Software Design Document Venn Diagram 2311

Jeffrey Walker

Joseph Spagnuolo

Marco Carusoni

Arjun Kaura

Supervised by Dr Vassilios Tzerpos

1. Introduction

1.1 Purpose of this Document

This document is intended to outline the software's class diagram, sequence diagram, and possible maintenance scenarios for future use.

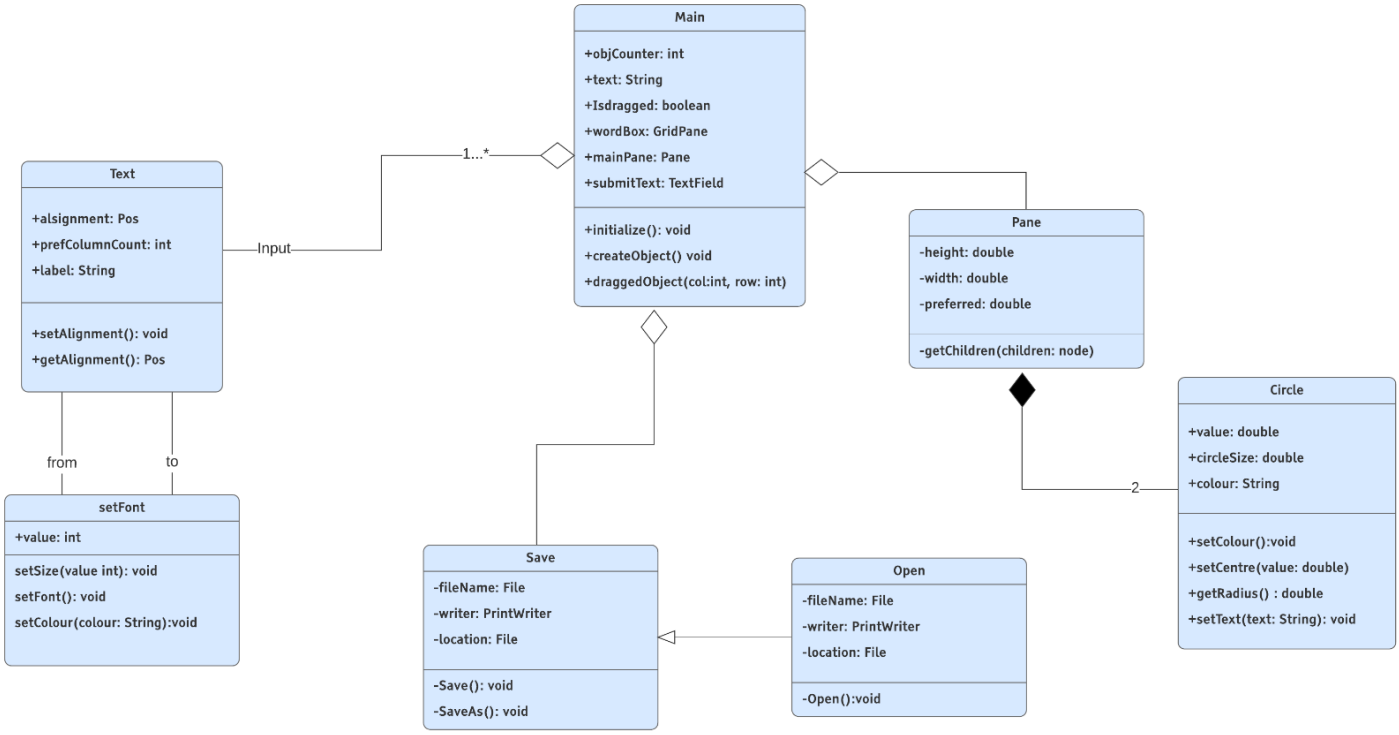
1.2 Overview

The class diagram will depict the interaction between important classes and methods used by the software. The sequence diagram will depict the interaction of important objects the software will use during runtime. The maintenance scenarios will outline possible changes that could implemented for future use. The venn diagram software was created using JavaFX and scene builder.

2. Diagrams

2.1 Main Class Diagram

The Venn Diagram software features will give the user the ability to create, save, load, and print venn diagrams all with customizable options. The base case of every venn diagram will include 2 large circles that intersect each other. This will also include a title for each circle and insertable text specified by the user. But some classes and methods will be omitted in the class diagram for the purpose of this document. The intention is to give an outline to give an idea of how the system is designed.



2.2 Main

This section will go over some classes the software most commonly uses. Some code will be omitted for generality.

The main class holds important variables such as objectCounter, text, and isDragged variables. It also initializes a primary stage used by JavaFX. The method start() also loads an initial fxml file which is used to compose a JavaFX GUI.

```
public void start(Stage primaryStage) {
            try {

                  Parent root =
FXMLLoader.load(getClass().getResource("CoverPage.fxml"));

                  Scene scene = new Scene(root);
                  primaryStage.setScene(scene);
```

2.3 Initialize

This method initializes the word boxes to be inserted to the pane each time the program loads. Variables such as numCols and numRows are used to constrain the possible number of word box cells that text can be inserted into. This makes it possible for words to be contained within the circle pane without leaking outside.

```
            int numCols = 7;
            int numRows = 17;

            for (int i = 0; i < numCols; i++) {
                  ColumnConstraints colConstraints = new ColumnConstraints();
                  colConstraints.setHgrow(Priority.SOMETIMES);
                  WordBox.getColumnConstraints().add(colConstraints);
            }

            for (int i = 0; i < numRows; i++) {
                  RowConstraints rowConstraints = new RowConstraints();
                  rowConstraints.setVgrow(Priority.SOMETIMES);
                  WordBox.getRowConstraints().add(rowConstraints);
```

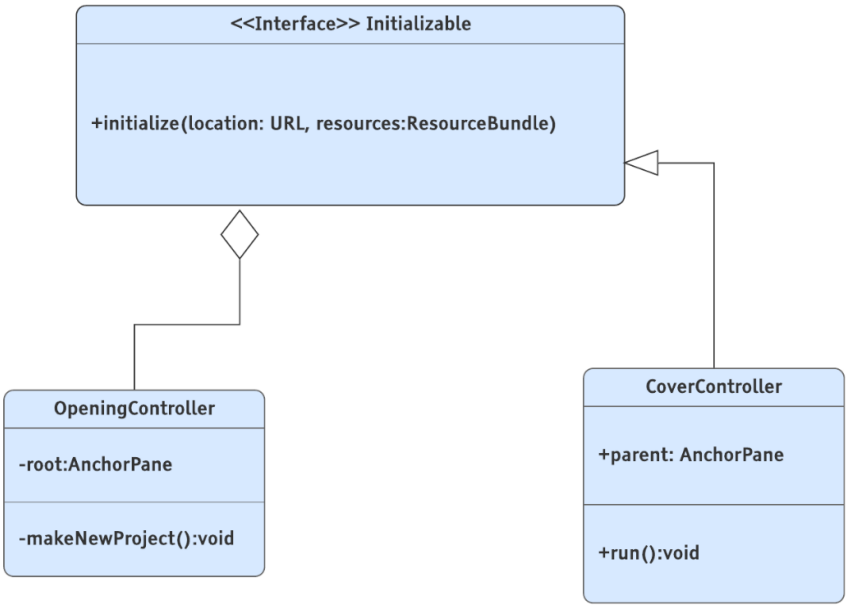
2.4 Save

This save class has method save() and saveAs which is used to save the contents of a given venn diagram. This method uses a the PrintWriter class to scan each line as a String. The String is then inserted line by line then saved as a text document. The location of each text object is also saved with variable coord. This keeps track of each numerical value when the text is dragged to a given cell.

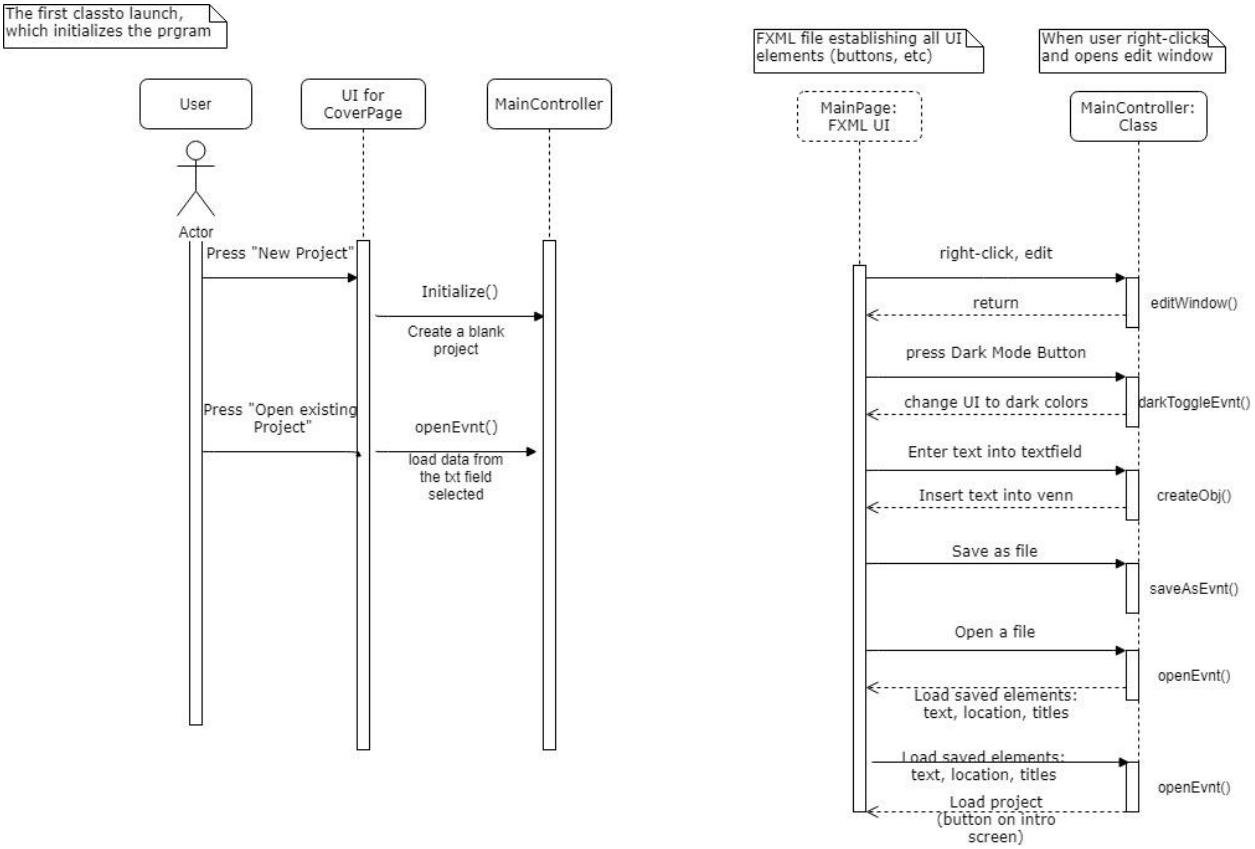
```
private void save() throws FileNotFoundException {

            if (filename != "") {

                  File file = new File(filename);
                  file.delete();

                  PrintWriter writer = new PrintWriter(filename);

                  int i = 0;
                  while (textObjects[i] != null) {

                        writer.println(textObjects[i] + "," + coord[i]);
                        i++;

                  }

                  writer.println("newLine");

                  writer.println(Title.getText());
                  writer.println(subTitle1.getText());
                  writer.println(subTitle2.getText());
                  writer.println(cpkVen1.getValue());
                  writer.println(cpkVen2.getValue());

                  writer.close();
```

2.5 Controller Class Diagram

Without loss of generality, there are classes and methods which are omitted. But this diagram will outline the general design of controller classes used by the software.

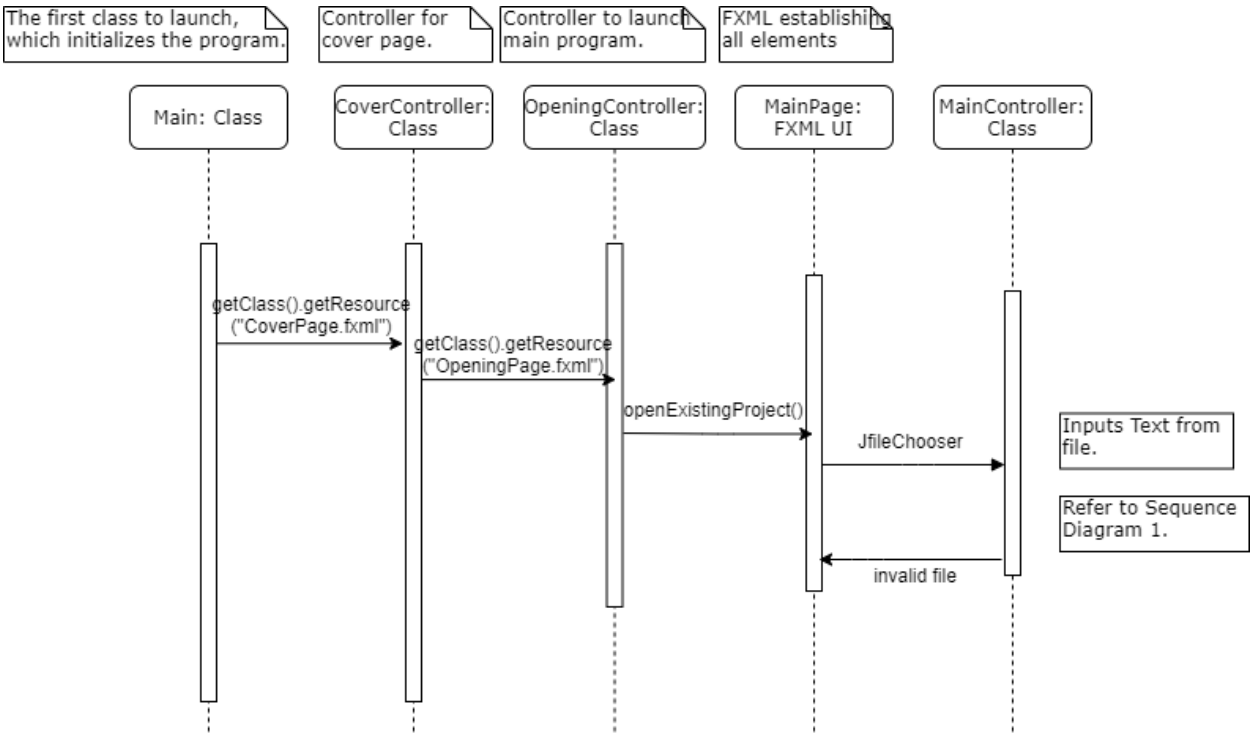


2.6 Sequence Diagram



The first three classes are Main (simply launching and setting up initial window), CoverController (controller for the initial window which is the welcome page with option to load or create new project), and the OpeningController which loads the actual program itself: MainPage. MainPage is FXML – user sees this as UI and interacts with it, affecting the MainController class.

The following diagram is for the feature of loading a file from the front page:



3. Maintenance Scenarios

3.1 General

This section will outline various maintenance scenarios by the developers. These scenarios will give insight to what could be added to the software for future use.

3.2 Extra Venn Diagram Circle

With this updated change, users will be able to insert a 3rd circle to their current venn diagram. This would be an upgrade from the current version which handles 2 circles. The current fxml file which handles the shapes inside the GUI would need to be updated. 2 circle objects are created then their placement is specified by a separator layout. An additional circle object should be created with its own separator layout. The addition is bolded in the given code.

<Circle fx:id="Ven1" fill="#b5b5ff80" layoutX="326.0" layoutY="350.0" radius="234.0" stroke="BLACK" strokeType="INSIDE" />
<Circle fx:id="Ven2" fill="#ffb8b880" layoutX="590.0" layoutY="350.0" radius="234.0" stroke="BLACK" strokeType="INSIDE" />
<Circle fx:id="Ven1" fill="#b5b5ff80" layoutX="326.0" layoutY="350.0" radius="234.0" stroke="BLACK" strokeType="INSIDE" />

<Separator layoutX="901.0" layoutY="-12.0" orientation="VERTICAL" prefHeight="686.0" prefWidth="16.0" />
<Separator layoutX="6.0" layoutY="-12.0" orientation="VERTICAL" prefHeight="686.0" prefWidth="16.0" />

<Separator layoutX="6.0" layoutY="-12.0" orientation="VERTICAL" prefHeight="686.0" prefWidth="16.0" />

3.3 Theme Changes

Currently, the software only supports a default theme (white) and a dark mode(black). The updated software could take user input to specify code. With the current software, there is no parameters for the method. With the updated version user can supply a colour then Title.setStyle, NumVen1, and NumVen2 would then set to that argument. The addition is bolded in the given code.

```
private void darkToggleEvnt(colour String) {

darkToggle.setText("Dark Mode");
                  Title.setStyle("-fx-text-inner-color: colour;");
                  NumVen1.setStyle("-fx-text-inner-color: colour;");
                  NumVen2.setStyle("-fx-text-inner-color: colour;");

                  mainPane.setBackground(
                              new Background(new
BackgroundFill(Color.web("#f2f2f2"), CornerRadii.EMPTY, Insets.EMPTY)));

            }
```

3.3 Saving in different formats

Currently, the software only supports saving files as .txt. In the updated software, users could be given more options for saving in different formats. This could include .pdf, .doc, etc. The addition is bolded in the given code.

```
JFileChooser fileChooser = new JFileChooser();
            fileChooser.setFileFilter(new FileNameExtensionFilter("Text
file", "txt"));
```

```
JFileChooser fileChooser = new JFileChooser();
            fileChooser.setFileFilter(new FileNameExtensionFilter("PDF file",
"pdf"));
```