The Metro Map Maker ™

Software Design Description

Author: Austin Biegler
Debugging Enterprises ™
November 2017
Version 1.0

Abstract: This document describes the design for an application that allows users to create elaborate metro maps.

Based on IEEE Std 1016™ -2009 document format

1 Introduction

This is the Software Design Description (SDD) for the Metro Map Maker™ application. Note that this document format is based on the IEEE Standard 1016-20017 recommendation for software design.

1.1 Purpose

This document is to serve as the blueprint for the construction of the Metro Map Maker application. This design will use UML class diagrams to provide complete detail regarding all packages, classes, instance variables, class variables, and method signatures needed to build 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

Metro Map Maker is an independent application. However, it does use a framework that allows for a more seamless way of accessing data in different parts of the program. The framework described also allows for multiple similar applications to be created. This framework is named Desktop Java Framework.

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.

IEEE – Institute of Electrical and Electronics Engineers, the “world’s largest professional association for the advancement of technology”.

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.

Desktop Java Framework - The software framework to be developed in tandem with the Metro Map Maker app such that additional apps can easily be constructed.

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

Sprite has its own position and velocity, so each will be its own Sprite that knows what Sprite Type it belongs to.

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

1.4 Reference

IEEE Std 830TM-1998 (R2009) – IEEE Standard for Information Technology – Systems Design – Software Design Description

1.5 Overview

This Software Design Description document provides a working design for the Metro Map Maker software application as described in the Metro Map Maker Software Requirements Specification. Note that all parties in the implementation stage must agree upon all connections between components before proceeding with the implementation stage. Section 2 of this document will provide the Package-Level Viewpoint, specifying the packages and frameworks to be designed. Section 3 will provide the Class-Level Viewpoint, using UML Class Diagrams to specify how the classes should be constructed. Section 4 will provide the Method-Level System Viewpoint, describing how methods will interact with one another. Section 5 provides deployment information like file structures and formats to use. Section 6 provides a Table of Contents, an Index, and References. Note that all UML Diagrams in this document were created using the VioletUML editor.

2 Package-Level Design Viewpoint

Framework to be used in its construction. In building both we will heavily rely on the Java API to provide services. Following are descriptions of the components to be built, as well as how the Java API will be used to build them.

2.1 Metro Map Maker

The figures below will show the classes made for the main app class and its relation to the rest of the classes and framework, the css style class, the data classes and their relations, the file classes and thier relations, and the GUI classes and their relations as well. Any classes that use Java API have also been given their relation to the appropriate Java classes.

Figure 2.1 specifies how the MapApp class is related to the rest of the program.

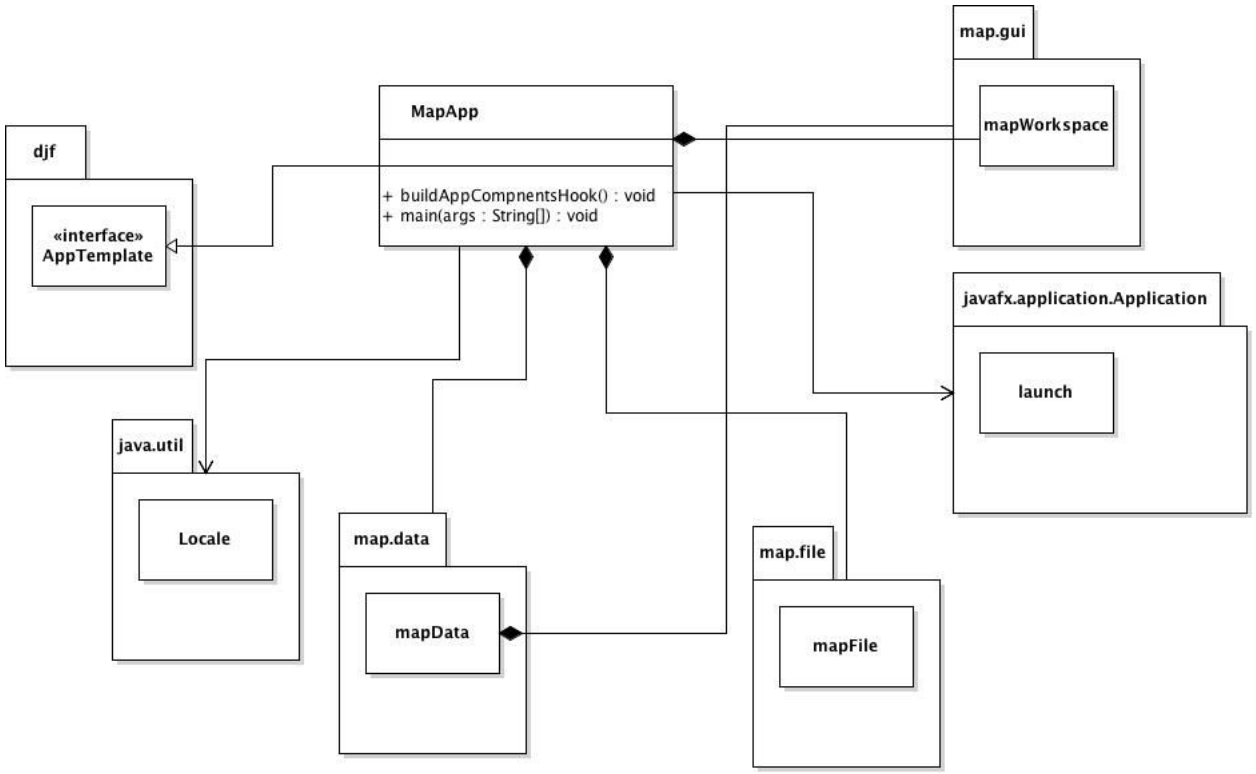

Figure 2.1: MapApp

Figure 2.2 shows the string constants created to help tie into the actual style sheet.

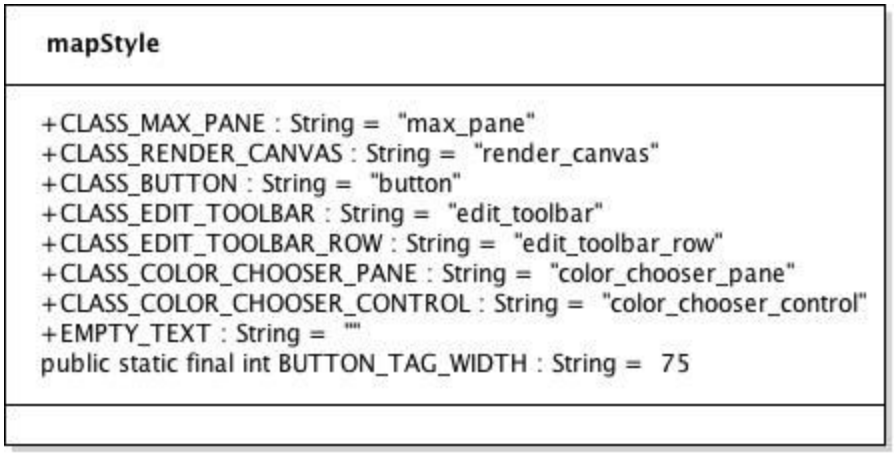

Figure 2.2: mapStyle

Figure 2.3 shows the data classes associated with the program.

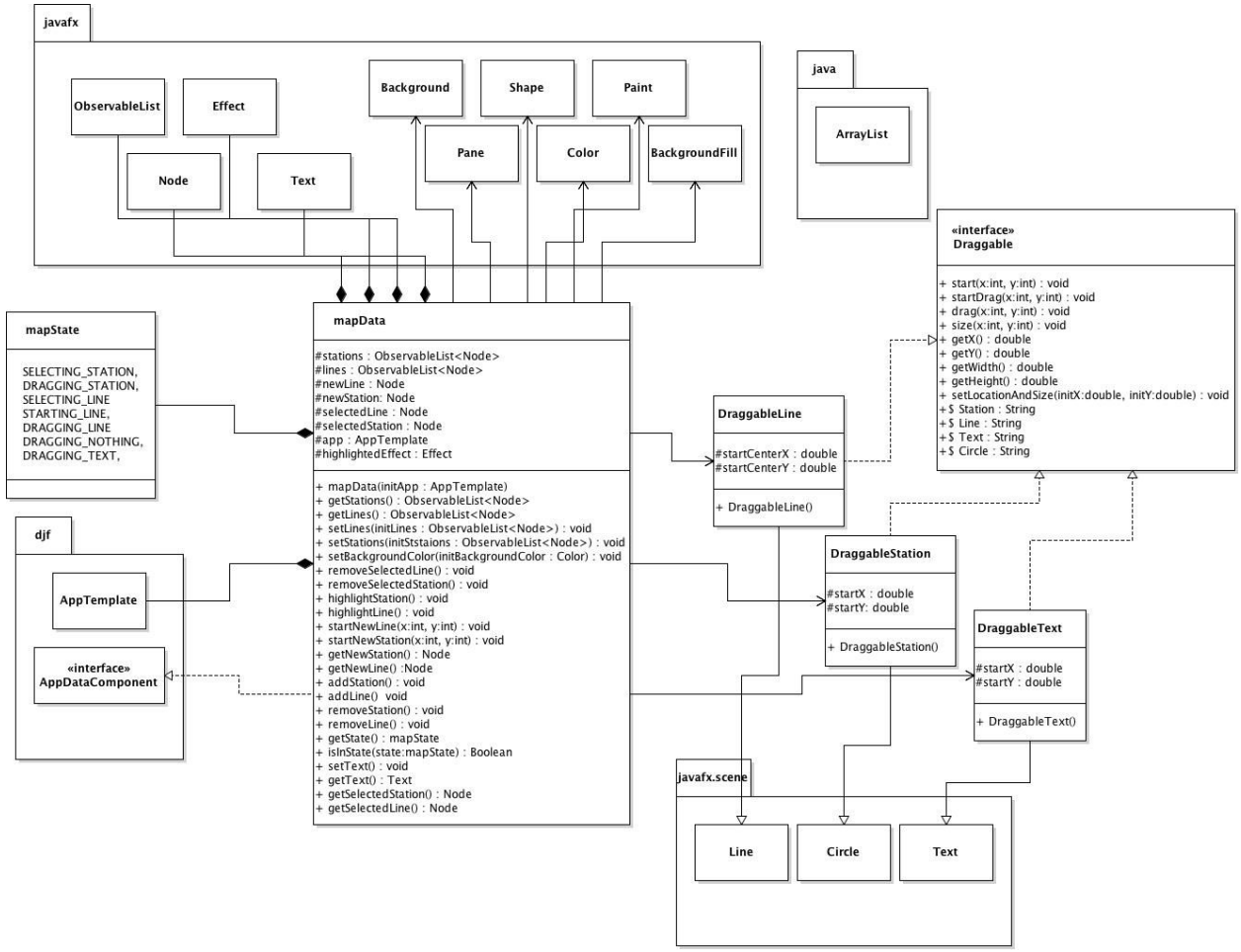

Figure 2.3: mapState, mapData, Draggable, DraggableLine, DraggableStation, DraggableText

Figure 2.4 shows the mapFiles class and the attributes included in order to make, save to, load from, and export data to a JSON file.

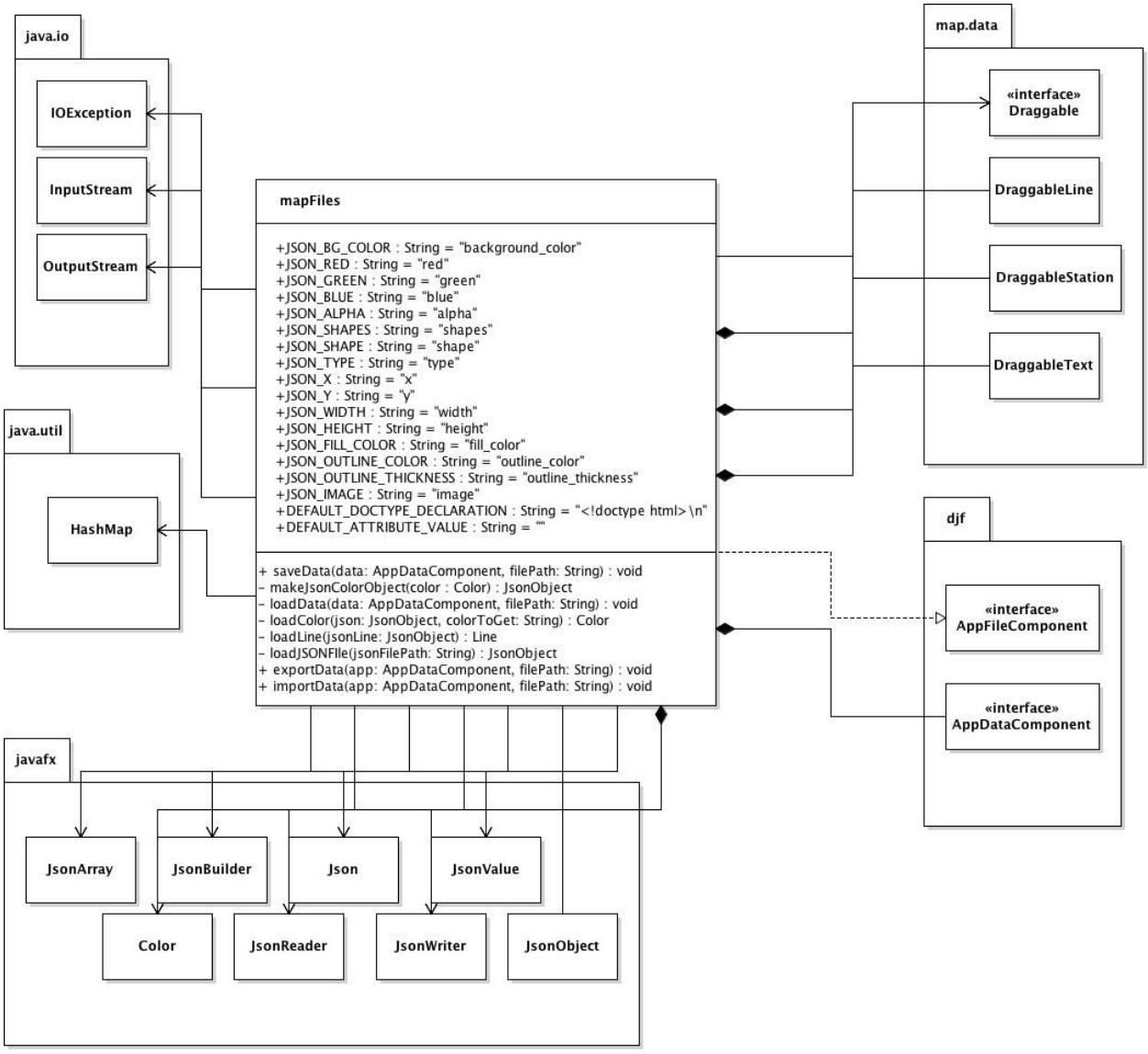

Figure 2.4: mapFile

Figure 2.5-2.7 will show all of the classes within the GUI package of the application. Classes include mapWorkspace, MapEditController, and CanvasController.

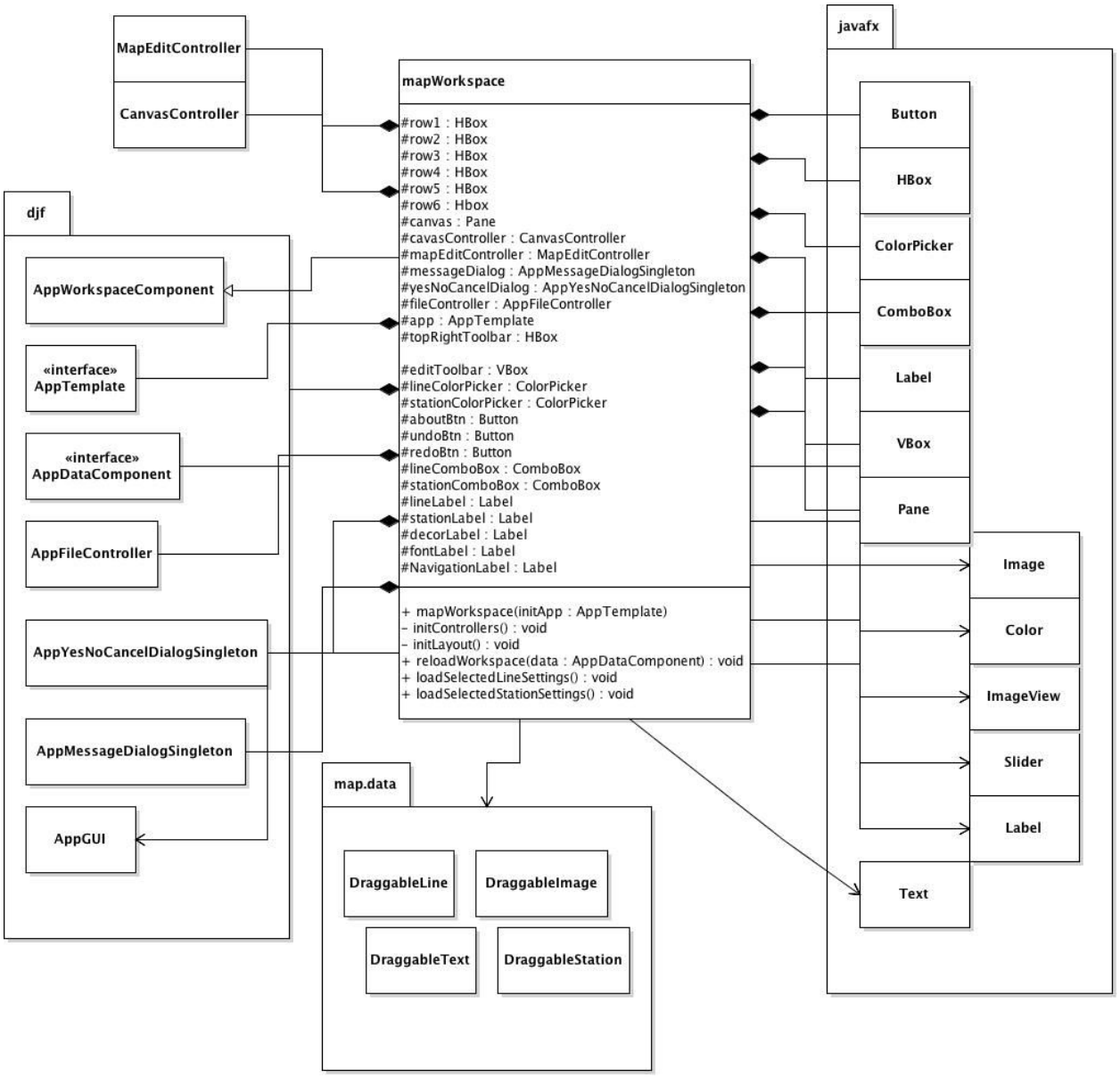

Figure 2.5: mapWorkspace

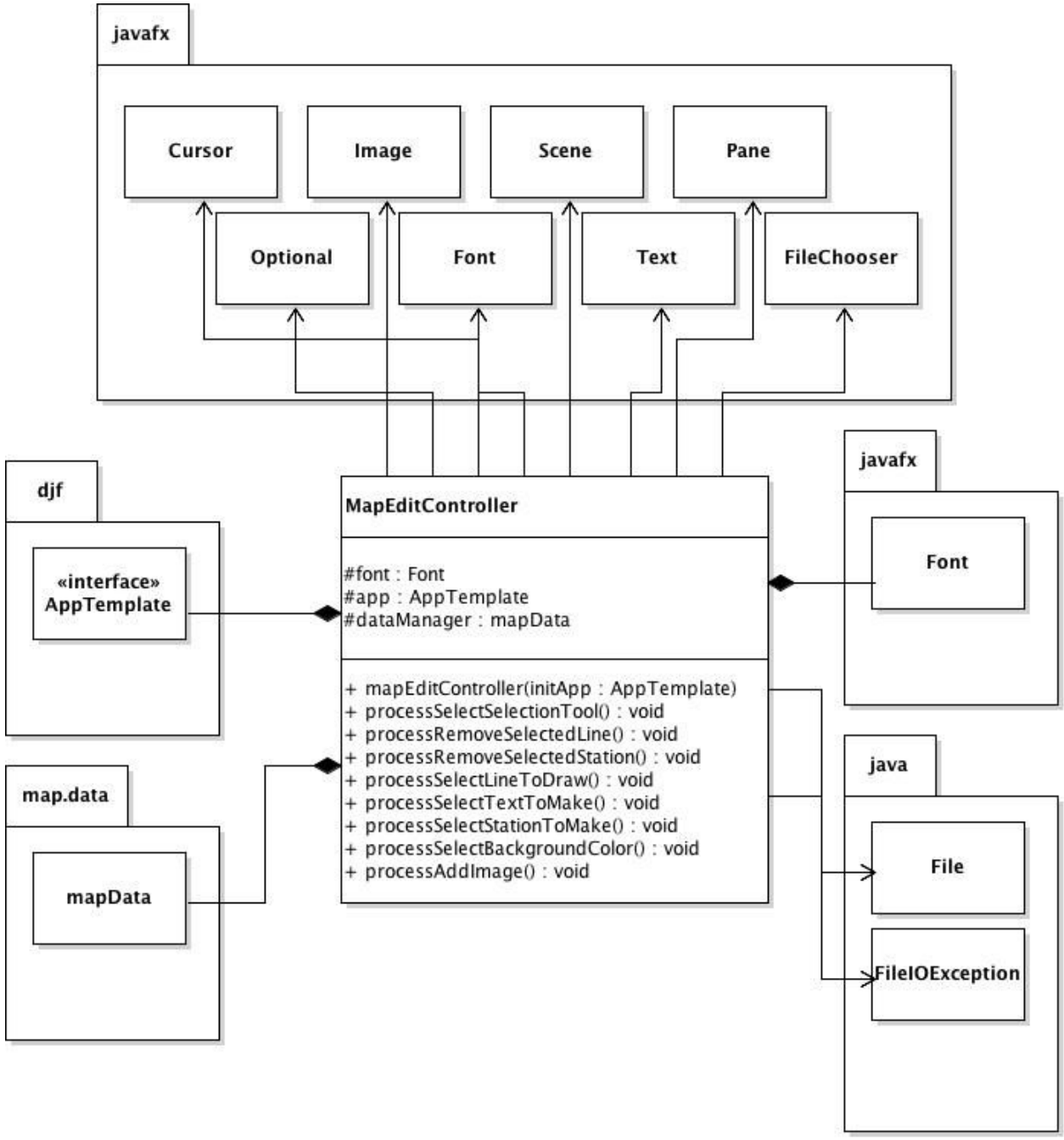

Figure 2.6: MapEditController

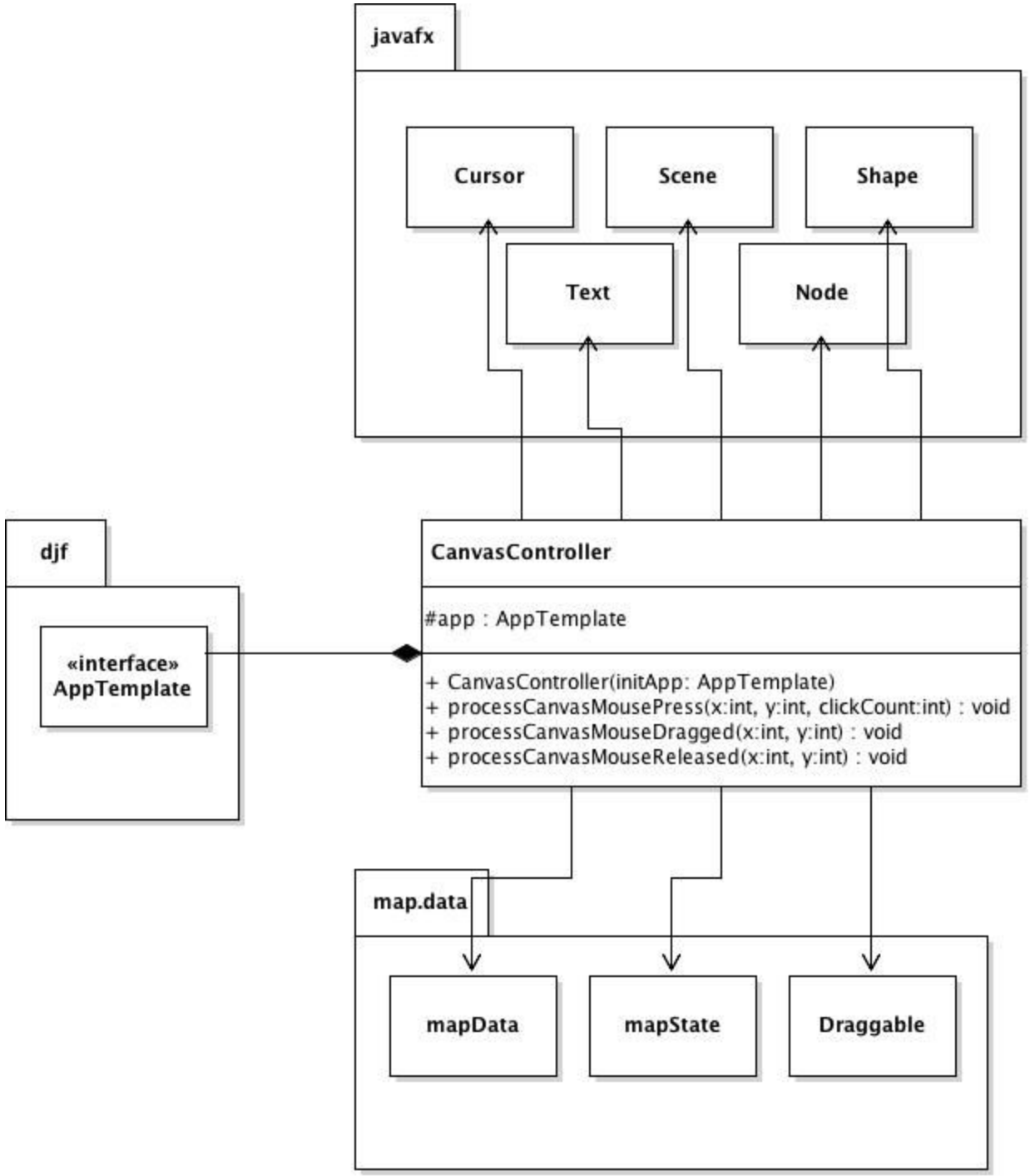

Figure 2.7: CanvasController

3 Use case sequence diagrams

The following UML Class Diagrams show relationships between certain classes for different use cases. The method calls have been labeled to easily see which functions are connecting the two classes together.

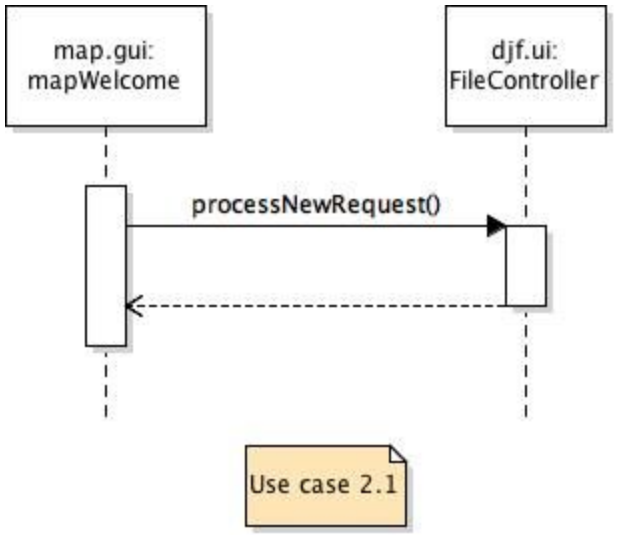

Figure 3.1: UML Sequence for the Use Case 2.1 from the SRS

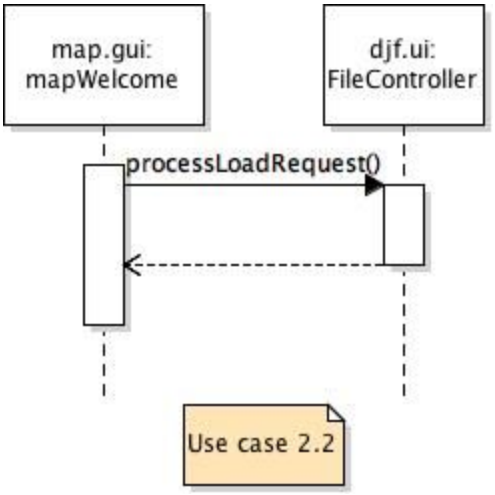

Figure 3.2: UML Sequence for the Use case 2.2 from the SRS

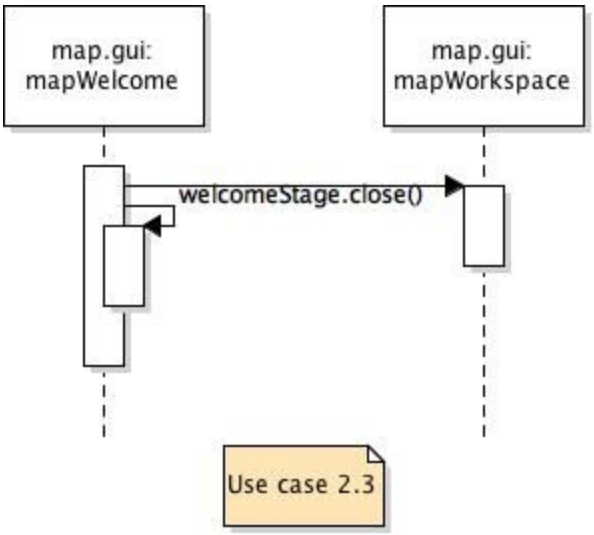

Figure 3.3: UML Sequence for the Use case 2.3 from the SRS

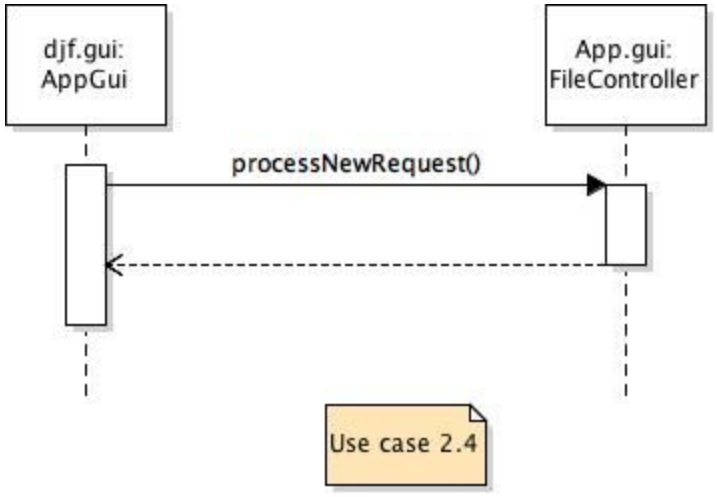

Figure 3.4: UML Sequence for the Use case 2.4 from the SRS

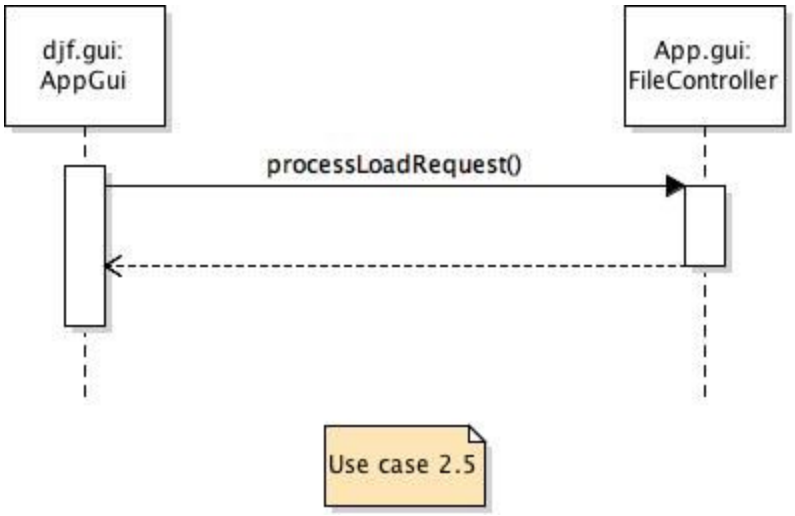

Figure 3.5: UML Sequence for the Use case 2.5 from the SRS

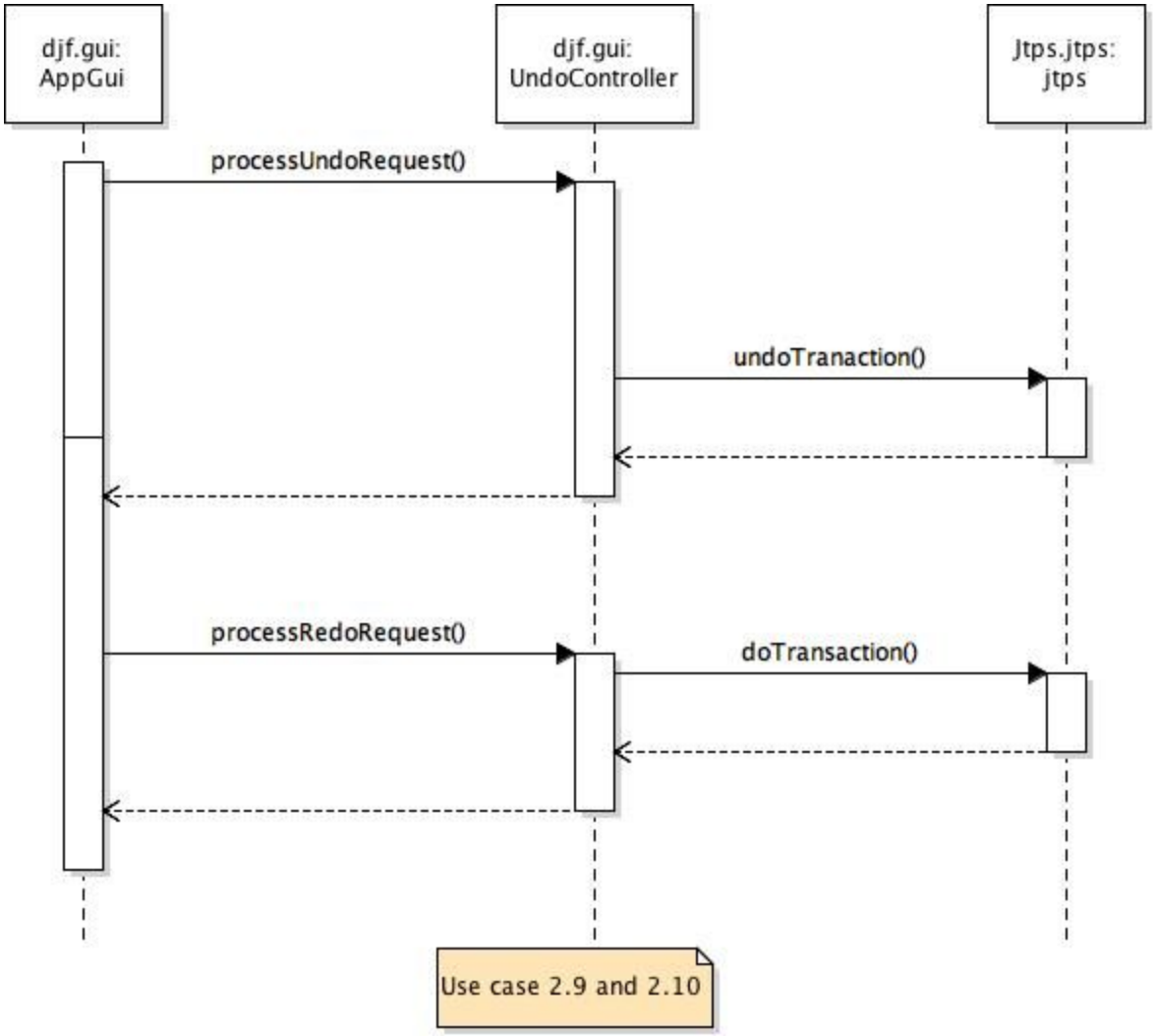

Figure 3.6: UML Sequence for the Use case 2.9 and 2.10 from the SRS

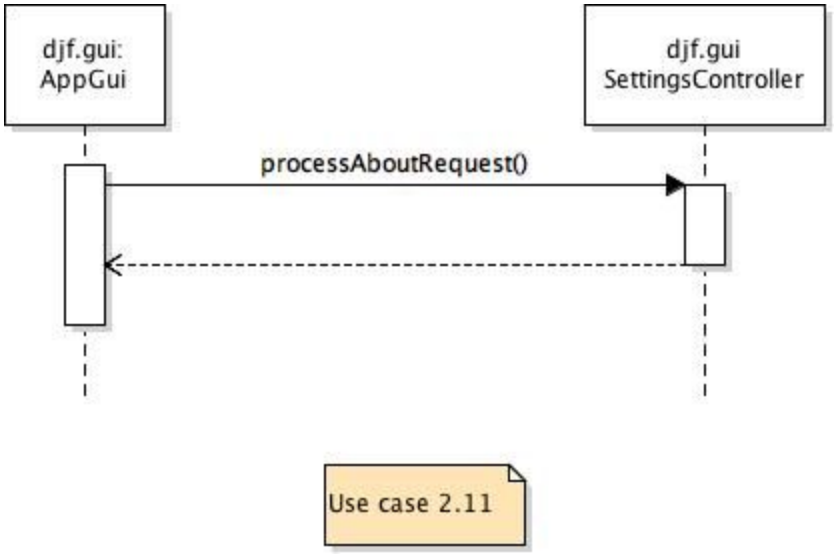

Figure 3.7: UML Sequence for the Use case 2.11 from the SRS

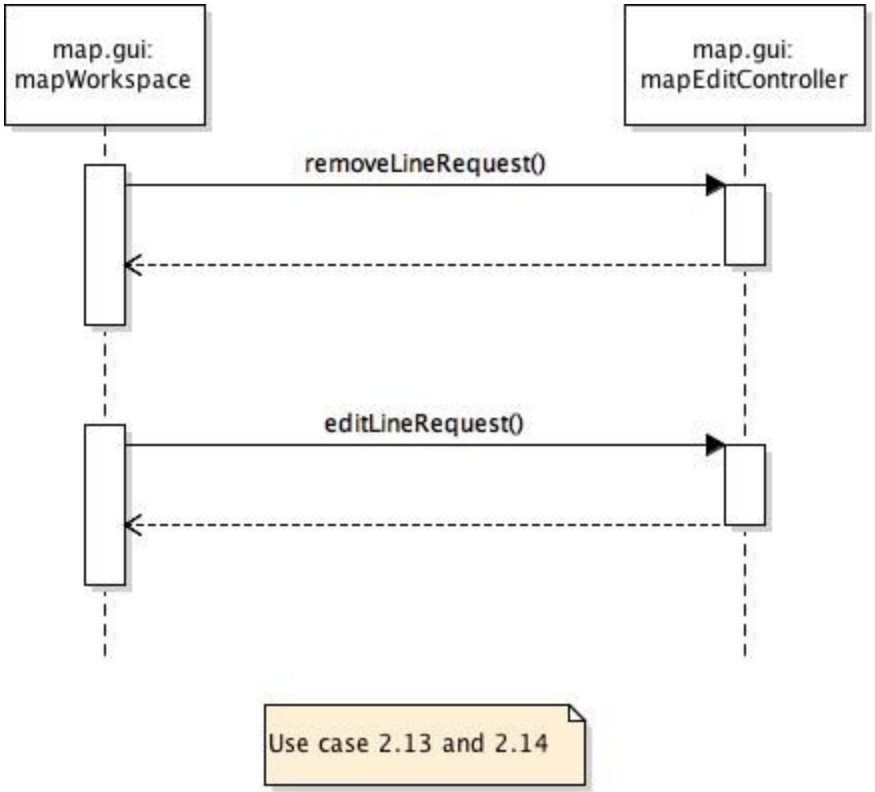

Figure 3.8: UML Sequence for the Use case 2.13 and 2.14 from the SRS

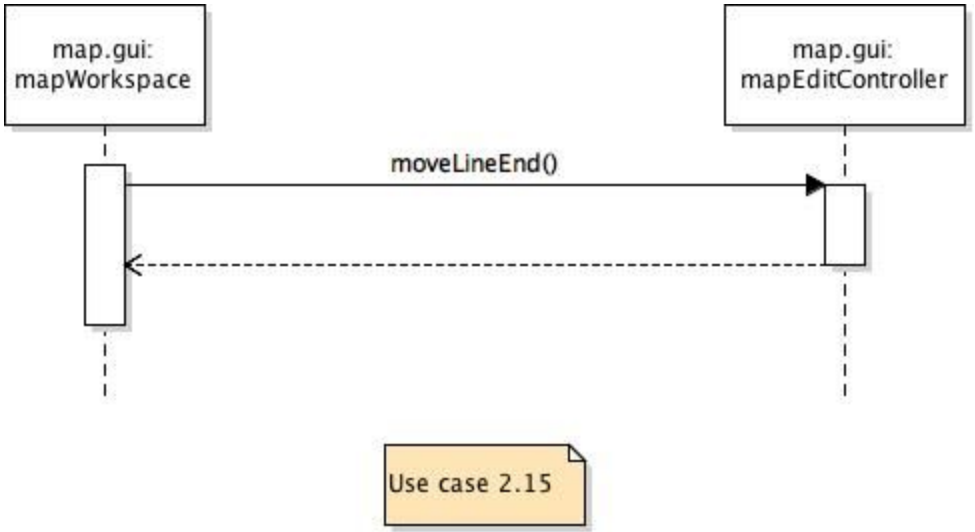

FIgure 3.9: UML Sequence for the Use case 2.15 from the SRS

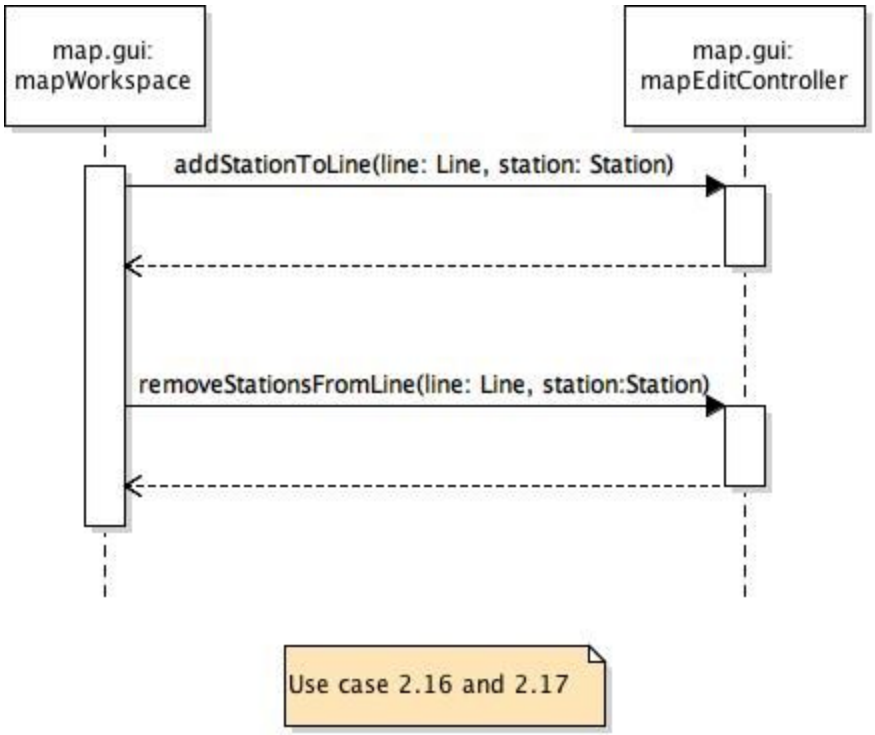

Figure 3.10: UML Sequence for the Use case 2.16 and 2.17 from the SRS

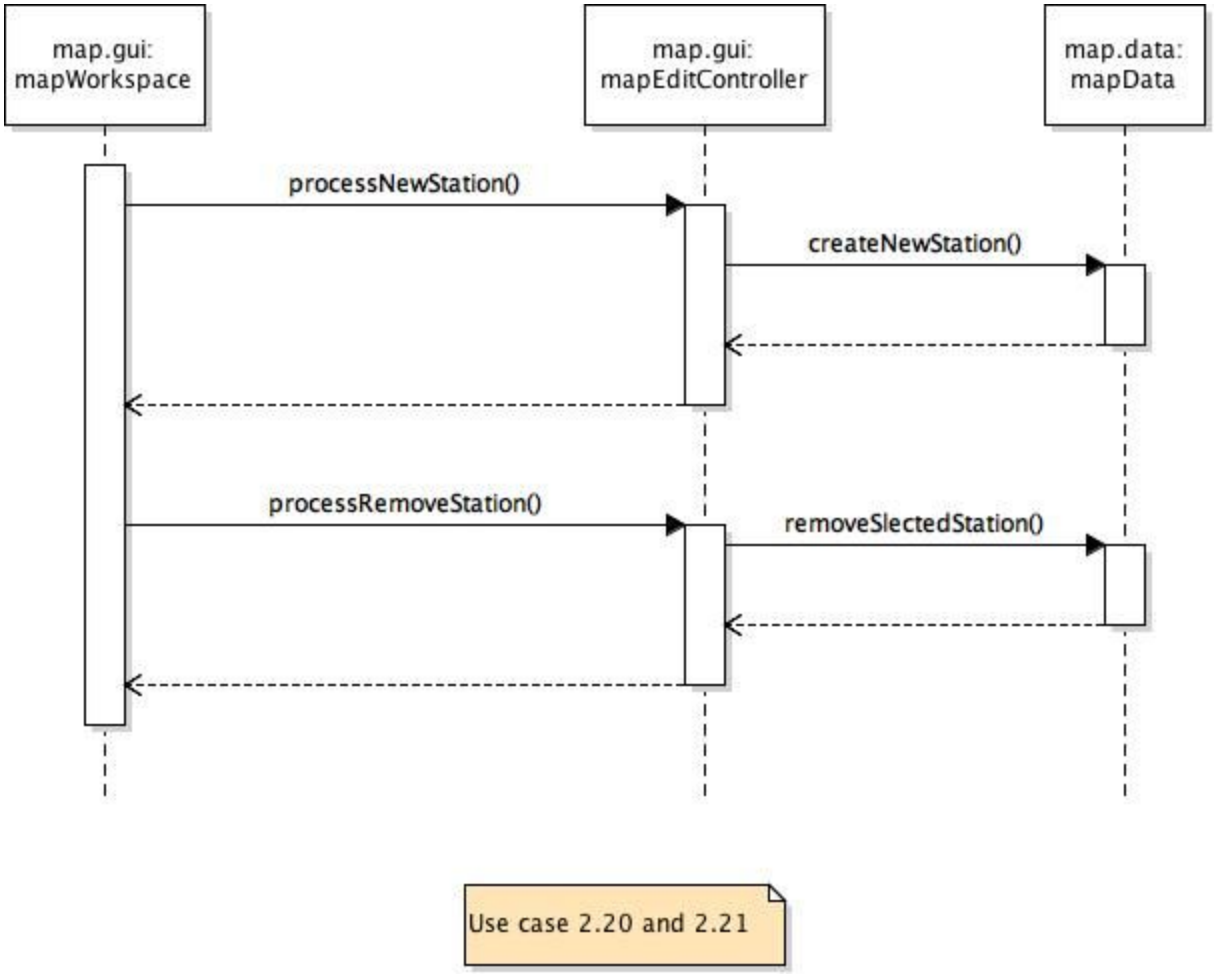

Figure 3.11: UML Sequence for the Use case 2.20 and 2.21 from the SRS

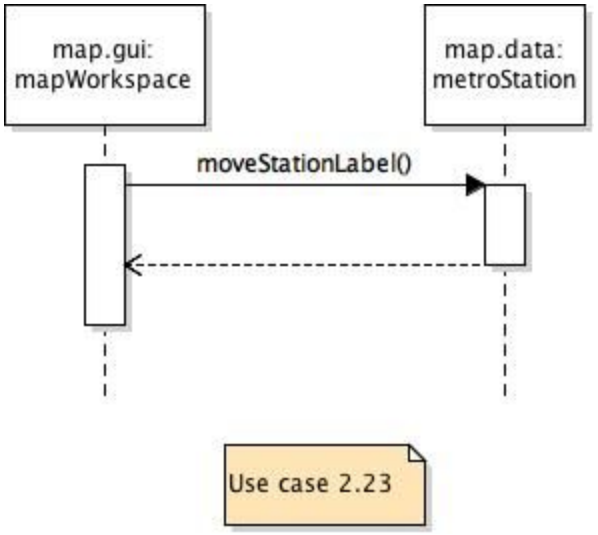

Figure 3.12: UML Sequence for the Use case 2.23 from the SRS

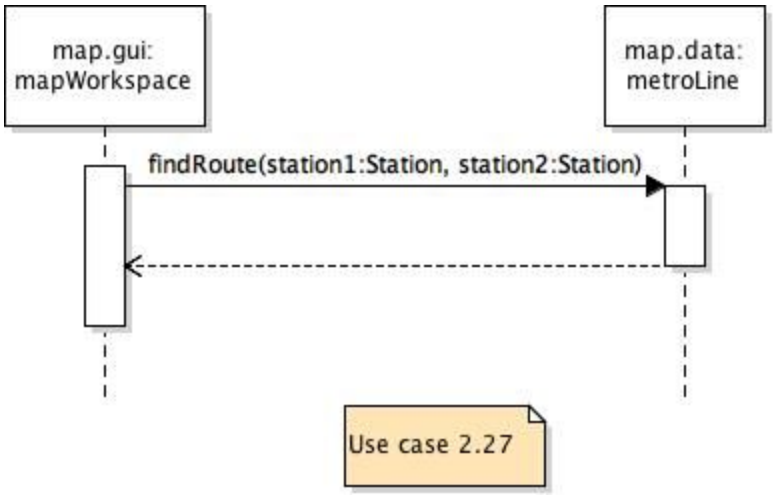

Figure 3.13: UML Sequence for the Use case 2.27 from the SRS

4 FIle Structure

The application uses an organized file structure. The reason it needs such structure is so the program appropriately access saved work, as well as save to the appropriate folder. Exporting to a separate folder will also need to be implemented. There is also a images folder so that any image the user adds can be stored for access every time the workspace needs to load a photo. The .jar (actual application file) is stored here as well.