EECS2311: Software Development Project

Winter 2020

Design Document

VennEver

Group 19

Muneer Nekiar

Sidharth Sudarsan

Areeba Abidi

Indermohan Gill

22 March 2020

1. Introduction

1.1 Purpose

Venn diagrams are used to display data sets in a visual manner. This software's purpose is to generate a diagram representing mathematical or logical sets pictorially as circles, common elements of the sets being represented by the areas of overlap among the circles. This document exists so developers can improve on this application.

1.2 Scope

VennEver is to help users represent sets of data. Given two data sets the application is capable of finding similar data and displaying it in a venn diagram format. The application can also display data if the user wants to add their own intersection data. The application is also capable of displaying the given data sent as numbers.

1.3 Definitions, Acronyms, Abbreviations

VennEver - Name of the application
Canvas - The AnchorPane containing the circles

2. System Operation

Figure 1 and *Figure 2* are sequence diagrams for key use cases of the application

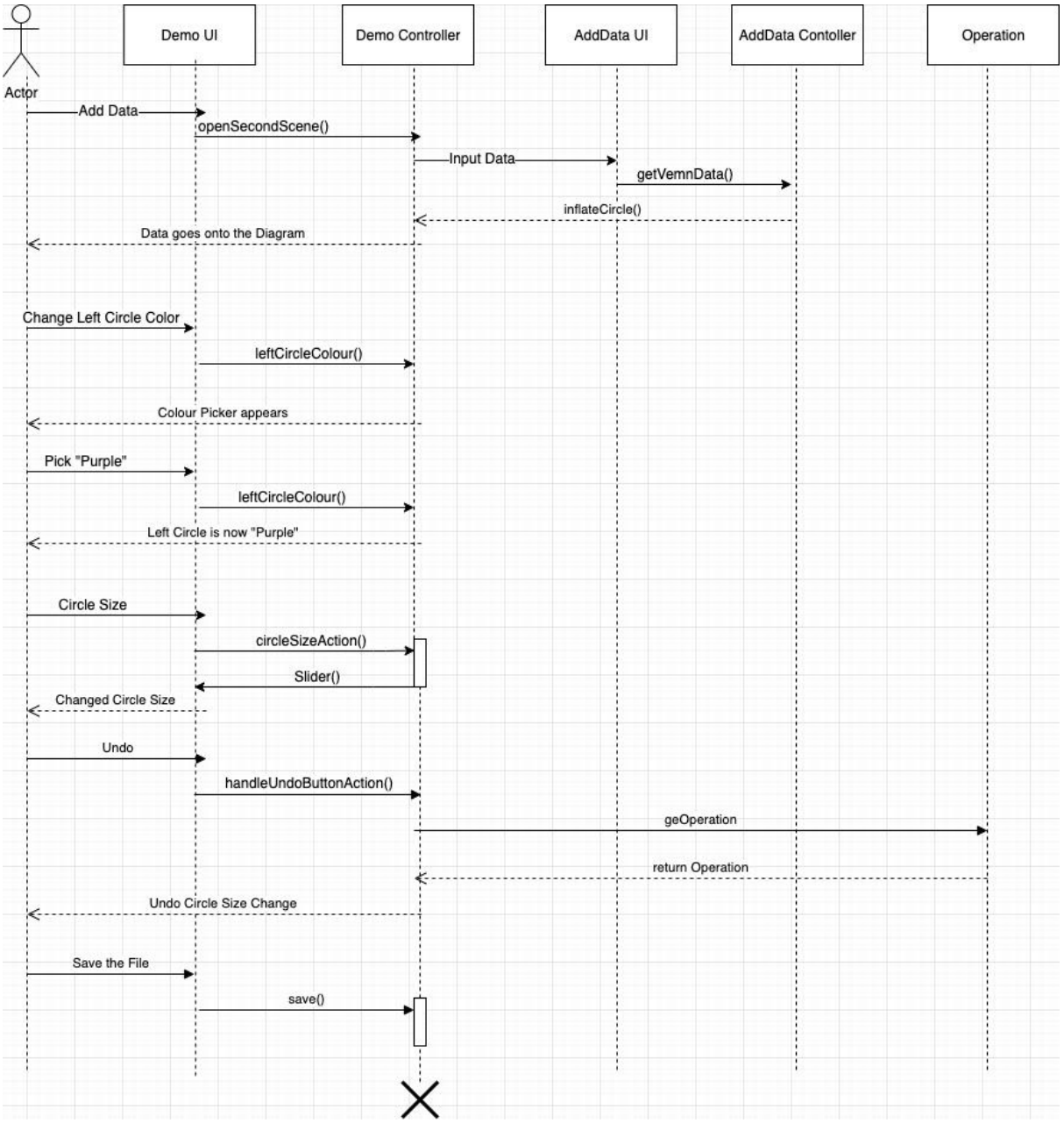

Actor
Demo UI
Demo Controller
AddData UI
AddData Contoller
Operation
Add Data
openSecondScene()
Input Data
getVemnData()
inflateCircle()
Data goes onto the Diagram
Change Left Circle Color
leftCircleColour()
Colour Picker appears
Pick "Purple"
leftCircleColour()
Left Circle is now "Purple"
Circle Size
circleSizeAction()
Slider()
Changed Circle Size
Undo
handleUndoButtonAction()
geOperation
return Operation
Undo Circle Size Change
Save the File
save()

Figure 1

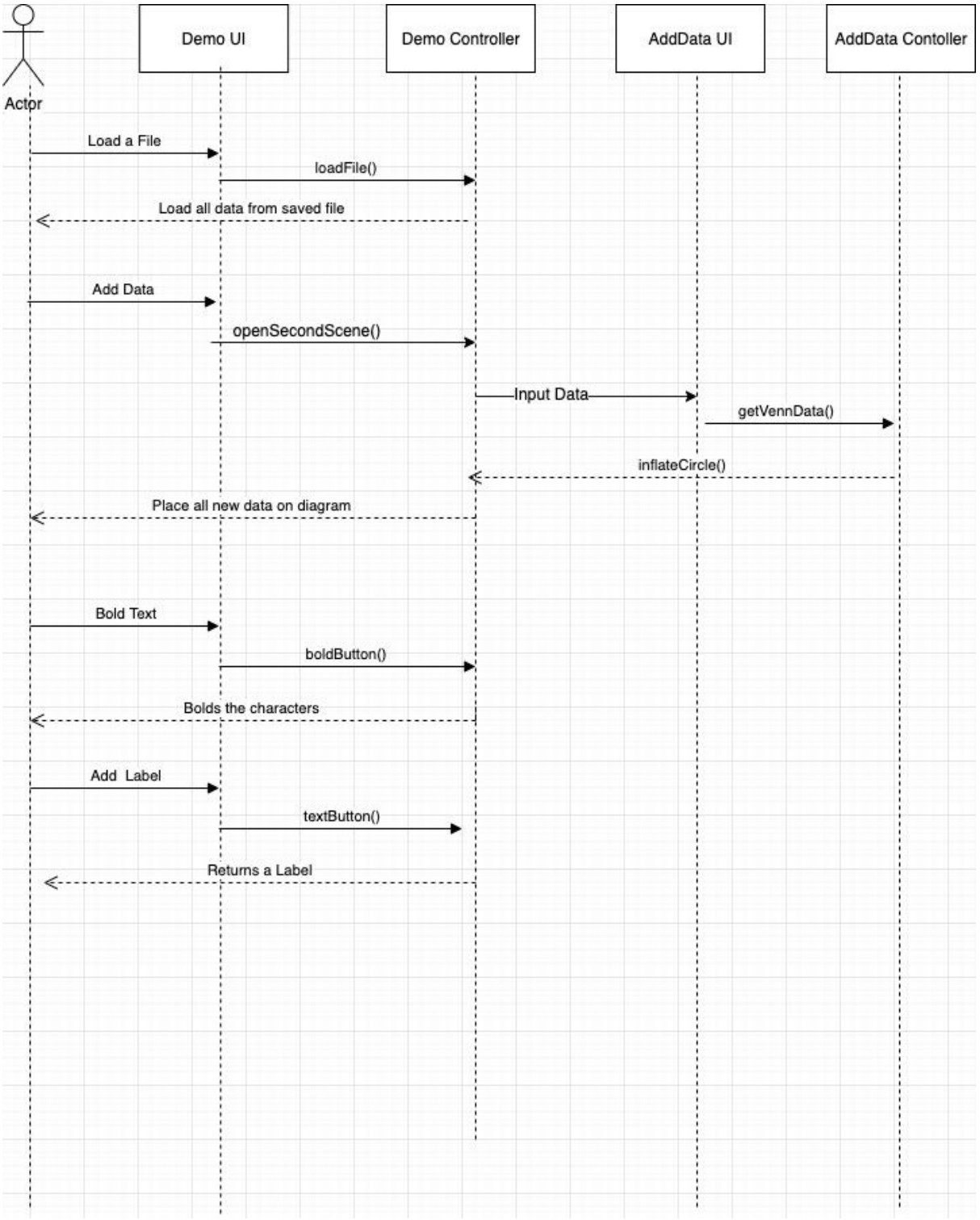
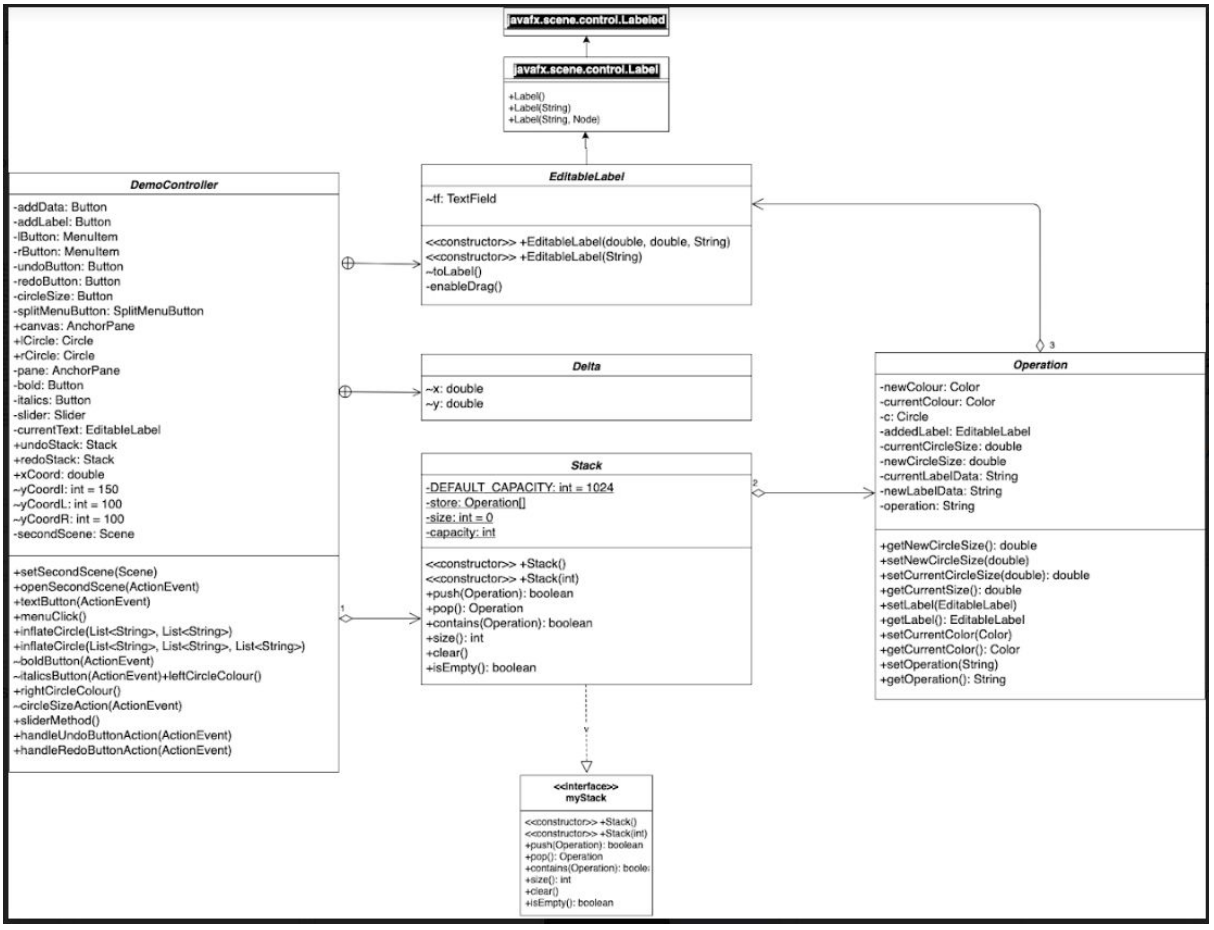

Actor
Demo UI
Demo Controller
AddData UI
AddData Contoller
Load a File
loadFile()
Load all data from saved file
Add Data
openSecondScene()
Input Data
getVennData()
inflateCircle()
Place all new data on diagram
Bold Text
boldButton()
Bolds the characters
Add Label
textButton()
Returns a Label

Figure 2

3. VennEver

Figure 3 and Figure 4 depicts the UML model for the VennEver Interface.
Figure 3 depicts the UML model for demoController. Figure 4 depicts the UML model for the addDataController. Most of the methods in these class diagrams represent callback functions for user input events.

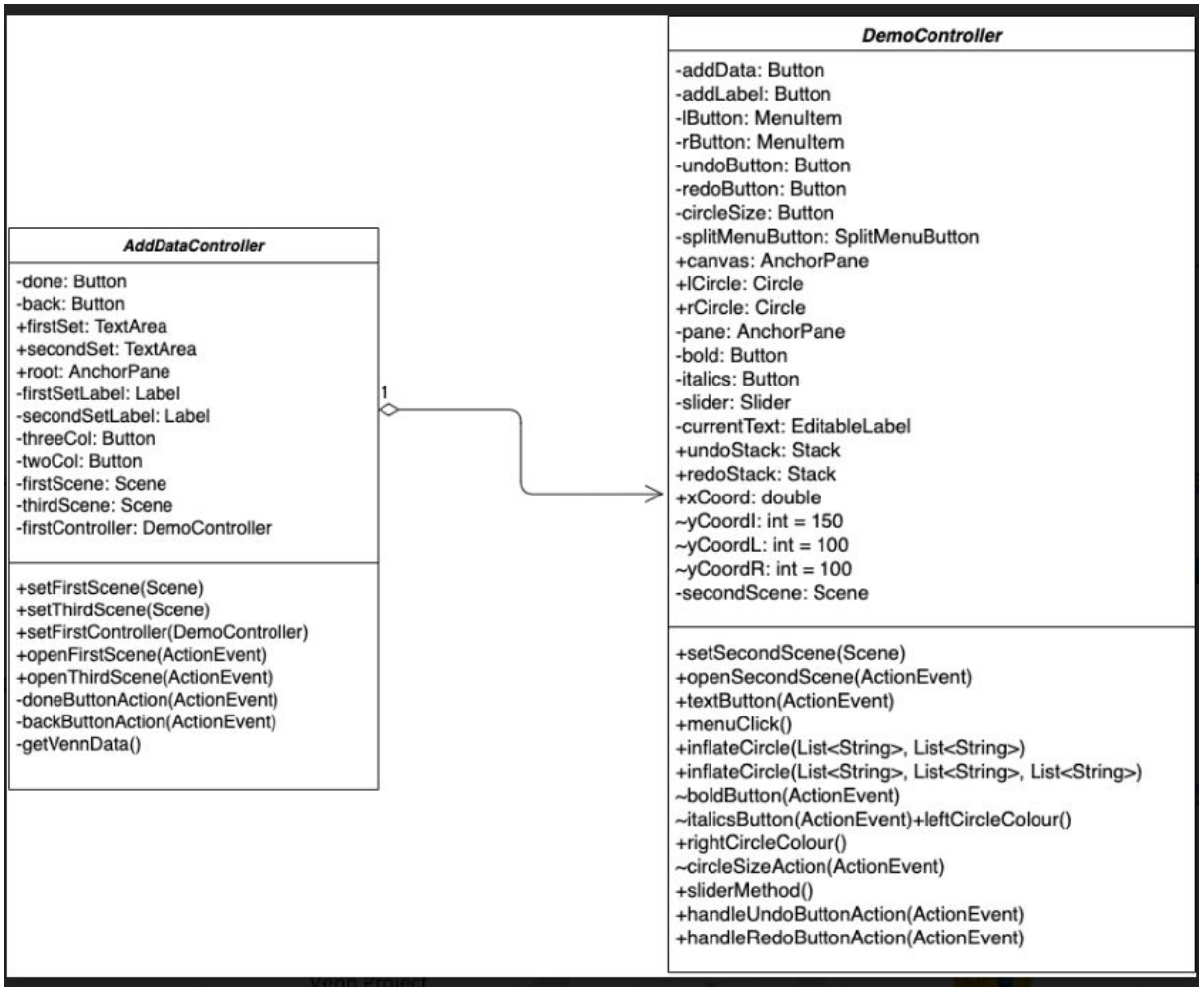


inflateCircle(List<String> first, List<String> second)	
Input:	Two String lists
Output:	Void
Description:	After the ‘ADD DATA’ button lets the user add data set values, it creates 2 lists. The inflateCircle function takes the two lists as an input, creates text boxes for each element in the lists, Any matching elements between the 2 lists will be added to a new list. Then the list elements are displayed on to the appropriate circle section. This method also sets global variables for later use.

handleUndoButtonAction(ActionEvent e)	
Input:	Onclick event. When Undo is pressed
Output:	Void
Description:	Undo’s the last event that occured on the canvas of the application.

public void leftCircleColour()	
Input:	
Output:	Void
Description:	Using the color picker the user can select a color for the left circle, the method sets the value of the new color, and the color is changed.

circleSizeAction(ActionEvent event)	
Input:	Onclick event. When ‘CIRCLE SIZE’ is pressed
Output:	Void
Description:	Upon click a slider appears, the circleSizeAction calls the slider method which deals with changing the size of circle



getVennData()	
Input:	
Output:	Void
Description:	Grabs the data from the user input, sets them to a string variable then breaks apart the variable at each ‘/n’ and adds it to 2/3 seperate lists. Calls on other methods to change or add data to the original sense, then. Sets back the original scene.

4. Maintenance Scenarios

The following is a perfective maintenance scenario:

Support three sets in the venn diagram

The application is not intended to be used in case of more than three sets of data since venn diagrams become complex and unreadable in that case. However, a third set can be supported in the application by following these steps.

1. *Demo.fxml* is the XML file that composes the GUI of the main editor view. Edit this file appropriately to add the third circle and give it an *fx id* on SceneBuilder.
2. Add this circle as a field in *DemoController* (controller class of *Demo.fxml*) with the variable name as its *fx id*.

To make the third circle functional, the controls with respect to its colour, size and data need to be tweaked.

3. **Colour**: Add the third circle to the colour split menu by editing *menuClick()* in *DemoController*. Add a *thirdCircleColour()* very similar to the corresponding method for the other two circles.
4. **Size**: Edit the *sliderMethod()* in *DemoController* to set the coordinates of the third circle appropriately on being tweaked.
5. **Data**: The *AddData.fxml* is the XML file for the view that adds data to the diagram. Add a third field to receive data for that set in this file. Update the *AddDataController* class accordingly to support these additions. The *inflateCircle(List<String> first, List <String> second)* method in *DemoController* populates the circle with received data. This needs to be adjusted to compute the intersection sets appropriately. The coordinates for populating the data in this method also need to be tweaked to ensure smooth support of three data sets.

The following is an adaptive maintenance scenario:

Changing the JDK of the Project

The application's JDK may not have been downloaded by everyone causing an error when launching the app. To fix this error, one must be able to change the JDK on the project.

1. Right-click on the project and hover over "Build Path", then click "Configure Build Path".
2. Click on the JRE System Library then click "Edit" on the right of the window.
3. Select "Alternate JRE" and then click "Installed JREs..." in its right.
4. Choose the JRE you have installed and click "Apply and Close" and choose "Apply and Close" until you return back to the first window, then click "Finish".

These changes will require changes to be made to the build.gradle file.

5. Now open the build.gradle file from the Package Explorer.
6. Navigate to the java tag on the build.gradle file and set the sourceCompatibility and targetCompatibility accordingly to the version you wish to upgrade to.