Test Document of Venn Diagram

EECS2311 W20

GRB #11

SUBMITTED BY: Tianpei Liao
Arda Temel
Lynn Al Agilly
Jeyaprashanth Sivasubramaniam

YORK UNIVERSITY

Table of Contents

Introduction 2

Implemented Classes 2

Testing Implementation 4

Coverage 7

Summary 9

Introduction

Venn Diagram application provides a user-friendly interface with a Venn diagram that allows users to create and update entries and customize within the Venn diagram. The system provides a wide range of features such as adding/removing data entries, updating existing entries' attributes (color, shape, text), drag and drop and categorize the entries within the diagram, Undo and Redo and important feature Test mode. We tested to make sure the software did not crash under simple user mistakes and run time. The total testing coverage from the JUnit test cases is around 83.4**%** but there have been several manual tests to ensure the functionality of the user interface is flexible across different OS platforms such as Windows and Mac OS and also our system has distinguished interface and functions.

Implemented Classes

Major Classes	**Description**
Main	Start application, initialize contents, Load Main Scene, Add New Entry, Entry templates, Configure buttons and images
VennController	Configure Venn diagram contents and initialize circle and Control mouse events, key events and import multiple new entries
EditController	Edit existing entries and its attributes
GetDataController	Create new entries, changes Sample Preview label when creating new entry
SetCircle	Configure the circle attributes and methods
DraggableText	Configure the draggable text, text's color, setting tooltip, Configure the mouse events

Action	Interface class for undo and redo functions. Which have abstract methods such as execute (), undo () and getName ()
CommandManager	Control class for undo and redo methods. It has all implemented methods inherited from parent class.
Add, Delete and Drag	Model classes which have inherited methods from its parent Action interface class. They have execute (), undo () and getName () methods and also class constructor.
SaveLoad	Controller class which has partial method which are important for main window action methods. All import, export, save and load methods are implemented in this class.
QueueStack<T>	Model class which has stack methods such as push (), pop () and clear ().
App.fxml, getData.fxml and EditText.fxml	Those .fxml file are constructed by scene builder, which implements the layout and the looking of Venn diagram and also provides mouse clicking linking to function in java.

Testing Implementation

The following test cases were derived through examining various attributes of the application.

1. **TC: Main GUI visual presentation and all UI contents in fixed layout**
   Users successfully launch the application without any error and menu item, buttons, Venn diagram and its colour contents are properly visible.

   When a user double-clicks the app/executable jar file the system launches the main window within a recent amount of time. In MainTest class a method called testWidgets () tests the contents in the main window.

2. **TC: Successfully add new entry**
   The user clicks the "plus" sign icon button in the main window then a second window called "Add New Entry "will popup through that user input text, color, and shape of the entry then click the create button. When a user input a new text, color for new entry the samplePreview label also changes accordingly. Created entries will be visible in the main window.

   Using GUI testFx under MainTest class an implemented testGetData () method automates this process and creates a single new entry and is added to the main window. Automated actions are done by FxRobot javafx library.

3. **TC: User does not input text for new entry**
   Users are responsible for input text for new entries. When a user does not input the text and try to create the new entry using the create button or **ENTER** key pressing the system won't allow the creation.

   Using GUI testFx we tested by without writing anything in the text field and clicking the create button and ENTER key press.

4. **TC: Successfully edit the existing entry**
   The user double clicks the existing entry in the main window then a second window called "Edit New Entry" will popup through that user edit the text, color, and shape of the entry then click the edit button. When a user edits the text, color for new entry the samplePreview label also changes accordingly. Edited entries will be updated in the main window.

Using GUI testFx under MainTest class an implemented testEditData () method automates this process and edit the existing selected single new entry and is updated in the main window. Automated actions are done by FxRobot javafx library.

5. **TC: Successfully delete the existing entry**
   User drag and drop the existing entry on top of the delete button icon then the released entry will be deleted from Draggable text array list and removed from main window.

   Using GUI testFx under MainTest class an implemented testDeleteEntry () method automates this process. Create an entry and drag and drop on top of the delete icon button and deleted entry removed from the main window.

6. **TC: User drag existing entry across the windows, over the Venn diagram area**
   User drag and drop the existing entry across the screen window and over the Venn diagram circle area. When dragged object over the circle area the receiving circle are should lit up and corresponding area's #elements count will be increased. And when dragged object remove from the circle area that area should lit down and corresponding area #elements count will be decreased.

   Using GUI testFx under MainTest class an implemented testDraggedEntry () method automates this process.

7. **TC: Prevent entries overlapping**
   Because users can have more than one entry they possibly drag and drop the existing entry on top of another entry. When overlapping happened the overlapped object auto arranged next to the previous entry.

   Using GUI testFx under MainTest class an implemented testDraggedEntry () method automates this process. When overlapping happened from Venn Controller event handler method triggered and prevented from two objects overlapping. And get the previous entry scene location and arrange the dragged object next to another entry.

8. **TC: Add multiple entries from file import**
   Users import more than one entry by file import from .txt format files. Select import option from file menu and open selected file from file opener. System will read the imported file and input the entries to the main window next to each other with default color and shape.

9. **TC: User dragged the created entry out of window**
   When a user drags and drop the entry outside the screen size the system restricts that entry from dragging outside the window.

   Using GUI testFx under MainTest () class an implemented boudaryTest () method automates this process. This test creates four new entry labels and drag each entry using a mouse event outside of the window and check this by all sides such as right, left, up and down.

   We tested this feature completely by manual and using GUI testFx under MainTest class an implemented testImportedData () method automates this process and test up to open file explorer for import.

10. **TC: User undo import/add action**
    After user's importing or adding items, if the user wants to undo the action by pressing z keyword on keyboard. The function undo () in CommandManager class has been called, where item has been removed from list, but on the stack (), but since the system stores two QueueStack on Command manager, the user is able to redo the action.

11. **TC: User redo import/add action**
    Same as undo, if the user redo the add/import items action, the user will gain the previous item which just been deleted, The function redo() in CommandManager class has been called, where item has been removed from list, but on the stack(), but since the system stores two QueueStack on Command manager, the user is able to redo the action.

12. **TC: User upload the answer sets**
    User can import user sets from text file and start Test mode. Once user import the file by clicking "Get Labels" button, system will display informative messages about adding additional answer sets and how to delete them. Uploaded labels will be visible in main window.

13. **TC: User can submit the answers for evaluation**
    User can submit the final answer for evaluation. If all the submitted answers are correct system will display success message. Otherwise system will notify the user by changing the background color of the labels. Correct → Green and Wrong → Red.

14. **TC: User can add additional answer set for the Test**
    User able to append additional answer sets with previous ones. Using edit menu "Add Answers "option insert new set of answer with previous from text file.

15. **TC: Delete answer sets and exit from the Test mode**
    User can delete the answer set and exit from the Test mode. Once user delete the answer sets the labels in the main screen will be deleted and system will move to normal state.

16. **TC: User can save the current state of the system**
    User able to save the current entries details to text file. The existing entries text, colour, size and location will be saved in text file. If user save the elements in same text file it will be replaced by new update.

17. **TC: User can load the previous state of the diagram**
    User able to load the previous work to the system and continue their editing. Using file menu load option, they can import the text file. User cannot load the empty file and also loaded file should be in correct format. (should have all the parameter values)

18. **TC: User can export the current system state as image file**
    User can export the system state as image file .png format. From file menu, export option they can export the current screen as image file

19. **TC: User can view Help document page**
    User able to get information about complete system and its implementation details and documentation through "About" menu. It will open latest product release web page and it contains all necessary information.

Coverage

Main: responsible for initializing and loading widgets and components of the main screen window and triggered the related controller classes and methods to launch the main GUI "App.fxml". In the main class we have two methods showAddStage () and showEditStage () both are responsible for launching the "AddNewEntry" and Edit Data" UI. Testing only missed exception handling branches. Hence our overall coverage is about

And in GetDataController class there are methods such as create Text () for creating new entries and changePrev () method for changing the SamplePreview**. 76.8%** for the main class.

VennController: handles the initialization and loads the controller events and action listeners when the main window launches. Whenever a user interacts with main windows components events handlers are triggered and responded. It has the main method Initializations () which will set the Venn diagram components such as circle size, name, color, border thickness etc. and also sets SetCircle class properties.

Some of the if statements branch and CaptureData () method which import multiple entries from file import and exportData () which is export existing entries into text file are not covered by our implemented test cases. Still we have obtained **83.5%**

DraggableText: is responsible for all draggable text setters and getters methods. Handle all mouse and key events. Tooltip handling. Also handle the properties of draggable text objects such as background color, border radius, text. Prevent entries drag and drop Collison. We have **89.0%** for this class.

SetCircle: is responsible for implementing the circle property and initializing the SetCircle components when the main GUI launches. This class is tested when we test the main class using GUI test. It has only circle attributes which are used to set circle properties. So, we have **58.2%** coverage for this class.

GetDataController: is responsible for creating a new entry. When a user clicks the create a button createText () method will be triggered and created text implement in the main window. And also, it has a changePrev () method to change SamplePreview label according to user input. Here also we missed some if branches only. We have **96.2%** coverage for this class.

EditController: is responsible for editing existing entries. When a user clicks the edit button an edit () method will be triggered and updated entry implemented in the main window. And also, it has a changePrev () method to change SamplePreview label according to user input. Here also we missed some if branches only. We have **92.4%** coverage for this class.

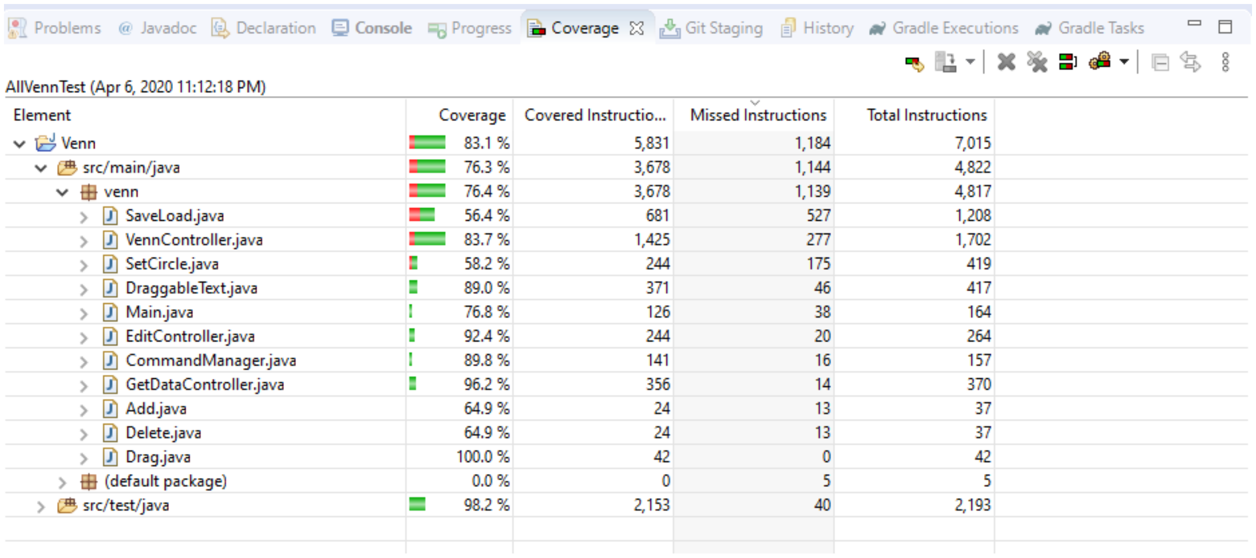
CommandManager: is responsible for keep record of user action history and record each command which are executed by user. When user execute commands CommandManager class record those events in QueueStack list and also record each action in list whenever user undo or redo it will execute previous or forward command in the list. Here also we missed some if branches only. We have **89.8%** coverage for this class.

Add, Delete and Drag: these are model classes responsible for undo and redo operations. Specifically handle when add, delete, and drag operations are operated. Coverage for this classes are. Add → **64.9%,** Delete → **100.0%,** Drag → **100.0%**

SaveLoad: This class responsible for all import, export, save and load functions. Whenever user make above mentioned operations VennController class initial methods will call the associated methods in this class. We have **56.5%** coverage for this class.

Summary

Our current system is completely tested automatically and manually. GUI and functions tests are sufficiently enough that proves current systems working in acceptable conditions. And also, we have an overall **83.4% Test Coverage** rate. The amount of missing coverage percentage is only because of some unreachable branches in exceptions and the classes which are implemented with methods have file chooser coding. So those missed branches are manually tested to increase the quality of the product. We fixed more than **95%** of bugs from previous release. As the program currently stands it is impossible for to unexpectedly terminate, behave unexpectedly or provide different output for above given functions and features.



AllVennTest (Apr 6, 2020 11:12:18 PM)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
Venn	83.1 %	5,831	1,184	7,015
src/main/java	76.3 %	3,678	1,144	4,822
venn	76.4 %	3,678	1,139	4,817
SaveLoad.java	56.4 %	681	527	1,208
VennController.java	83.7 %	1,425	277	1,702
SetCircle.java	58.2 %	244	175	419
DraggableText.java	89.0 %	371	46	417
Main.java	76.8 %	126	38	164
EditController.java	92.4 %	244	20	264
CommandManager.java	89.8 %	141	16	157
GetDataController.java	96.2 %	356	14	370
Add.java	64.9 %	24	13	37
Delete.java	64.9 %	24	13	37
Drag.java	100.0 %	42	0	42
(default package)	0.0 %	0	5	5
src/test/java	98.2 %	2,153	40	2,193