Contents

1 Introduction

This is the Software Design Description (SDD) for the Metro Map Maker application.

1.1 Purpose

This document will serve as the blueprint for the construction of the Metro Map maker application. This document will use UML class diagrams to provide detail regarding packages, classes, instance variables, class variables, and method signatures needed for the application. In addition, UML Sequence diagrams will be used to specify object interactions post-initialization of the application, meaning in response to user interactions or timed events.

1.2 Scope

The Metro Map Maker will be a desktop application that will be built from the Desktop Java Framework (DJF).

1.3 Definitions, acronyms, and abbreviations

Class Diagram - A UML document format that describes classes graphically. Specifically, it describes their instance variables, method headers, and relationships to other classes.

Framework - In an object-oriented language, a collection of classes and interfaces that collectively provide a service for building applications or additional frameworks all with a common need.

Java – A high-level programming language that uses a virtual machine layer between the Java application and the hardware to provide program portability.

Sequence Diagram - A UML document format that specifies how object methods interact with one another.

UML - Unified Modeling Language, a standard set of document formats for designing software graphically.

Desktop Java Framework (djf) - The framework that the Metro Map Maker application will be built using.

1.4 References

Metro Map Maker SRS - Software requirements specification for the Metro Map Maker software application.

1.5 Overview

This Software Design Description document provides a design for the Metro Map Maker software application described in the Metro Map Maker SRS. The Java API being used to create the application are described in section 2, as well as their purpose. Section 3 provides an overview of the class design for the Metro Map Maker. Section 4 contains information on the use cases the program has been required. Section 5 contains information on how the program will store data. Section 6 contains supplementary information that may be useful to the reader. Note that all of the UML Diagrams used in this document were created in the VioletUML editor.

2 Package-Level Design Viewpoint

This application will be designed using the DJF. The software will also use Java API.

2.1 Metro Map Maker overview

The Metro Map Maker will use the following components directly from the DJF. These classes will not be discussed in detail in this document.

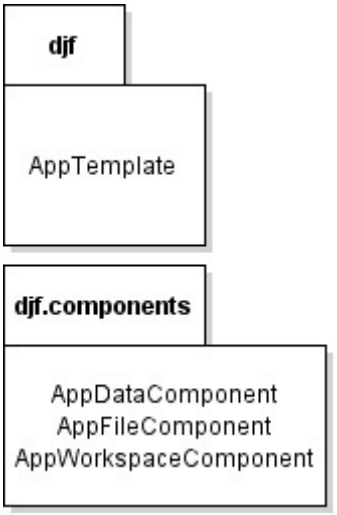

Figure 1: djf Framework directly used

2.2 Java API Usage

The design for the Metro Map Maker will make use of the following classes specified in the following figure

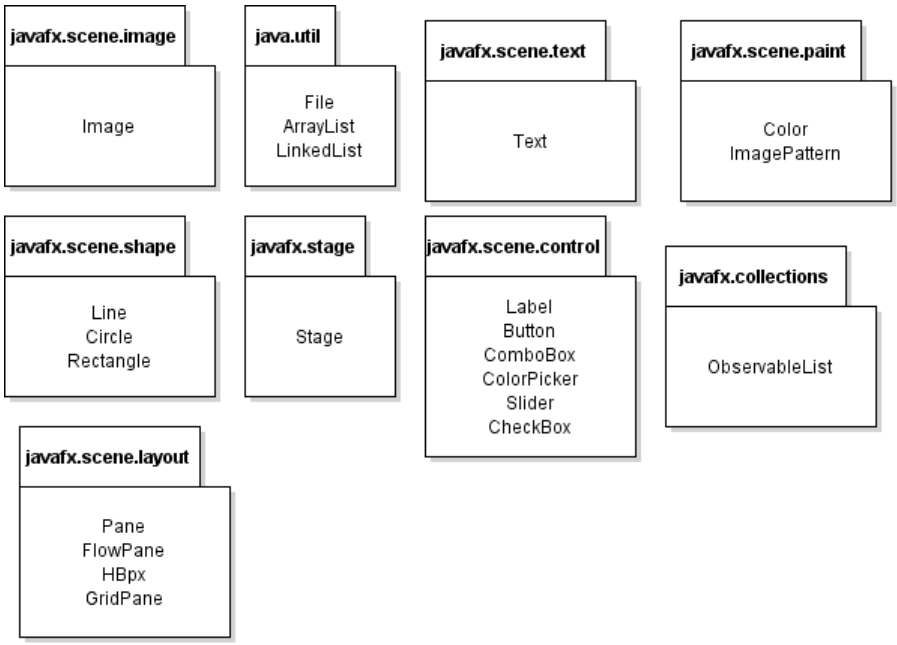

Figure 2: The Java API being used

2.3 Java API Usage Descriptions

Class/Interface	Use
File	This will be used for importing and exporting saved data for the Metro Map Maker, as well as importing images into a map
ArrayList	This class will be used to store groups of Data
LinkedList	This class is used as the parent class of the MetroLine class. This seemed appropriate as a Metro Line would be an ordered collection with a designated head and tail.

Table 1: Uses for the classes in the Java API's java.util package

Class/Interface	Use
Line	This class will be used to draw the connections between Metro Stations on the map
Circle	This class is used to indicate on the map where the Metro Stations are located
Rectangle	This class is used to allow for the user to add images to the metro map they are editing

Table 2: Uses for the classes in the Java API's javafx.scene.shape package

Class/Interface	Use
ObservableList	This class is used to allow for MMMData and MMMWorkspace to interact with each other fairly well.

Table 3: Uses for the classes in the Java API's javafx.collections package

Class/Interface	Use
Image	This class will be used to add images to the metro map being edited, as well adding an image background to a map

Table 4: Uses for the classes in the Java API's javafx.scene.image package

Class/Interface	Use
Text	This class will be extended so that the user will be able to add labels to the metro map being edited

Table 5: Uses for the classes in the Java API's javafx.scene.text package

Class/Interface	Use
Color	This class will be used to change the fill color of shapes on the metro map, as well as set the background color.
ImagePattern	This class is used in the scenario where the user wishes to add an image to the map being edited

Table 6: Uses for the classes in the Java API's javafx.scene.paint package

Class/Interface	Use
Stage	This class will be extended for all of the dialog windows which are outside of the main editor. This includes the WelcomeDialog, EnterTextDialog, etc.

Table 7: Uses for the classes in the Java API's javafx.stage package

Class/Interface	Use
Label	This class is used to label windows, and other objects in the GUI
Button	This class is used frequently and it provides the user with many options to control and edit their metro map
ComboBox	This class is used to allow the user to select certain types of data.
ColorPicker	This class is used to offer to the user a large variety of colors to use as the fill for shapes on the metro map or for the background color
Slider	This class will allow the user to have a finer control over the selection of certain variables in the application.
CheckBox	This class will be used to allow the user to set certain editing options which only have two options available

Table 8: Uses for the classes in the Java API's javafx.scene.control package

Class/Interface	Use
Pane	The main container for all the objects in a metro map
FlowPane	This container is used so to automatically organize certain toolbars in the main window
HBox	This class will be used as the main container for all of the toolbars in the editor
GridPane	This will be used as the container for certain toolbars

Table 9: Uses for the classes in the javafx.scene.layout package

3 Class-Level Design Viewpoint

As stated before, the design will encompass the Metro Map Maker application and Desktop Java Framework. Since no modifications are being made to the Desktop Java Framework, the UML documentation for the Desktop Java Framework will not be provided. For the purpose of fitting all the diagrams to the pages, the diagrams will be split up.

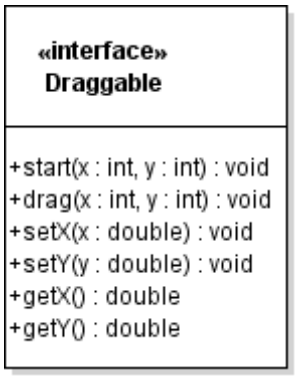

Figure 3: Draggable

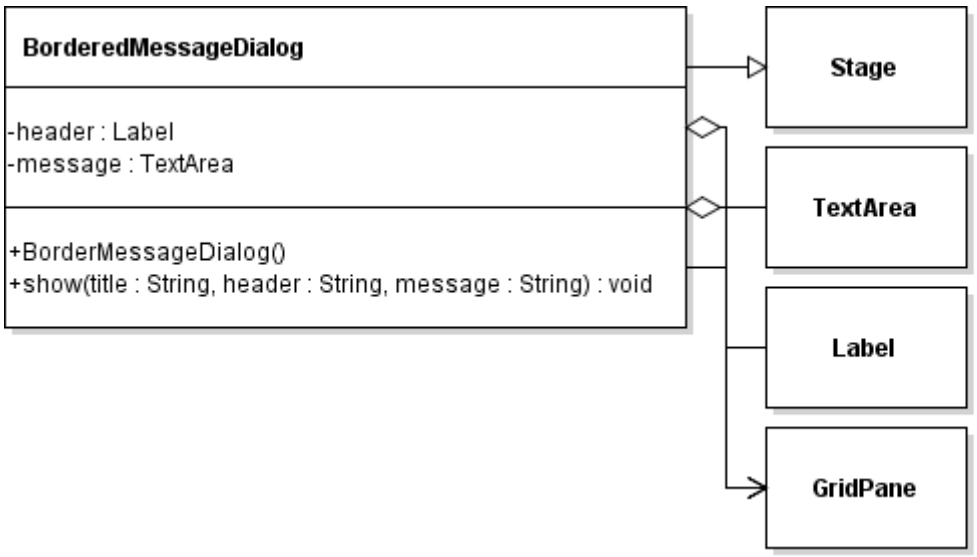

Figure 4: BorderedMessageDialog

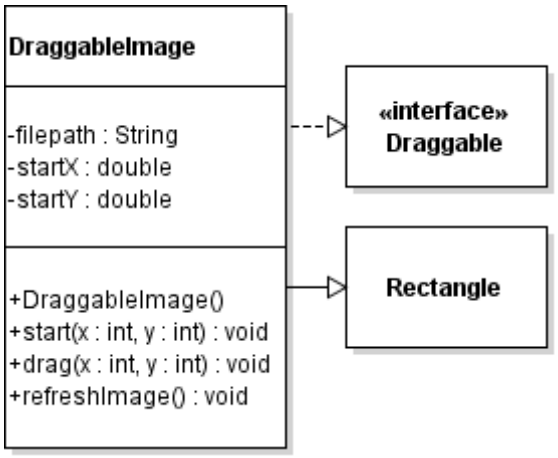

Figure 5: DraggableImage

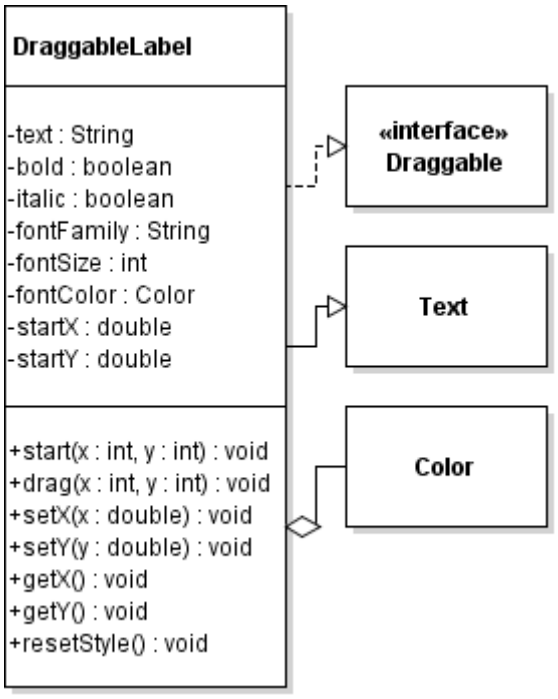

Figure 6: DraggableLabel

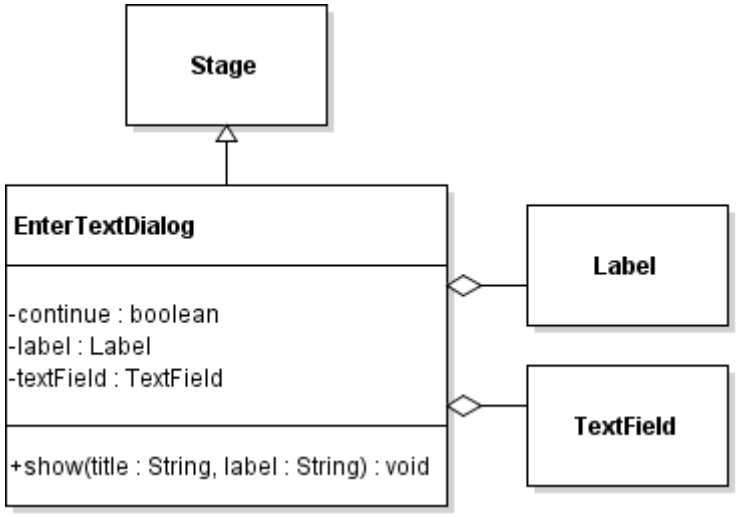

Figure 7: EnterTextDialog

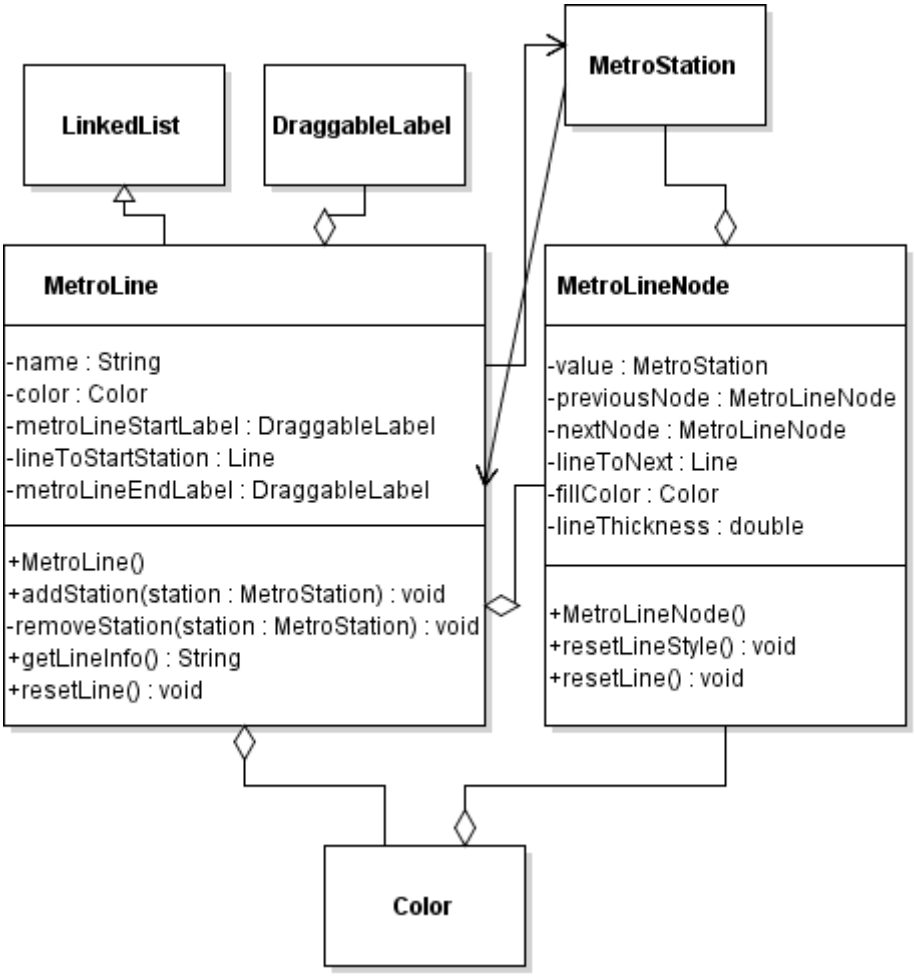

Figure 8: MetroLine and MetroLineNode classes

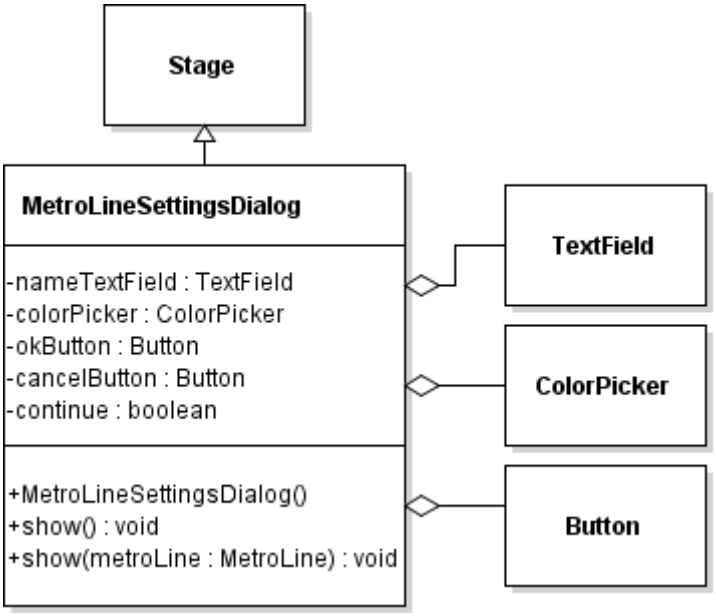

Figure 9: MetroLineSettingsDialog

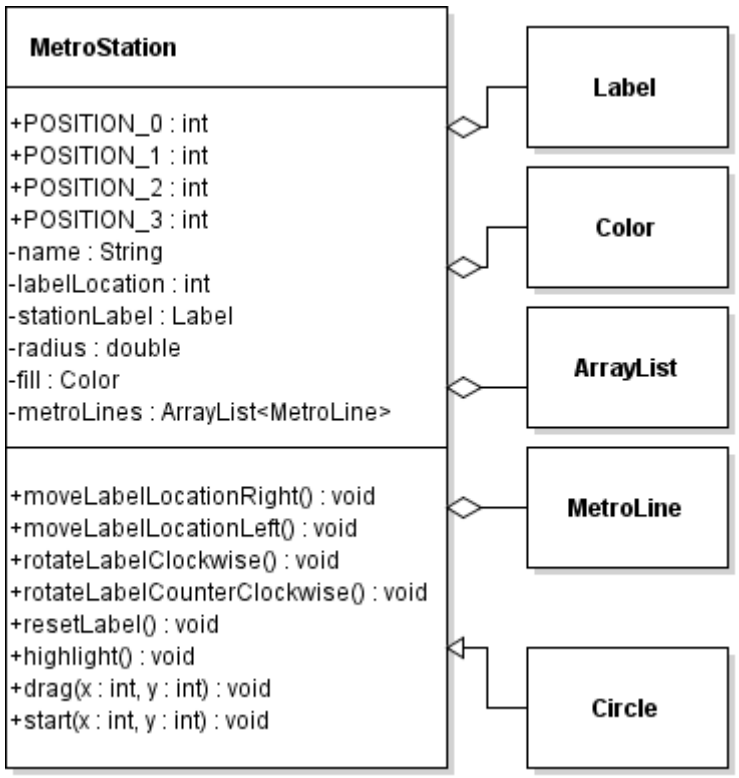

Figure 10: MetroStation

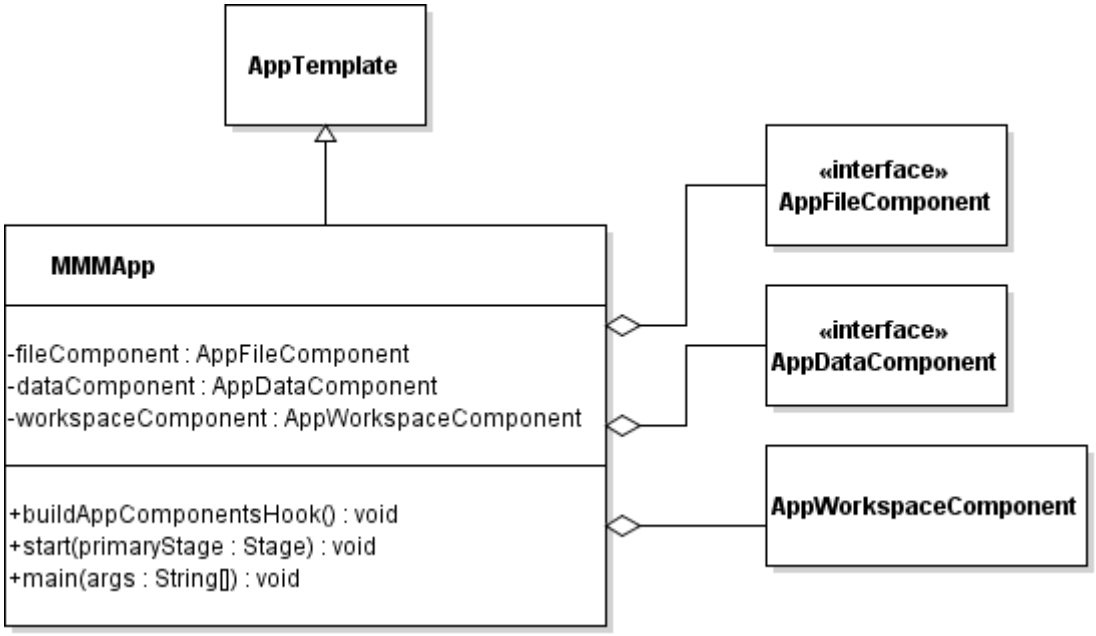

Figure 11: MMMApp

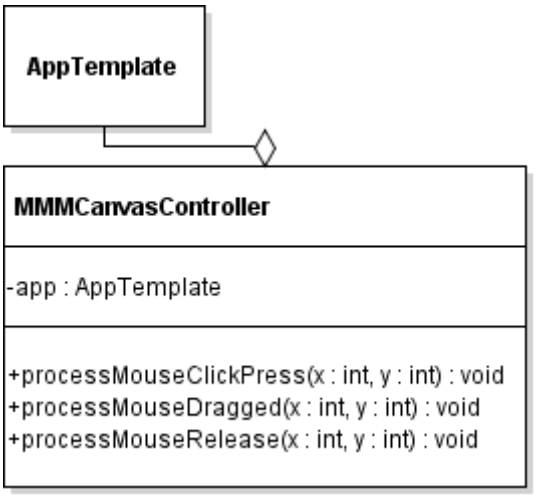

Figure 12: MMMCanvasController

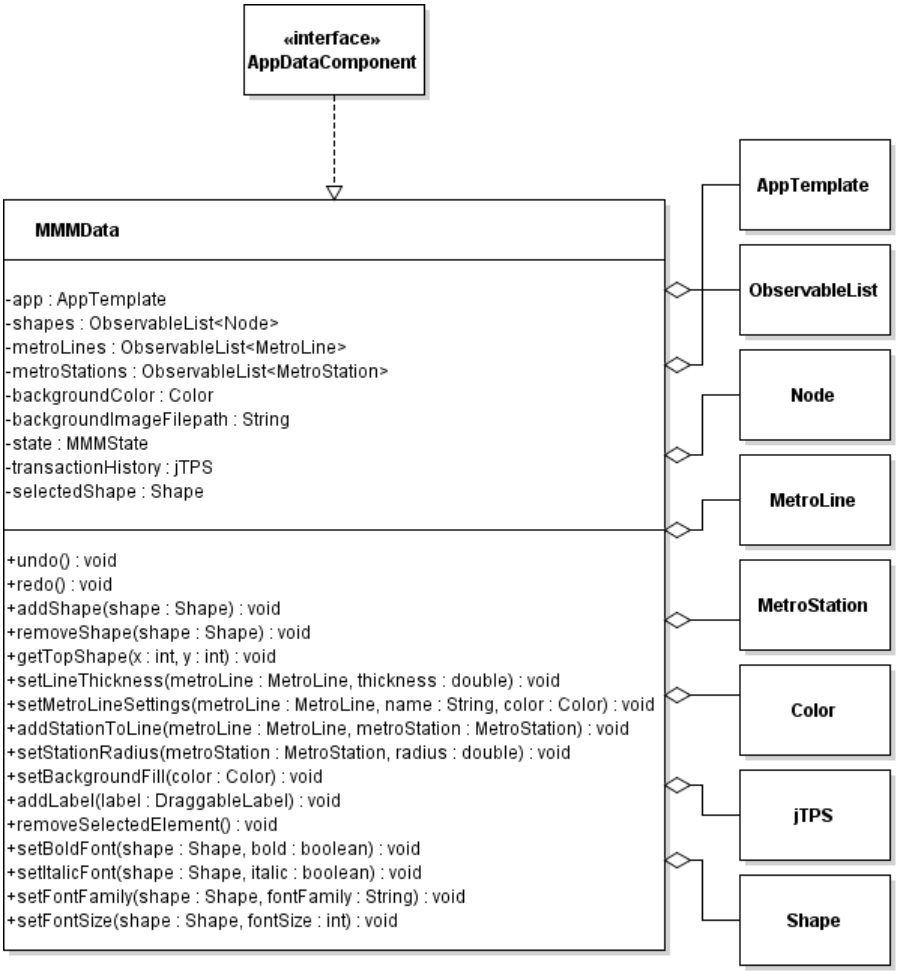

Figure 13: MMMData

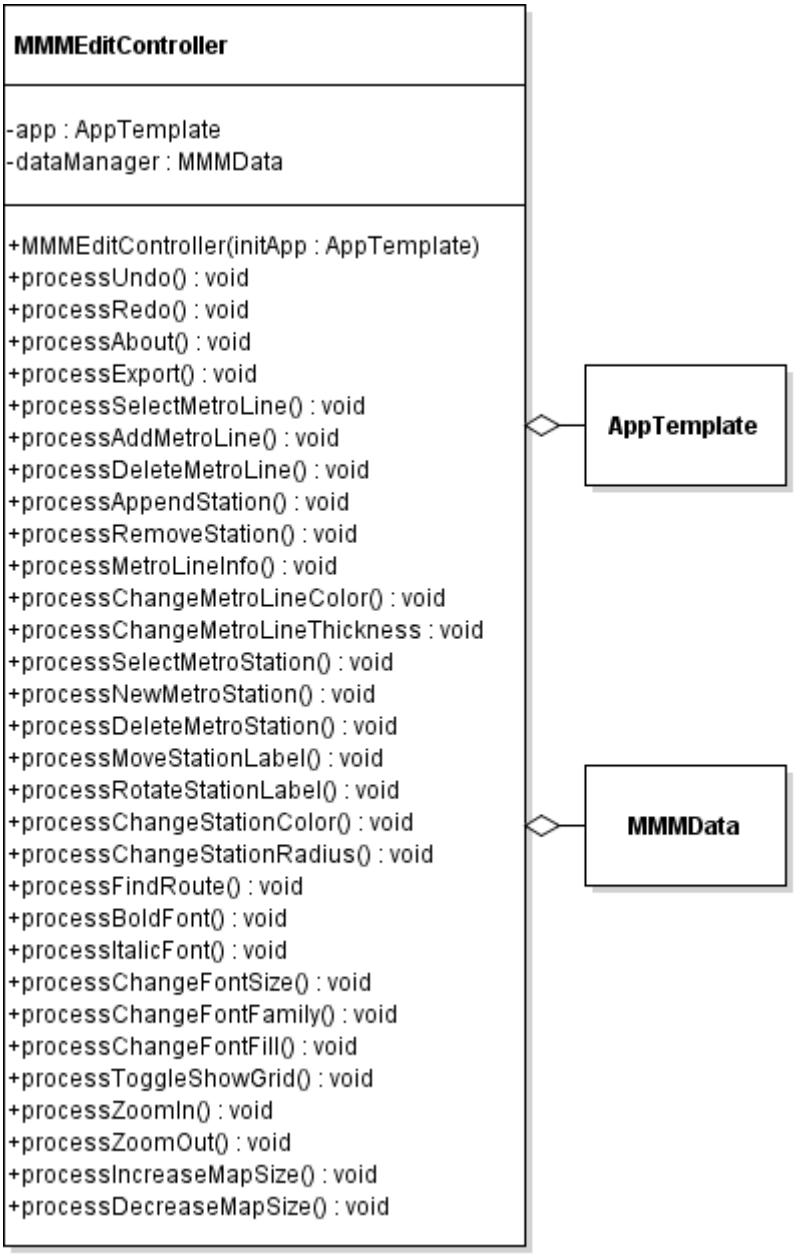

Figure 14: MMMEditController

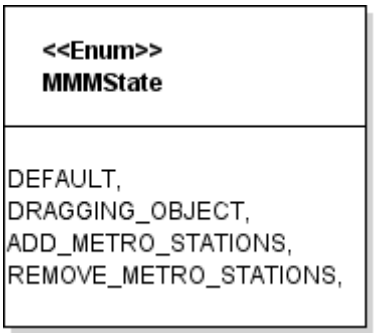

Figure 15: MMMState Enum

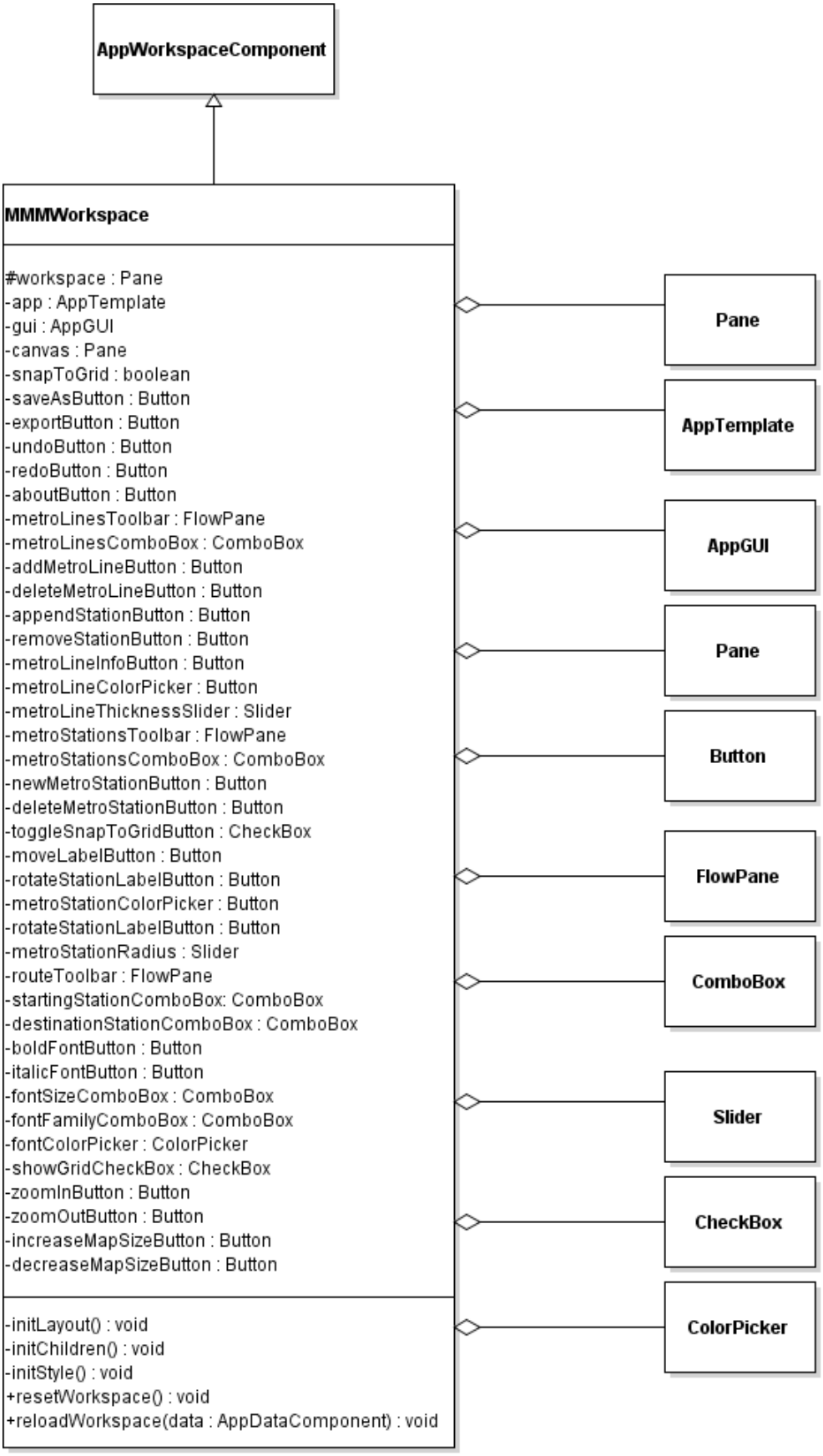

Figure 16: MMMWorkspace

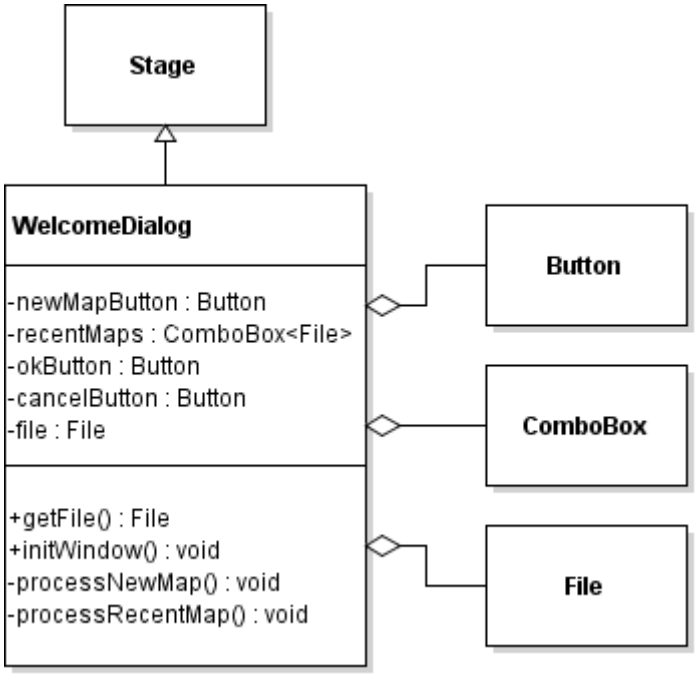

Figure 17: WelcomeDialog

4 Method Level Design Viewpoint

The following diagrams will describe the methods that will be called within the code.

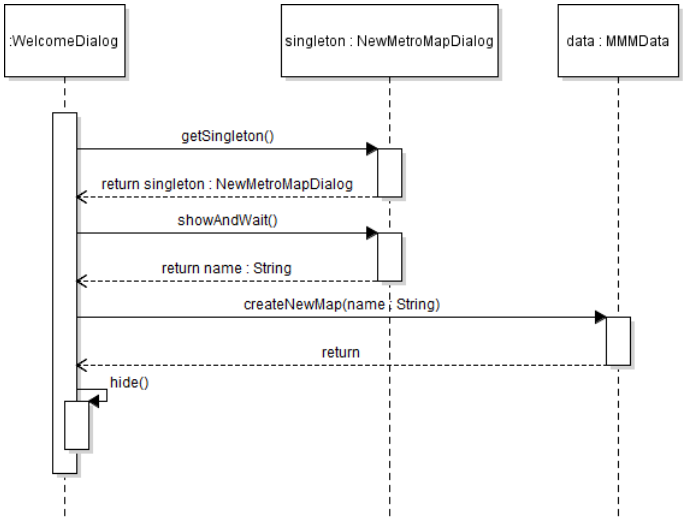

Figure 18: Create new map from the Welcome Dialog

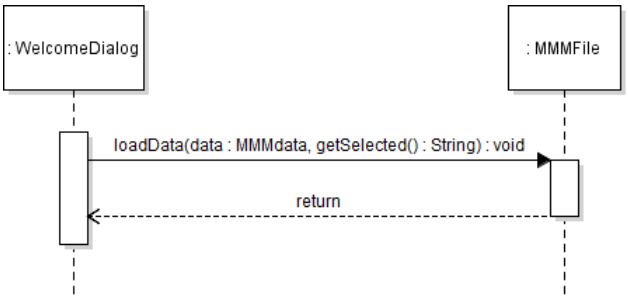

Figure 19: Select Recent Map to Load

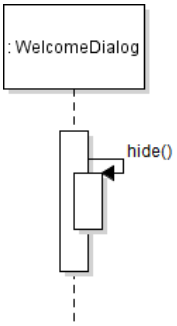

Figure 20: Close Welcome Dialog

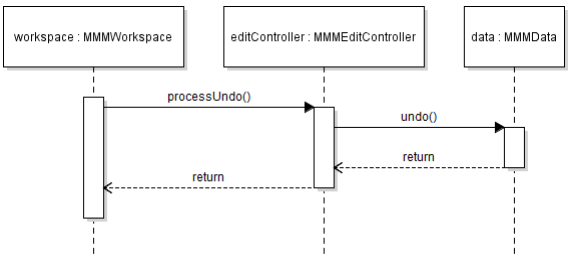

Figure 21: Undo Edit

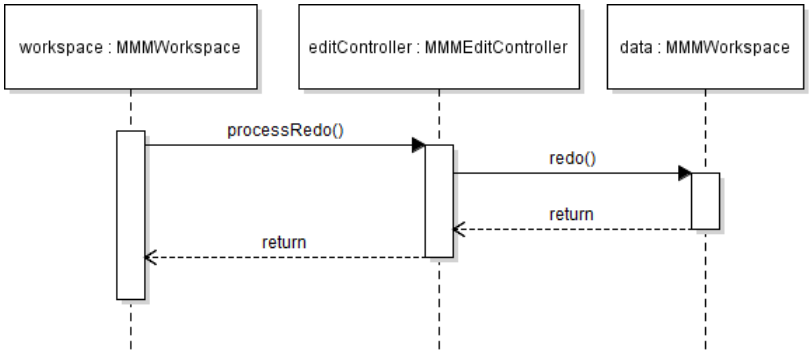

Figure 22: Redo Edit

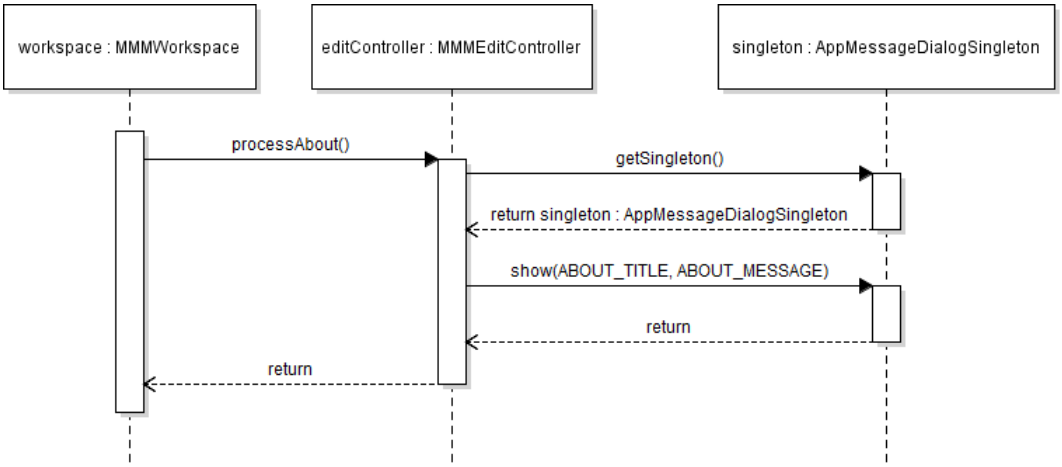

Figure 23: Learn About Application

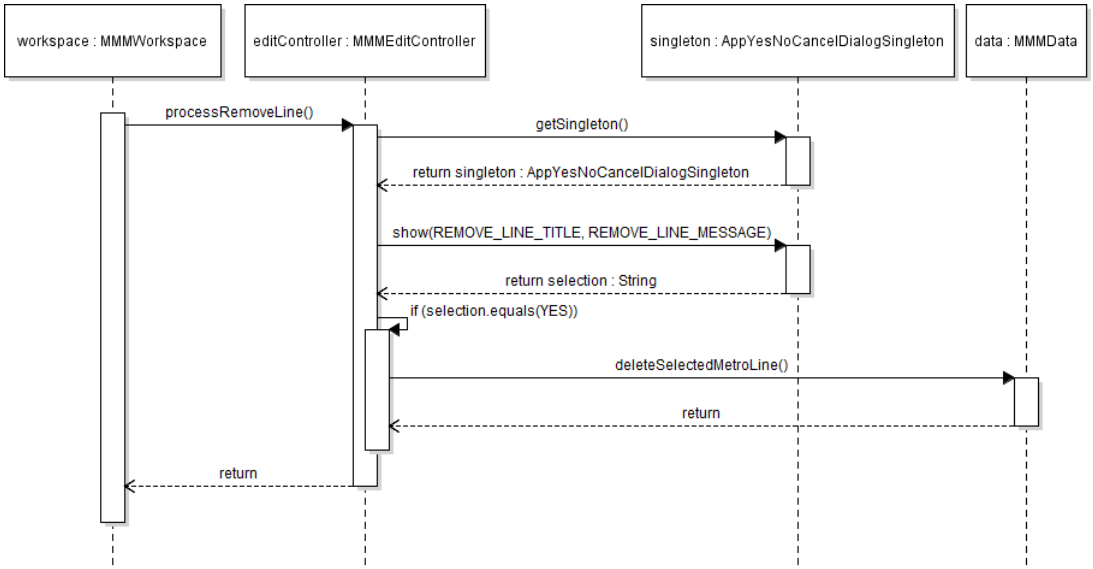

Figure 24: Remove Metro Line

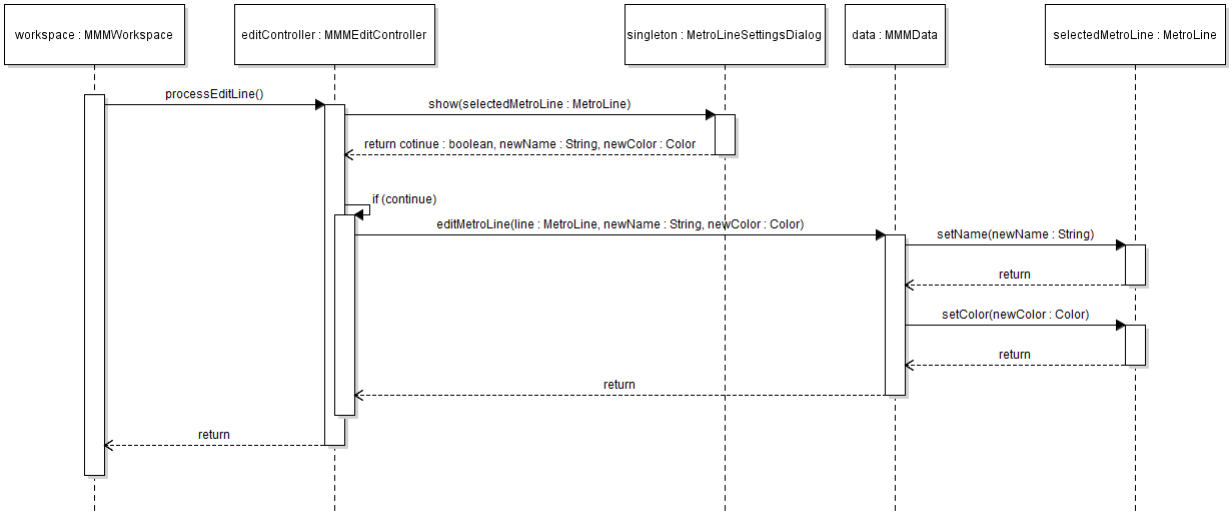

Figure 25: Edit Line

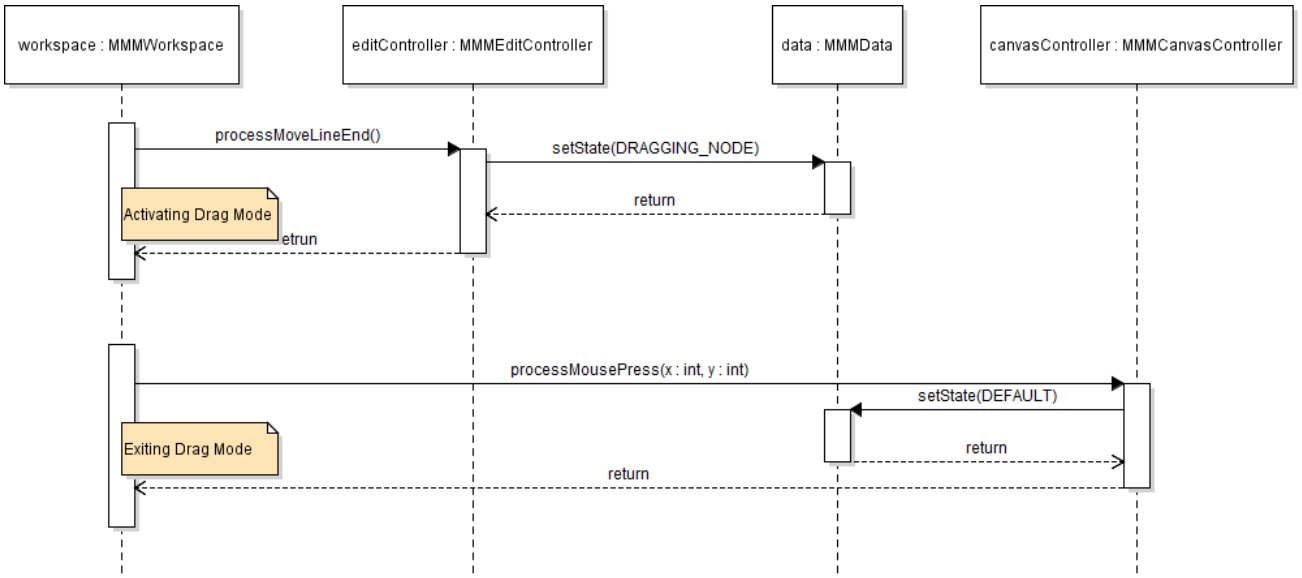

Figure 26: Move Line end points

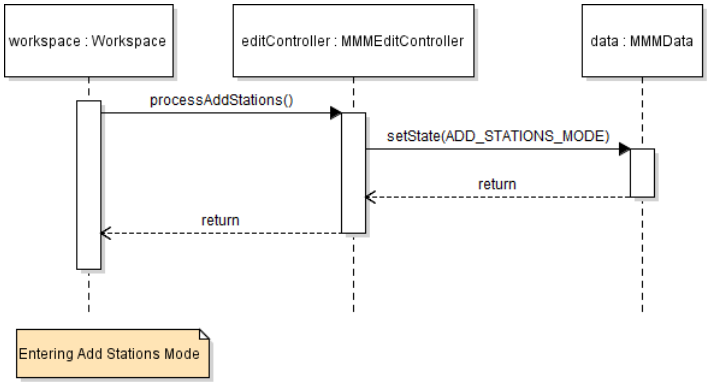

Figure 27: Add Stations to Line - Entering Add Stations Mode.

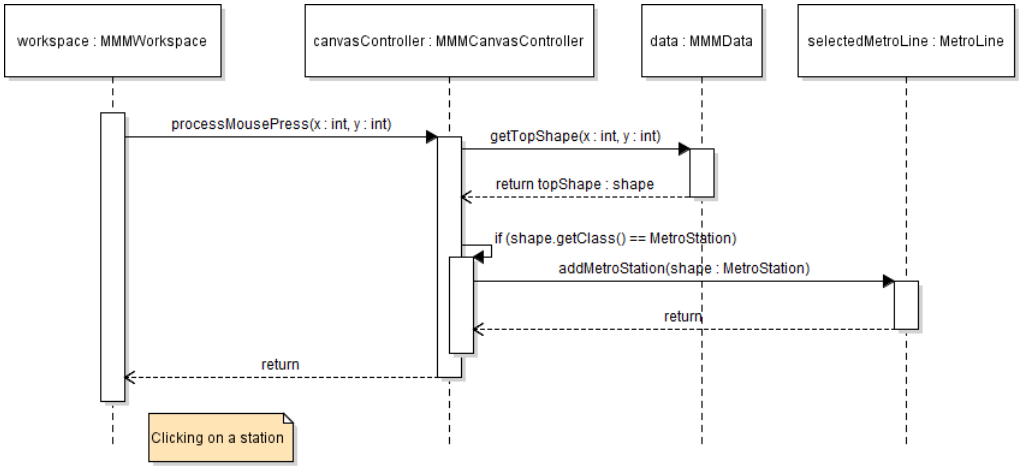

Figure 28: Add Stations to Line - Adding stations to a Metro Line.

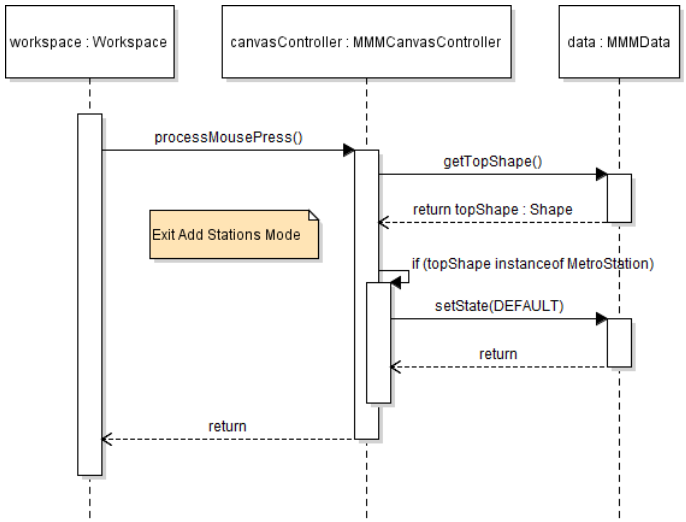

Figure 29: Add Stations to Line - Exiting Add Stations Mode.

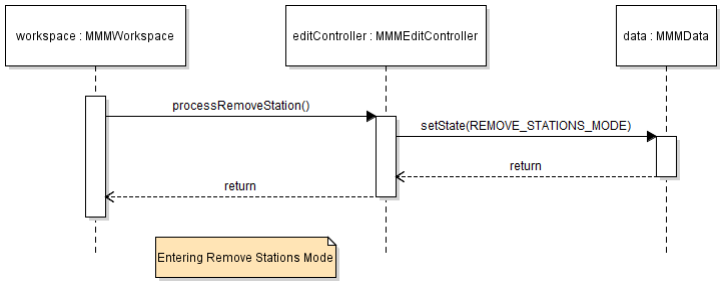

Figure 30: Remove Stations - Entering Remove Stations Mode

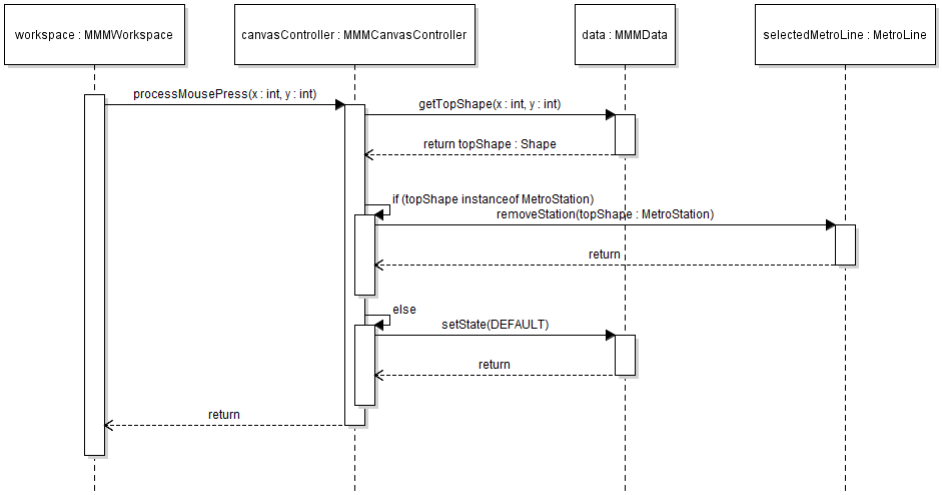

Figure 31: Remove Stations - Removing Stations and Exiting Remove Stations Mode.

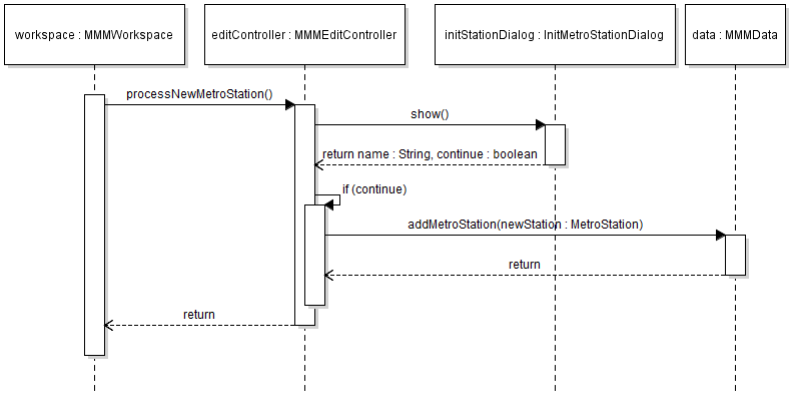

Figure 32: Add New Station

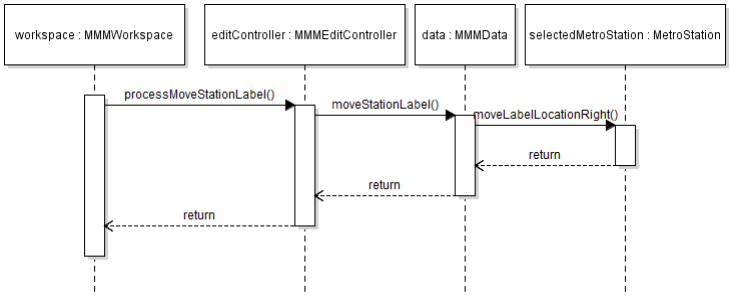

Figure 33: Move Station Label

5 File Structure and Formats

The Desktop Java Framework will be provided inside the MetroMapMaker.jar. This needs to be imported into the necessary project for the Metro Map Maker application.

5.1 Application Data

Various forms of data will be kept for the Metro Map Maker. We will have various language properties (stored in an .xml file) stored inside the data folder along with a schema.

Images will be stored in an image folder for ease of access.

5.2 Exporting Data

A Metro Map Maker file will be saved using the JSON strucutre. For each map the following data fields are saved:

Map

- Background Color
- Background Image (If one is chosen)

Metro Station

- Name
- X-Coordinate
- Y-Coordinate
- Fill Color
- Radius
- Label Rotation
- Label Position

Metro Line

- Name
- Fill Color
- Line Thickness
- Start Label X-Coordinate
- Start Label Y-Coordinate
- End Label X-Coordinate
- End Label Y-Coordinate
- Metro Station Stops

Labels

- Text
- X-Coordinate
- Y-Coordinate
- Bold Font
- Italic Font

- Font Family
- Font Size
- Font Fill

Images

- Image filepath
- X-Coordinate
- Y-Coordnate

6 Supporting Information

N/A