The Metro Map Maker
Software Design Description

Author: Dan Kim
Course: CSE 219
Instructor: Ritwik Banerjee
Homework 3

1 Introduction

Finding your way around a new city can be challenging so many people look to the Internet for help. Cities with subway systems typically provide maps to help one navigate from one stop to another across a number of intersecting lines. These maps let one chart which lines to take and how many stops it will be before one arrives and the user can typically choose between multiple routes.

The **Metro Map Maker (i.e. M3)** application will provide the user with a set of tools to build graphical representations of city subway systems with named lines and named stops and intersecting lines and landmarks. It will also provide a means for calculating the best route to take to journey from one particular station to another. Finally, it will provide an export feature such that it may export a generated map and associated metro system information to a format that can be used by a corresponding Web application that will be able to make use of it.

1.1 Purpose

The purpose of this document is to specify how our Metro Map Maker program should look and operate. The intended audience for this document is all the members of the development team, those who will design the maps for use with the Web application, and the potential users of such an application. This document serves as the blueprint for how the map creation tool should ultimately be constructed. Upon completing the reading of this document, one should clearly visualize how the application will look and operate.

1.2 Scope

For this project the goal is for users to easily make and edit subway maps. There will be an emphasis on ease of use. Note that there will be a common export format that will be provided for exported subway system data such that all maps can be used by a uniform application.

1.3 Definitions, acronyms, and abbreviations

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.

GUI – Graphical User Interface, visual controls like buttons inside a window in a software application that collectively allow the user to operate the program.

IEEE – Institute of Electrical and Electronics Engineers, the "world's largest professional association for the advancement of technology".

JavaScript – the default scripting language of the Web, JavaScript is provided to pages in the form of text files with code that can be loaded and executed when a page loads so as to dynamically generate page content in the DOM.

Stylesheet – a static text file employed by HTML pages that can control the colors, fonts, layout and other style components in a Web page.

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

Use Case Descriptions – A formal format for specifying how a user will interact with a system.

1.4 References

Metro Map Maker SRS - Software Requirement Specification for the Metro Map Maker application provided by Ritwik Banerjee.

IEEE Std 830™-1998 (R2009) – IEEE Recommended Practice for Software Requirements Specification

Zombiquarium SDD - Sample SDD provided by Richard McKenna for CSE 219 students

1.5 Overview

This Software Design Description will clearly define how to design the Metro Map Maker application using UML according to the Metro Map Maker Software Requirement 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

This design will encompass the Metro Map Maker application, the jTPS library, the DesktopJavaFramework framework, and the PropertiesManager framework in its construction. We will also rely on the Java API to build them.

2.1 Application Overview

MetroMapMaker, jTPS, DesktopJavaFramework, and PropertiesManager will be developed together. Figures 2.10 to 2.13 specifies all the components to be developed and the packages they are placed in.

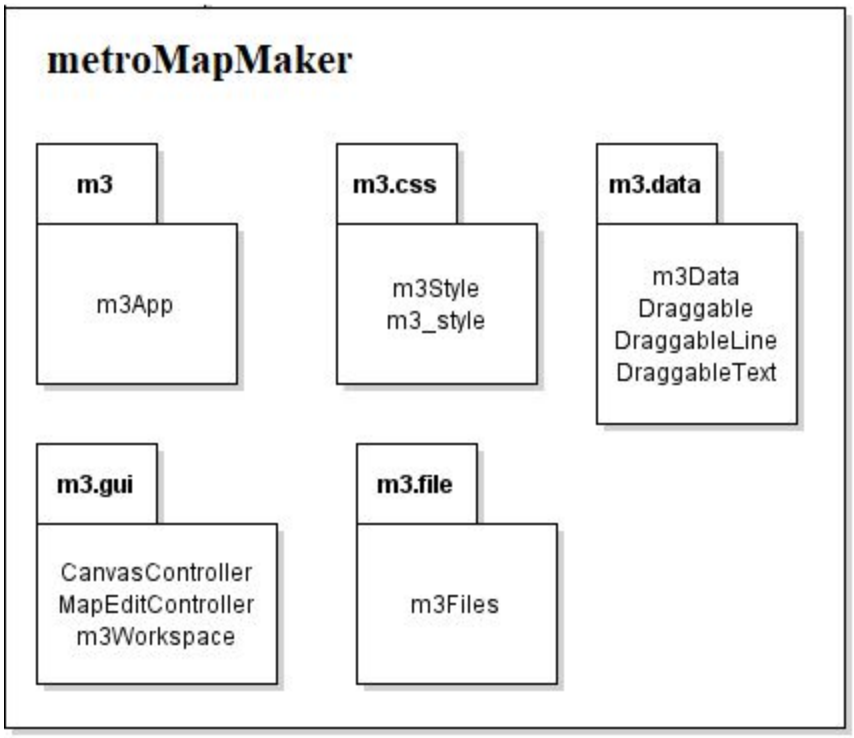

Figure 2.10: MetroMapMaker Package Overview

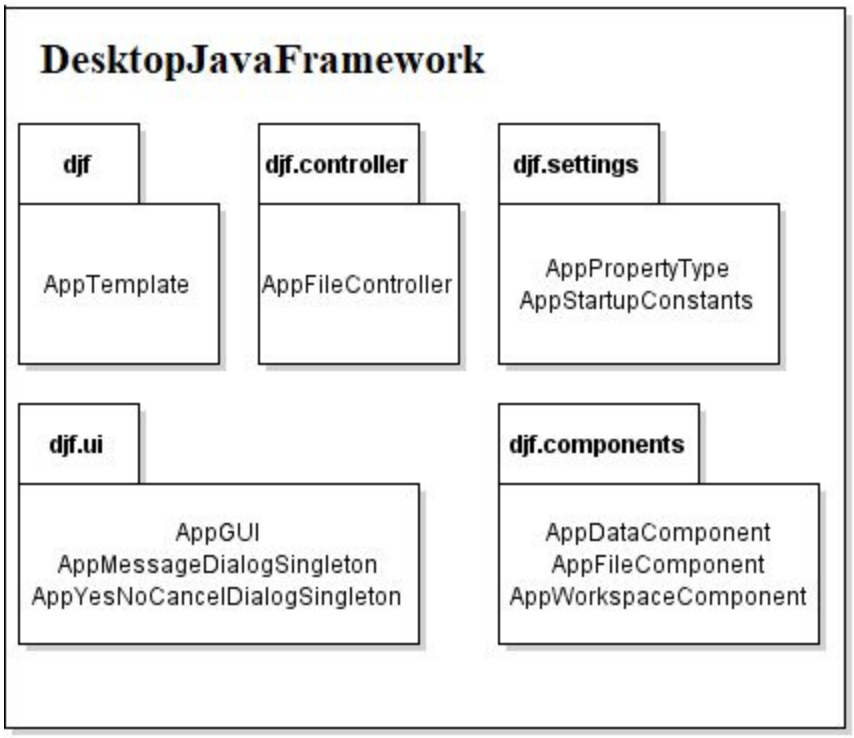

Figure 2.11: DesktopJavaFramework Package Overview

Figure 2.12 & 2.13: jTPS and PropertiesManager Package Overview

2.2 Java API Usage

Our design will utilize the following classes provided by the Java API.

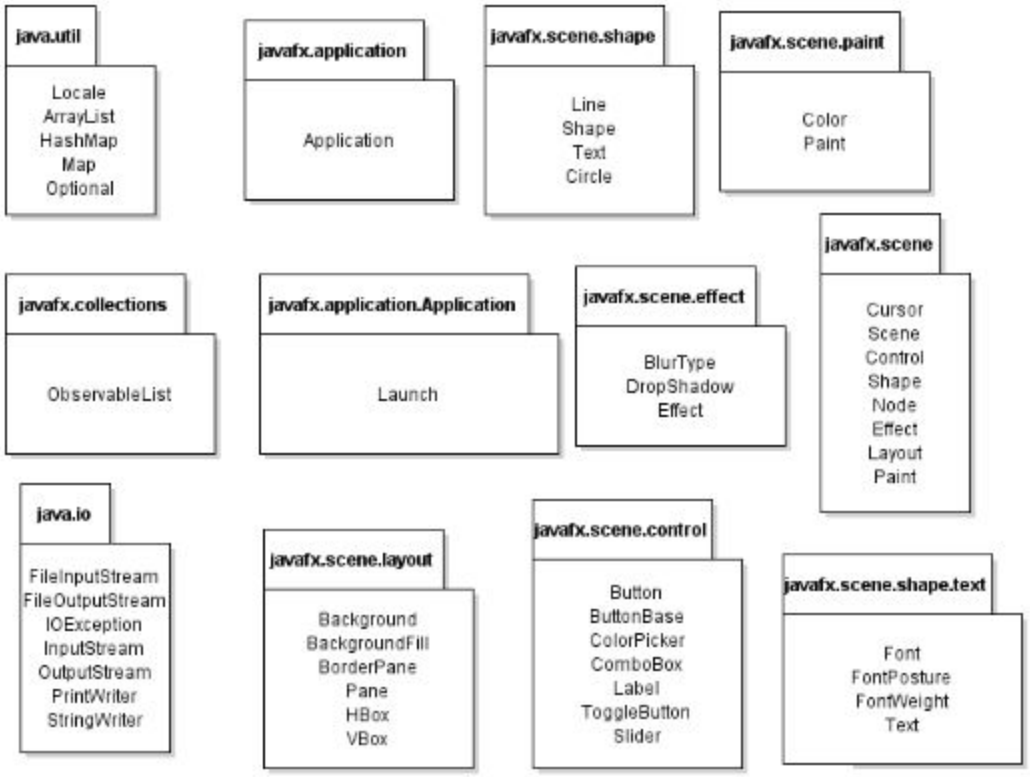

Figure 2.2: Java API Packages and Classes To Use

2.3 Java API Usage Descriptions

Class	**Use**
Locale	Utilized for initialization of launch
ArrayList	Easier way of grouping elements together in an array
HashMap	Sorting structure to implement key/vault aspects
Map	Basic structure for our canvas
Optional	To use for alerts

Table 2.1 : Java API's java.util package

Class	Use
Line	Basic shape for map routes
Shape	Used for subclasses
Text	Text for any shape
Circle	Basic shape for map stations

Table 2.2: Java API's javafx.scene.shape package

Class	Use
Cursor	Selecting shapes within the map canvas
Scene	Framework for windows
Control	Parent class for subclasses required
Shape	Used for subclasses
Node	Utilized for grouping more specific shapes
Effect	For highlighting
Layout	For the editing the style
Paint	To add color into our styles

Table 2.3 : Java API's javafx.scene

3 Class-Level Design Viewpoint

The following UML class diagrams provide an overview of the metroMapMaker application, the DesktopJavaFramework framework, the jTPS library, and the PropertiesManager. Due to the complexity of the project, the class diagrams will be presented as a series of diagrams going from overview diagrams to detailed diagrams.

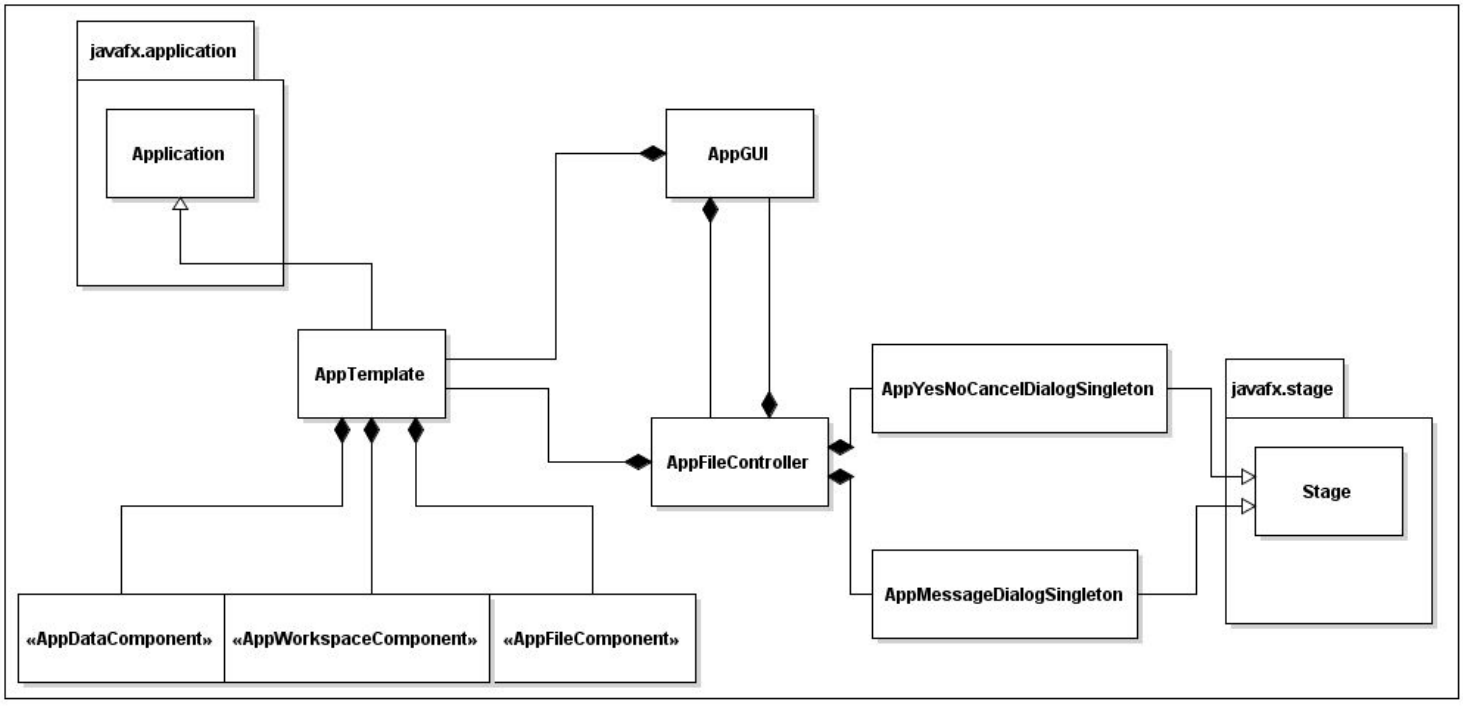

Figure 3.1: DesktopJavaFramework Overview UML Class Diagram

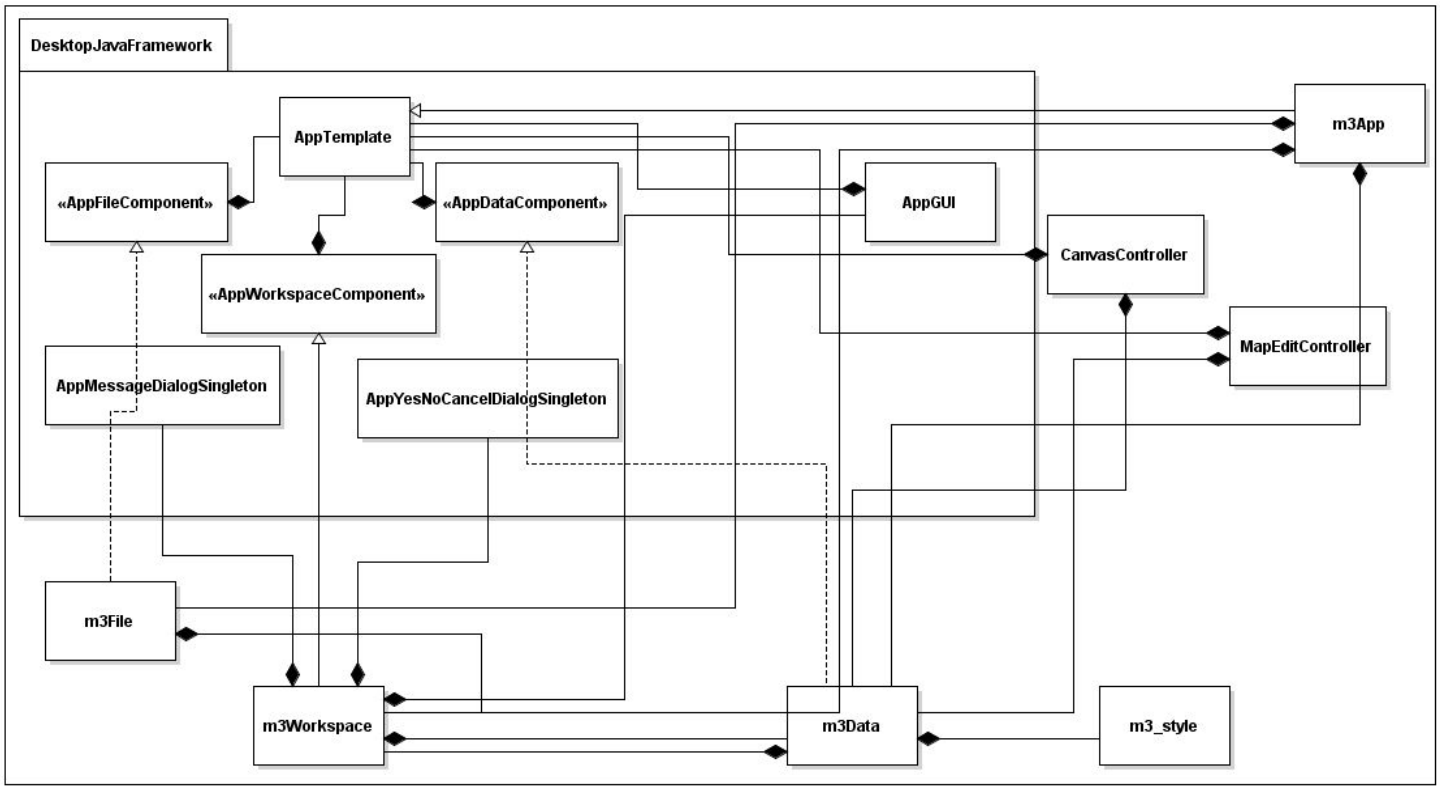

Figure 3.2: MetroMapMaker Overview UML Class Diagram

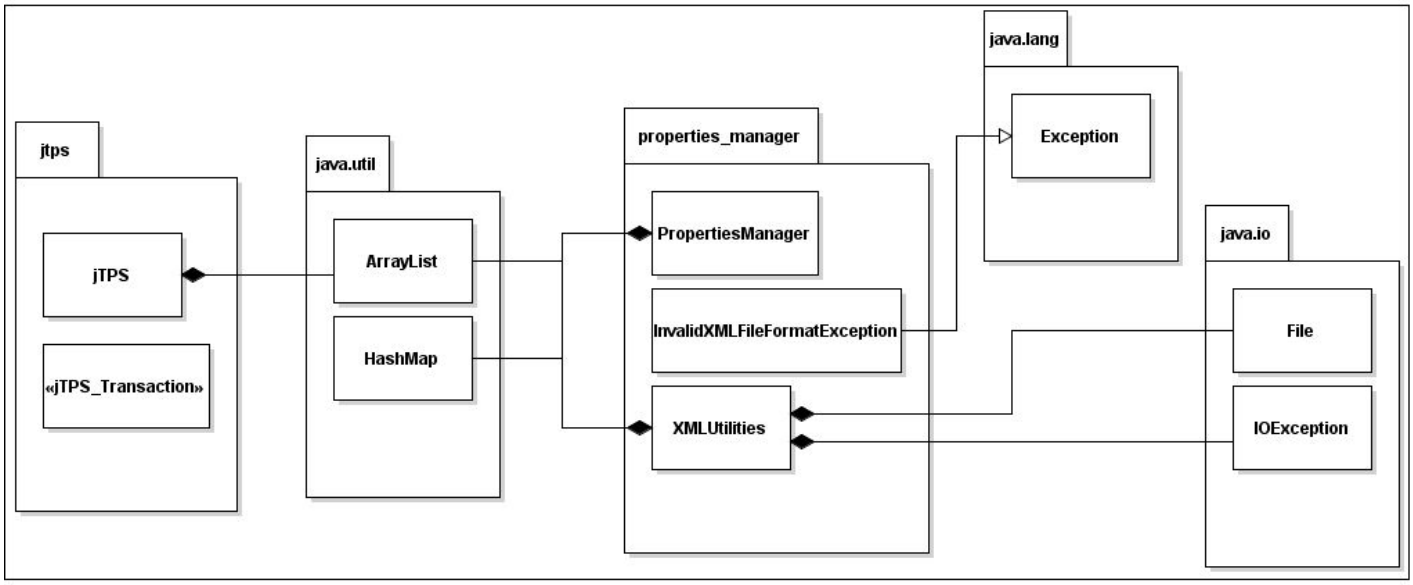

Figure 3.3: jTPS and PropertiesManager Overview UML Class Diagram

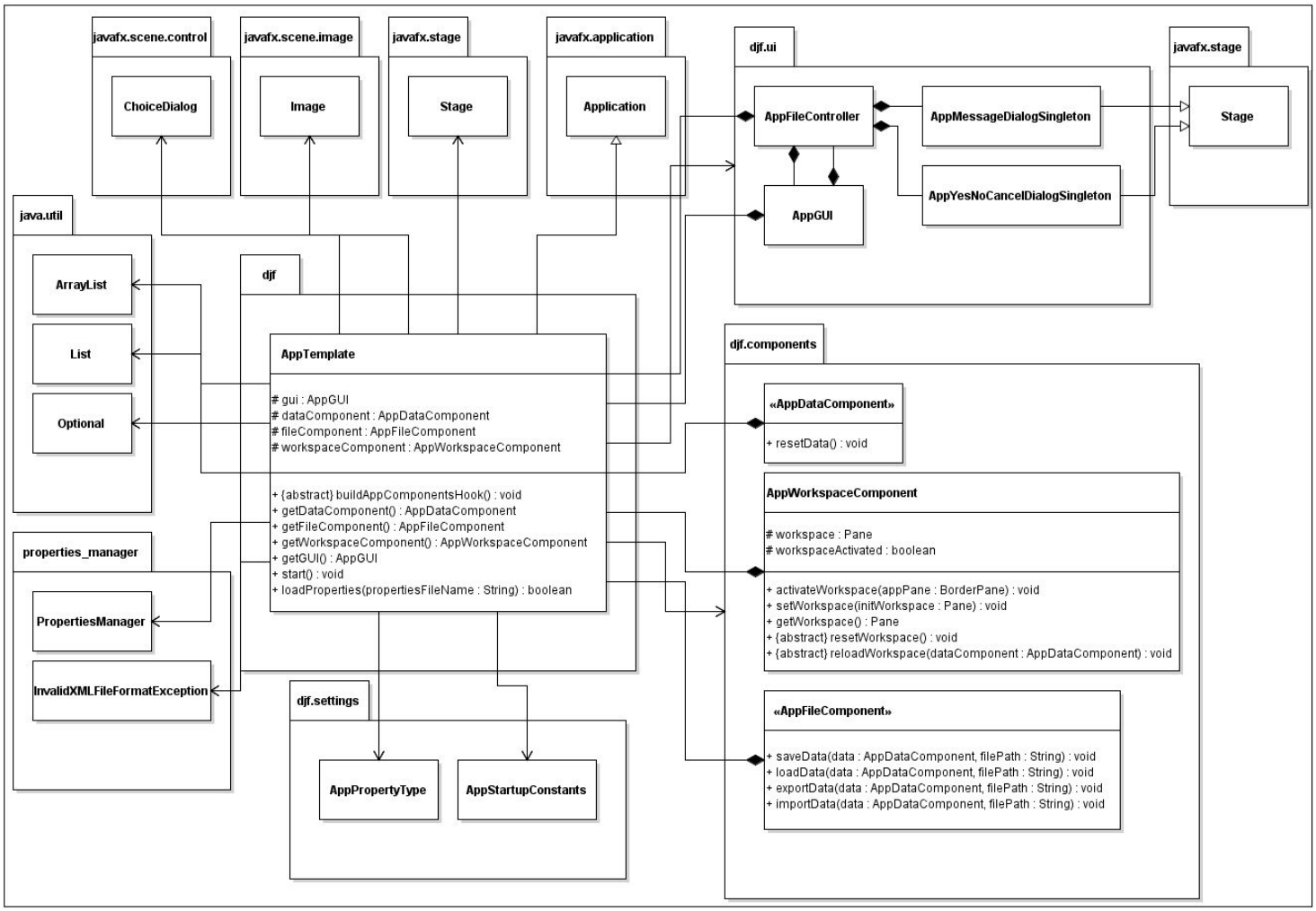

Figure 3.4 : Detailed AppTemplate and Components UML Class Diagram

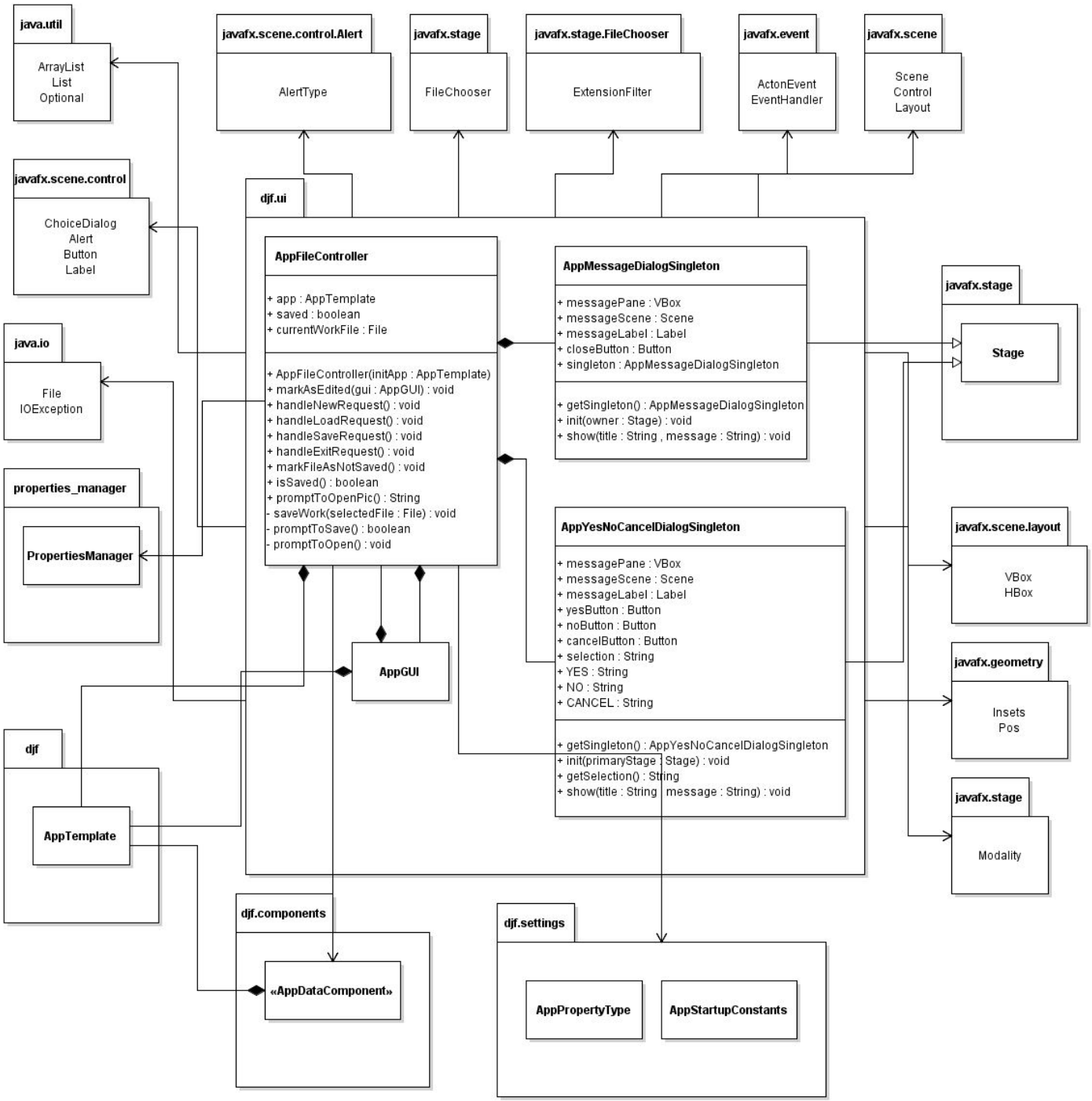

Figure 3.5 : Detailed AppFileController and Dialogs UML Class Diagrams

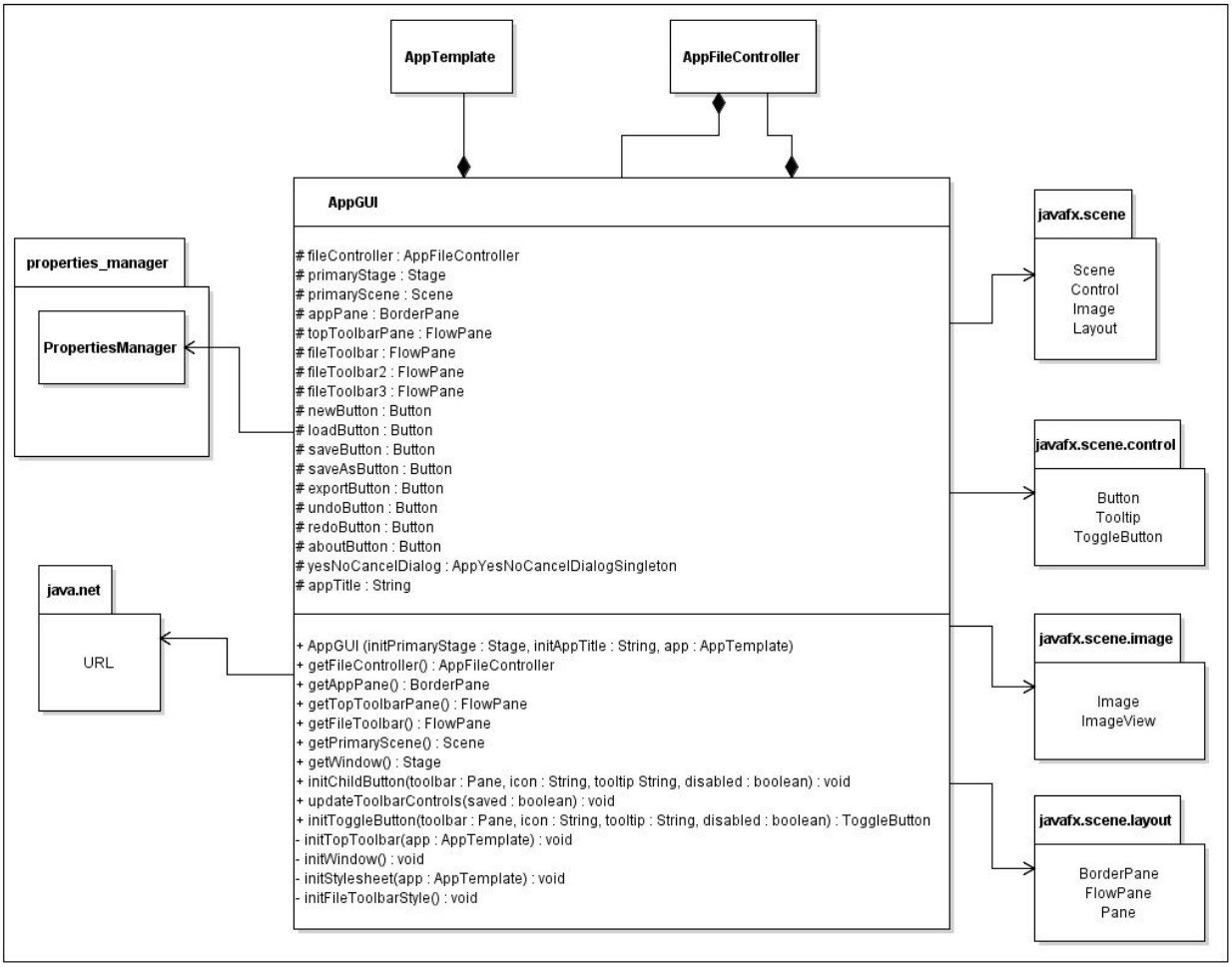

Figure 3.6 : Detailed AppGUI UML Class Diagram

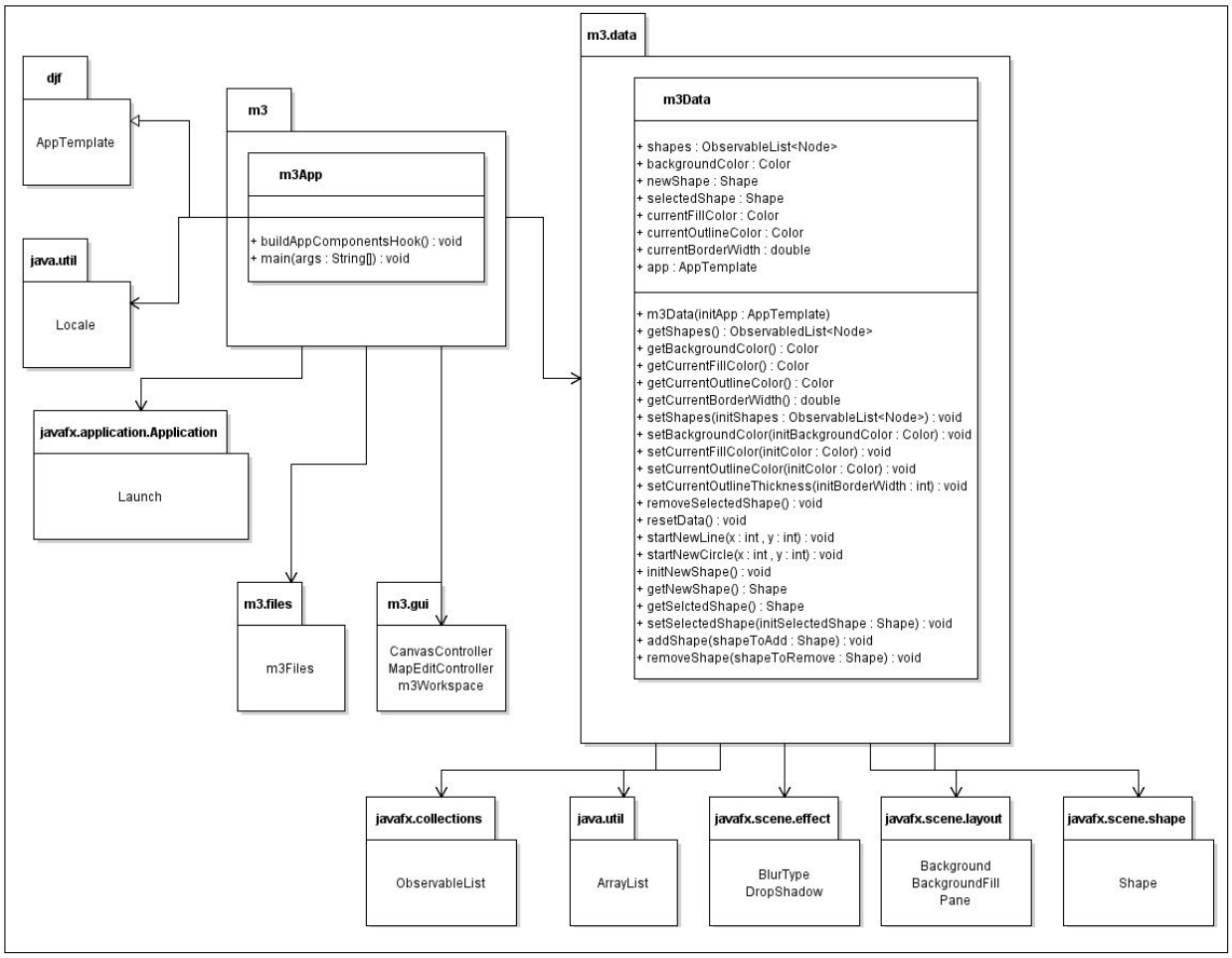

Figure 3.7 : Detailed m3App and m3Data UML Class Diagrams

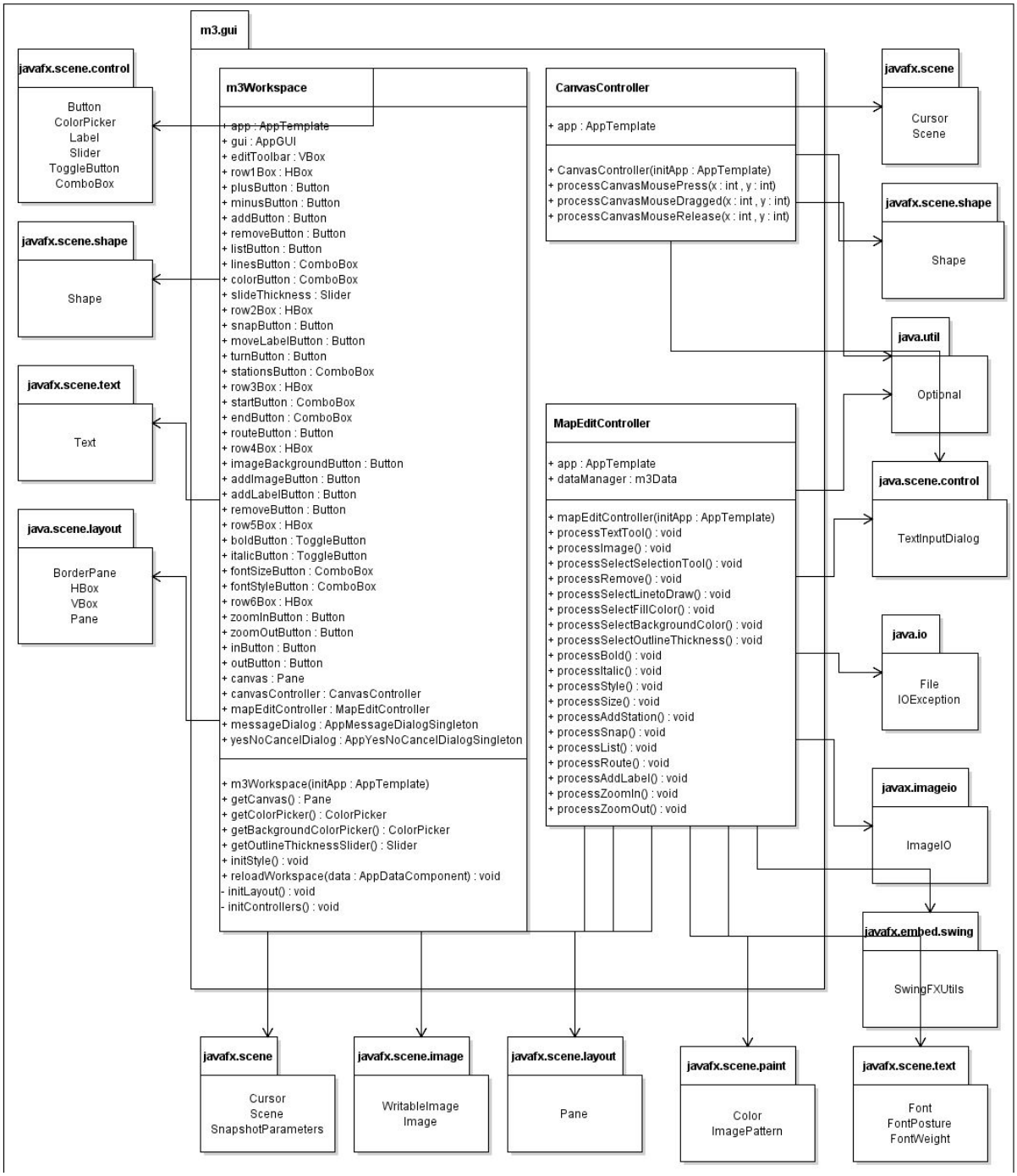

Figure 3.8 : Detailed m3Workspace and Controllers UML Class Diagrams

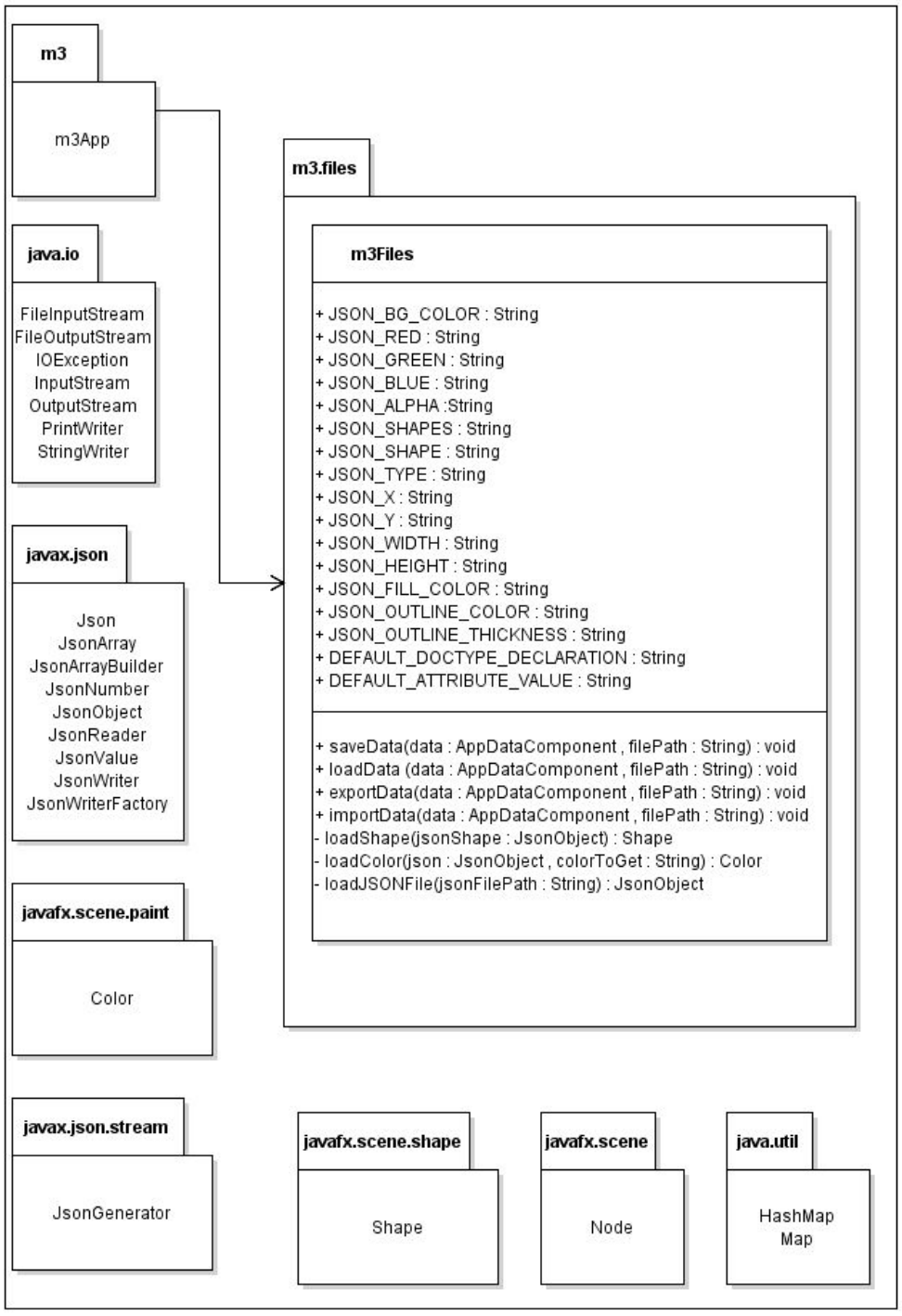

Figure 3.9 : Detailed m3Files UML Class Diagrams

4 Method-Level Design Viewpoint

The following UML sequence diagrams describe the methods called within the code that need to be developed.

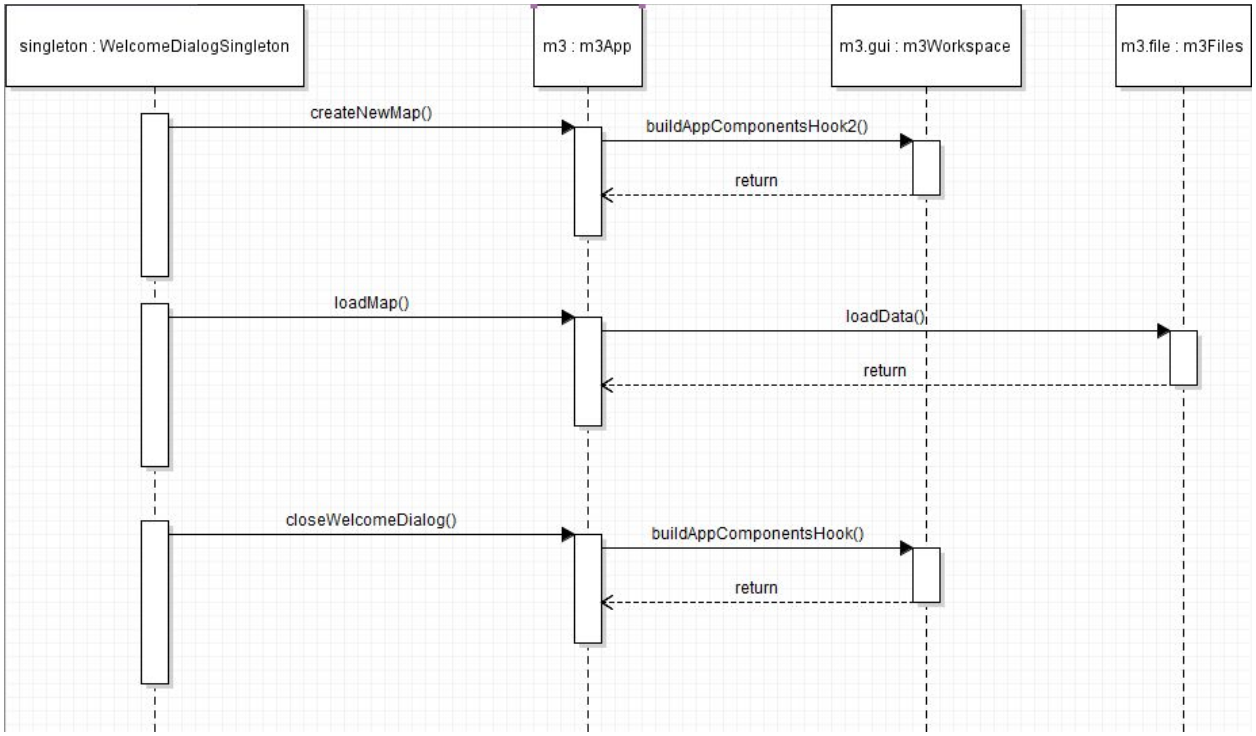

Figure 4.1: Welcome Dialog UML Sequence Diagram

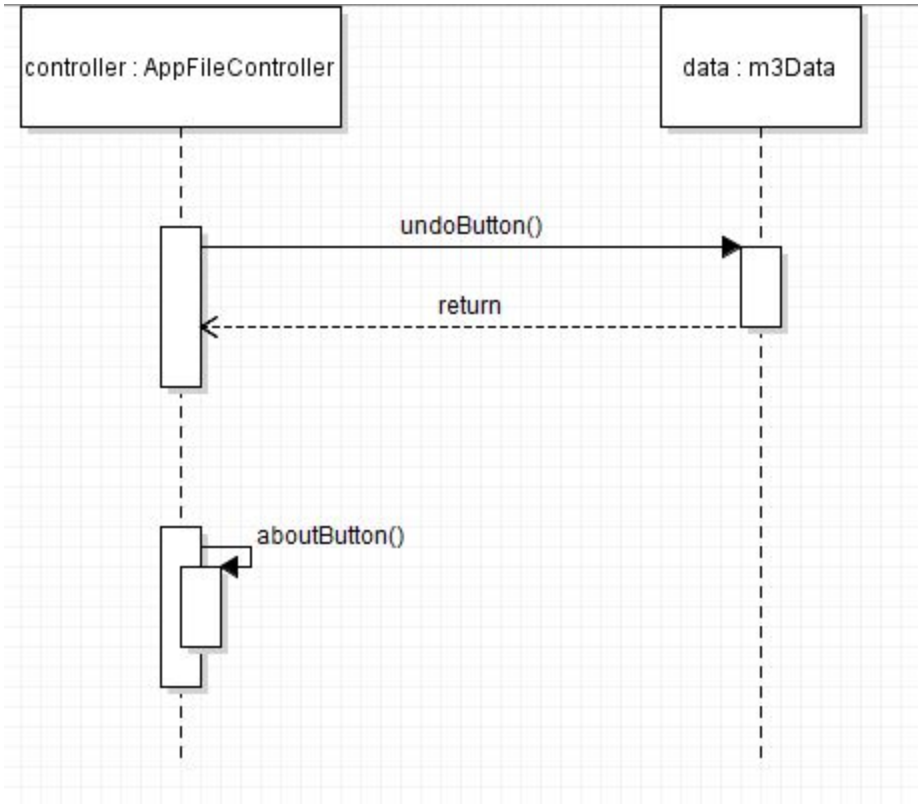

Figure 4.2: Undo, About UML Sequence Diagram

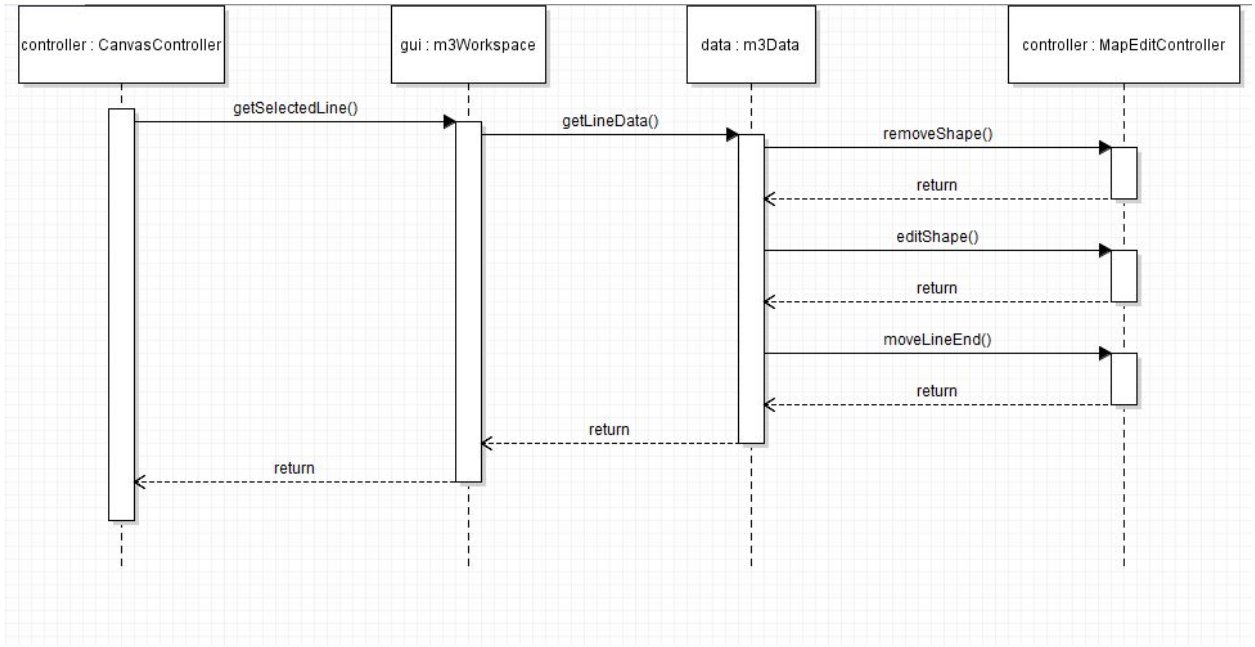

Figure 4.3: Editing Line UML Sequence Diagrams

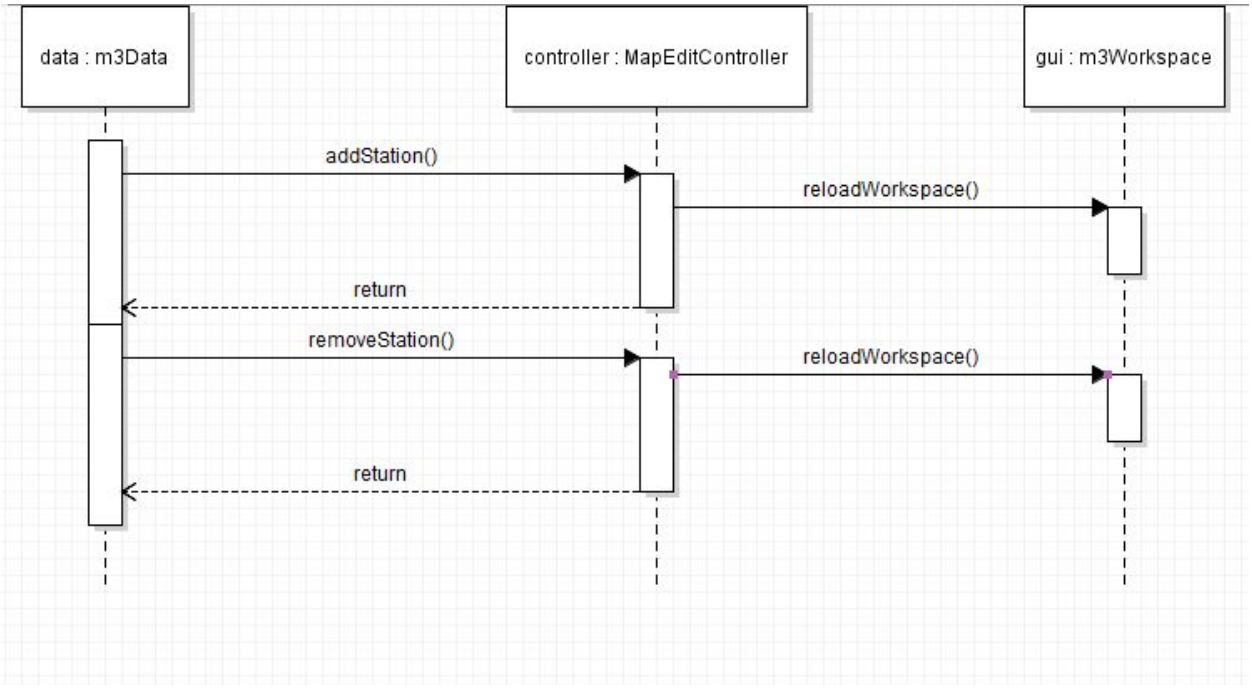

Figure 4.4: Station to Line Interaction UML Sequence Diagram

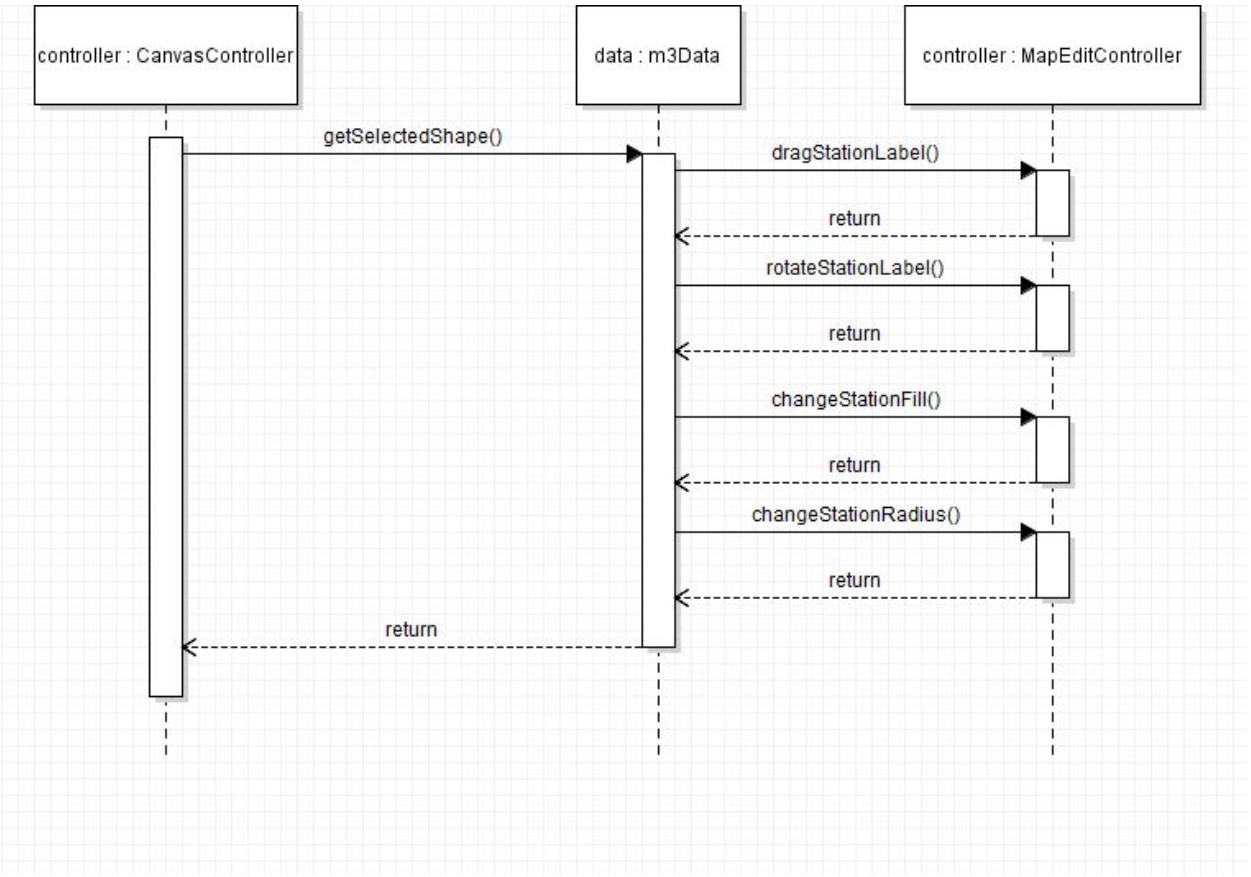

Figure 4.5: Metro Stations Toolbar UML Sequence Diagram

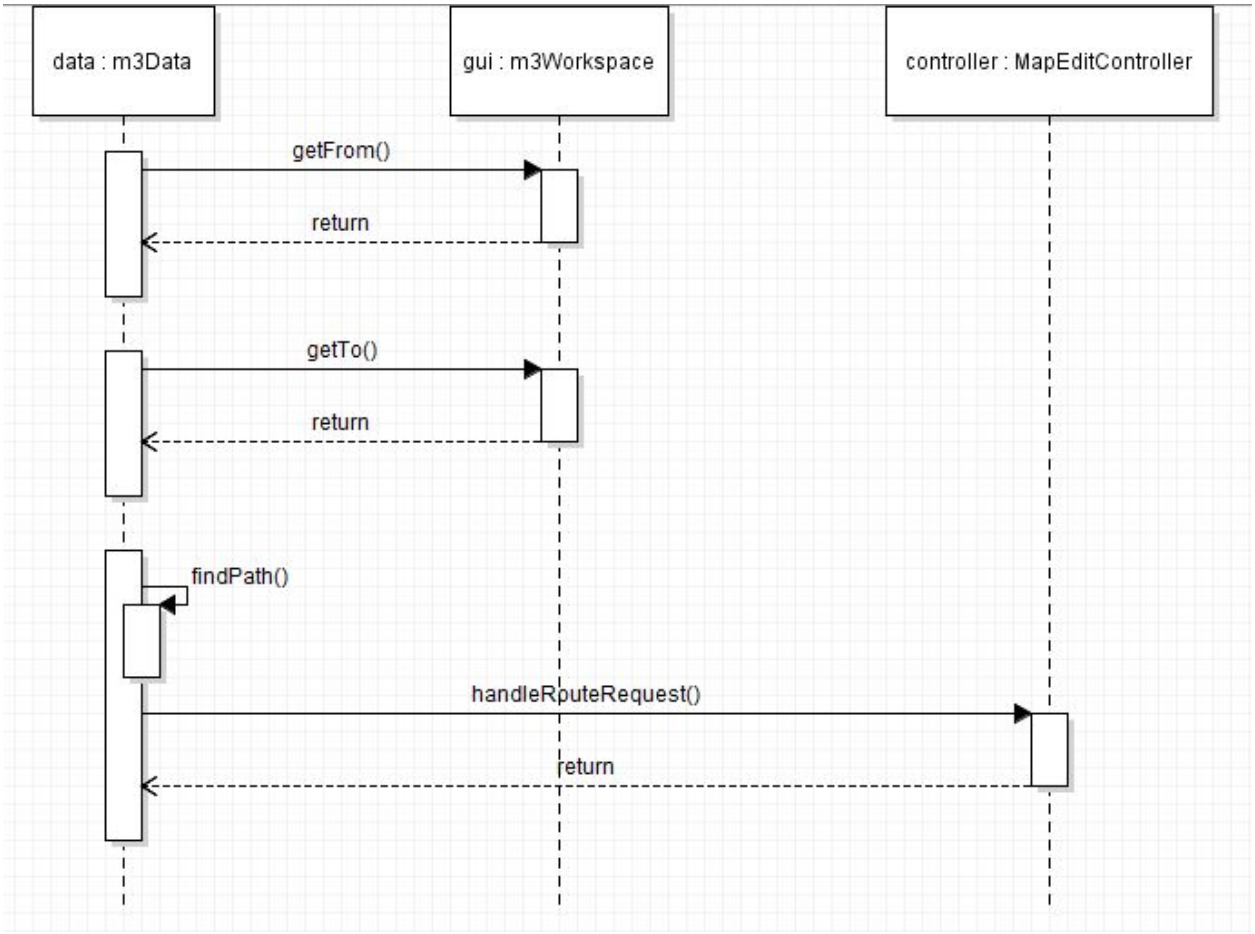

Figure 4.6: Station Router UML Sequence Diagram

5 File Structure and Formats

Note that the DesktopJavaFramework, jTPS, and PropertiesManager modules will be imported for the MetroMapMaker application. Figure 5.1 specifies the necessary file structure the application will use. Within the build lies the generated classes, whereas within the data lies the necessary XML files containing information for various icon images. The dist file contains the libraries required while the images file contains all the images we need. The MetroMapMaker application will extract this information and use it as the starting point for a new build and the nbproject which contains various property files. Finally the src contains important css and data files used within the application.

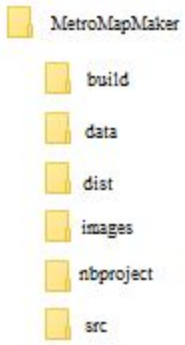

Figure 5.1: File Structure of MetroMapMaker

6 Table of Contents

1. Introduction
    1. Purpose
    2. Scope
    3. Definitions, acronyms, and abbreviations
    4. References
    5. Overview
2. Package-Level Design Viewpoint
    1. Metro Map Maker Overview
    2. Java API Usage
    3. Java API Usage Descriptions
3. Class-Level Design Viewpoint
4. Method-Level Design Viewpoint
5. File Structure and Formats
6. Table of Contents