The *Regio Vinco*™

The Map Editor

Software Design Description

Author: Shen Shao
June, 2016
Version 1.0

Abstract: This document describes the software design for *Regio Vinco* Map Editor, a editor helps constructing maps for the game *Regio Vinco*.

Based on IEEE Std 1016TM-2009 document format

1 Introduction

This is the Software Design Description (SDD) for the Map Editor application of the game *Regio Vinco*. The software uses Simple App Framework to perform basic file operations. Note that this document format is based on the IEEE Standard 1016-2009 recommendation for software design.

1.1 Purpose

This document is to serve as the blueprint for the construction of the *Regio Vinco* Map Editor 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

For this project the goal is to let a map designer easily make maps that look precisely the way they want. This design contains design descriptions for the use of Simple App Framework and the development of Map Editor. Note that Java is the target language for this software design.

1.3 Definitions, acronyms, and abbreviations

Regio Vinco – The *Regio Vinco* game is a game for learning about the world and its geography. The game will let the user select the map of their choice to play and will keep track of player accomplishments for all provided maps.

Map – A map of the game *Regio Vinco* could be a country, a continent, or a world map. It contains following information: the geography of the region, the border, the capitals, flags, leaders and anthem of that region.

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.

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

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

Use Case Diagram – A UML document format that specifies how a user will interact with a system. Note that these diagrams do not include technical details. Instead, they are fed as input into the design stage (stage after this one) where the appropriate software designs are constructed based in part on the Use Cases specified in the SRS.

1.4 References

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

***Regio Vinco*™ Map Editor SRS** – Software Requirements Specification for the Rigio Vinco game application.

1.5 Overview

This Software Design Description document provides a working design for the *Regio Vinco* Map Editor software application as described in the *Regio Vinco* Map Editor 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

In building this application 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 Map Editor Overview

The Map Editor will be designed and developed in tandem. Figure 2.1 specifies all the components to be developed.

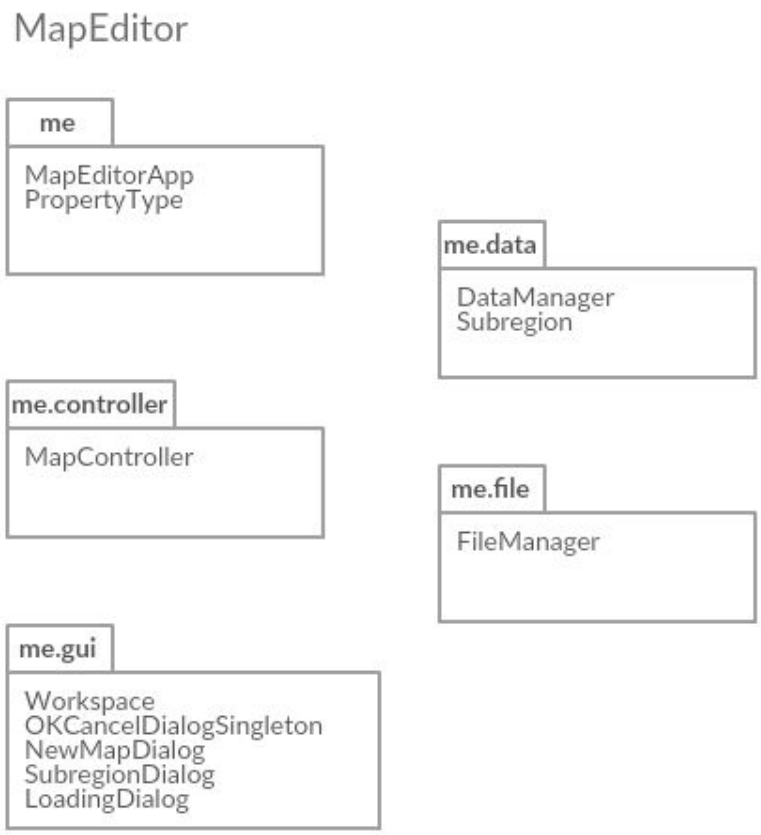


Figure 2.1: Design Packages Overview

2.2 Java API Usage

The Map Editor will be developed using the Java programming languages. As such, this design will make use of the classes specified in Figure 2.2.

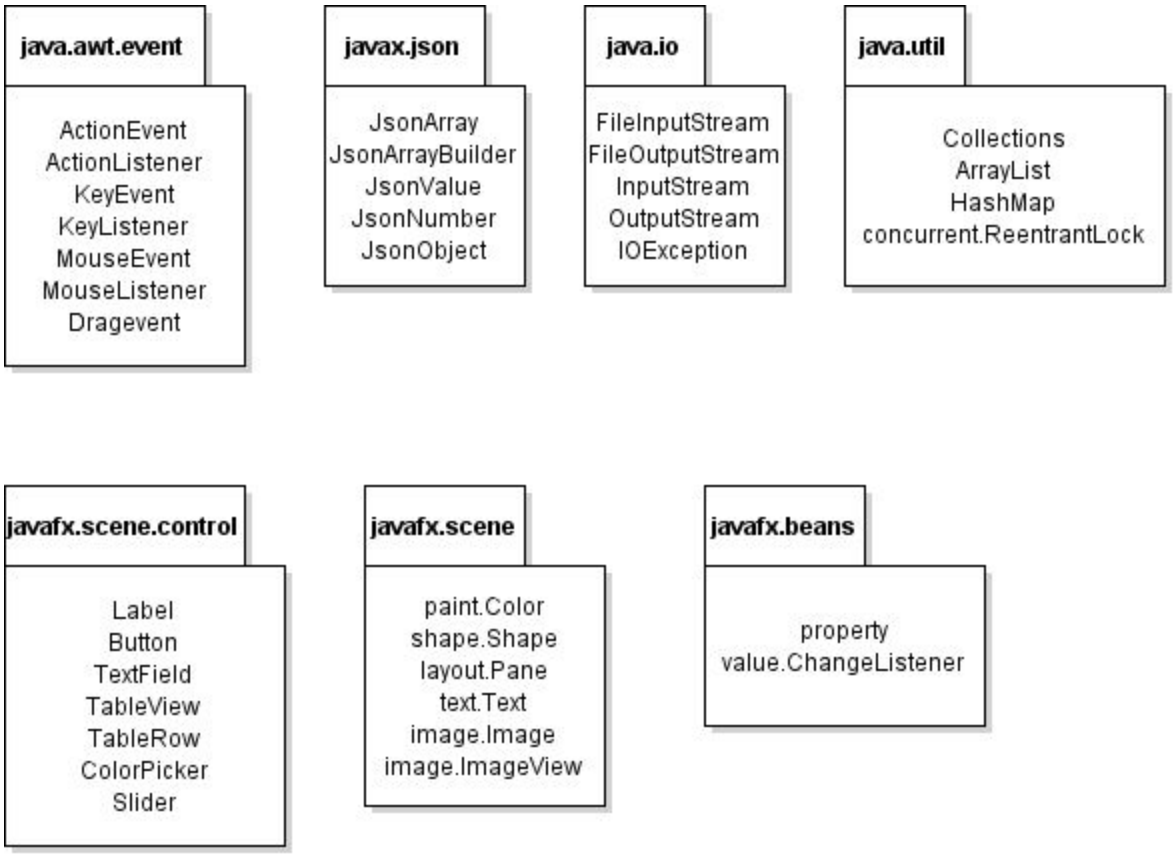


Figure 2.2: Java API Packages and Classes To Be Used

2.3 Java API Usage Descriptions

Java.awt.event: For getting information about an action event, then respond to an event, which own custom implementation of listener interface will be provided.
Javax.json: For saving and reading Map and Subregion data. Including map coordinate data, region name and other details.
Java.io: For inputting/outputting data to files.
Java.util.Collections: For listing objects like Subregions and Images.

Java.util.concurrent.ReentrantLock: For locking the thread of progress bar while loading large coordinate data.
Javafx.scene: For setting up and showing the application GUI.
Javafx.beans: For reading and writing to application GUI nodes and observe its change to unable/disable related functions.

2.4 Other Framework Usage

This design will make use of SimpleAppFramework, PropertyManager and XMLUtilities.

SimpleAppFramework: For building basic function and GUI of application: New, Save, Load, Export and Exit.
PropertyManager: For loading and then accessing application properties for our application.
XMLUtilities: For extracting XML file elements and its attributes, to get the data needed.

3 Class Level Design Viewpoint

The following UML class diagrams reflects the design of Map Editor. Due to the complexity of the project, the design will be presented using a series of diagrams going from overview diagrams down to detailed ones.

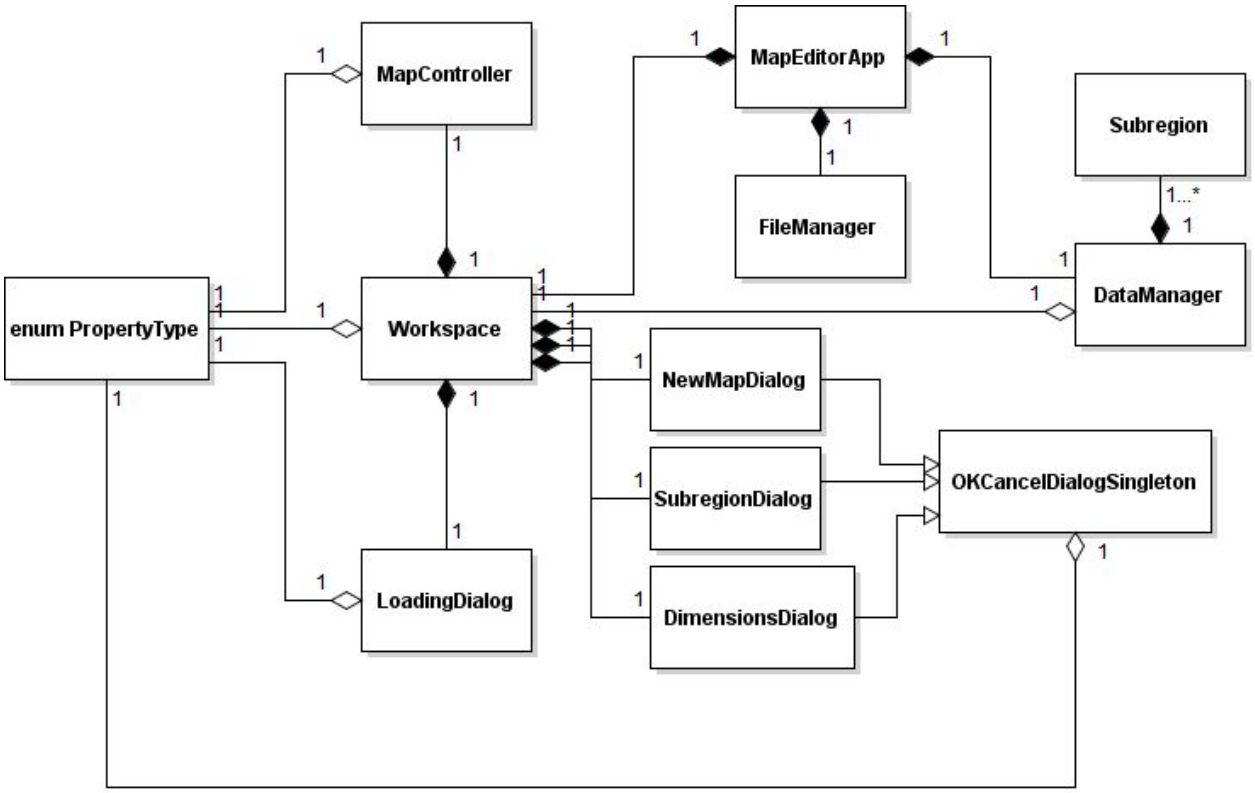


Figure 3.1: Map Editor Application Overview UML Class Diagram

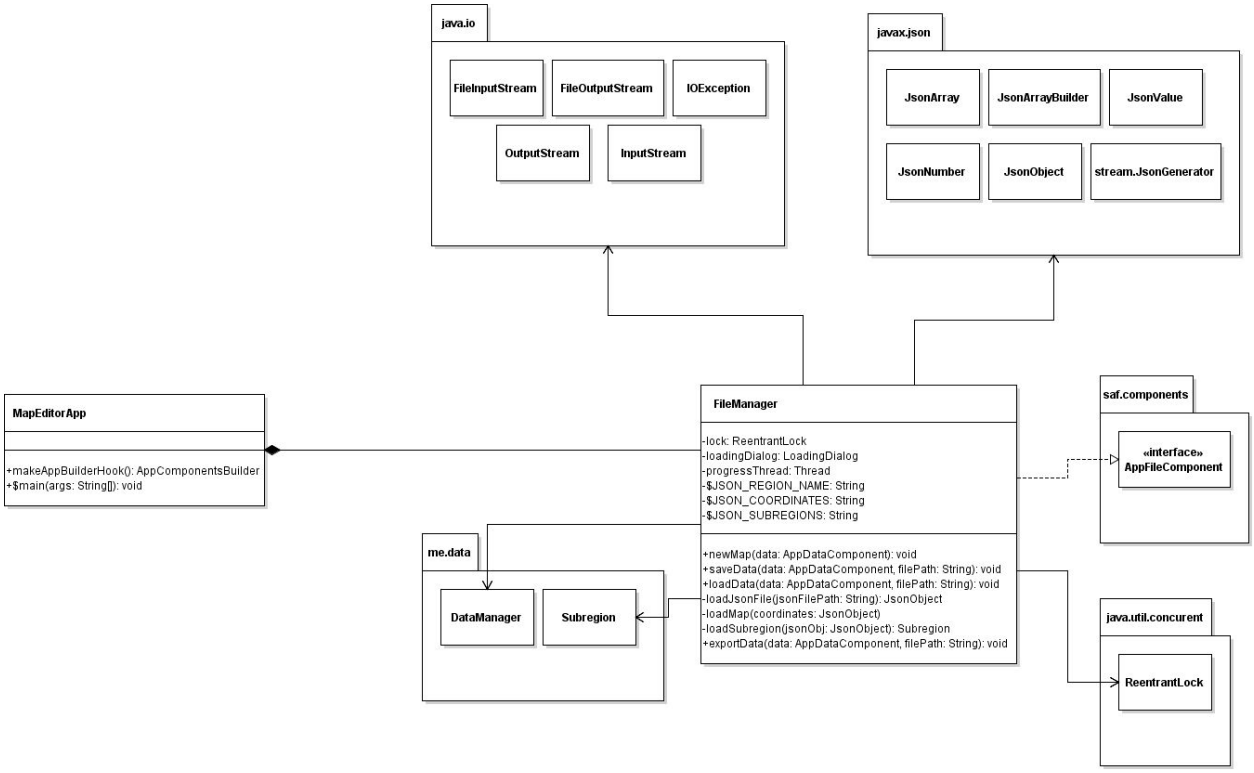


Figure 3.2: Detailed MapEditorApp and FileManager UML Class Diagram

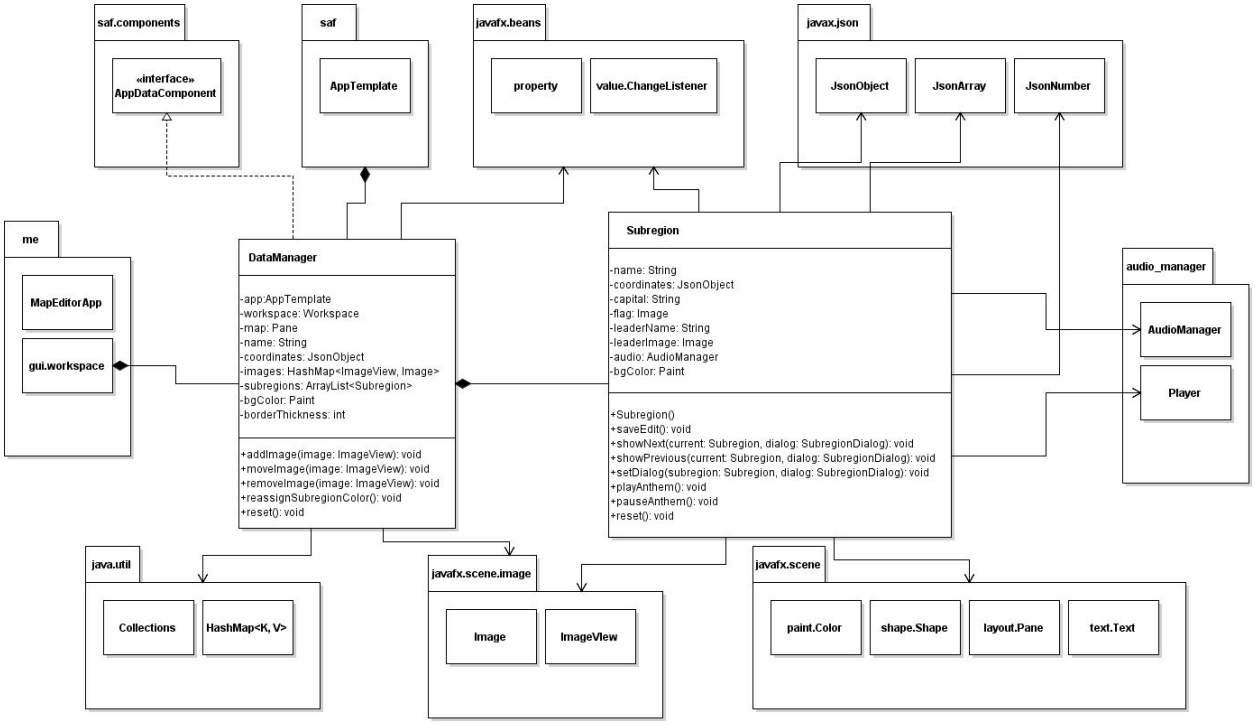


Figure 3.3: Detailed DataManager and Subregion UML Class Diagram

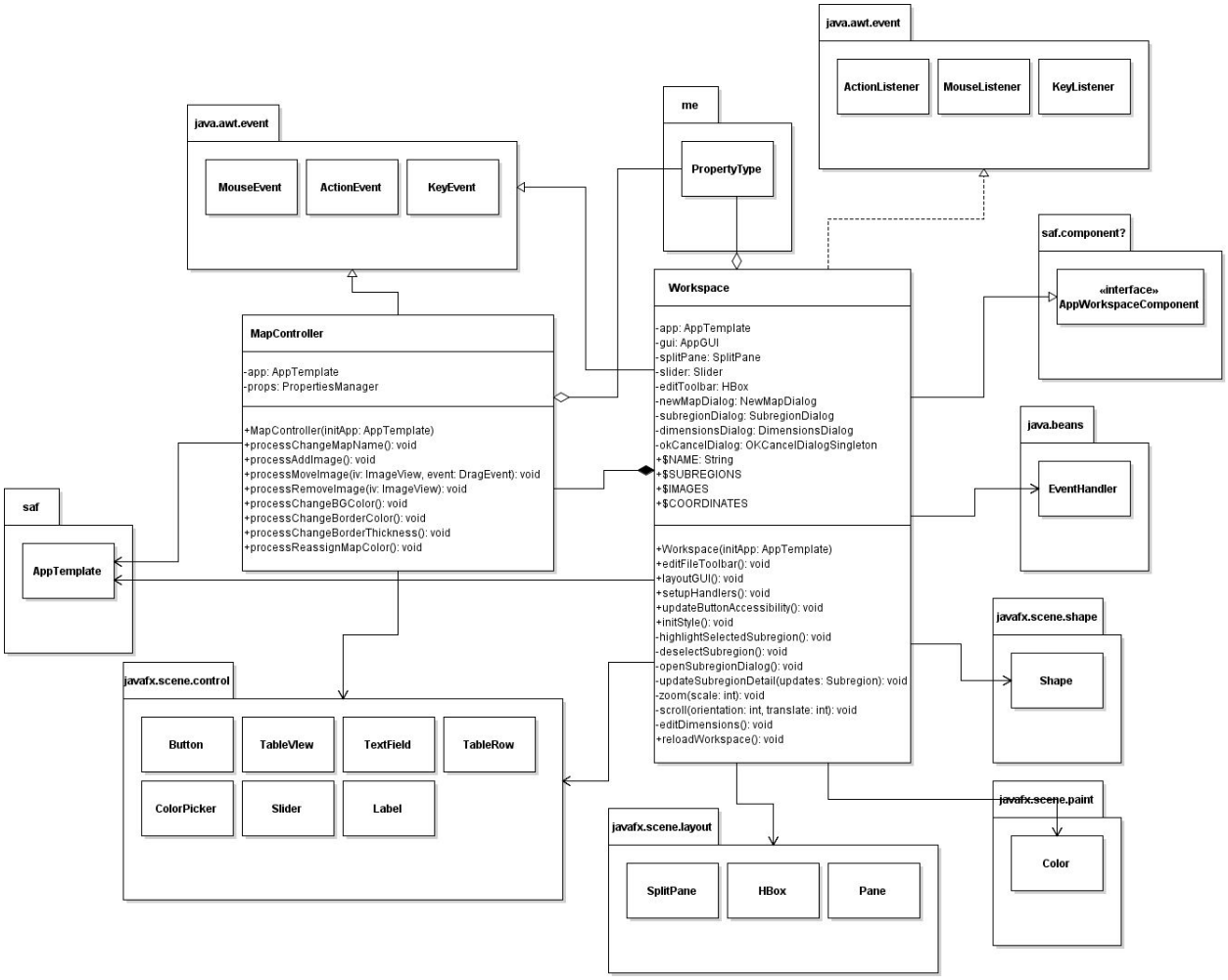


Figure 3.4: Detailed Workspace and MapComtroller UML Class Diagram

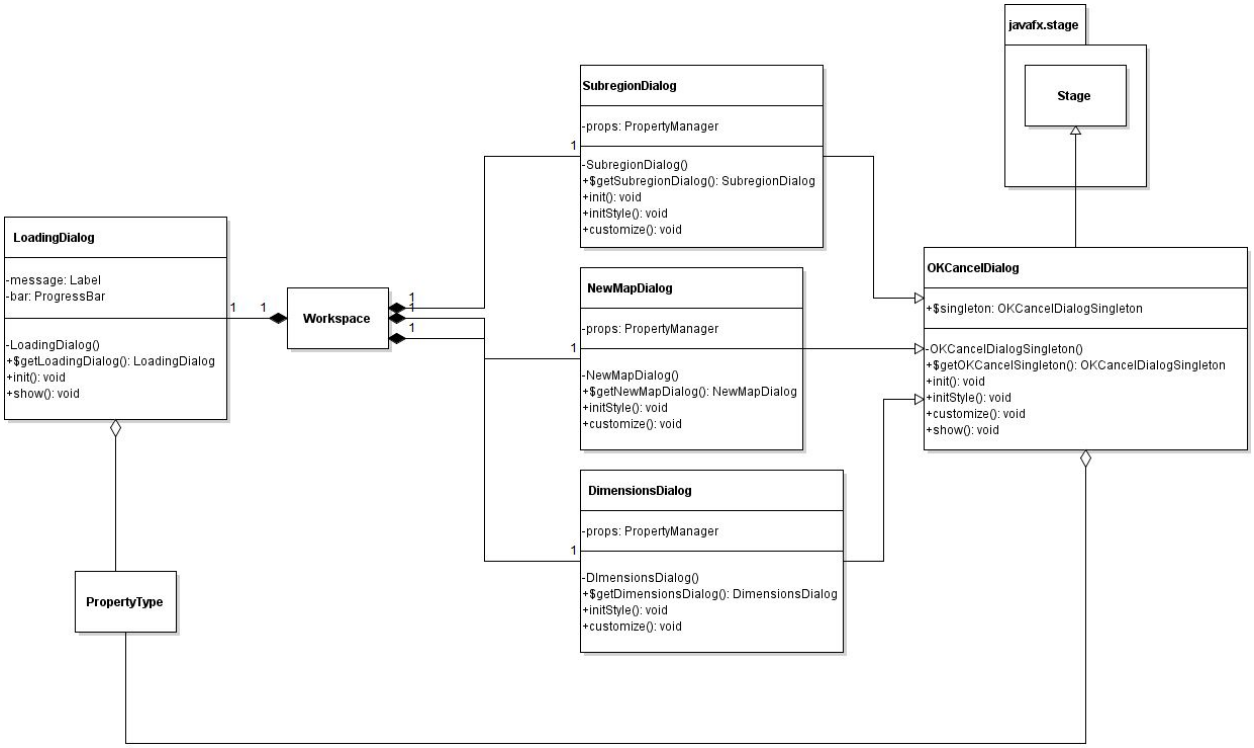
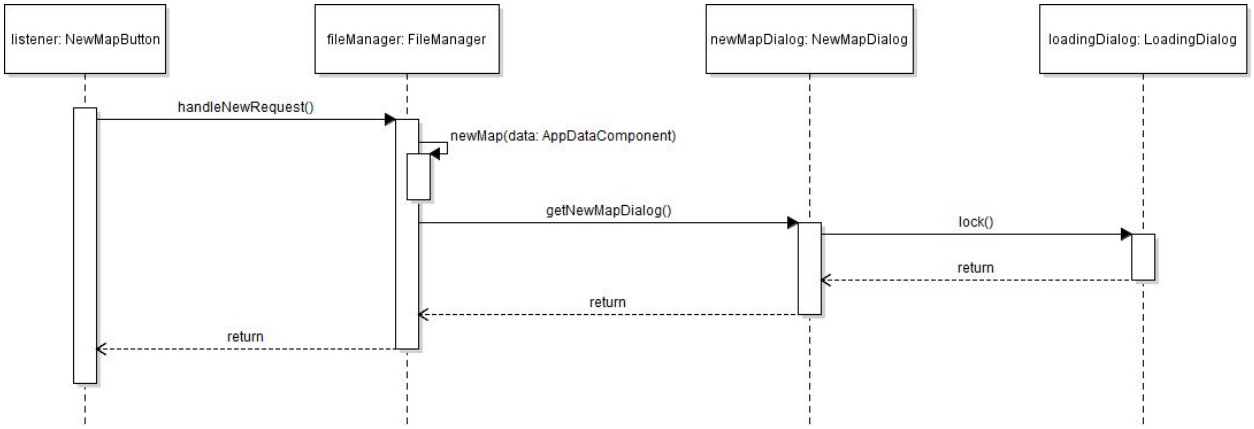


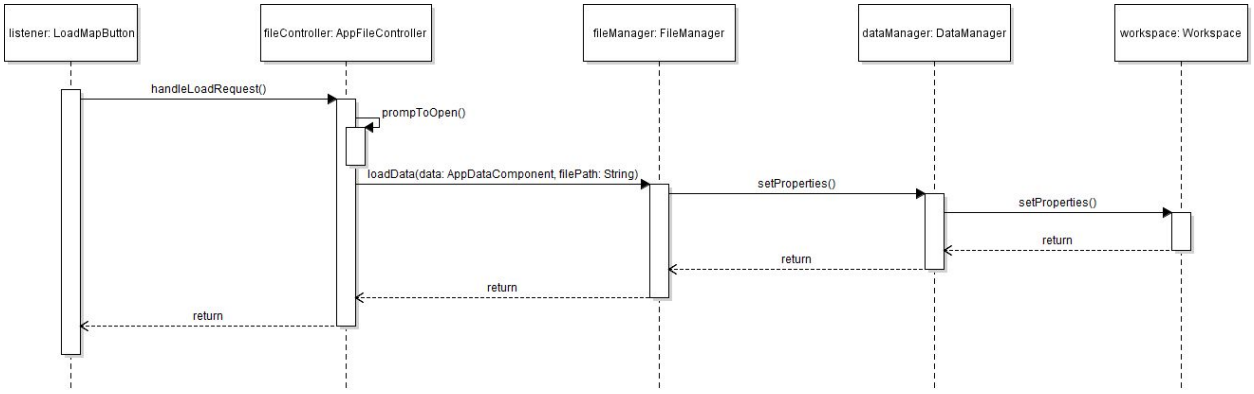
Figure 3.5: Detailed NewMapDialog, SubRegionDialog, LoadingDialog and OKCancelDialog UML Class Diagram

4 Method-Level Design Viewpoint

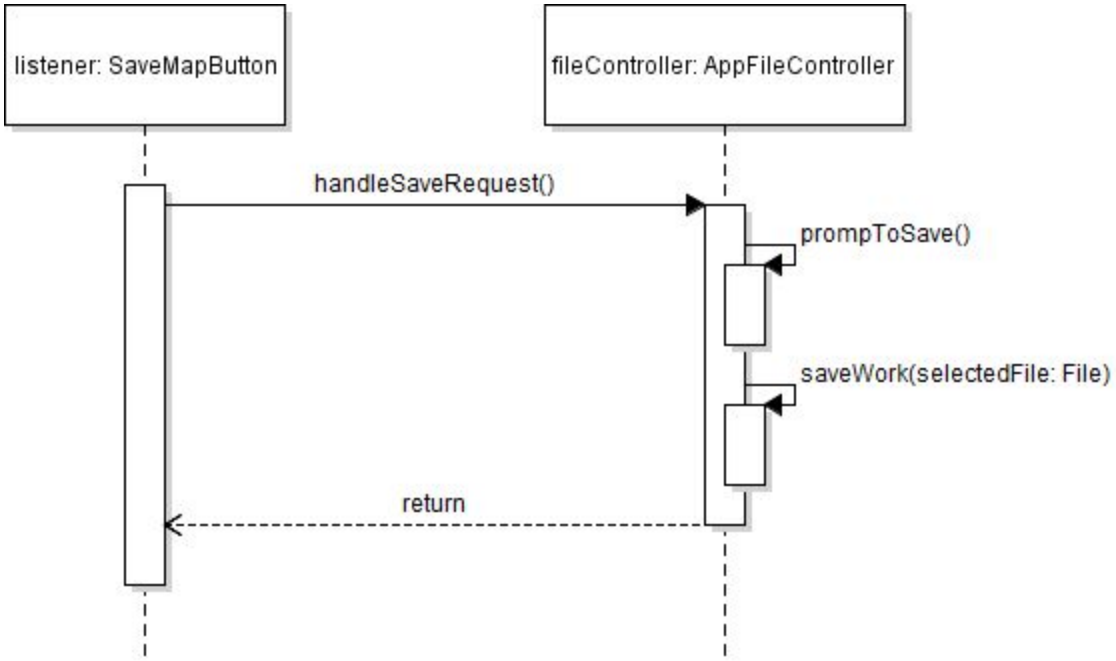
The following UML Sequence Diagrams describe the methods called within the code to be developed in order to provide the appropriate event responses.



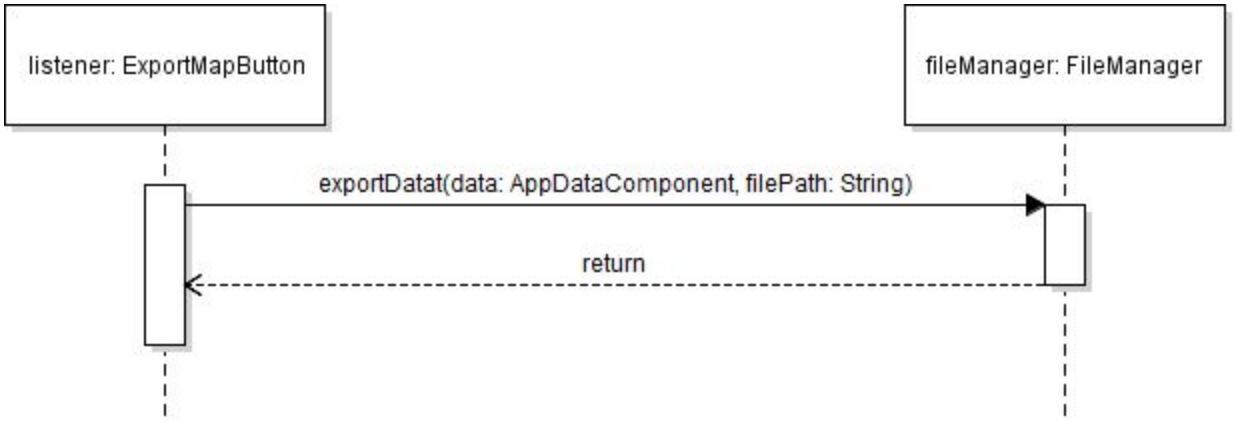
4.1 New Map UML Sequence Diagrams



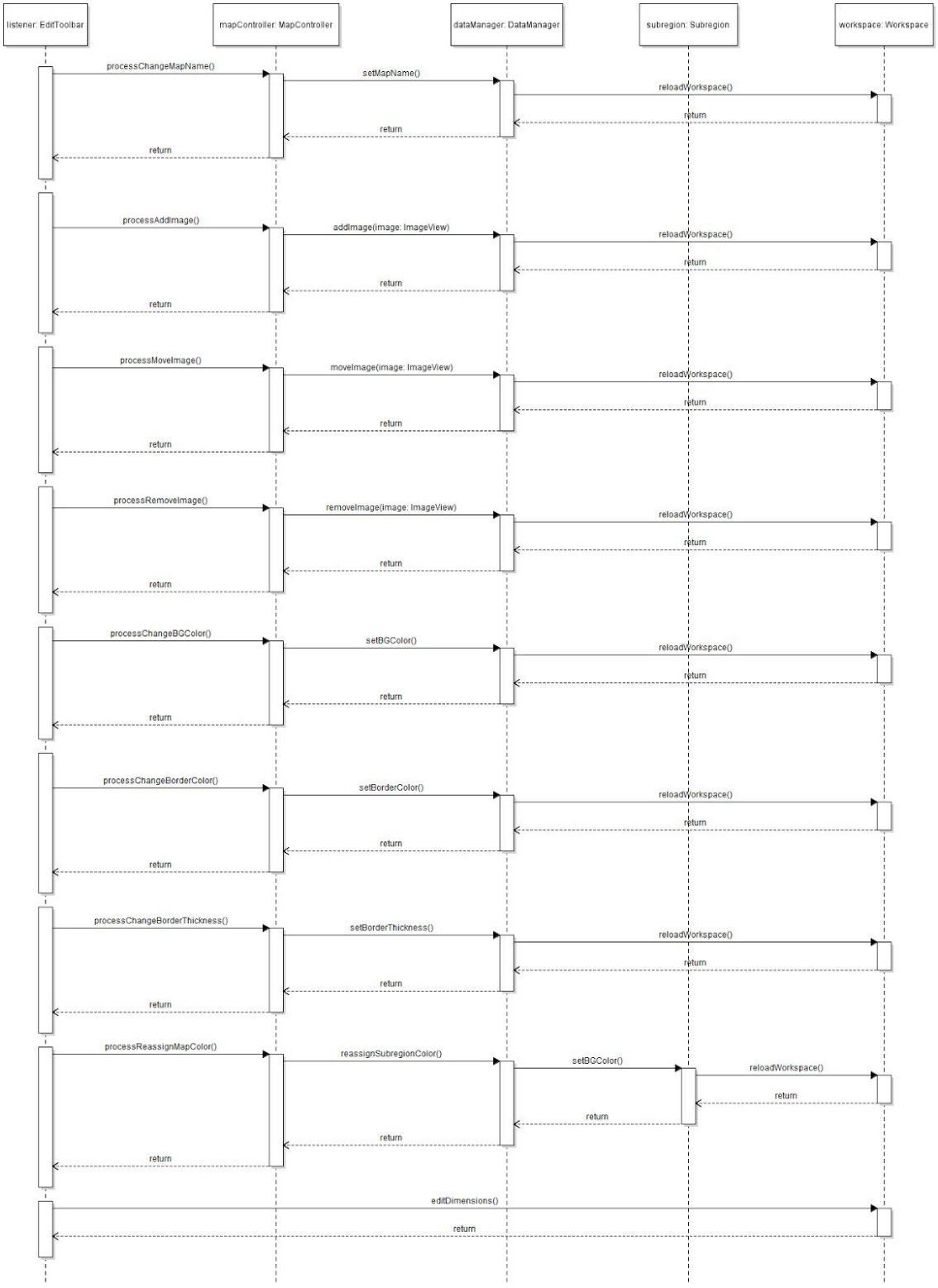
4.2 Load Map UML Sequence Diagrams



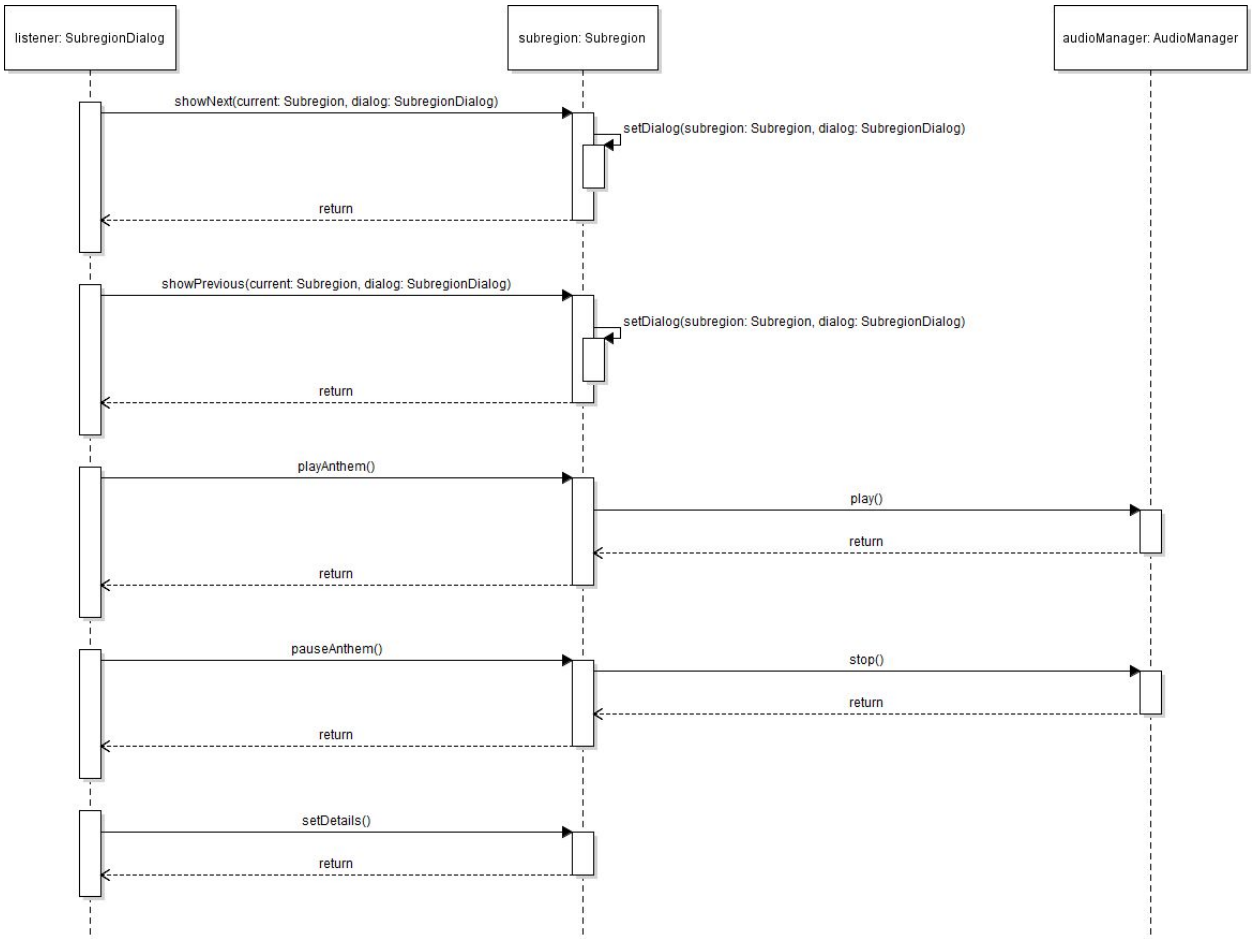
4.3 Save Map UML Sequence Diagrams



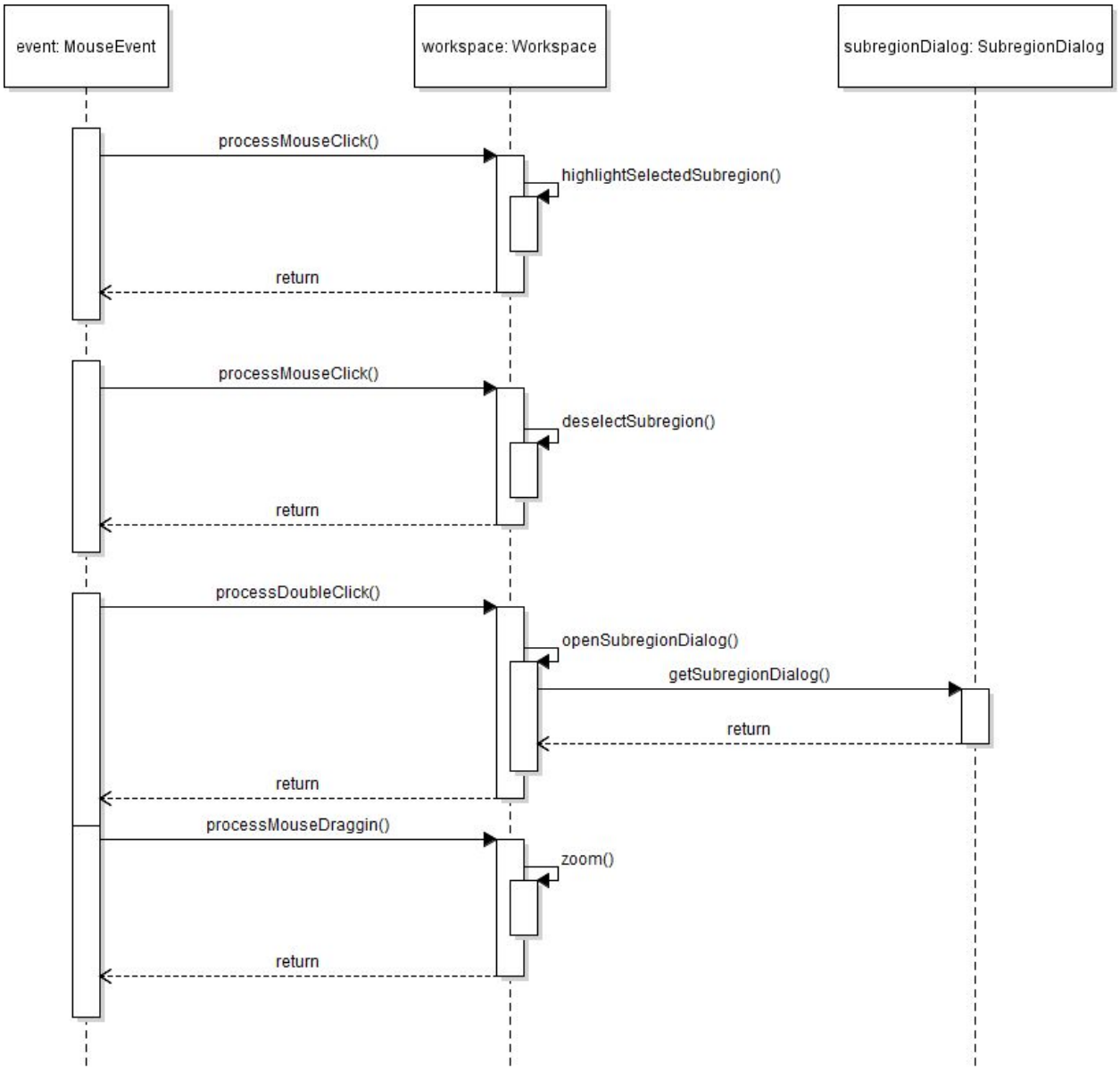
4.4 Export Map UML Sequence Diagrams



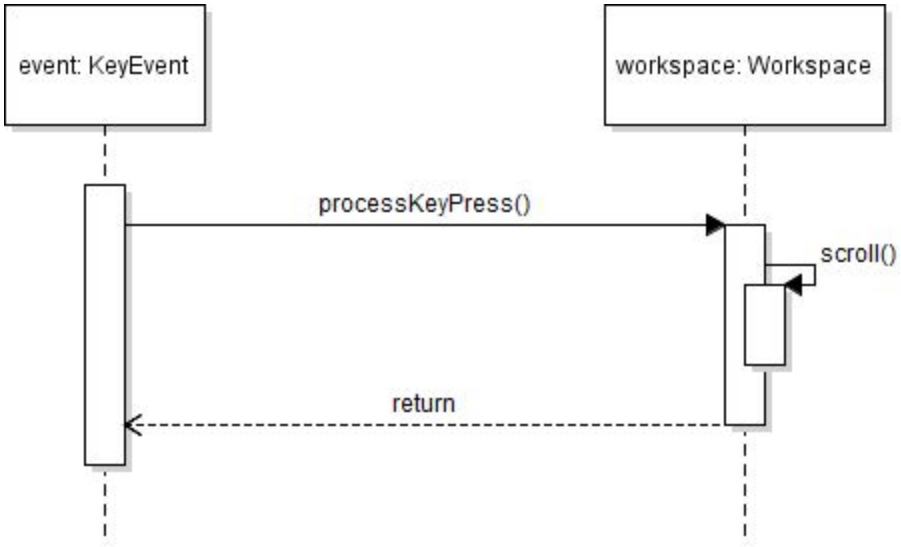
4.5 Edit Toolbar Functions UML Sequence Diagrams



4.6 Subregion Dialog Function UML Sequence Diagrams



4.7 Mouse Events UML Sequence Diagrams



4.8 Key Events UML Sequence Diagrams

5 File Structure and Formats

Below specifies the necessary file structure the launched application should use. Note that all necessary audios should go in the audio directory and images should go in the image directory. The <RegionName>.json contains the map coordinates and name, leader name, etc.

-MapEditor
    -<RegionName>
        -audio
        -image
        -<RegionName>.json
    -MapEditor.jar

The me_properties.xml provides the file and state names for all properties in the application. The file is a xml file that can be read by framework XMLUtilities and PropertyManager. The setting of the properties are shown below:

Property Names:

X_BUTTON: The label of the button
X_<NODE>: Label for certain kinds of nodes
X_ICON: The image file name of the icon
X_LABEL: The text of the label
X_HEADING: The label of a node
X_TITLE: The title of the dialog box
X_MESSAGE: The message in the dialog box

6 Supporting Information

Note that this document should serve as a reference for those implementing the code, so we'll provide a table of contents to help quickly find important sections.

6.1 Table of contents

6.2 Appendixes

N/A