Testing Document

Group 4

Andrew Hocking (215752835)
Nourin Abd El Hadi (216107021)
Anika Prova (216474306)
Nabi Khalid (216441677)

Table of Contents

Overview of Testing JavaFX Applications with TestFX

Venn was tested using TestFX; an open source GUI testing framework for JavaFX-based applications. TestFX is a relatively new testing framework that is still in its alpha stage of development, however it is very stable in its current state and provides a convenient and user-friendly solution to perform automated unit tests for the GUI of JavaFX applications.

Configuring TestFX

We used Gradle to add a dependency for TestFX in our project. This has added two jar files to our project: *testfx-junit-4.0.15-alpha.jar* and *testfx-core-4.0.15-alpha.jar*.

Writing the ControllerTest Class for our Controller Class

A TestFX test class must extend the TestFX class ApplicationTest. This requires the implementation of the *start* method. TestFX works with JUnit. We have an *@Before* method called *setUpClass* which creates a new ControllerTest object and runs the *ApplicationTest.launch* method, which starts the application. We also have an *@After* method called *teardown*, which closes the application window and releases any keys or mouse buttons that may have been simulated during the tests.

We also have a helper method called *find* which uses the very useful *lookup* method from TestFX that can return any Node visible on screen while the application is running based on either its FXID or based on a String. For example, a button labelled “Save” with an ID of “saveButton” could be found using either *find(“Save”)* or *find(“#saveButton”)*.

Unit Tests

testAddingItems()

This test tests the action of adding items to the diagram. The test simulates the user clicking on the text field above the item list, typing in 10 items and hitting *enter* after each, and dragging 9 of them into the diagram. It then checks whether the items were dragged into their expected locations in the diagram (left circle, right circle, or intersection), and whether the one item that was not dragged is still in the item list.

testCircleSize()

This test tests the action of changing the size of the circles in the diagram. The test simulates the user opening the settings panel and dragging the sliders to change the size, and checks whether the circle has been resized accordingly. It then simulates typing numbers in the text

field and checks the same. It also simulates putting invalid text into the text field and checks whether that has been properly handled.

testColorPickers()

This test tests the action of changing all the colours in the diagram. The test simulates the user adding three items to the diagram, opening the settings panel, clicking all the colour pickers, and selecting random values for each. It then checks whether the items dragged into the diagram are the correct colours, the background is the correct colour, the circles are the correct colours, and the titles are the correct colour based on the new colours picked.

testDeleteButton()

This test tests the action of deleting items from the diagram and from the item list. The test simulates the user adding 3 items to the item list and dragging 2 of them into the diagram. It clicks on the item still in the list and clicks the delete button, then checks whether it was removed from the list. Then it clicks on each of the other two items and clicks the delete button, checking that the items were deleted.

testDetailView()

This test tests the action of changing the title and description of an item in the diagram. The test simulates the user adding an item to the diagram, double clicking on it, changing the title and description of the item, and clicking save. It then checks whether the title and were properly saved. Then, it simulates double clicking on the item, changing the title and description, and clicking cancel. It then ensures that the title and description of the item were not changed.

testKeyboardShortcuts()

This test tests that all the keyboard shortcuts in the diagram work. It simulates a user pressing all the keyboard shortcuts available and checks whether the desired effect was achieved for each one.

testMovingItems()

This test tests the action of moving items within the diagram. The test simulates the user adding one item to the diagram, and dragging it to different locations. It then checks whether the items were dragged into their expected locations in the diagram (left circle, right circle, or intersection) after each drag.

testRemoveButton()

This test tests the action of removing items from the diagram. The test simulates the user adding three items to the diagram, clicking on one, and clicking the remove button. Then it ensures the item is no longer in the diagram and is back in the list. It then simulates selecting both the other items at once and clicking the remove button. It then ensures that these two items are also no longer in the diagram and are back in the list.

testTitles()

This test tests the action of adding titles to the diagram. The test simulates the user clicking on the text fields above the diagram and typing titles for each. It then checks that the titles have been saved.

testUndoRedo()

This test tests the undo and redo function for every action in the diagram. The test simulates the user adding items, changing its title and description, moving items, removing items, deleting items, changing colours, changing circle sizes, and changing titles. It simulates the user undoing and redoing all these actions and ensures the actions are properly inverted and reverted.

Testing Coverage

Our testing currently covers 59.2% of the code. The code that is not covered falls into one of three categories: untaken catch statements, code that references files, and code that need not be tested.

The untaken catch statements are mostly catch statements that are there just in case, but are never likely to be executed. For example, we handle the FileNotFoundException when referencing files we know for a fact to exist, so these catch statements are never used.

The code that references files includes any code that handles images that would be dragged into the diagram, the code for saving or loading files, and the code for importing and exporting files. There is unfortunately no way to automate these tests in a sensible way using TestFX, so they have been thoroughly tested manually. Hopefully in the future TestFX provides a better way to test this type of action.

There are only two examples of code that need not be tested: the code that shows the About screen, and the code that opens the link to the online user manual. This code is very basic and is easily tested by simply clicking on the menu items. It is unnecessary to devote time to testing these two simple features.