Design Documentation for Venn Diagram Application

Release Version: 2.0

Release date:

Group 4

Table of Contents

Section 1: Class Diagram of the Venn Diagram

Section 2: Sequence Diagram for Activity 1

Section 3: Sequence Diagram Activity 2

Section 4: Sequence Diagram Activity 3

Section 5: Sequence Diagram Activity 4

Section 6: Sequence Diagram Activity 5

Section 7: Software Maintenance Scenarios

Section 1: Class Diagram of the Venn Diagram

This UML Diagram shows the main classes of the appplication and their methods. Main class contains the UX application window.

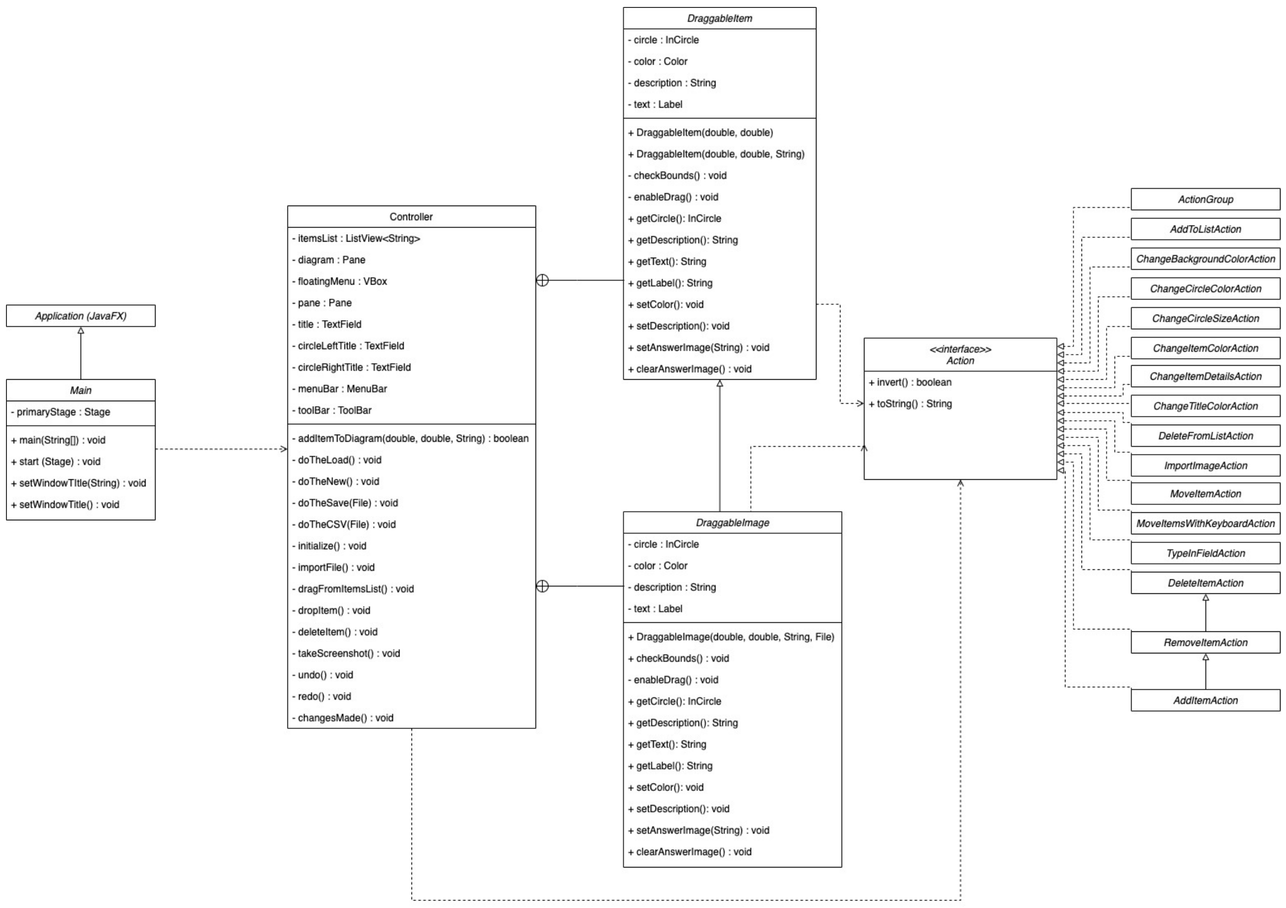
The Controller class contains all the methods required to functionalise the features of the app. The attributes of the controller class consists of the UI components. The methods use or alter these attributes depending on the feature we are trying to execute. This diagram contains the key methods.

The Main class has directional connection with the Controller class. It is a has-a relationship and depicts aggregation.

The controller class calls the DraggableItem class. It depicts a 'has-a' relationship (aggregation).

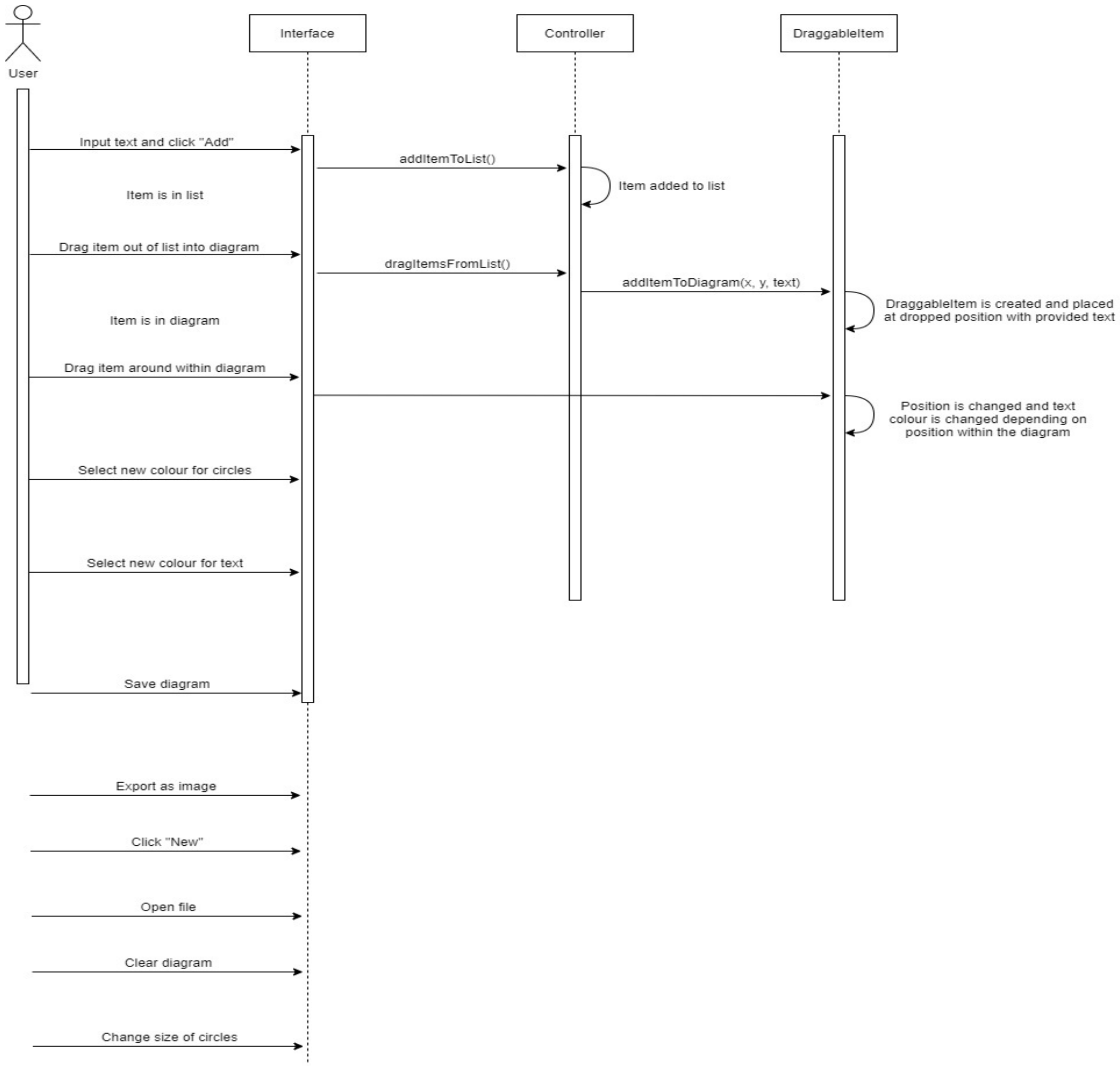
DraggableItem class extends the StackPane class in javaFX. The stackpane class is used for alighing the nodes within the stack pane.

Draggable item class is used to create draggable textbox objects that can be dragged and dropped into the diagram. The DraggableItem instances have 3 customizable constructors and one default contructor that can be used to create draggable textboxes within the UI.



Section 2: Sequence Diagram for Activity 1

The following Sequence diagram depicts the sequence of inputting text in the list, and then drag the text in the VennDiagram.


User
Interface
Controller
DraggableItem
Input text and click "Add"
addItemToList()
Item added to list
Item is in list
Drag item out of list into diagram
dragItemsFromList()
addItemToDiagram(x, y, text)
DraggableItem is created and placed at dropped position with provided text
Item is in diagram
Drag item around within diagram
Position is changed and text colour is changed depending on position within the diagram
Select new colour for circles
Select new colour for text
Save diagram
Export as image
Click "New"
Open file
Clear diagram
Change size of circles

User and Interface

User inputs texts and clicks add to insert item into the list.
User can drag item out of the list and into the diagram or can drag item out of the list and around within the diagram.

Interface and Controller

addItemtoList method is called for the first action.
dragItemFromList method is called for the second action.

<u>Returns</u>: item is added to the list

Controller and DraggableItem

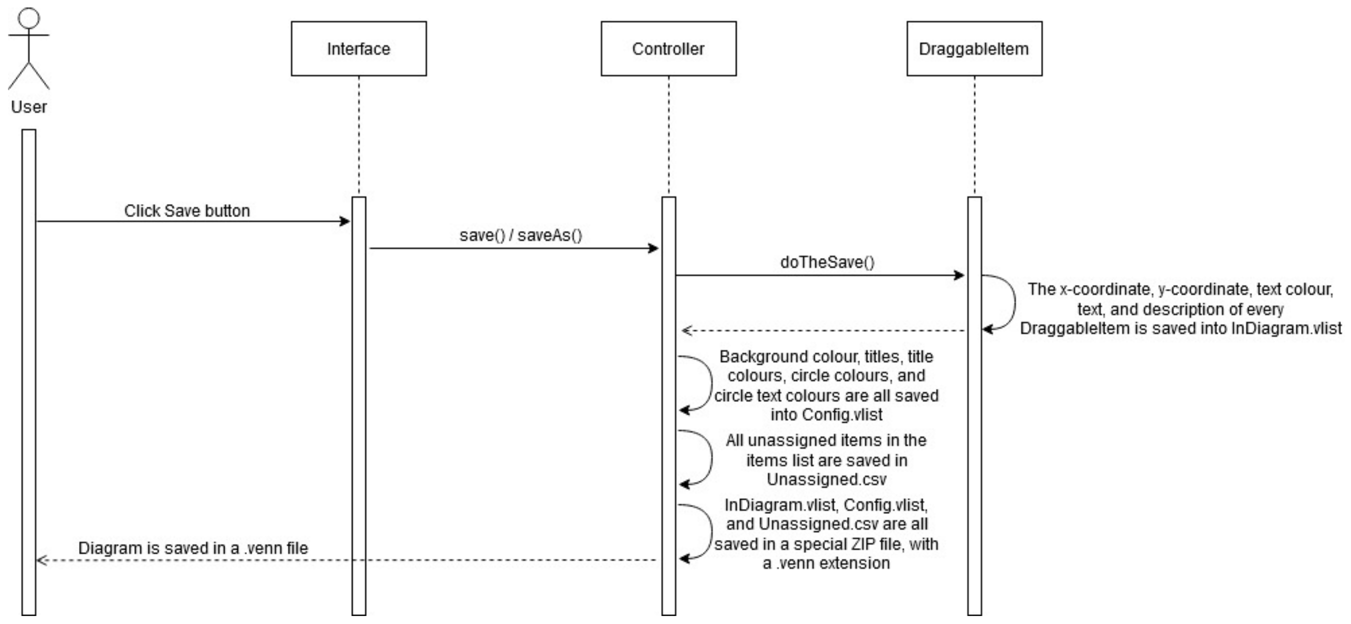
A draggable item instance is created using the customizable constructor that has height, width and text parameters to passed in.

<u>Returns</u>: Draggble item is created and placed at dropped position with the text.

Position is changed depending on the location within the diagram.

Section 3: Sequence Diagram Activity 2

The following Diagram depicts the sequence of Save As and/or Save function in the Venn Diagram. It shows the stages of method calls in various classes after the user selects to perform these functions from the interface.


User
Interface
Controller
DraggableItem
Click Save button
save() / saveAs()
doTheSave()
The x-coordinate, y-coordinate, text colour, text, and description of every DraggableItem is saved into InDiagram.vlist
Background colour, titles, title colours, circle colours, and circle text colours are all saved into Config.vlist
All unassigned items in the items list are saved in Unassigned.csv
InDiagram.vlist, Config.vlist, and Unassigned.csv are all saved in a special ZIP file, with a .venn extension
Diagram is saved in a .venn file

User and Interface

User clicks save button.

Returns: Diagram is saved in a .venn file is saved

Interface and Controller

Save or SaveAs method is called.

Returns: All the customizations are saved into Config.vlist

All unassigned items are saved in Unassigned.csv

InDiagram.vlist, Config.vlist and Unassigned.csv are saved in a zip file with .venn extension.

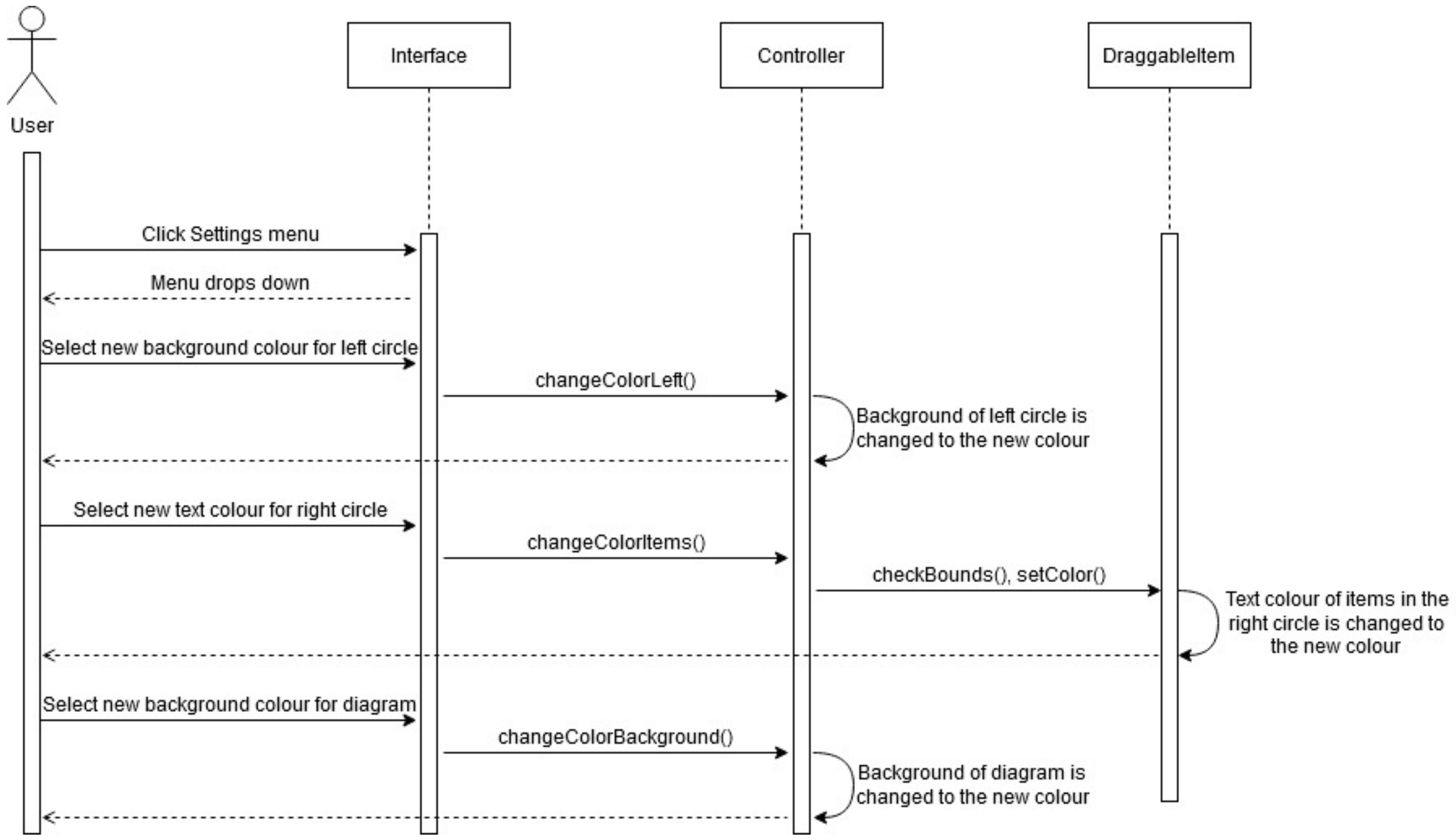
Controller and DraggableItem

doTheSave method is called

<u>Returns:</u> All the changes made to the draggable item, such as texts added, color changes made are saved in InDiagram.vlist

Section 4: Sequence Diagram Activity 3

The following sequence diagram shows the stages of customizing colour of the left and right


User
Interface
Controller
DraggableItem
Click Settings menu
Menu drops down
Select new background colour for left circle
changeColorLeft()
Background of left circle is changed to the new colour
Select new text colour for right circle
changeColorItems()
checkBounds(), setColor()
Text colour of items in the right circle is changed to the new colour
Select new background colour for diagram
changeColorBackground()
Background of diagram is changed to the new colour

circles.

User and Interface

Action 1 User clicks the settings menu.

Action 2 User selects background colour for the left circle.

Action 3 User selects new text color for the right circle.

Action 4 User selects new background color for the diagram.

Interface and Controller

For Action 2 changeColorLeft() method is called.

For Action 3 changeColorItems() method is called.

For Action 4 changeColorBackground() method is called.

<u>Returns</u>: background color of the left circle is changed

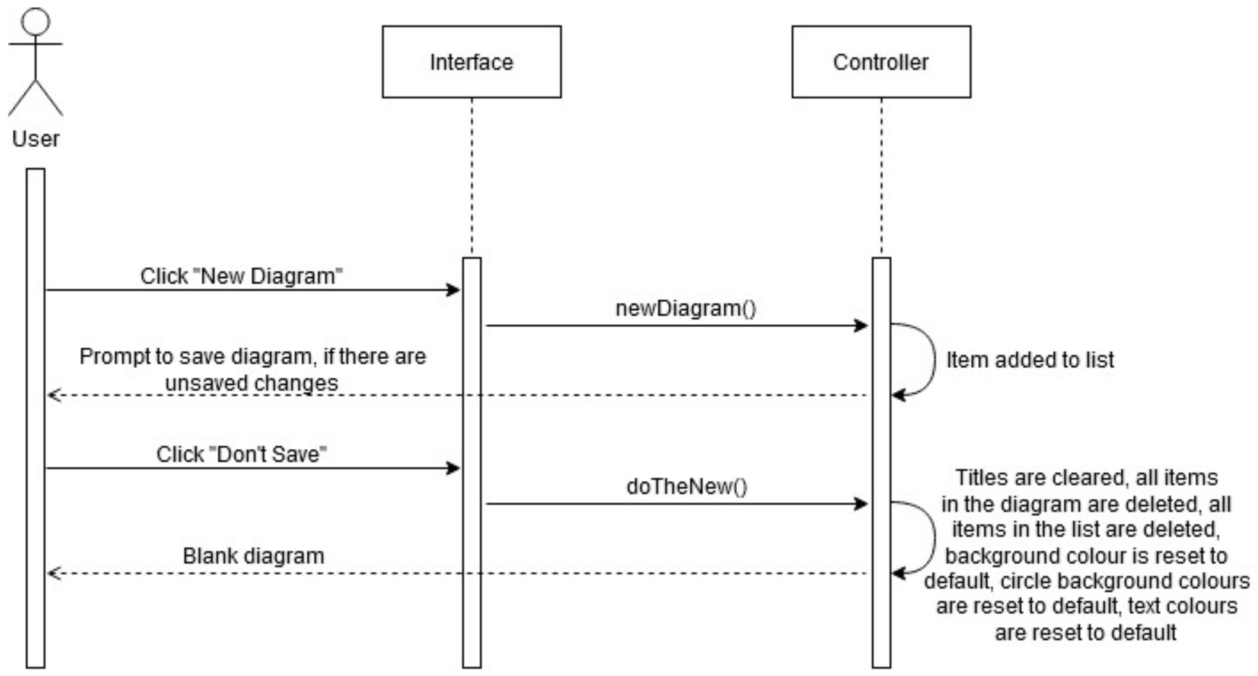
background color of the the diagram is changed.

Controller and DraggableItem
For Action 3 checkBounds() and setColor() method are called.

Returns: Text color of the items in the right circle is changed to new color.

Section 5: Sequence Diagram Activity 4

The stages of creating a new Venn Diagram.


User
Interface
Controller
Click "New Diagram"
newDiagram()
Item added to list
Prompt to save diagram, if there are unsaved changes
Click "Don't Save"
doTheNew()
Titles are cleared, all items in the diagram are deleted, all items in the list are deleted, background colour is reset to default, circle background colours are reset to default, text colours are reset to default
Blank diagram

User and Interface

Action 1 User clicks New Diagram button.
Returns: A prompt to save diagram's unsaved changes.

Action 2 User clicks Don't save
Returns: A new blank diagram is returned.

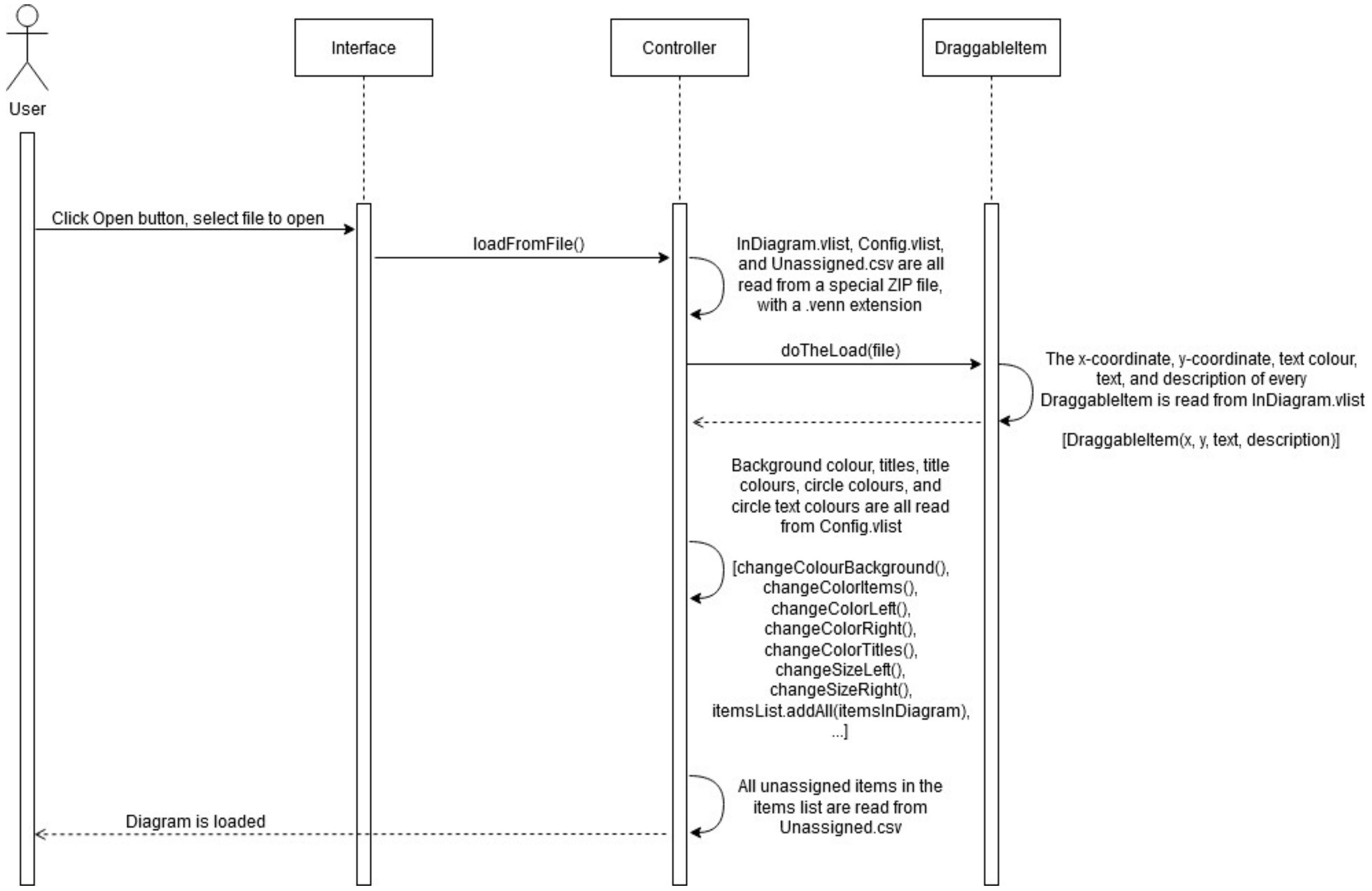
Interface and Controller

For Action 1 newDiagram() method is called.

For Action 2 doTheNew() method is called.
Returns: All the titles are cleared, items in the diagram are deleted, items in the list are deleted, and all customizable features are set to default.

Section 6: Sequence Diagram Activity 5

The stages of importing loading a file into the Venn Diagram application.


User
Interface
Controller
DraggableItem
Click Open button, select file to open
loadFromFile()
InDiagram.vlist, Config.vlist, and Unassigned.csv are all read from a special ZIP file, with a .venn extension
doTheLoad(file)
The x-coordinate, y-coordinate, text colour, text, and description of every DraggableItem is read from InDiagram.vlist
[DraggableItem(x, y, text, description)]
Background colour, titles, title colours, circle colours, and circle text colours are all read from Config.vlist
[changeColourBackground(), changeColorItems(), changeColorLeft(), changeColorRight(), changeColorTitles(), changeSizeLeft(), changeSizeRight(), itemsList.addAll(itemsInDiagram), ...]
All unassigned items in the items list are read from Unassigned.csv
Diagram is loaded

User and Interface

User clicks Open button

Selects the file of to open

Returns: The diagram is loaded.

Interface and Controller

loadFromFile() method is called

Returns: InDiagram.vlist, Config.vlist and Unassigned.csv are read from the zip file with a .ven extension.

The customizations like background and text colours are read from the the Config.vlist.

Respective methods to change the customizations accordingly.
The unassigned items are read from Unassigned.csv

Controller and DraggableItem
doTheLoad (file) method is called.

Returns: x coordinate, y coordinate, text colour and text of the draggable item on the list is read from InDiagram.vlist

Section 7: Software Maintenance Scenarios

This section includes the Corrective Maintenance issues that addresses the existing bugs that we aim to fix in the next release.

It also includes Perfective Maintenance scenarios that we can include to perfect the existing application.

Corrective Maintenance Scenarios:

- If the application is opened in a full screen mode, the bottom of the Venn diagram is cropped out of the full screen mode.
  To fix this issue, we can add scroll bars in the application window. Implement zooming function in the previous release.

- The textboxes may overlap if one textbox is dragged on top of the other. Add a condition before the setDropCompleted method to see if the borders of the textboxes overlap. If they do, use the setVisible(false) method to make the textbox added later invisible.

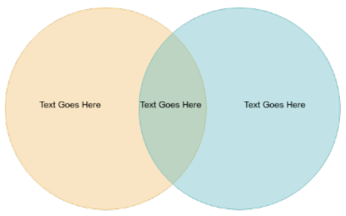
Perfective Maintenance Scenario:

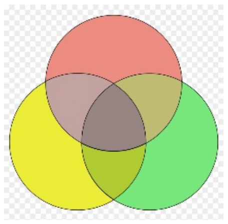
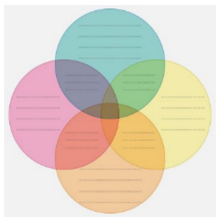
Feature 1: Multiple Venn templates

- An updated release of the app would have several layouts of a Venn. For example, a three circle and a four circle Venn Diagram

- To implement this feature, we need to change the class hierarchy of the java application, we need to have a 2 more classes, each of which contains the appropriate UI components. These classes will be called by the Main class. Each of these classes will share the same controller class to navigate the components and deploy the features

- The same feature can also be implemented using a add circle feature. The add circle allows us to add 2/3/4 circles depending on the demand of the user. The circles can be dragged to create an overlapping zone. This will not require a complete reformation of the class hierarchy but we only need to add an addCircle() method in the Controller class. In addition, we need to tailor this change into the existing the methods and UI components

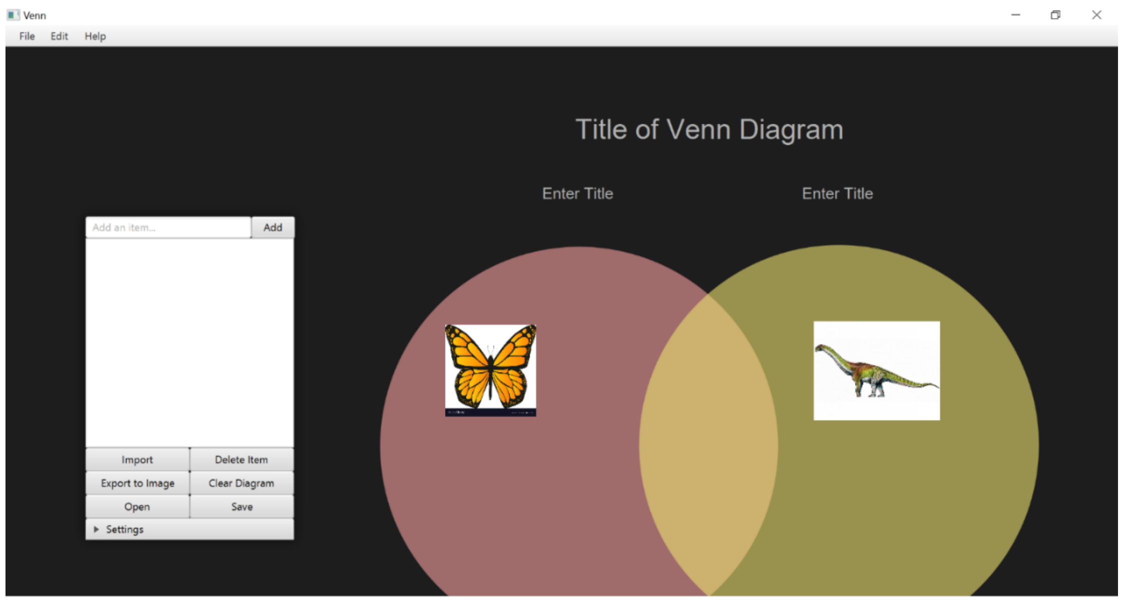


Feature 2: Importing Image into Venn Diagram:

- Another feature that would enhance the application is to be able to import image and drag and drop them within the Venn Diagram
- To execute the feature, we need to create a DraggableImage class that extends DraggableItem and override methods in Draggable Item to handle image format
- Finally make an addImageToDiagram method in the Controller class

This is what the feature might look like:


Venn
File
Edit
Help
Title of Venn Diagram
Enter Title
Enter Title
Add an item...
Add
Import
Delete Item
Export to Image
Clear Diagram
Open
Save
Settings