Testing Document

for

EECS 2311

VennEver

Version 2.0

Prepared by Group 19

Muneer Nekiar

Sidharth Sudarsan

Areeba Abidi

Indermohan Gill

11/4/2020

Change Log

Version	Change Date	Description
1.0	5 March 2020	Initial documentation setup
2.0	5 April 2020	Final revision

1. Introduction

The testing strategy for the application was to examine the logical functionality of the application.

1.1 Scope

1.1.1.1 In Scope

Logical aspects of the application will be tested. Changes will be made accordingly.

1.1.1.2 Out of Scope

Event driven testing and testing the security of the application .

1.2 Quality Objective

The objective of the application is to produce an application that is capable of creating aveen diagram. The user should have the freedom to use different ways to input the data. And the logical aspect of the application should always produce a functioning venn diagram.

Some objectives of your testing project could be

- Make sure matching values between set A and and set B are printed in the overlapping area of the Venn diagram.
- All sets should be unique
- All sets should be distinct

2. Test Methodology

2.1 Overview

The logical aspect of the application will be the focus for the testing. Upon exploration of the assignment requirement, it was discovered that it is not required to test the GUI of the application.

2.2 Test Deliverables

Test Case 1: testValuesGotTransfered: check to see if the data from add Tables got converted to a list for manipulation.

Expected:

- Size of the arrayList to equal to global value keeping a track of number of elements in each set

Actual:

- Returns true. Meaning global variable is equal to the length of the new value

Test Case 2: Adding data to the diagram with one identical text.

Expected:

- The text in First Set to be inside the left circle and the text in Second Set to be in the right circle.
- The identical text to only be in the middle of the two circles.

Actual:

- Puts the text into their respected circles.
- Text that is identical appears in the middle. And is removed from the other two sets

Test Case 3: testValuesGotTransferedSet1() testing to see individual values match the values in the global array

Expected:

- Anything repeated between set1 and set2 will be added to set3/intersection.
- Specific values are found in the right arrayList

Actual:

- Returns true if a certain value that is added to the instance array is found in the correct set.

Test Case 4: testSpecificValueAddedIntersection(): testing to see individual values match the values in the global array specifically for interaction/third set. Since they can be added explicitly and implicitly.

Expected:

- Anything repeated between set1 and set2 will be added to set3/intersection.
- Third set is also part of intersection
- Specific values are found in the right arrayList

Actual:

- Returns true if a certain value that is added to both set1 and set2 the instance array is found in the intersection set or if it exists in the third set.

Test Case 5: testSpecificValueAddedIntersectionLen(): test the length of the intersection array based on the explicit arrays. This means that intersection length = the length of interaction + third array

Expected:

- Anything repeated between set1 and set2 will be added to set3/intersection.
- Third set is also part of intersection
- Specific values are found in the right arrayList
- Length of intersection = intersection + third array

Actual:

- Returns true if a length of intersection = the length of this instance's third set + intersection.

Test Case 6: checkThirdArray(): check to see if something added third array, will appear in intersection

Expected:

- Anything repeated between set1 and set2 will be added to set3/intersection.
- Third set is also part of intersection
- Specific values are found in the right arrayList
- Third array values will appear in intersection

Actual:

- Returns true if a certain third array value is in the intersection.

Test Case 7: afterIntersectionSet1(): check to see if the intersection value is removed from set1

Expected**:**

- Anything repeated between set1 and set2 will be added to set3/intersection.
- Third set is also part of intersection
- Specific values are found in the right arrayList
- Set1 will not value the intersecting value

Actual**:**

- Returns true if a certain value is not in set1 because it is now in intersection.

Test Case 8: checkSet2Contains():

Expected**:**

- Anything repeated between set1 and set2 will be added to set3/intersection.
- Third set is also part of intersection
- Specific values are found in the right arrayList
- Set1 will not value the intersecting value
- If a value is only present in the second set expected = true

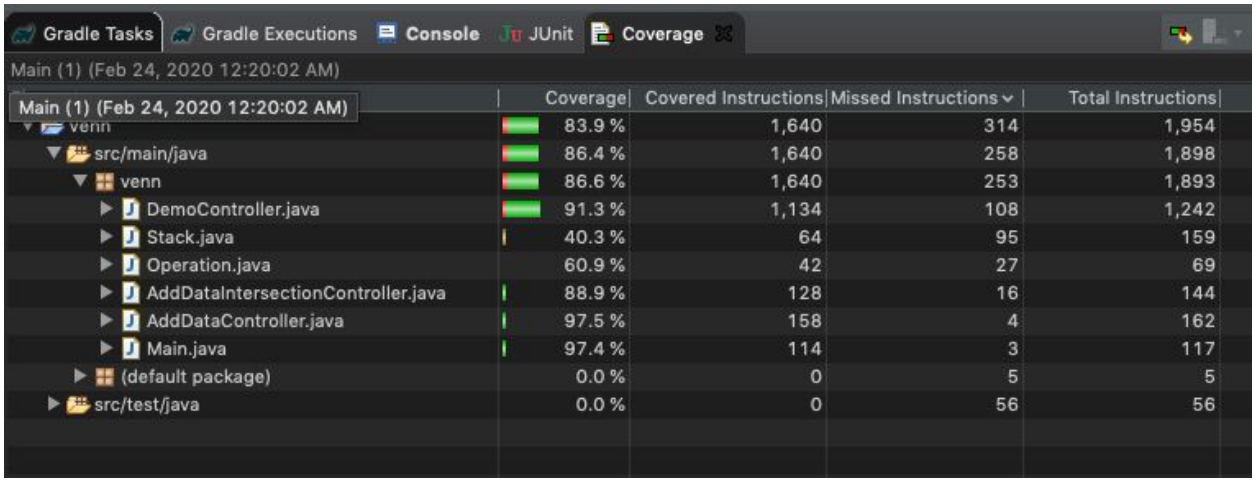
Actual

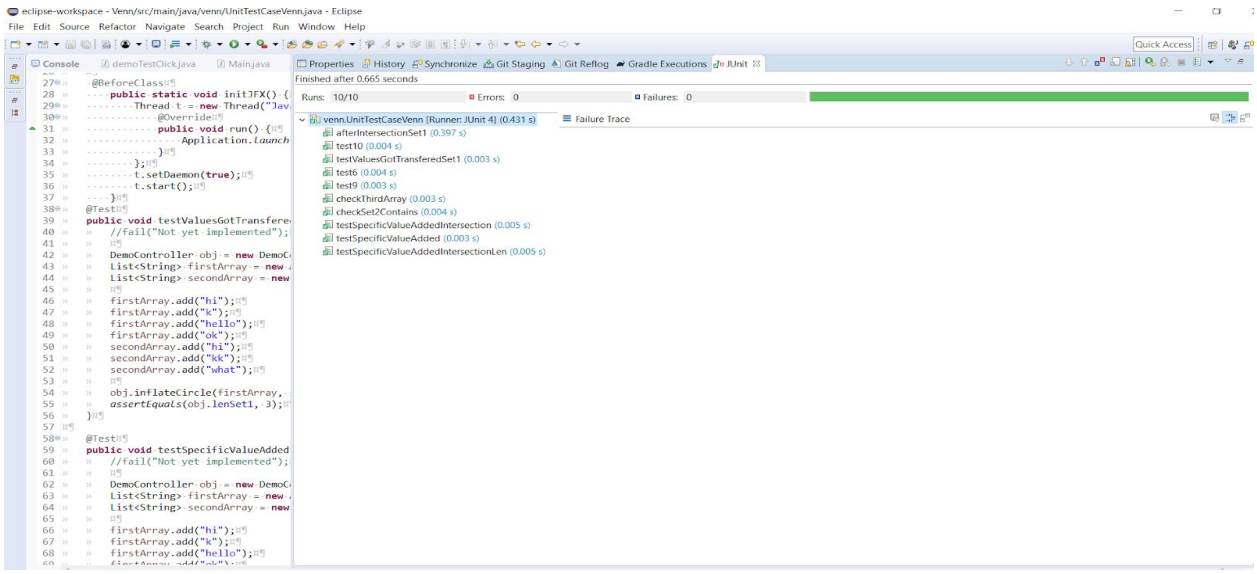
- Returns true if set2 contains a specific value

2.3 Test Completeness

Here you define the criterias that will deem your testing complete.
For instance, a few criteria to check Test Completeness would be

- 100% test coverage
- All Manual & Automated Test cases executed
- All open bugs are fixed or will be fixed in next release


Gradle Tasks
Gradle Executions
Console
JUnit
Coverage
Main (1) (Feb 24, 2020 12:20:02 AM)
Main (1) (Feb 24, 2020 12:20:02 AM)
Coverage
Covered Instructions
Missed Instructions
Total Instructions
venn
83.9 %
1,640
314
1,954
src/main/java
86.4 %
1,640
258
1,898
venn
86.6 %
1,640
253
1,893
DemoController.java
91.3 %
1,134
108
1,242
Stack.java
40.3 %
64
95
159
Operation.java
60.9 %
42
27
69
AddDataIntersectionController.java
88.9 %
128
16
144
AddDataController.java
97.5 %
158
4
162
Main.java
97.4 %
114
3
117
(default package)
0.0 %
0
5
5
src/test/java
0.0 %
0
56
56


eclipse-workspace - Venn/src/main/java/venn/UnitTestCaseVenn.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Console
demoTestClick.java
Main.java
Properties
History
Synchronize
Git Staging
Git Reflog
Gradle Executions
JUnit
Finished after 0.665 seconds
Runs: 10/10
Errors: 0
Failures: 0
venn.UnitTestCaseVenn [Runner: JUnit 4] (0.431 s)
afterIntersectionSet1 (0.397 s)
test10 (0.004 s)
testValuesGotTransferedSet1 (0.003 s)
test6 (0.004 s)
test9 (0.003 s)
checkThirdArray (0.003 s)
checkSet2Contains (0.004 s)
testSpecificValueAddedIntersection (0.005 s)
testSpecificValueAdded (0.003 s)
testSpecificValueAddedIntersectionLen (0.005 s)
Failure Trace

2.4 Test Environment

It mentions the minimum hardware requirements that will be used to test the Application.

Following software's are required in addition to client-specific software.

- Windows 8 and above
- Eclipse
- JUnit library

Conclusion

The test cases provided above test everything functional in the Venn Diagram tool. The test cases check the core functionality of how a Venn Diagram tool should work. What happens when there is a matching text is in both sets and or if data is added explicitly the third set consists of its own values plus the intersection values. Another thing tested is the ability of the diagram to retain data and be able to add new data without any issues. All in all, these test cases are sufficient in testing our tool for errors and bugs.