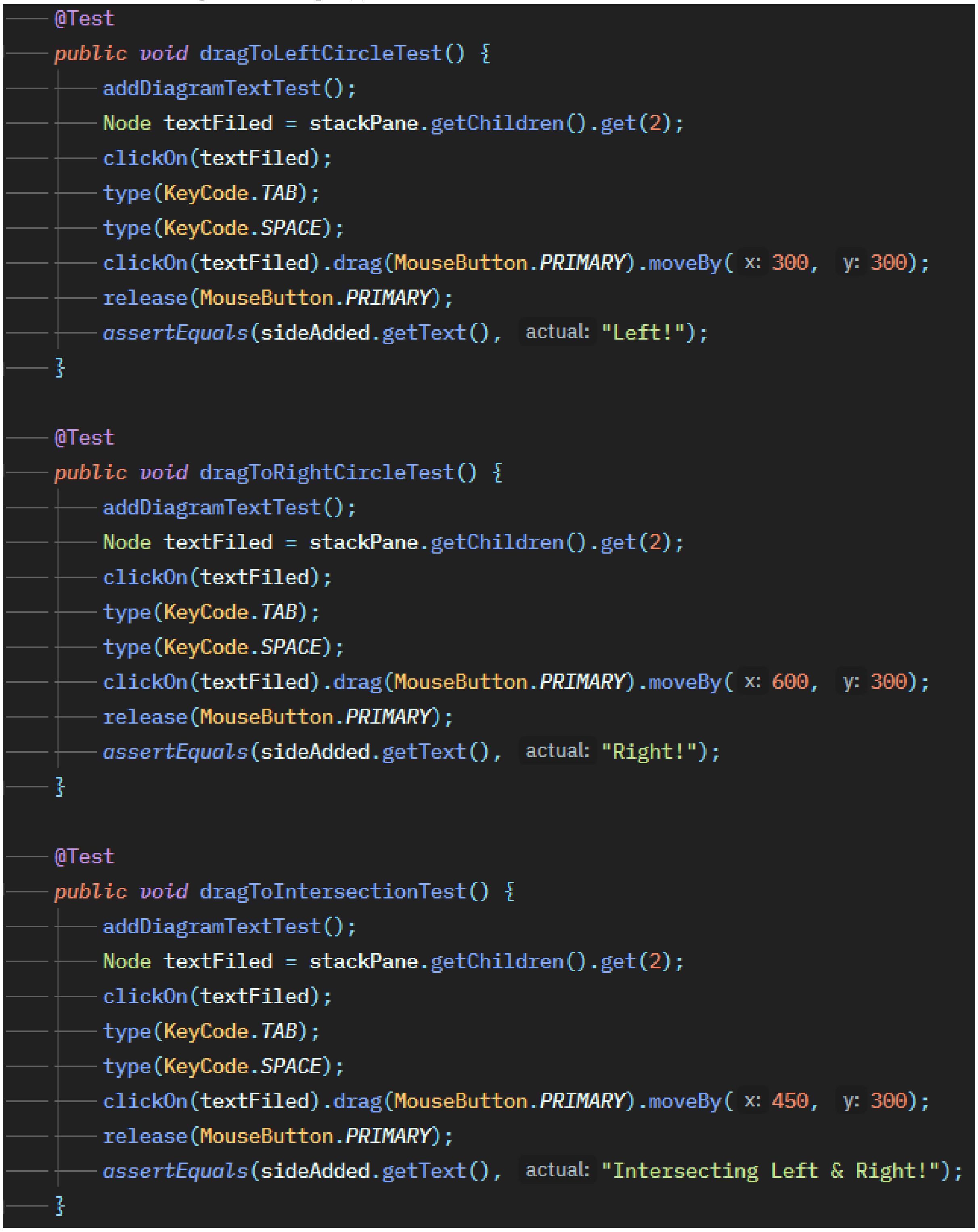
Derivation: These test cases were to ensure the correctness of the customization of the appearance of diagram texts in the left Venn circle.

Implementation: These tests also uses TestFX's function to move the mouse to the specified location and then clicks in order to change the colour. Then asserting the value of actual colour to the expected value.

Test Status: **PASSED**

4.2.9 Tests Concerning Text Dragging

We conducted several tests concerning dragging the diagram text to the different position of the Venn circle. The following is an example(s) of such a test.



```
@Test
public void dragToLeftCircleTest() {
    addDiagramTextTest();
    Node textFiled = stackPane.getChildren().get(2);
    clickOn(textFiled);
    type(KeyCode.TAB);
    type(KeyCode.SPACE);
    clickOn(textFiled).drag(MouseButton.PRIMARY).moveBy( x: 300,  y: 300);
    release(MouseButton.PRIMARY);
    assertEquals(sideAdded.getText(),  actual: "Left!");
}

@Test
public void dragToRightCircleTest() {
    addDiagramTextTest();
    Node textFiled = stackPane.getChildren().get(2);
    clickOn(textFiled);
    type(KeyCode.TAB);
    type(KeyCode.SPACE);
    clickOn(textFiled).drag(MouseButton.PRIMARY).moveBy( x: 600,  y: 300);
    release(MouseButton.PRIMARY);
    assertEquals(sideAdded.getText(),  actual: "Right!");
}

@Test
public void dragToIntersectionTest() {
    addDiagramTextTest();
    Node textFiled = stackPane.getChildren().get(2);
    clickOn(textFiled);
    type(KeyCode.TAB);
    type(KeyCode.SPACE);
    clickOn(textFiled).drag(MouseButton.PRIMARY).moveBy( x: 450,  y: 300);
    release(MouseButton.PRIMARY);
    assertEquals(sideAdded.getText(),  actual: "Intersecting Left & Right!");
}
```

Figure 15 - Test Text Dragging

Description of Test Case: These test cases test dragging the diagram text to different location on the Venn diagram, including the left circle, right circle and intersecting left and right circle.

Derivation: These test cases were to ensure the correctness dragging the diagram text and the side added feature is working as expected.

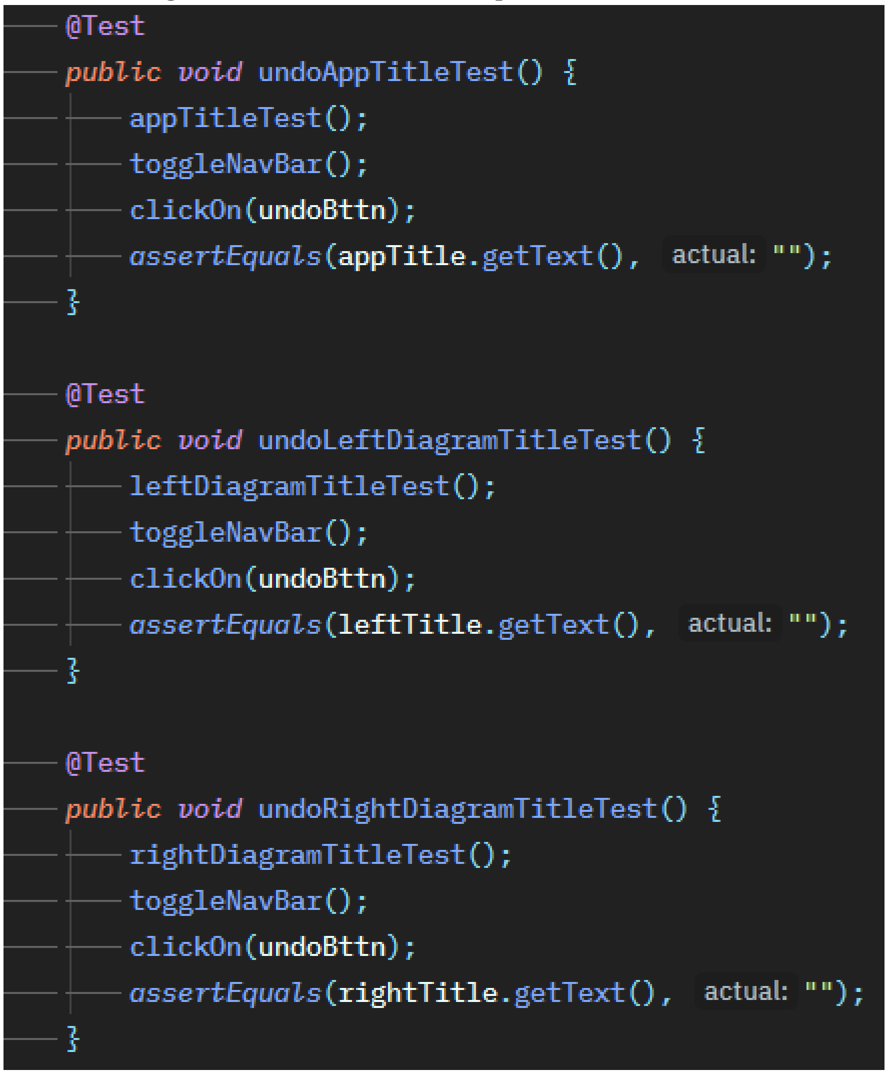
Implementation: In each of these tests cases, we first add a diagram text to the diagram, then uses TextFX's function to drag it to the specified location. Then asserting the side added text to the expected value.

Test Status: PASSED

Document Continues Below…

4.2.9 Tests Concerning Undo Command

Note that, for every command that has been introduced, there is a test case to test the undo command of this action, due to the large number of the test cases, partial screenshot is shown.



```
@Test
public void undoAppTitleTest() {
    appTitleTest();
    toggleNavBar();
    clickOn(undoBttn);
    assertEquals(appTitle.getText(), actual: "");
}

@Test
public void undoLeftDiagramTitleTest() {
    leftDiagramTitleTest();
    toggleNavBar();
    clickOn(undoBttn);
    assertEquals(leftTitle.getText(), actual: "");
}

@Test
public void undoRightDiagramTitleTest() {
    rightDiagramTitleTest();
    toggleNavBar();
    clickOn(undoBttn);
    assertEquals(rightTitle.getText(), actual: "");
}
```

Figure 16 - Test Undo Command

Description of Test Case: The undo command test cases test the undo command for each of the test cases that has been introduction above.

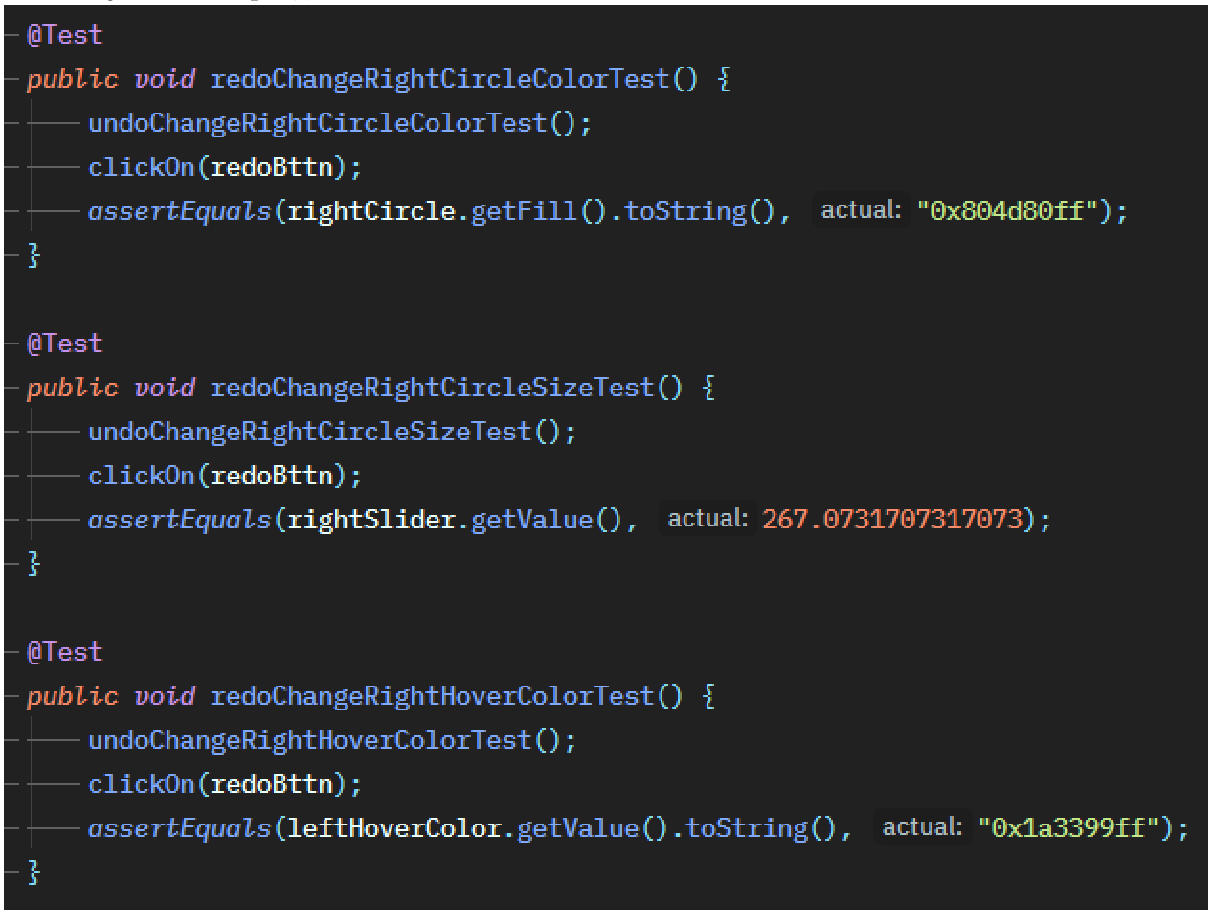
Derivation: These test cases were to ensure the correctness of the correctness of the undo command.

Implementation: The undo tests first invoke the corresponding test cases defined previously, then programmatically clicks the undo button, then checks if the actual value is correct.

Test Status: **PASSED**

4.2.10 Tests Concerning Redo Command

Note that, for every command that has been introduced, there is a test case to test the redo command on this action, due to the large number of the test cases, we will only show the partial screenshot. The following is an example(s) of such a test.



```
@Test
public void redoChangeRightCircleColorTest() {
    undoChangeRightCircleColorTest();
    clickOn(redoBttn);
    assertEquals(rightCircle.getFill().toString(),  actual: "0x804d80ff");
}

@Test
public void redoChangeRightCircleSizeTest() {
    undoChangeRightCircleSizeTest();
    clickOn(redoBttn);
    assertEquals(rightSlider.getValue(),  actual: 267.0731707317073);
}

@Test
public void redoChangeRightHoverColorTest() {
    undoChangeRightHoverColorTest();
    clickOn(redoBttn);
    assertEquals(leftHoverColor.getValue().toString(),  actual: "0x1a3399ff");
}
```

Figure 17 - Test Redo Command

Description of Test Case: The redo command test cases test the redo command for each of the undo command test cases that has been introduction above.

Derivation: These test cases were to ensure the correctness of the correctness of the redo command.

Implementation: The redo tests first invoke the corresponding undo test cases defined previously, then programmatically clicks the redo button, then checks if the actual value is correct.

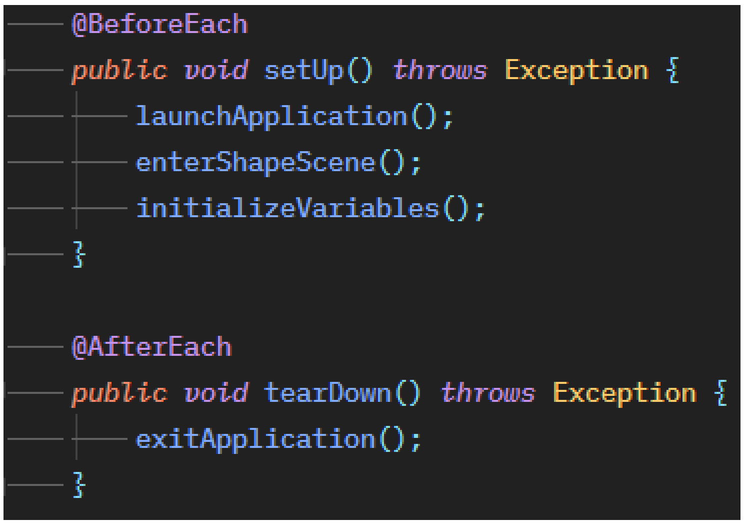
Test Status: PASSED

4.3 KeyboardShortCutTest

This test file targets the keyboard shortcut functionality that's supported in shapeScene. Mainly undo shortcut and redo shortcut. The test class also extends the *MainTest*.

BeforeEach and AfterEach

Similar to *ShapeSceneControllerTest*, before each test cases in this class starts, we will first launch the main application, enter shapeScene and initialize the necessary variables that will be used in the test cases. And after each test cases, we will exit the application.



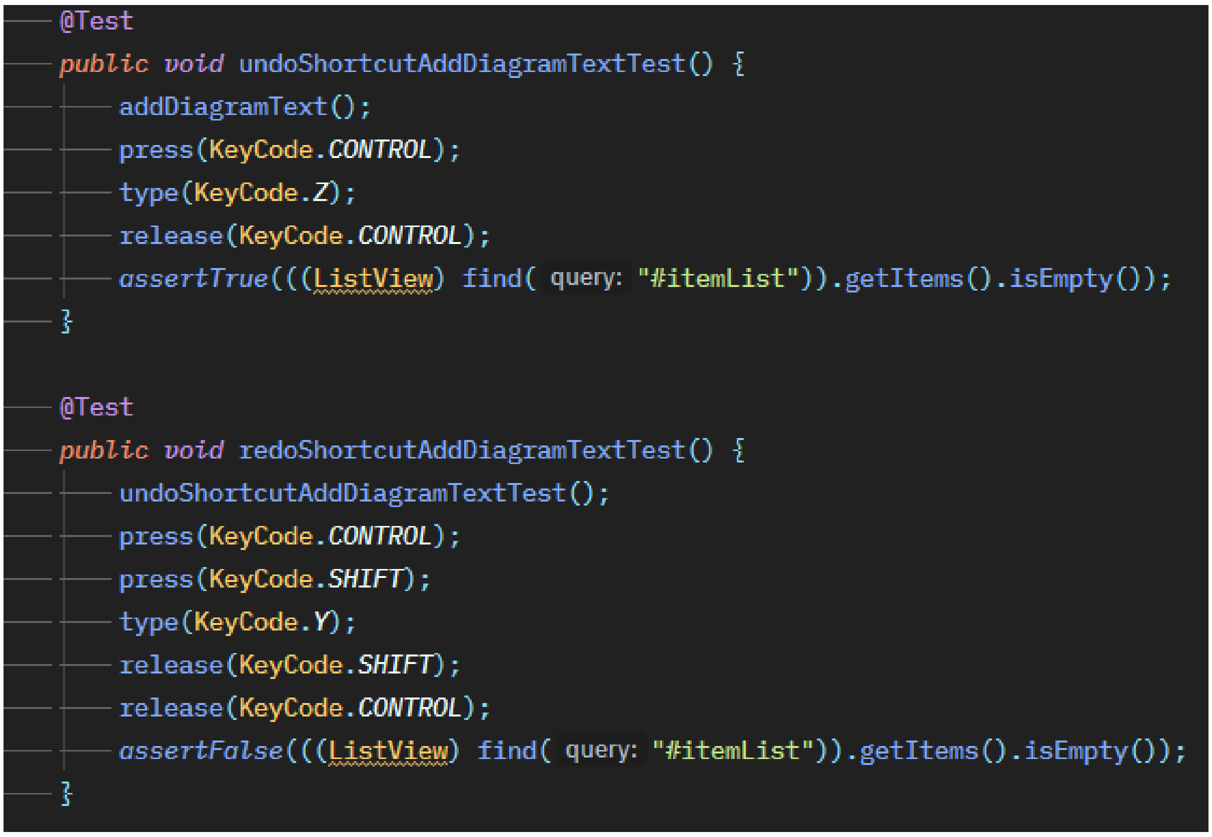
```java
@BeforeEach
public void setUp() throws Exception {
    launchApplication();
    enterShapeScene();
    initializeVariables();
}

@AfterEach
public void tearDown() throws Exception {
    exitApplication();
}
```

Figure 18 - KeyboardShortCutTest BeforeEach and AfterEach

Document Continues Below…

4.3.1 Tests Concerning undo/redo Shortcut

As we have already done undo/redo command test in *ShapeSceneControllerTest*, we will only be testing the keyboard shortcut once for each undo and redo command. The following is an example of such a test.



```
@Test
public void undoShortcutAddDiagramTextTest() {
    addDiagramText();
    press(KeyCode.CONTROL);
    type(KeyCode.Z);
    release(KeyCode.CONTROL);
    assertTrue(((ListView) find( query: "#itemList")).getItems().isEmpty());
}

@Test
public void redoShortcutAddDiagramTextTest() {
    undoShortcutAddDiagramTextTest();
    press(KeyCode.CONTROL);
    press(KeyCode.SHIFT);
    type(KeyCode.Y);
    release(KeyCode.SHIFT);
    release(KeyCode.CONTROL);
    assertFalse(((ListView) find( query: "#itemList")).getItems().isEmpty());
}
```

Figure 19 - undo/redo Shortcut Test

Description of Test Case: These test cases focused on testing keyboard shortcut for undo/redo the action of adding a diagram text.

Derivation: These test cases were to ensure no error occurs for the keyboard shortcut of the undo/redo command.

Implementation: These test cases also were all implemented using TextFX's functions. It programmatically enters the keyboard shortcut for the undo and redo command.

Test Status: **PASSED**

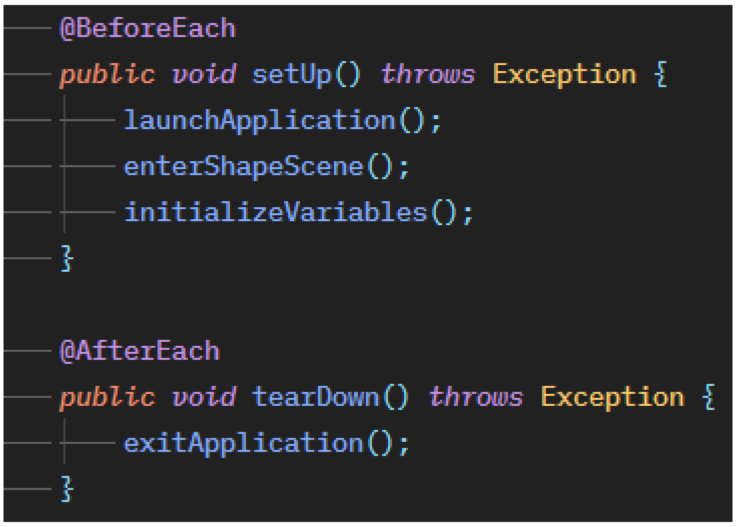
4.4 ContextMenuTest

This suite of test cases test the context menu in shape scene, which is a crucial part of the software as the context menu includes the option to remove the diagram text.

Hence, we also developed 14 test cases for the context menu to further ensure the usability and correctness of our software.

BeforeEach and AfterEach

Similar to *ShapeSceneControllerTest*, before each test cases in this class starts, we will first launch the main application, enter shapeScene and initialize the necessary variables that will be used in the test cases. And after each test cases, we will exit the application.



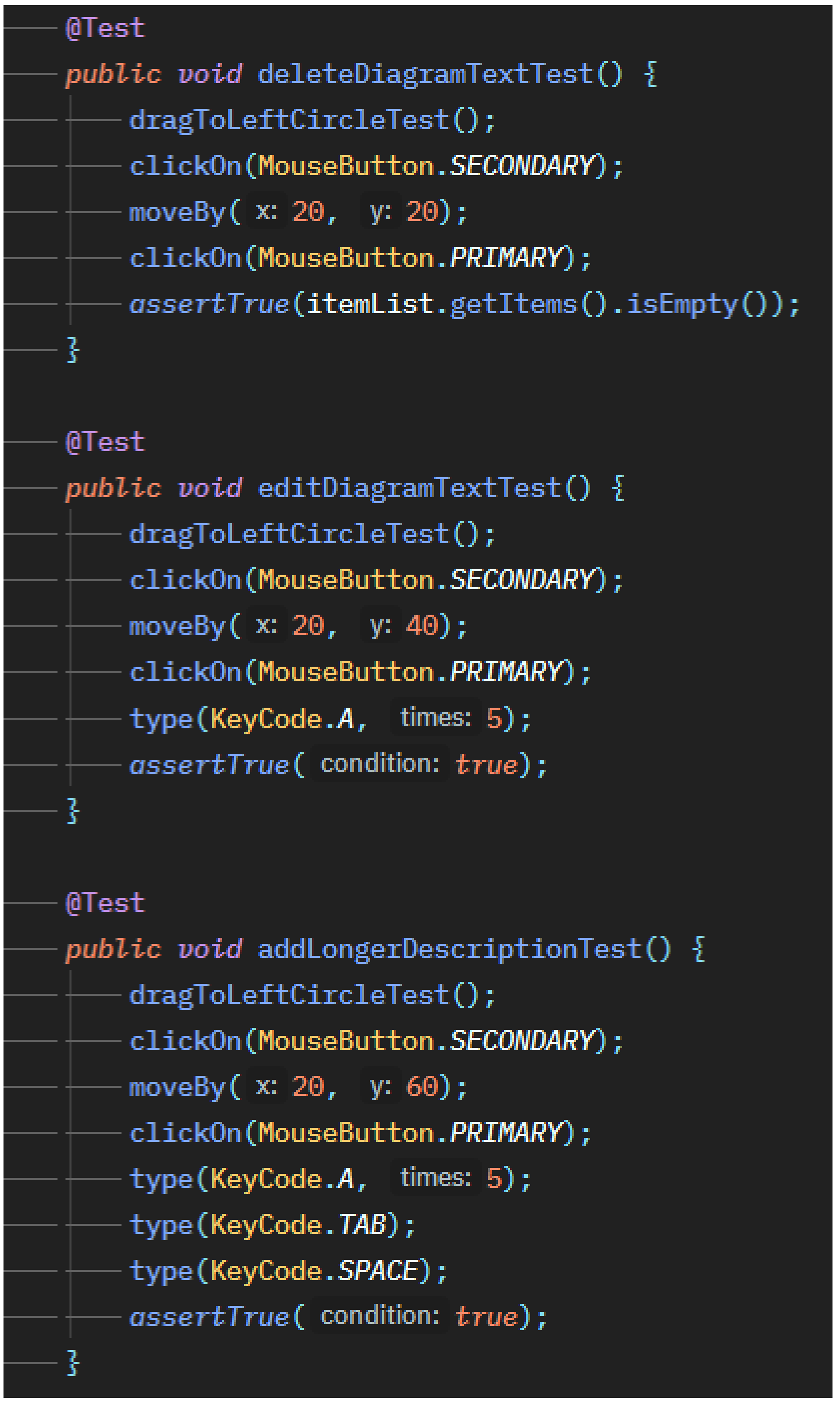
```java
@BeforeEach
public void setUp() throws Exception {
    launchApplication();
    enterShapeScene();
    initializeVariables();
}

@AfterEach
public void tearDown() throws Exception {
    exitApplication();
}
```

Figure 20 - ContextMenuTest BeforeEach and AfterEach

Document Continues Below…

4.4.1 Tests Concerning Diagram Text Context Menu

We conducted several tests concerning the context menu of the diagram text. The following is an example(s) of such a test.



```java
@Test
public void deleteDiagramTextTest() {
    dragToLeftCircleTest();
    clickOn(MouseButton.SECONDARY);
    moveBy( x: 20,  y: 20);
    clickOn(MouseButton.PRIMARY);
    assertTrue(itemList.getItems().isEmpty());
}

@Test
public void editDiagramTextTest() {
    dragToLeftCircleTest();
    clickOn(MouseButton.SECONDARY);
    moveBy( x: 20,  y: 40);
    clickOn(MouseButton.PRIMARY);
    type(KeyCode.A,  times: 5);
    assertTrue( condition: true);
}

@Test
public void addLongerDescriptionTest() {
    dragToLeftCircleTest();
    clickOn(MouseButton.SECONDARY);
    moveBy( x: 20,  y: 60);
    clickOn(MouseButton.PRIMARY);
    type(KeyCode.A,  times: 5);
    type(KeyCode.TAB);
    type(KeyCode.SPACE);
    assertTrue( condition: true);
}
```

Figure 21 - Diagram Text Context Menu Test

Description of Test Case: These test cases target the context menu of a user right clicks a diagram text.

Derivation: These test cases were to ensure no error occurs the context menu options regarding the diagram text.

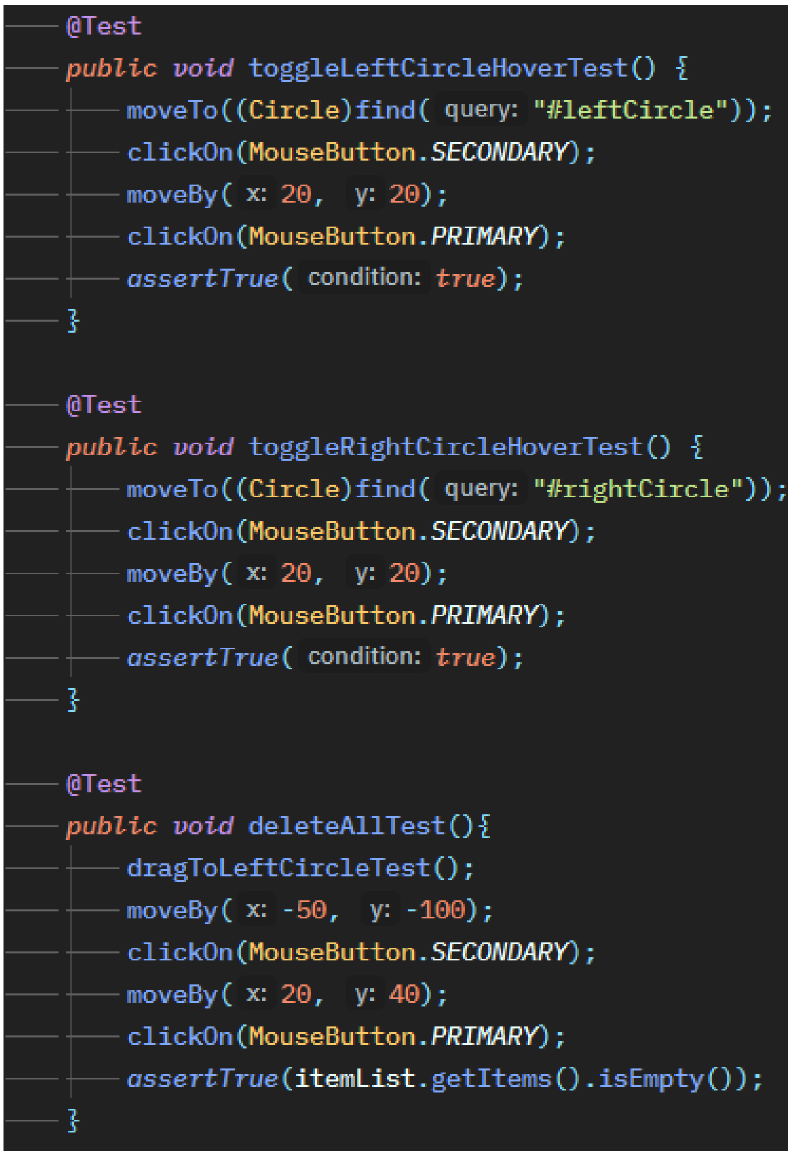
Implementation: These test cases also were all implemented using TextFX's functions. It programmatically clicks the diagram text and the context menu option.

Test Status: **PASSED**

Document Continues Below…

4.4.2 Tests Concerning Circle Context Menu

We conducted several tests concerning the context menu of the Venn circle. The following is an example(s) of such a test.



```java
@Test
public void toggleLeftCircleHoverTest() {
    moveTo((Circle)find( query: "#leftCircle"));
    clickOn(MouseButton.SECONDARY);
    moveBy( x: 20,  y: 20);
    clickOn(MouseButton.PRIMARY);
    assertTrue( condition: true);
}

@Test
public void toggleRightCircleHoverTest() {
    moveTo((Circle)find( query: "#rightCircle"));
    clickOn(MouseButton.SECONDARY);
    moveBy( x: 20,  y: 20);
    clickOn(MouseButton.PRIMARY);
    assertTrue( condition: true);
}

@Test
public void deleteAllTest(){
    dragToLeftCircleTest();
    moveBy( x: -50,  y: -100);
    clickOn(MouseButton.SECONDARY);
    moveBy( x: 20,  y: 40);
    clickOn(MouseButton.PRIMARY);
    assertTrue(itemList.getItems().isEmpty());
}
```

Figure 22 - Venn Circle Context Menu Test

Description of Test Case: These test cases target the context menu of a user right clicks a Venn circle.

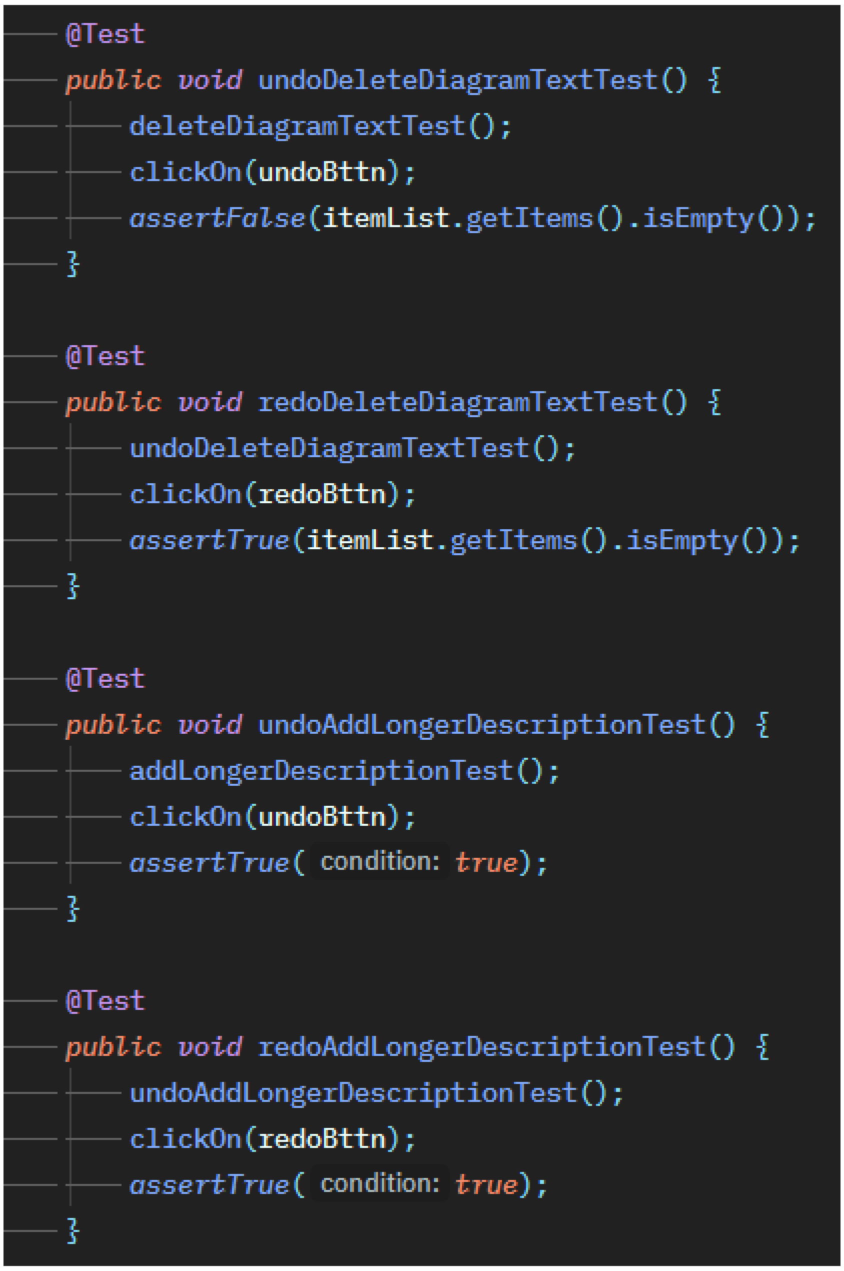
Derivation: These test cases were to ensure no error occurs the context menu options regarding the Venn circle.

Implementation: These test cases also were all implemented using TextFX's functions. It programmatically clicks the Venn circle and the context menu option.

Test Status: **PASSED**

4.4.3 Tests Concerning undo/redo Context Menu Actions

We will also test the undo/redo command for each of the context menu actions that is introduced above, note that we only shows part of the screenshots here. The following is an example of such a test.



```
@Test
public void undoDeleteDiagramTextTest() {
    deleteDiagramTextTest();
    clickOn(undoBttn);
    assertFalse(itemList.getItems().isEmpty());
}

@Test
public void redoDeleteDiagramTextTest() {
    undoDeleteDiagramTextTest();
    clickOn(redoBttn);
    assertTrue(itemList.getItems().isEmpty());
}

@Test
public void undoAddLongerDescriptionTest() {
    addLongerDescriptionTest();
    clickOn(undoBttn);
    assertTrue( condition: true);
}

@Test
public void redoAddLongerDescriptionTest() {
    undoAddLongerDescriptionTest();
    clickOn(redoBttn);
    assertTrue( condition: true);
}
```

Figure 23 - undo/redo Context Menu Actions Text

Description of Test Case: These test cases focused on testing the undo/redo command on the context menu actions.

Derivation: These test cases were to ensure no error occurs for the the undo/redo command of the context menu.

Implementation: These test cases also were all implemented using TextFX's functions. It first called the functions defined above and then clicks the undo/redo button.

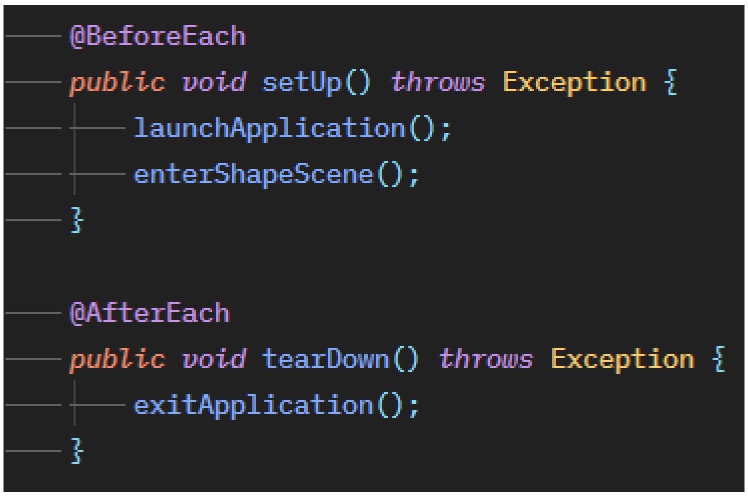
Test Status: **PASSED**

4.5 MenuBarTest

In this test file, we will be testing the functionalities of the menu bar items in shapeScene.

BeforeEach and AfterEach

Similar to *ShapeSceneControllerTest*, before each test cases in this class starts, we will first launch the main application, enter shapeScene. And after each test cases, we will exit the application.



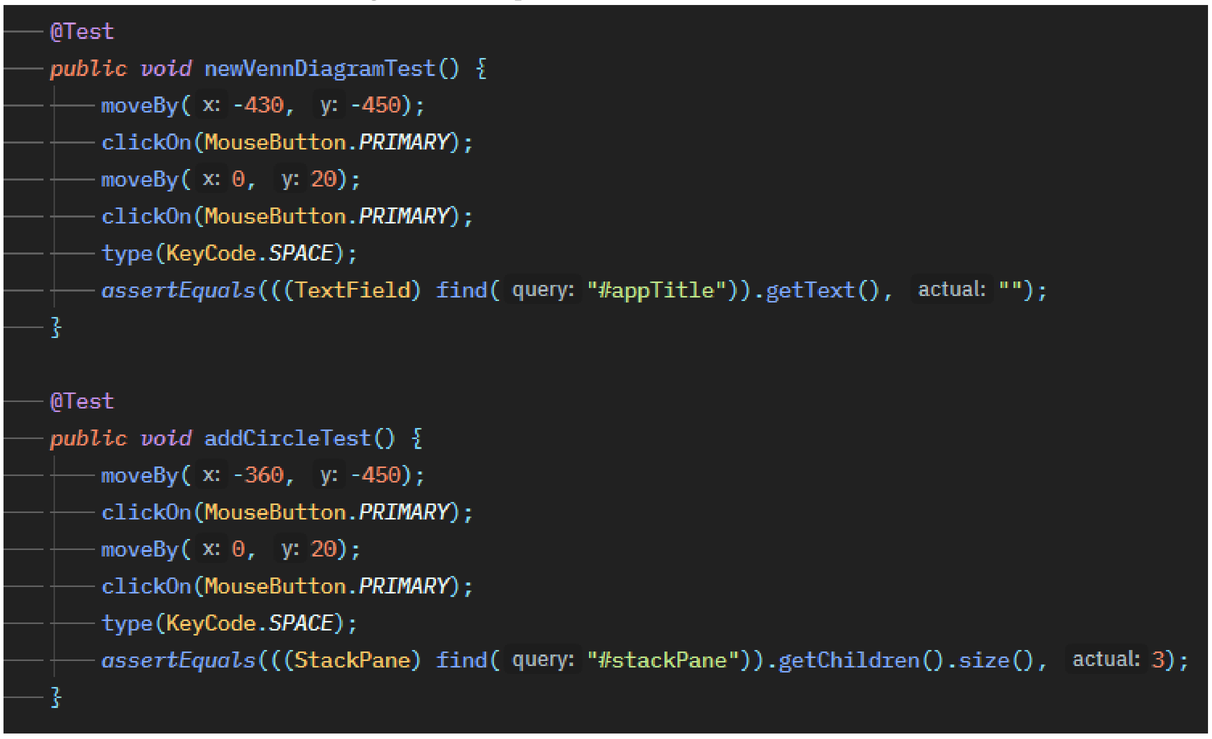
```
@BeforeEach
public void setUp() throws Exception {
    launchApplication();
    enterShapeScene();
}

@AfterEach
public void tearDown() throws Exception {
    exitApplication();
}
```

Figure 24 - MenuBarTest BeforeEach and AfterEach

Document Continues Below…

4.5.1 Tests Concerning Menu Bar Items

As we have already shown tests for undo/redo command in *ShapeSceneControllerTest*, hence we will not be testing the undo/redo menu bar items. However, we will test the newVennDiagram and addCircle button. The following is an example of such a test.



```java
@Test
public void newVennDiagramTest() {
    moveBy( x: -430,  y: -450);
    clickOn(MouseButton.PRIMARY);
    moveBy( x: 0,  y: 20);
    clickOn(MouseButton.PRIMARY);
    type(KeyCode.SPACE);
    assertEquals(((TextField) find( query: "#appTitle")).getText(),  actual: "");
}

@Test
public void addCircleTest() {
    moveBy( x: -360,  y: -450);
    clickOn(MouseButton.PRIMARY);
    moveBy( x: 0,  y: 20);
    clickOn(MouseButton.PRIMARY);
    type(KeyCode.SPACE);
    assertEquals(((StackPane) find( query: "#stackPane")).getChildren().size(),  actual: 3);
}
```

Figure 25 - Menu Bar Items Test

Description of Test Case: These test cases test the newVennDiagram and addCircle menu bar item.

Derivation: These test cases were to ensure no error occurs for the newVennDiagram and addCircle menu bar item.

Implementation: These test cases also were all implemented using TextFX's functions. It programmatically clicks the corresponding menu bar item.

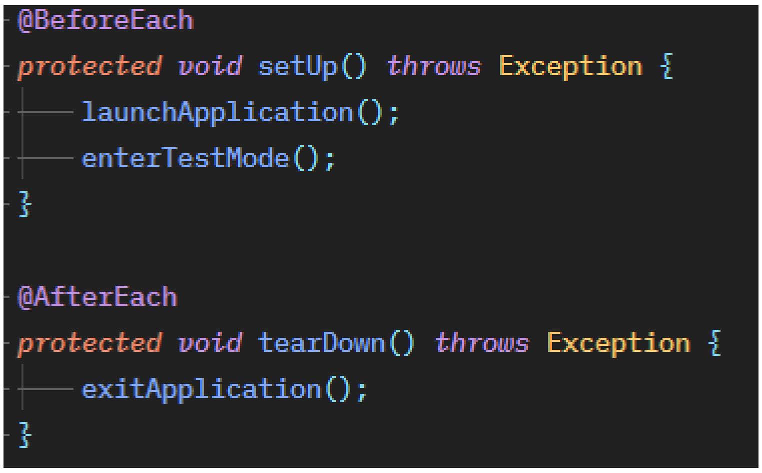
Test Status: **PASSED**

4.6 TestModeTest

In this test file, we will be testing the functionalities of the Test Mode of this software.

BeforeEach and AfterEach

Before each test, we will enter the Test Mode of this software, and then after each test case we will also exit the application.



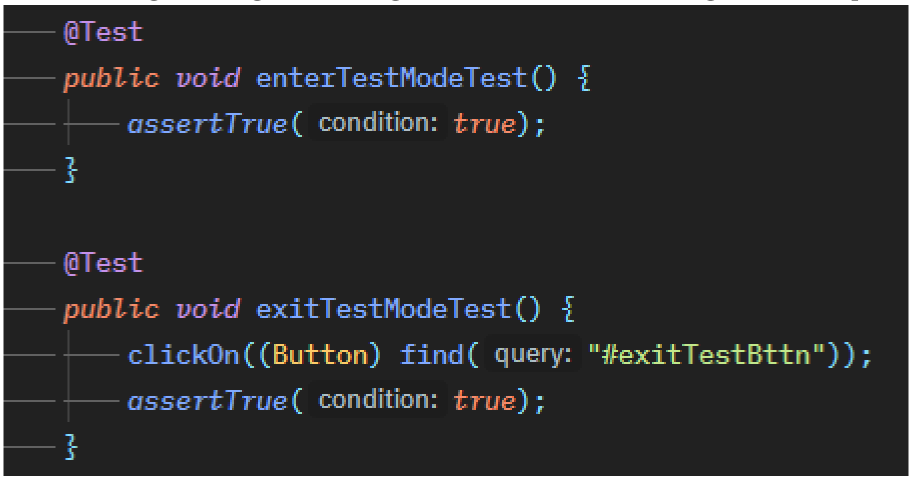
```
@BeforeEach
protected void setUp() throws Exception {
    launchApplication();
    enterTestMode();
}

@AfterEach
protected void tearDown() throws Exception {
    exitApplication();
}
```

Figure 26 - TestModeTestBeforeEach and AfterEach

Document Continues Below…

4.6.1 Tests Concerning Test Mode

Due to the Import Test File function requires a file chooser, we will not be testing that function. Instead, we will be testing entering and exiting Test Mode. The following is an example of such a test.



```
@Test
public void enterTestModeTest() {
    assertTrue( condition: true);
}

@Test
public void exitTestModeTest() {
    clickOn((Button) find( query: "#exitTestBttn"));
    assertTrue( condition: true);
}
```

Figure 27 - Test Mode Test

Description of Test Case: These test cases test enter Test Mode and exiting Test Mode.

Derivation: These test cases were to ensure no error occurs for entering and exiting TestMode.

Implementation: These test cases simply clicks the button to enter and exit Test Mode.

Test Status: **PASSED**

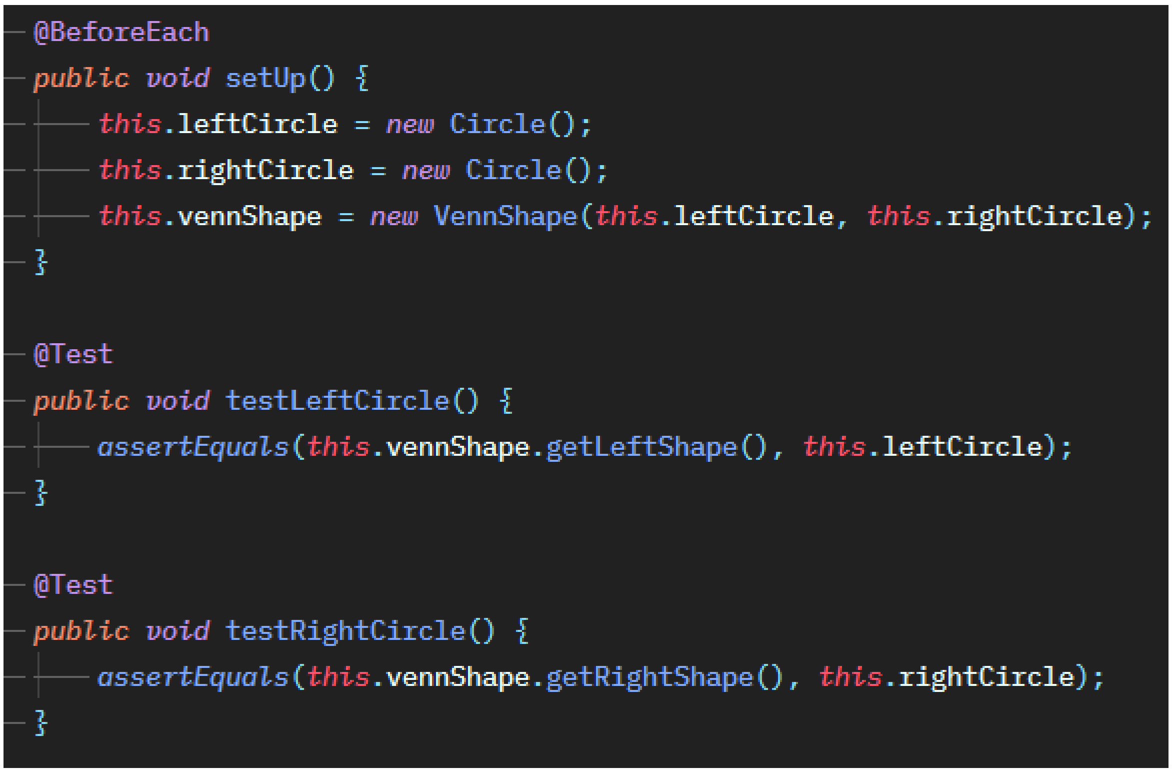
Document Continues Below…

4.7 VennShapeTest

This is the test case for our back-end logic of this software, which does not rely on TestFX. It uses only JUnit test cases for the tests.

4.7.1 Tests Concerning VennShape

Since the back-end logic is straightforward, hence we only procured simply yet complete test case for this part. The following is an example of such a test.



```java
@BeforeEach
public void setUp() {
    this.leftCircle = new Circle();
    this.rightCircle = new Circle();
    this.vennShape = new VennShape(this.leftCircle, this.rightCircle);
}

@Test
public void testLeftCircle() {
    assertEquals(this.vennShape.getLeftShape(), this.leftCircle);
}

@Test
public void testRightCircle() {
    assertEquals(this.vennShape.getRightShape(), this.rightCircle);
}
```

Figure 28 - VennShape Test

Description of Test Case: These test cases test creating the back-end VennShape object.

Derivation: These test cases were to ensure no error occurs for creating VennShape object.

Implementation: These test cases simply creates the object and test for function correctness.

Test Status: **PASSED**

End of Testing Document