

EECS 2311 Testing Document

Chidalu Agbakwa (216337784)

Shangru Li (214488993)

Jihal Patel (216376436)

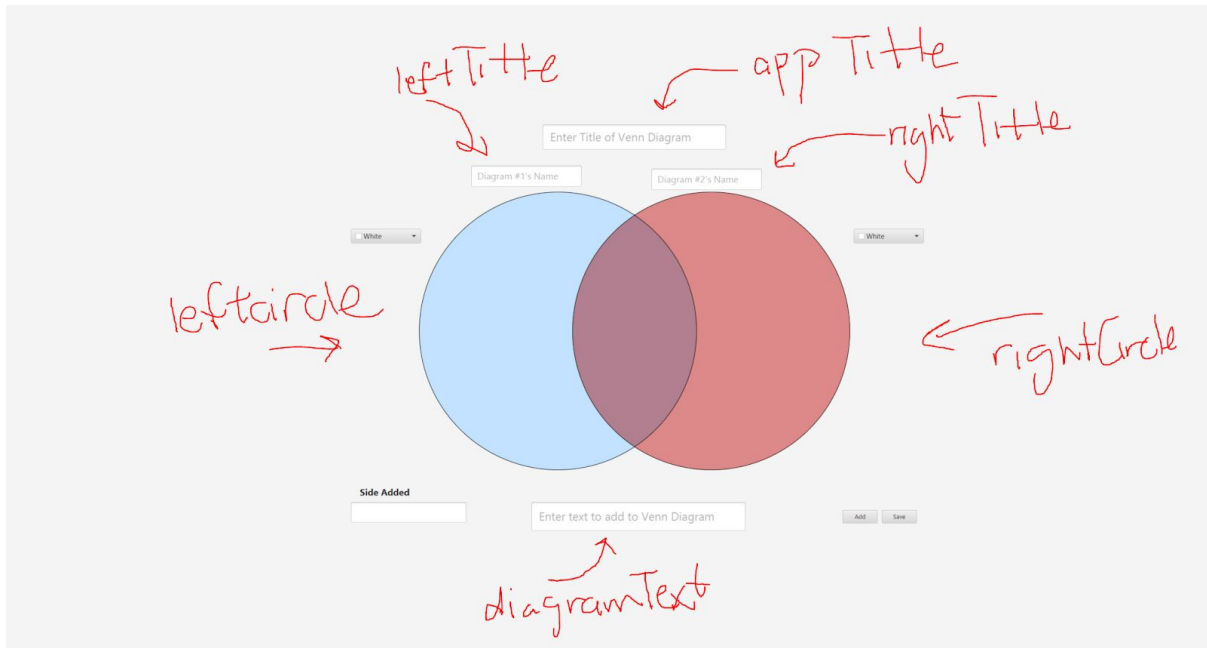
Robert Suwary (215446016)

Table of Contents

Program to be Tested	3
Overall Summary For GUI Testing Functionality	4
Testing Reading and Writing to Files	5
Test~testSave()	5
Test~testLoad()	6
In-Depth Summary of GUI Test-Cases	7
Test~testDrag_05()	7
Test~bottomTextFieldTest()	8
Test~test_appTitle()	9
Test~changeLeftColor()	10
Test~changeRightColor()	11
Testing~Drag and Drop placement	12
Test~testDrag_03()	13
Test~testDrag_06()	14
Test~testDrag_07()	15
Test~testLeftTitle1 & testRightTitle1()	16
Coverage Metrics	17

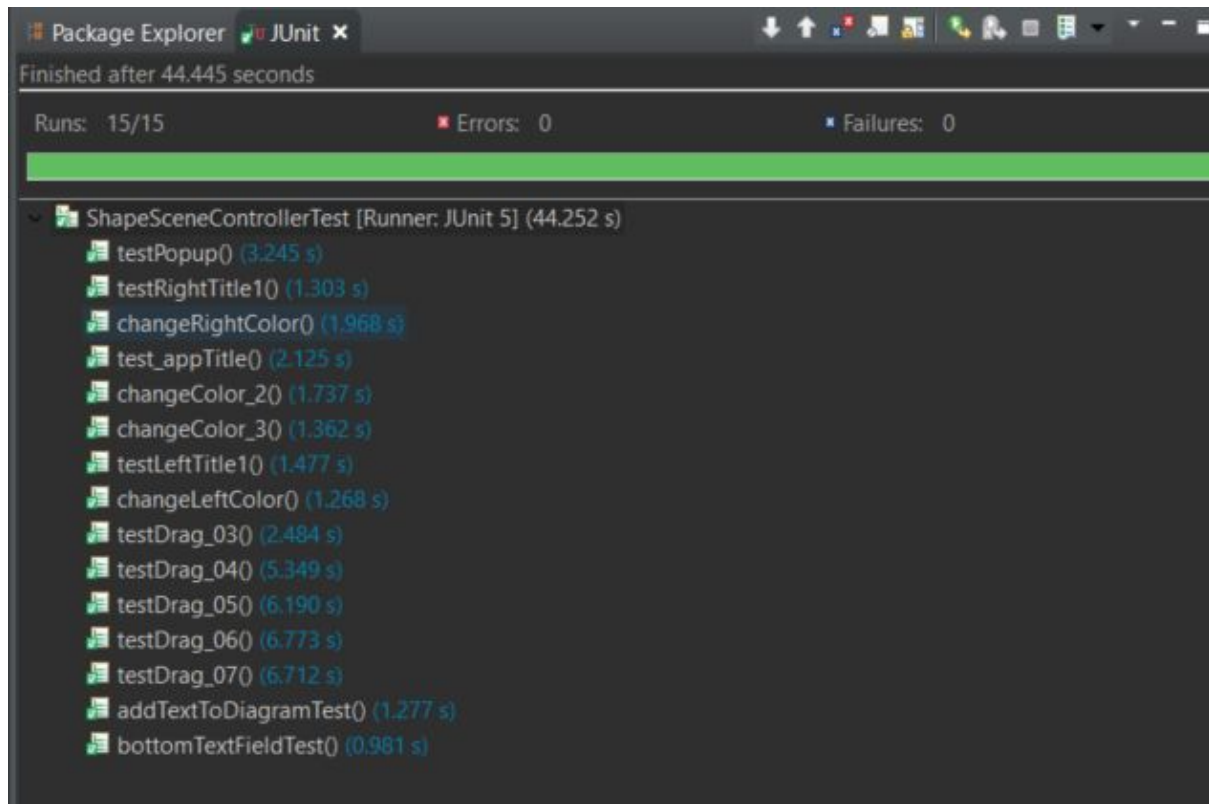
Program to be Tested

VennCreate



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

Overall Summary For GUI Testing Functionality



In testing GUI Functionality we ran 15 test cases, these test cases included drag and drop, populating TextFields, and other tests. We ran 15 test cases and all 15 of them passed.

Testing Reading and Writing to Files

Test~testSave()

```
@Test
public void testSave() throws Exception {

    String[] expected=new String[5];
    TextField tf = new TextField();
    for (int i = 0; i < expected.length; i++) {
        tf=new TextField();
        tf.setText("I am test");
        tf.setTranslateX(i);
        tf.setTranslateY(i);
        cont.getTextFields().add(tf);
        expected[i]=tf.getText()+" "+tf.getTranslateX()+" "+tf.getTranslateY();
    }
    cont.saveVenn(cont.getTextFields());
    FileReader fr=new FileReader(System.getProperty("user.dir")+"\\src\\main\\java\\application\\save.csv");
    BufferedReader br=new BufferedReader(fr);
    String[] s=new String[5];
    for (int i = 0; i < s.length; i++) {
        s[i]=br.readLine();
    }

    assertEquals(expected, s);
}
```

Description of Test Case: This test case was to ensure reading and writing of files to and from CSV worked correctly.

Derivation: This test case was to ensure that upon reading and writing of files, no Exceptions and errors would be thrown in our application.

Implementation: This test case was implemented by creating text fields and reading all their contents to a CSV file called save.csv. We made sure that the same text fields that we saved to the CSV were the same ones that were retrieved upon reading the CSV.

Test Status: **PASSED**

Test~testLoad()

```
@Test
public void testLoad() throws IOException {

    String[] expected=new String[5];
    TextField tf;
    cont=new ShapeSceneController();
    for (int i = 0; i < expected.length; i++) {
        tf=new TextField();
        tf.setText("I am test");
        tf.setTranslateX(i);
        tf.setTranslateY(i);
        cont.getTextFields().add(tf);
        expected[i]=tf.getText()+" "+tf.getTranslateX()+" "+tf.getTranslateY();
    }
    cont.saveVenn(cont.getTextFields());
    cont=new ShapeSceneController();
    cont.setStackPane(new StackPane());
    cont.loadVenn("save.csv");
    String[] s=new String[5];

    for (int i = 0; i < s.length; i++) {
        tf=cont.getTextFields().get(i);
        s[i]=tf.getText()+" "+tf.getTranslateX()+" "+tf.getTranslateY();
    }
    assertEquals(expected, s);
}
```

Description of Test Case: This test case was to ensure we could properly load text along with coordinates of where to place the text on our screen worked correctly.

Derivation: This test case was to ensure that upon reading and writing of files, no Exceptions and errors would be thrown in our application.

Implementation: This test case was implemented by creating text fields and reading all their contents to a CSV file called save.csv. We made sure that the same text fields that we saved to the CSV were the same ones that were retrieved upon reading the CSV.

Test Status: **PASSED**

In-Depth Summary of GUI Test-Cases

Test~testDrag_05()

```
@Test
public void testDrag_05() throws Exception {
    clickOn("#diagramText");
    write("This is a Junit Test SIDE ADDED YA");
    type(KeyCode.ENTER);
    Node tf = stackPane.getChildren().get(2);

    tf.setTranslateX(-174);
    Thread.sleep(1000);
    clickOn(leftCircle);
    Thread.sleep(1000);
    assertEquals(sideAdded.getText(), "Left!");
}
```

Description of Test Case: This test case ensured that the drag and drop feature, one of the most important features of our program, had the correct functionality. In addition, it ensured that when the textField was placed in the leftCircle, the sideAdded textField would output "Left!" ensuring that this was properly placed.

Derivation: This test case was derived

Implementation: It was implemented with a trivial assertEquals with JUnit 5

Test Status: **PASSED**

Test~bottomTextFieldTest()

```
@Test
public void bottomTextFieldTest() throws InterruptedException {
    clickOn("#diagramText");
    write("Stop it");
    assertEquals(diagramText.getText(), "Stop it");
}
```

Description of Test Case: This test case ensured that when a user were to click diagramText, and type a phrase, the correct text would indeed be inside the textField.

Derivation: We needed this test case in order to ensure in later steps, the user could drag and drop the text they just typed in the circles.

Implementation: It was implemented with a trivial assertEquals with JUnit 5.

Test Status: **PASSED**

Test~test_appTitle()

```
@Test
public void test_appTitle() throws InterruptedException {
    clickOn("#appTitle");
    write("This is my Title");
    assertEquals(appTitle.getText(), "This is my Title");
}
```

Description of Test Case: This test case ensured that when a user were to click appTitle, and type a phrase ("This is my Title"), the correct text would indeed be inside the textField.

Derivation: We needed this test case in order to ensure in later steps, the user could use this title to name their application.

Implementation: It was implemented with a trivial assertEquals with JUnit 5

Test Status: **PASSED**

Test~changeLeftColor()

```
@Test
public void changeLeftColor() throws InterruptedException {
    clickOn("#leftColorPicker").type(KeyCode.LEFT).type(KeyCode.LEFT).type(KeyCode.LEFT).type(KeyCode.LEFT).type(KeyCode.ENTER);
    assertEquals(leftCircle.getFill(), Color.valueOf("#ffffb3"));
}
```

Description of Test Case: This test case automated the process of a user choosing a circle colour to apply to their circle.

Derivation: This test case was to ensure that the leftColorPicker and rightColorPicker had the correct functionality, that allowed the user to choose their circle colour.

Implementation: It was implemented with a trivial assertEquals with JUnit 5. We made sure that using that sequence of mouse movements, the correct colour was chosen.

Test Status: **PASSED**

Test~changeRightColor()

```
@Test
public void changeRightColor() throws InterruptedException {
    clickOn("#rightColorPicker").type(KeyCode.LEFT).type(KeyCode.LEFT).type(KeyCode.LEFT).type(KeyCode.LEFT).type(KeyCode.LEFT).type(KeyCode.UP).type(KeyCode.UP).type(KeyCode.UP).type(KeyCode.UP).type(KeyCode.ENTER);
    assertEquals(rightCircle.getFill(), Color.valueOf("#fb366"));
}
```

Description of Test Case: This test case automated the process of a user choosing a circle colour to apply to their circle.

Derivation: This test case was to ensure that the leftColorPicker and rightColorPicker had the correct functionality, that allowed the user to choose their circle colour.

Implementation: It was implemented with a trivial assertEquals with JUnit 5. We made sure that using that sequence of mouse movements, the correct colour was chosen.

Test Status: **PASSED**

Testing~Drag and Drop placement

```
public class VennShapeTest {  
  
    private VennShape vennShape;  
    private Circle leftCircle;  
    private Circle rightCircle;  
  
    @Before  
    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);  
    }  
}
```

Description of Test Case: This Test case was to ensure that on Creation of our VennShape Object, the leftCircle and rightCircle of our application were in their right positions.

Derivation: The motivation for this test case was to ensure that later methods could access the left and rightCircle and have access to its fields. Therefore, we needed to test to make sure vennShape had all the correct functionality.

Implementation: It was implemented with a trivial assertEquals with JUnit 5. We made sure that using that sequence of mouse movements, the correct colour was chosen.

Test Status: **PASSED**

Test~testDrag_03()

```
@Test
public void testDrag_03() throws Exception {
    clickOn("#diagramText");
    write("This is a JUnit Test");
    type(KeyCode.ENTER);
}
```

Description of Test Case: This Test ensured correct functionality on entering text into the bottom textfield and submitting it by clicking enter.

Derivation: The motivation for this test case was to ensure that the main area would be populated with a text field upon clicking enter.

Implementation: This test-case was more of a visual test and didn't use assertion, instead we wanted to ensure that we could visually see the result of this test.

The Result:

Test Status: PASSED

Test~testDrag_06()

```
@Test
public void testDrag_06() throws Exception {
    clickOn("#diagramText");
    write("This is a Junit Test SIDE ADDED YA");
    type(KeyCode.ENTER);
    Node tf = stackPane.getChildren().get(2);

    tf.setTranslateX(-174);
    Thread.sleep(1000);
    clickOn(leftCircle).drag(MouseButton.PRIMARY).dropTo(rightCircle);
    Thread.sleep(1000);
}
```

Description of Test Case: This test case was to ensure upon adding text to diagramText TextField and clicking Enter (to put the text on the Venn diagram) that the drag feature would work.

Derivation: The motivation for this test case was to automate, the drag and drop movement of a user to ensure the drag functionality was correct.

Implementation: This test was more of a visual test to make sure that a user could drag a circle starting from x=-174 (leftCircles centre) to the rightCircle.

Test Status: **PASSED**

Test~testDrag_07()

```
@Test
public void testDrag_07() throws Exception {
    clickOn("#diagramText");
    write("This is a Junit Test SIDE ADDED YA");
    type(KeyCode.ENTER);
    Node tf = stackPane.getChildren().get(2);

    tf.setTranslateX(-174);
    Thread.sleep(1000);
    clickOn(leftCircle).drag(MouseButton.PRIMARY).dropTo(rightCircle);
    Thread.sleep(1000);
    assertEquals(sideAdded.getText(), "Intersection!");
}
```

Description of Test Case: This test case ensured that our sideAdded textField could correctly identify where the textField had been placed.

Derivation: One of the most important features of our program is to alert the user on every mouse drag-release where the text field has been added. This test was intended to test the functionality of the alert.

Implementation: We implemented this test by writing text into the bottom textfield (addTextToDiagram) and submitting it by clicking enter. We tested that by dragging the textField from the leftCircle to close to the right circle (was actually the intersection) that an assertion would be called.

Test Status: **PASSED**

Test~testLeftTitle1 & testRightTitle1()

```
@Test
public void testLeftTitle1() throws Exception {
    clickOn("#leftTitle");
    write("Benefits");

    assertEquals(leftTitle.getText(), "Benefits");
}

@Test
public void testRightTitle1() throws Exception {
    clickOn("#rightTitle");
    write("Doubts");
    assertEquals(rightTitle.getText(), "Doubts");
}
```





Description of Test Case: These test cases were very similar, so we will discuss them together.

Derivation: One of the most important features of our program is to alert the user on every mouse drag-release where the text field has been added. This test was intended to test the functionality of the alert.

Implementation: We implemented this test by writing text into the bottom textfield (addTextToDiagram) and submitting it by clicking enter. We tested that by dragging the textfield from the leftCircle to close to the right circle (was actually the intersection) that an assertion would be called.

Test Status: **PASSED**

Coverage Metrics

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
▼ Venn	 36.2 %	1,161	2,042	3,203
▸ src/main/java	 20.9 %	445	1,684	2,129
▼ src/test/java	 66.7 %	716	358	1,074
▸ tests	 66.7 %	716	358	1,074

In the package tests with our tests we had 66.7% coverage of test cases.