BuzzWord™
Software Design Description

Author: Po Yiu Ho (109723660)

McKilla Gorilla Inc.™

October, 2016

Version 1.0

Abstract: This is a document illustrating the software design for BuzzWord™, an educational word game similar to “Boggle”.

Based on IEEE Std 1016™-2009 document format

1 Introduction

This is the Software Design Description (SDD) for the BuzzWord™ game application. Note that this document format was written based on the IEEE Standard 1016™-2009 model for software design.

1.1 Purpose

This document fulfills the purpose of an architectural map to building the BuzzWord™ game. It will be planned out using UML class and sequence diagrams, which will provide details concerning all packages, classes, instance variables, class variables, and methods necessary for the construction of this project as well as respective object interactions in response to user input or timed events.

1.2 Scope

BuzzWord™ is a simple gaming application that aims for both fun and education. To build this game, a framework called the WordGame Framework will be used to help facilitate the construction of the project design. Note that Java is the target language for this software design.

1.3 Definitions, acronyms, and abbreviations

BuzzWord – The title of the game application being designed by this Software Design Description.

Class Diagram – A UML document format that describes classes graphically to describe their variables, methods, and relationships with other classes.

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

Educational Games – Games that assist people in learning about specific concepts through different levels that they can play

Framework – A collection of classes, interfaces, and enums that provide the building blocks for exclusive applications.

GUI – Graphical User Interface, or a program's visual indicators that both serve output and request for user input at the same time.

Java – A high-level programming language built for OS portability that uses a virtual machine to interpret and run compiled instructions.

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

Sequence Diagram – UML document format that specifies the flow of data as object methods interact with one another.

Use Case Diagram – UML document format that describes how certain inputs would affect the entire system as a whole.

Word Game Framework – The framework to be developed with the BuzzWord™ game application so that other word games can be easily constructed.

Word Games – Applications that involve fun activities incorporating the making, guessing, or selecting of words or letters.

1.4 References

IEEE Std 830™-1998 – IEEE Standard for Information Technology – Systems Design – Software Design Descriptions

BuzzWord™ SRS – Software Requirements Specification for the BuzzWord™ game application

1.5 Overview

This Software Design Description document provides a blueprint design for the BuzzWord™ game application as described in the BuzzWord™ Software Requirements Specification. Section 2 of this document will display the Package-Level Viewpoint, which specifies the packages and frameworks that the application will use. Section 3 will provide the Class-Level Viewpoint, which will use UML Class Diagrams to portray how classes will be built and how they connect to each other. Section 4 will be the sector that shows the Method-Level System Viewpoint, which will use Sequence Diagrams to represent how methods interact with each other. Section 5 will have deployment information as well as file structures and formats to use. Section 6 provides a Table of Contents, an Index, and References. All UML Diagrams in this document were created using the VioletUML editor.

2 Package-Level Design Viewpoint

The BuzzWord™ game application will include both BuzzWord itself and the WordGame Framework in its construction. Both projects will copiously utilize Java API classes to build them.

2.1 BuzzWord and WordGame Overview

The BuzzWord program and WordGame framework will be designed and developed together so that they can work in unison. Figure 2.1 displays every component and class to build in these two big source packages.

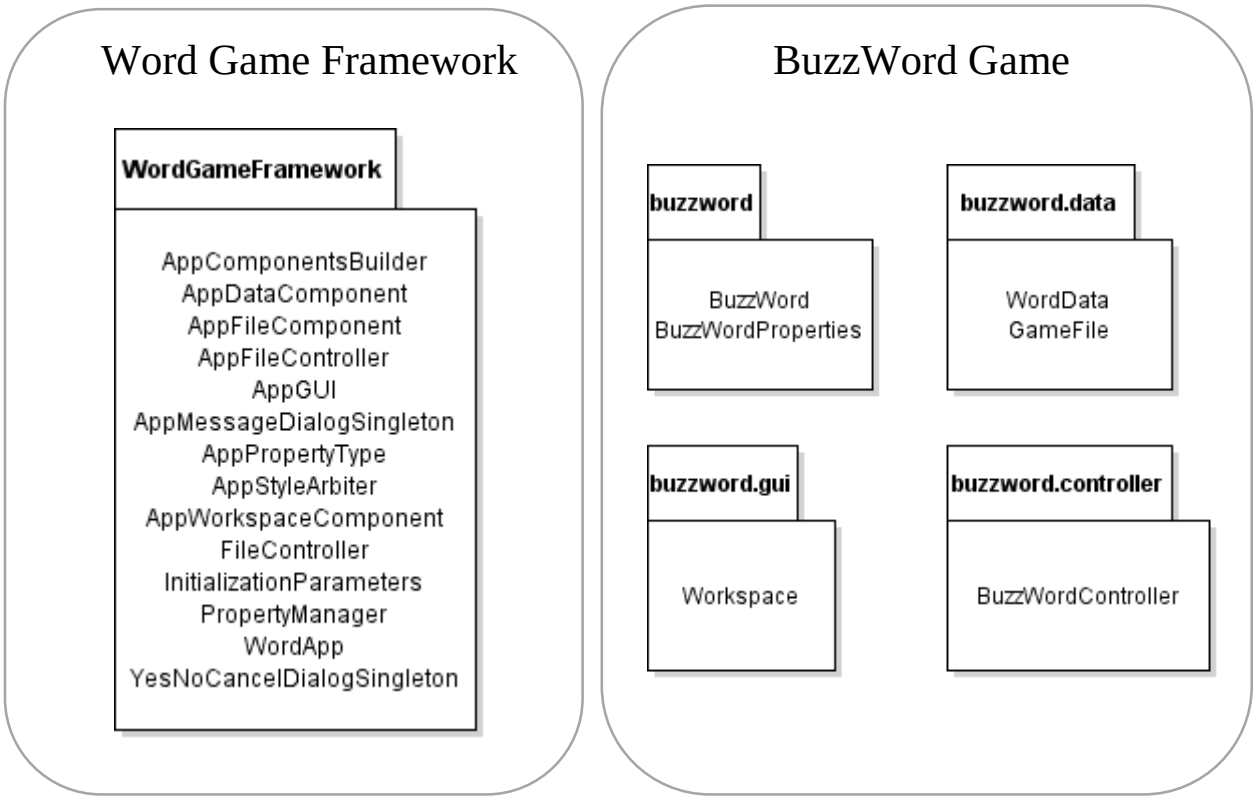

Figure 2.1: Design Packages Overview

2.2 Java API Usage

The BuzzWord application as well as the word game framework would both be developed using Java Application Programming Interfaces (APIs), which will become apparent in Figure 2.2.

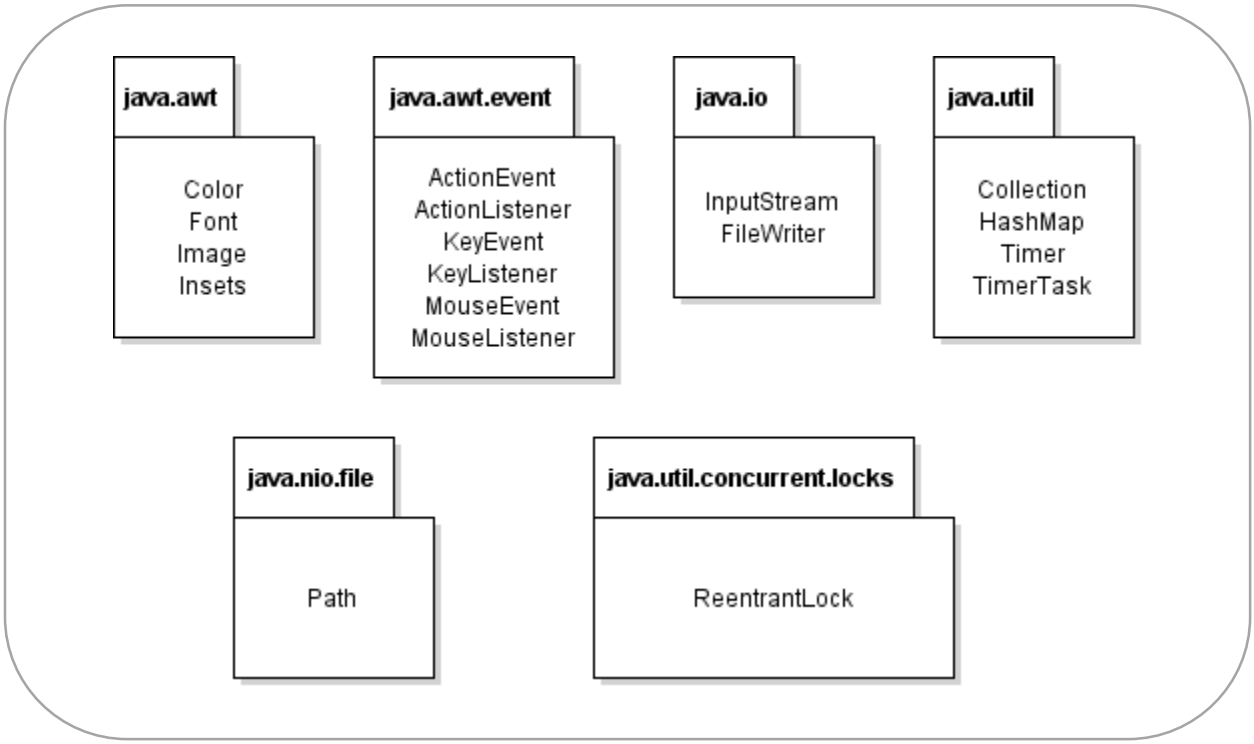

Figure 2.2: Java API Classes and Packages to be Used

2.3 Java API Usage Descriptions

Tables 2.1-2.7 below will describe how each Java API class/interface will be used.

Class/Interface	Use
Color	For setting the colors of GUI graphics and text
Font	For setting the fonts for application text
Image	For storing image data
Insets	For changing node margins

Table 2.1: Uses for classes in the Java API's java.awt package

Class/Interface	Use
ActionEvent	For getting information about events that nodes fire
ActionListener	For responding to action events; custom made
KeyEvent	For getting information about pressed key events
KeyListener	For responding to key events; custom made
MouseEvent	For getting information about mouse events
MouseListener	For responding to mouse events; custom made

Table 2.2: Uses for classes in the Java API's java.awt.event package

Class/Interface	Use
InputStream	For reading JSON files when loading data
FileWriter	For writing JSON files when saving data

Table 2.3: Uses for classes in the Java API's java.io package

Class/Interface	Use
Collection	For storing groups of data
HashMap	For storing key and value pairs (mostly for logins)
Timer	For executing certain tasks in fixed intervals (levels)
TimerTask	For the execution of certain tasks using Timer

Table 2.4: Uses for classes in the Java API's java.util package

Class/Interface	Use
Path	For storing data files (either saved games or profiles)

Table 2.5: Uses for classes in the Java API's java.nio.file package

Class/Interface	Use
ReentrantLock	For ensuring that only one thread has access to program data

Table 2.6: Uses for classes in the Java API's java.util.concurrent package

3 Class-Level Design Viewpoint

The design of this document will have both the BuzzWord game application and the WordGame Framework in its blueprints. The following UML Class Diagrams will specify the requirements of these two projects further. Because the project as a whole is complex, the class designs will proceed from least to most detailed.

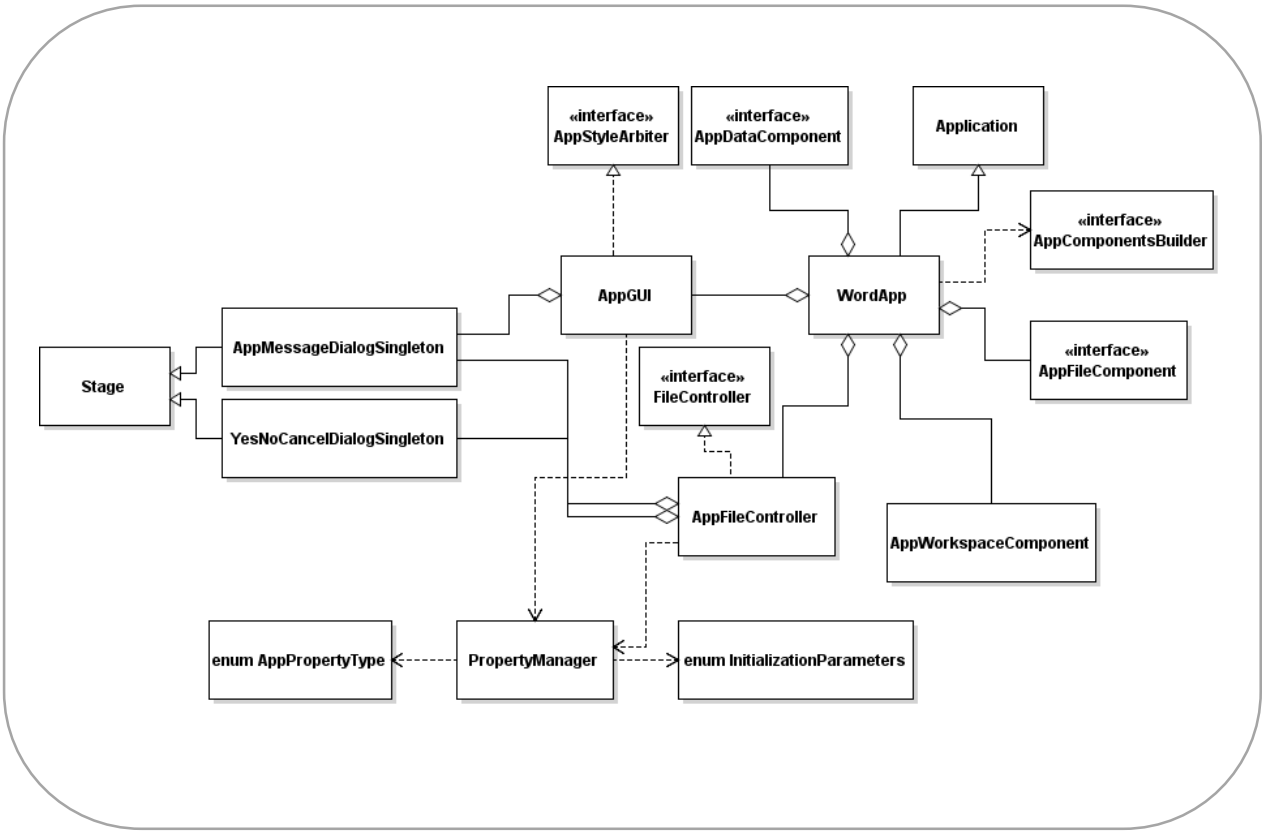

Figure 3.1: WordGame Framework Overview UML Class Diagram

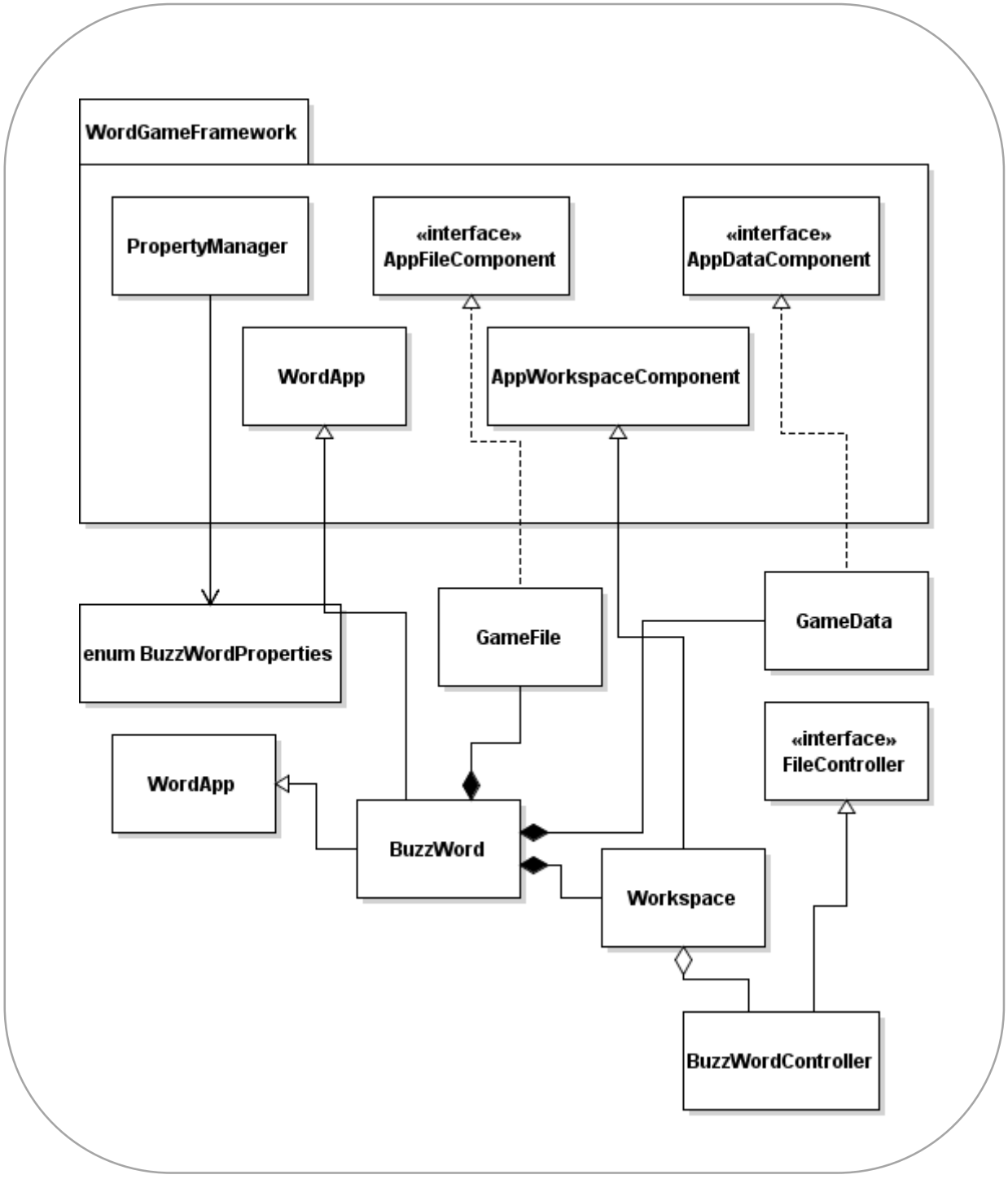

Figure 3.2: BuzzWord Overview UML Class Diagram

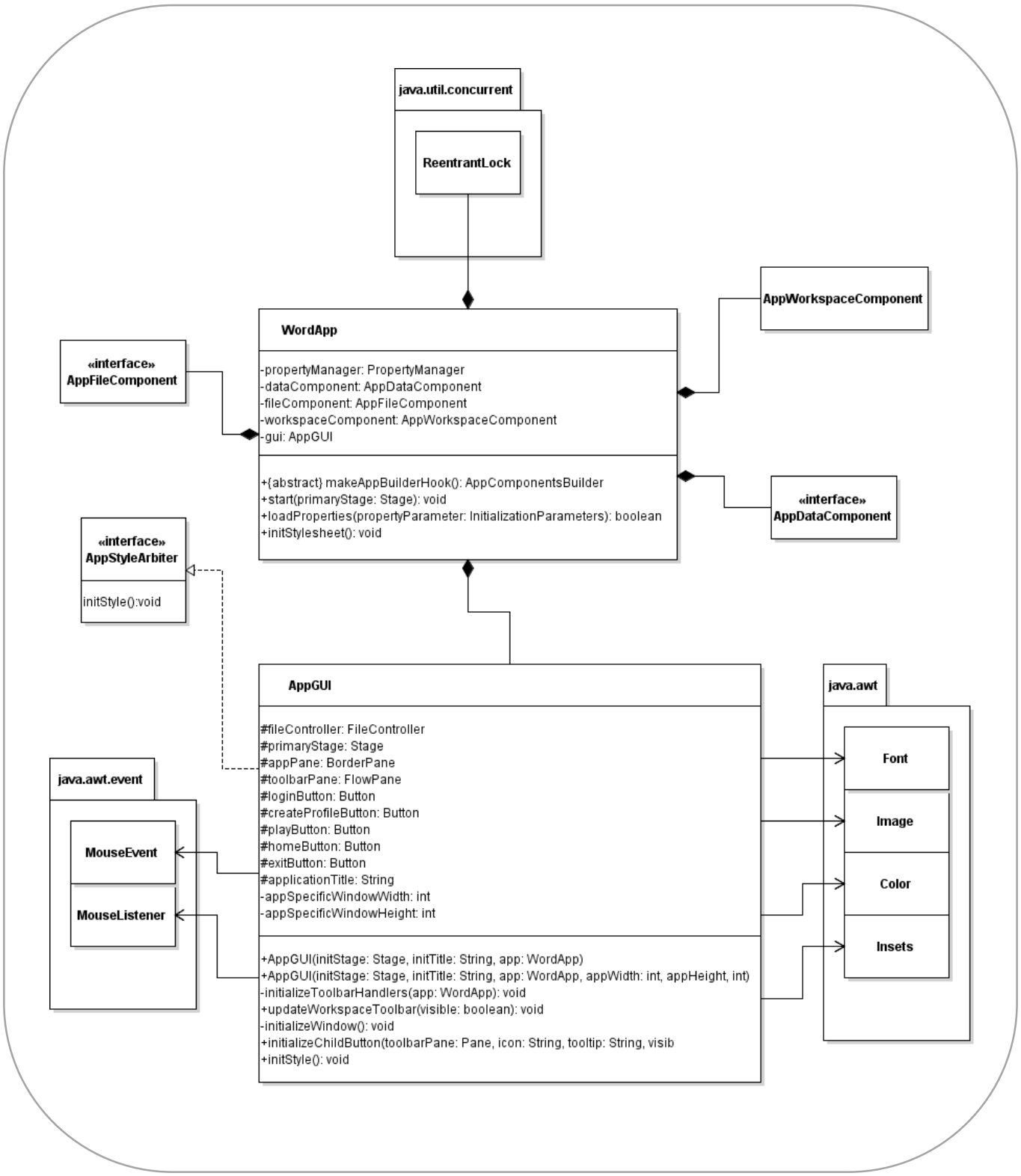

Figure 3.3: Detailed WordApp and AppGUI UML Class Diagrams

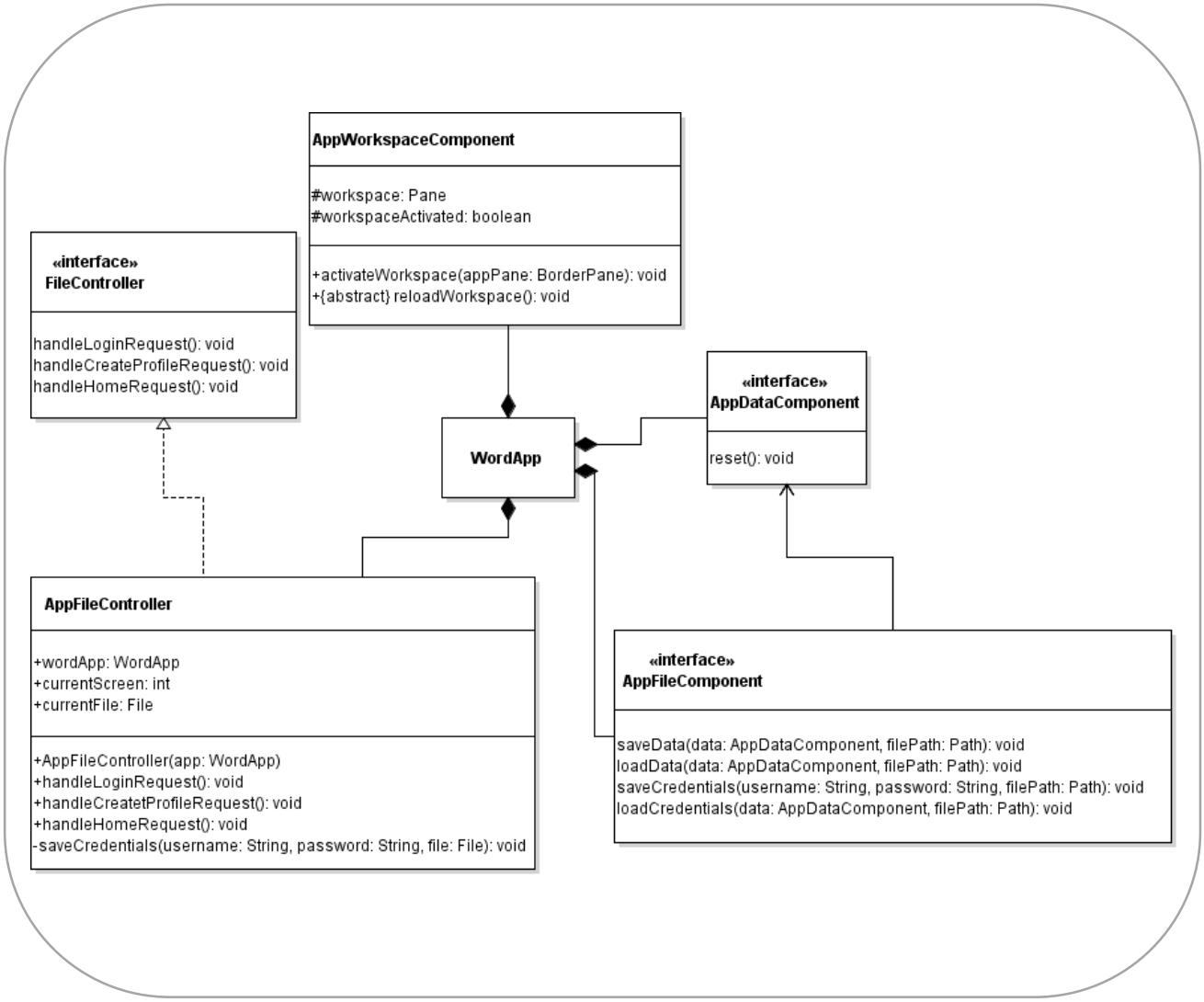

Figure 3.4: Detailed AppDataComponent, AppWorkspaceComponent, AppFileComponent, and AppFileController/FileController UML Class Diagrams

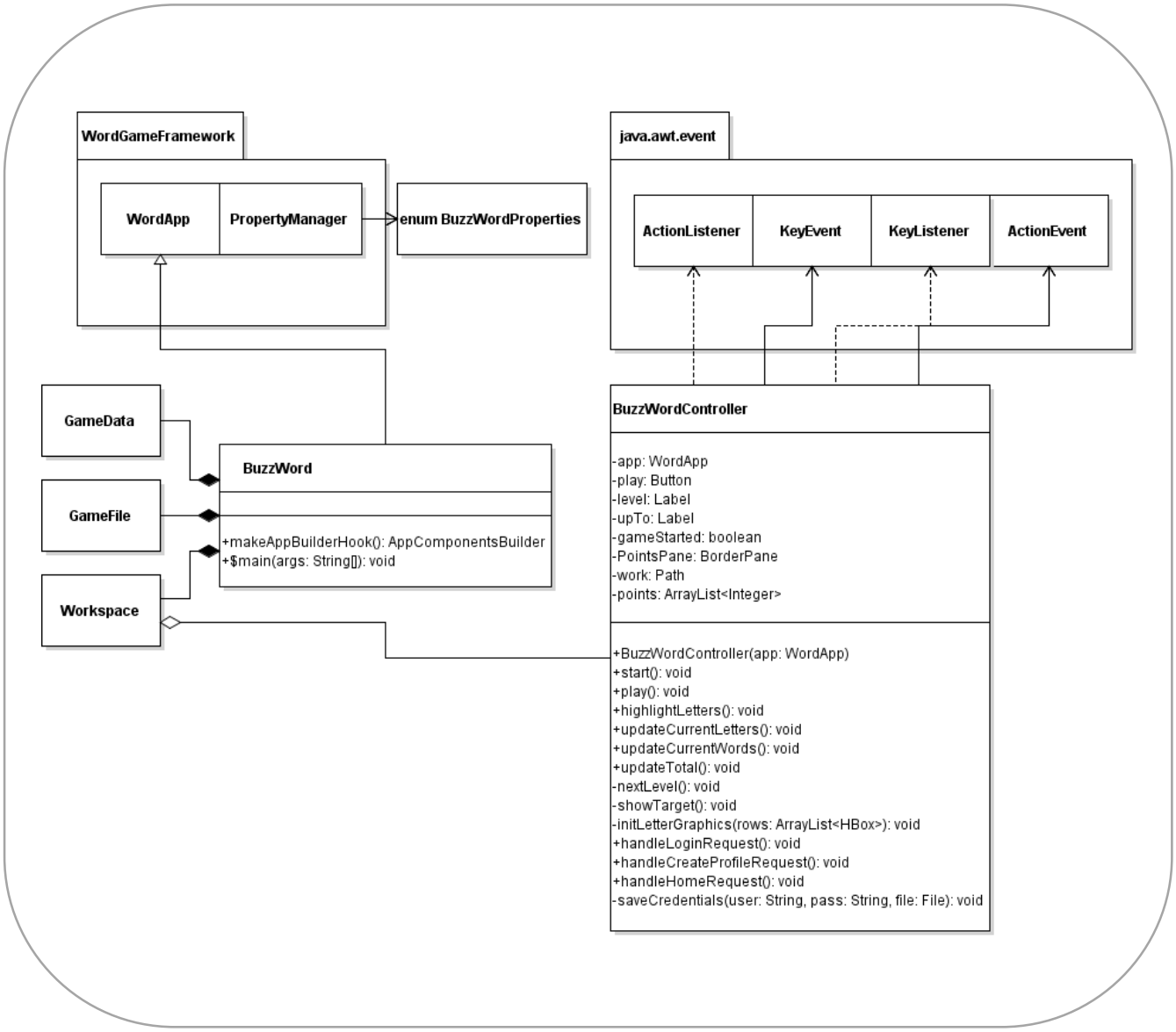

Figure 3.5: Detailed BuzzWord and BuzzWord Controller UML Class Diagrams

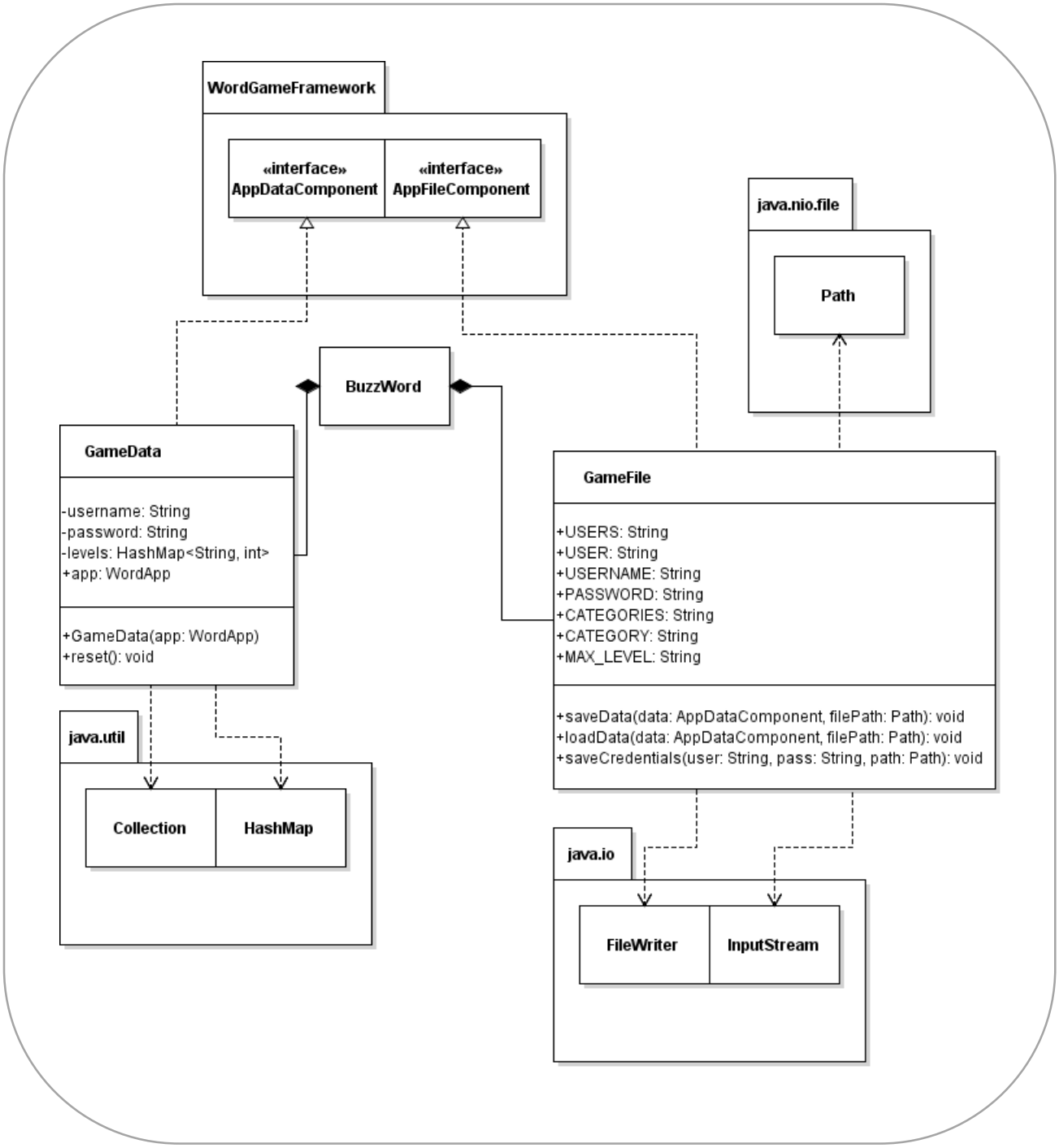

Figure 3.6: Detailed GameData and GameFile UML Class Diagrams

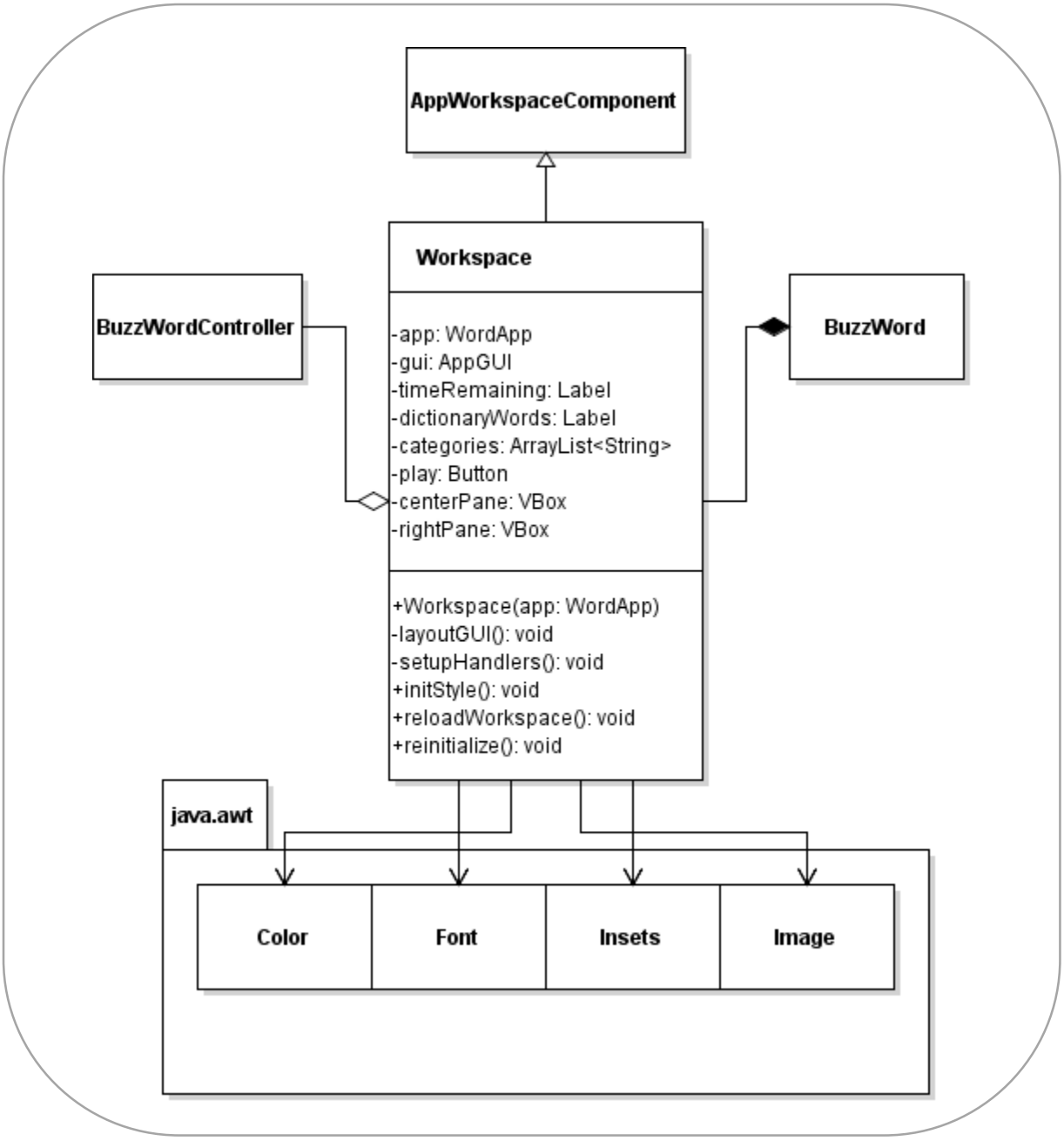

Figure 3.7: Detailed Workspace UML Class Diagram

4 Method-Level Design Viewpoint

The assigned classes and their relationships help determine how data will flow through the project as a whole. The following UML Sequence Diagrams will specify methods/events and their appropriate event responses.

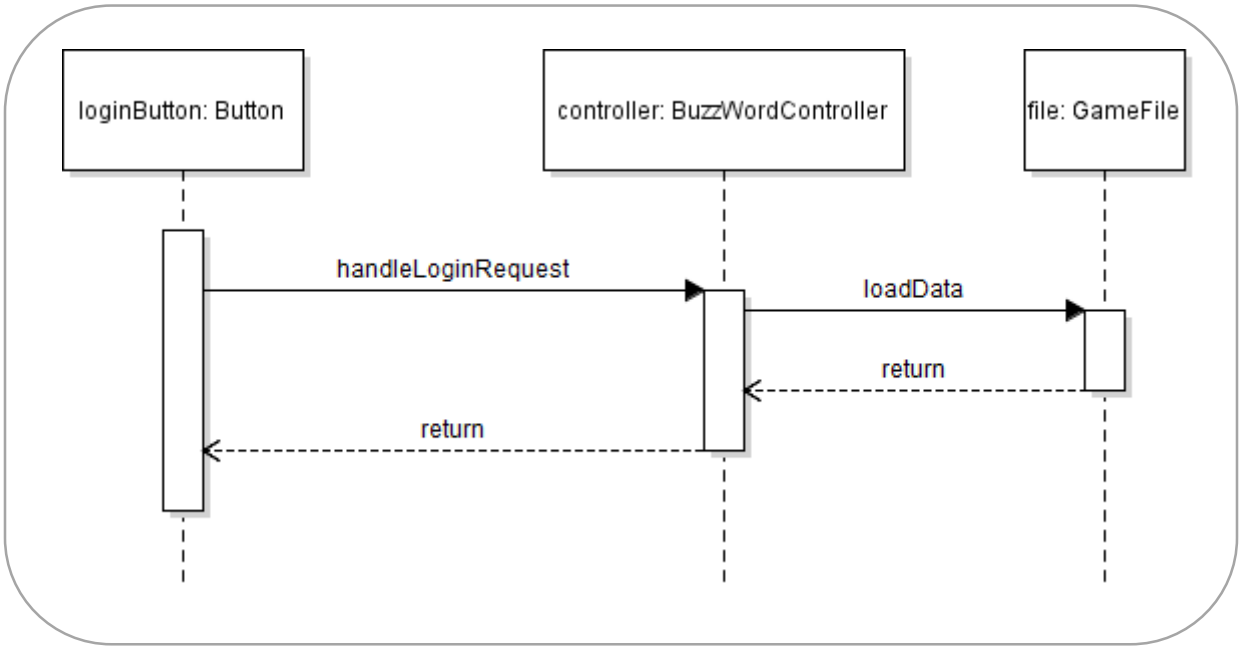

Figure 4.1: loginButton UML Sequence Diagram

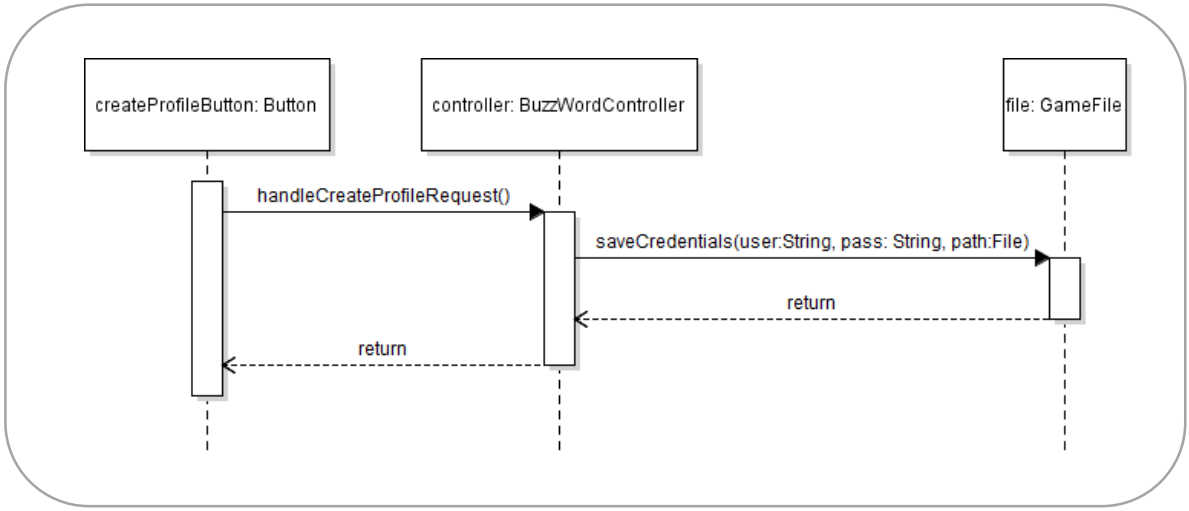

Figure 4.2: createProfileButton UML Sequence Diagram

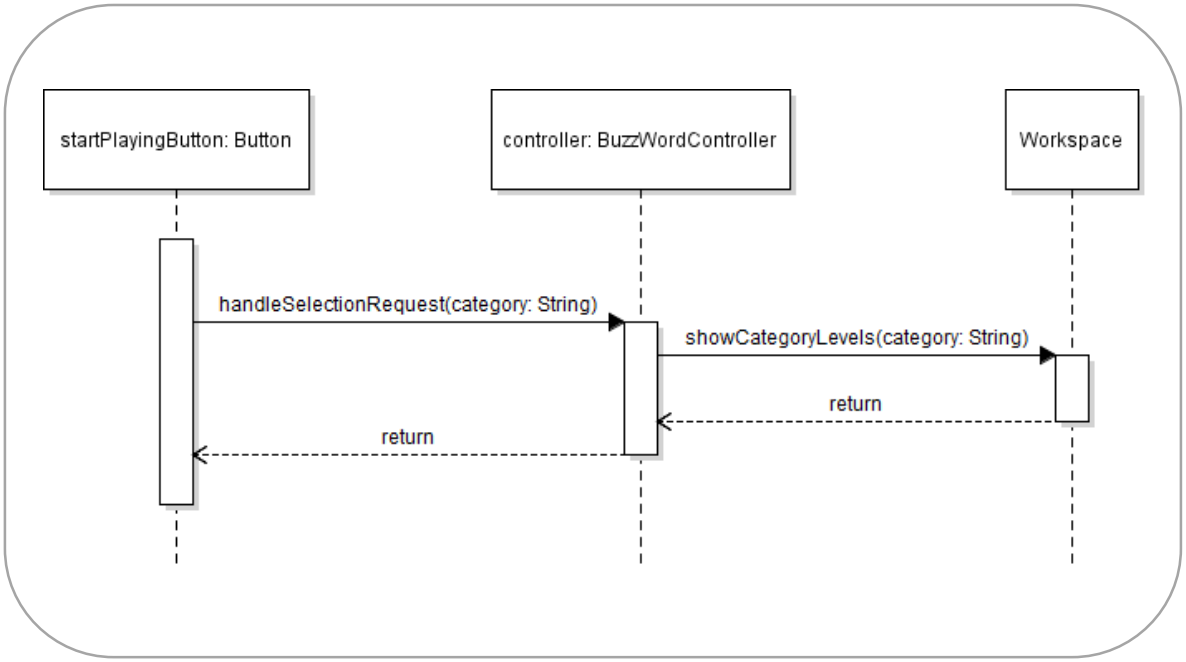

Figure 4.3: startPlayingButton UML Sequence Diagram

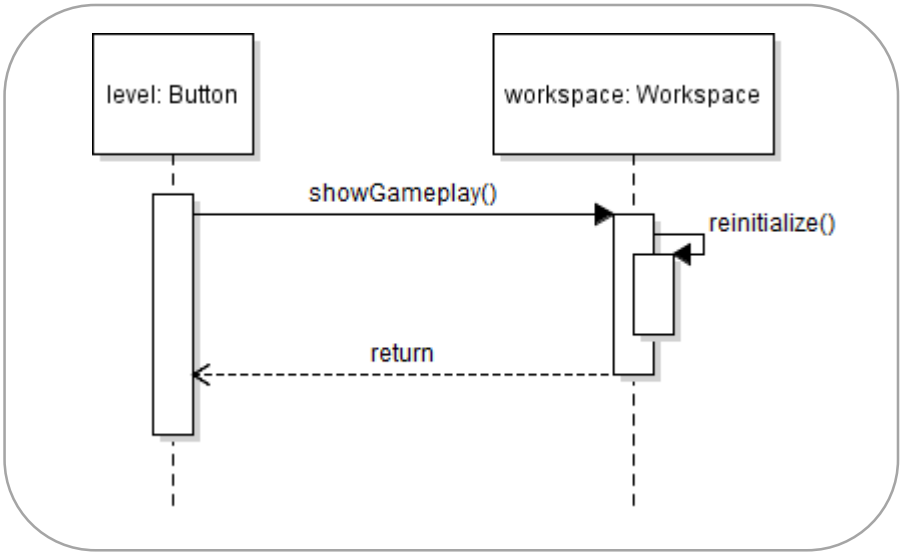

Figure 4.4: level buttons UML Sequence Diagram

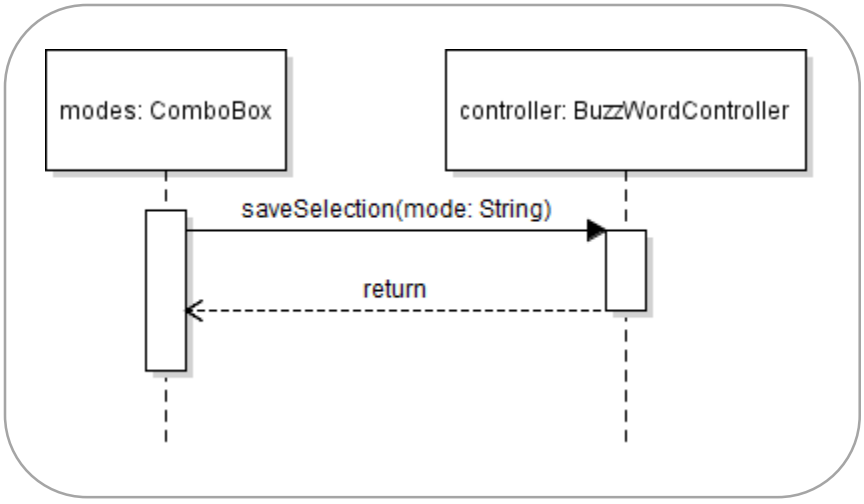

Figure 4.5: modes UML Sequence Diagram

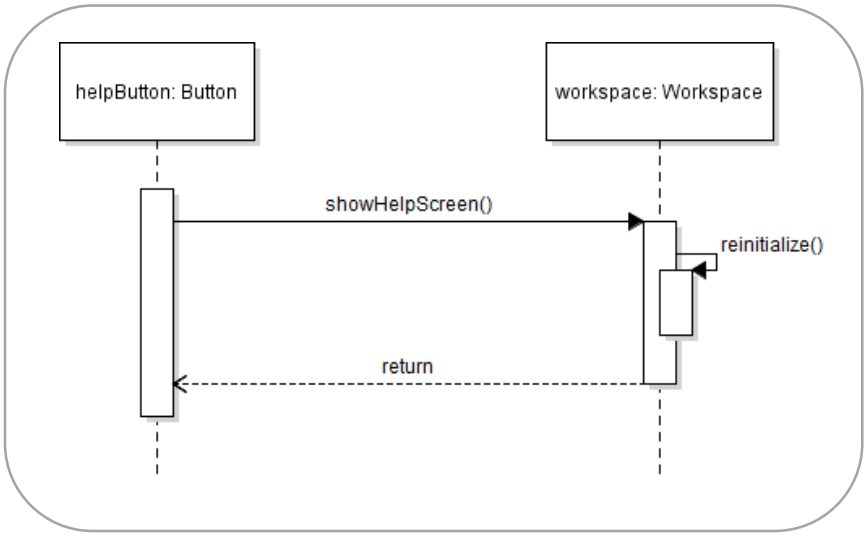

Figure 4.6: helpButton UML Sequence Diagram

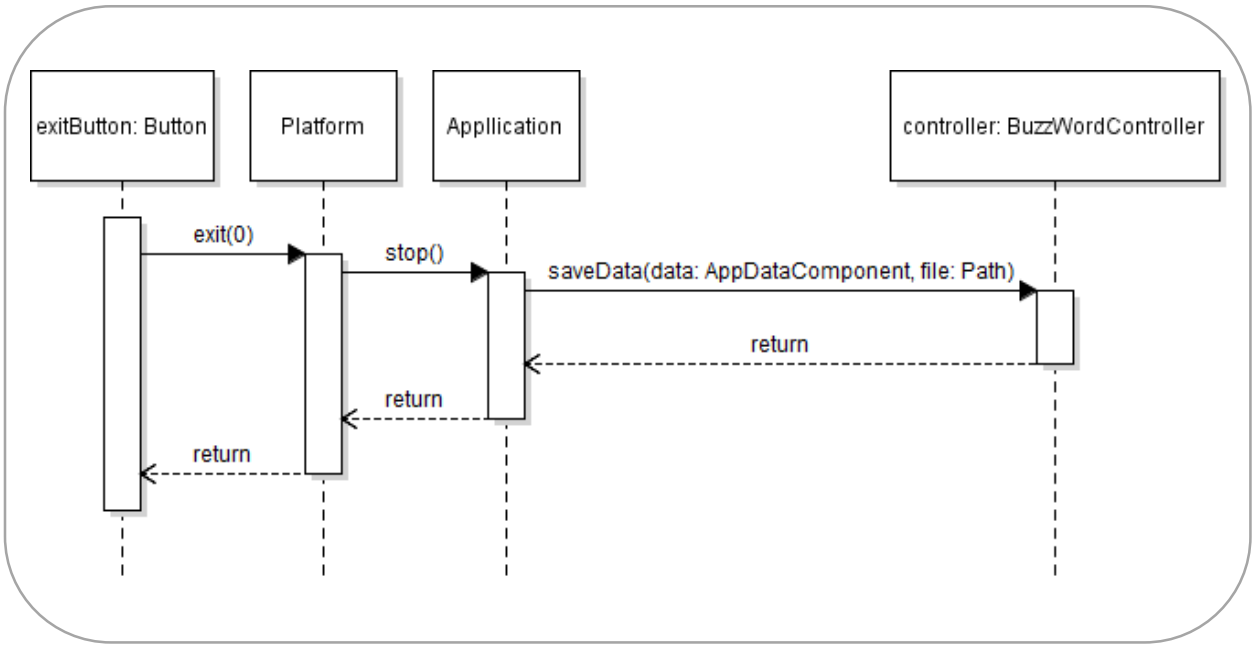

Figure 4.7: Exiting UML Sequence Diagram; will override Application.stop() method

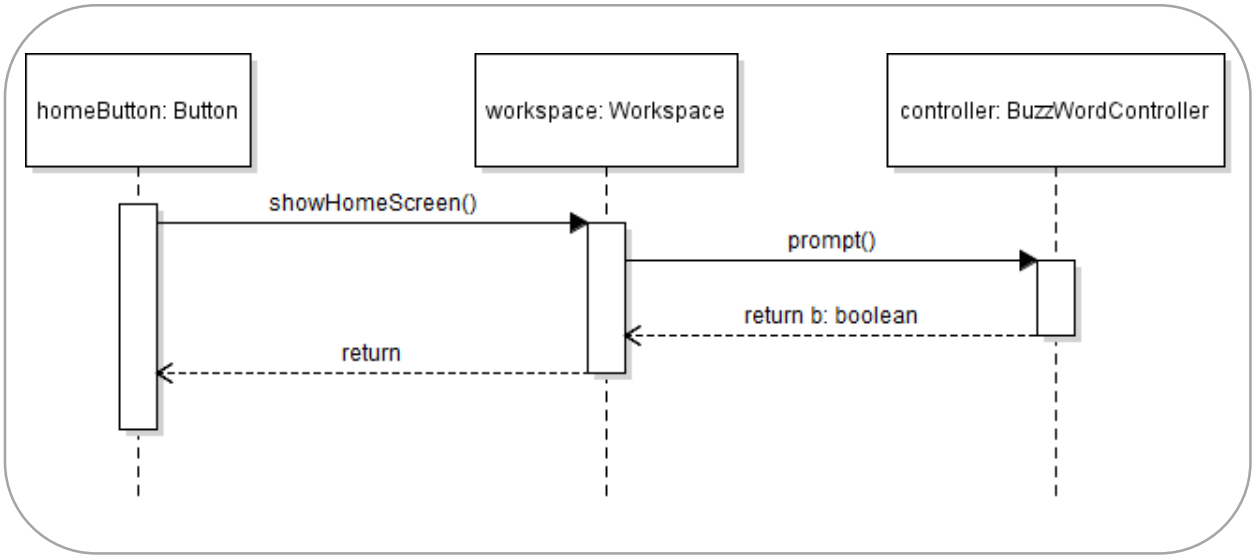

Figure 4.8: homeButton UML Sequence Diagram

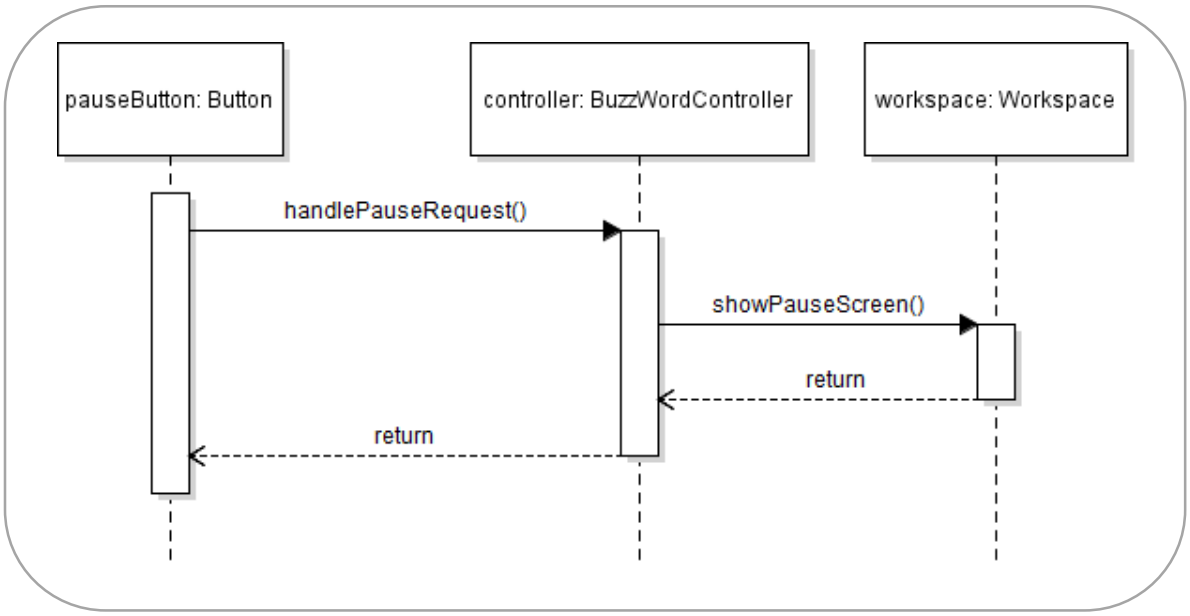

Figure 4.9: pauseButton UML Sequence Diagram

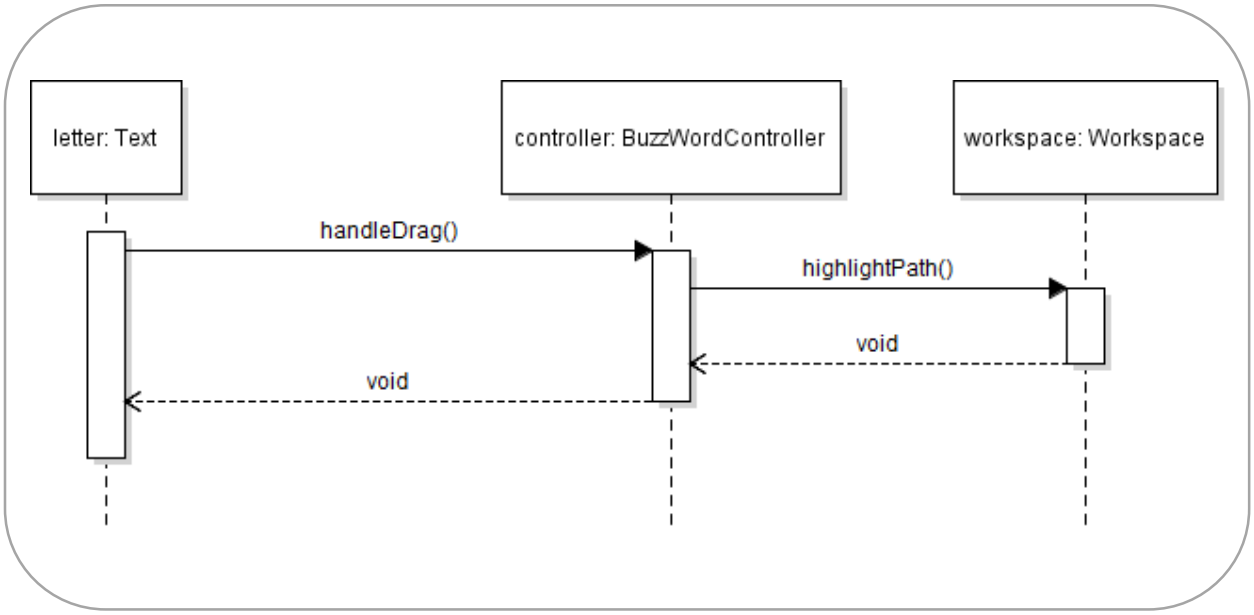

Figure 4.10: Selecting Text nodes by dragging UML Sequence Diagram

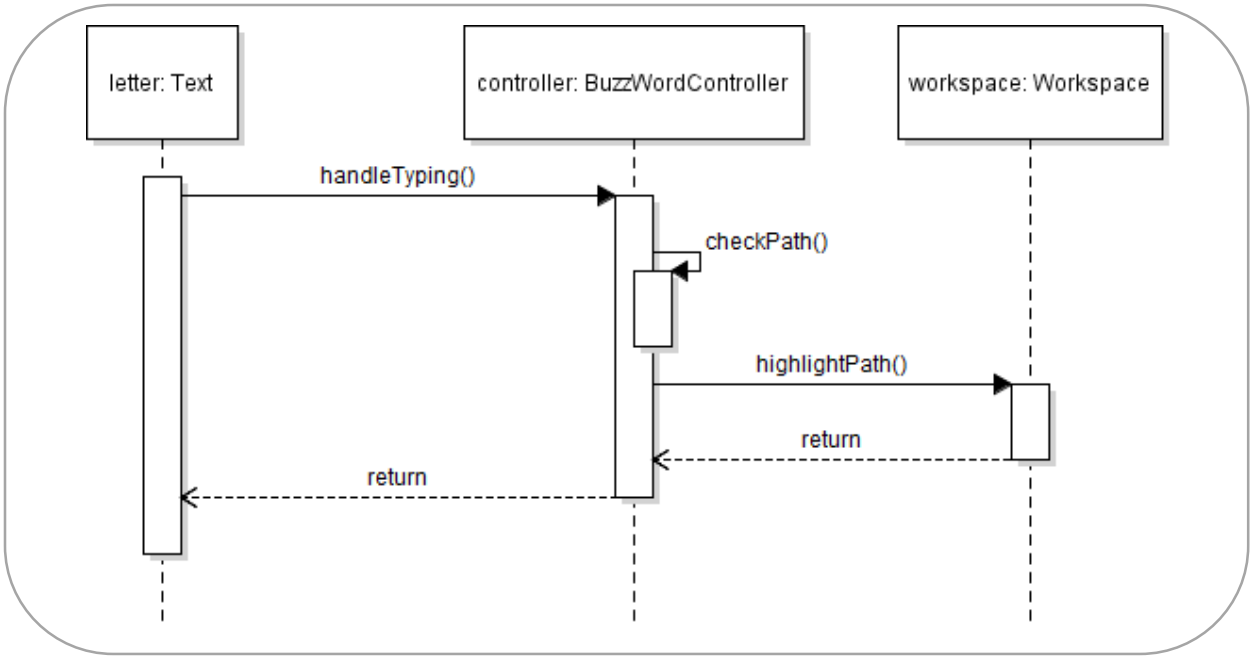

Figure 4.11: Selecting Text nodes by typing UML Sequence Diagram

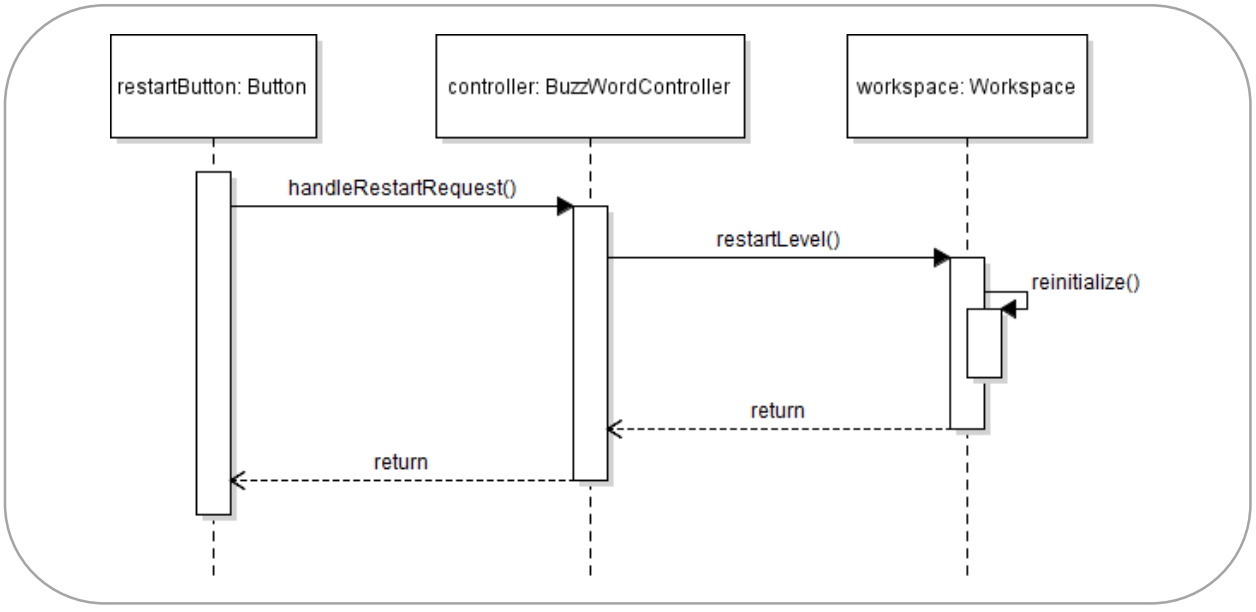

Figure 4.12: restartButton UML Sequence Diagram

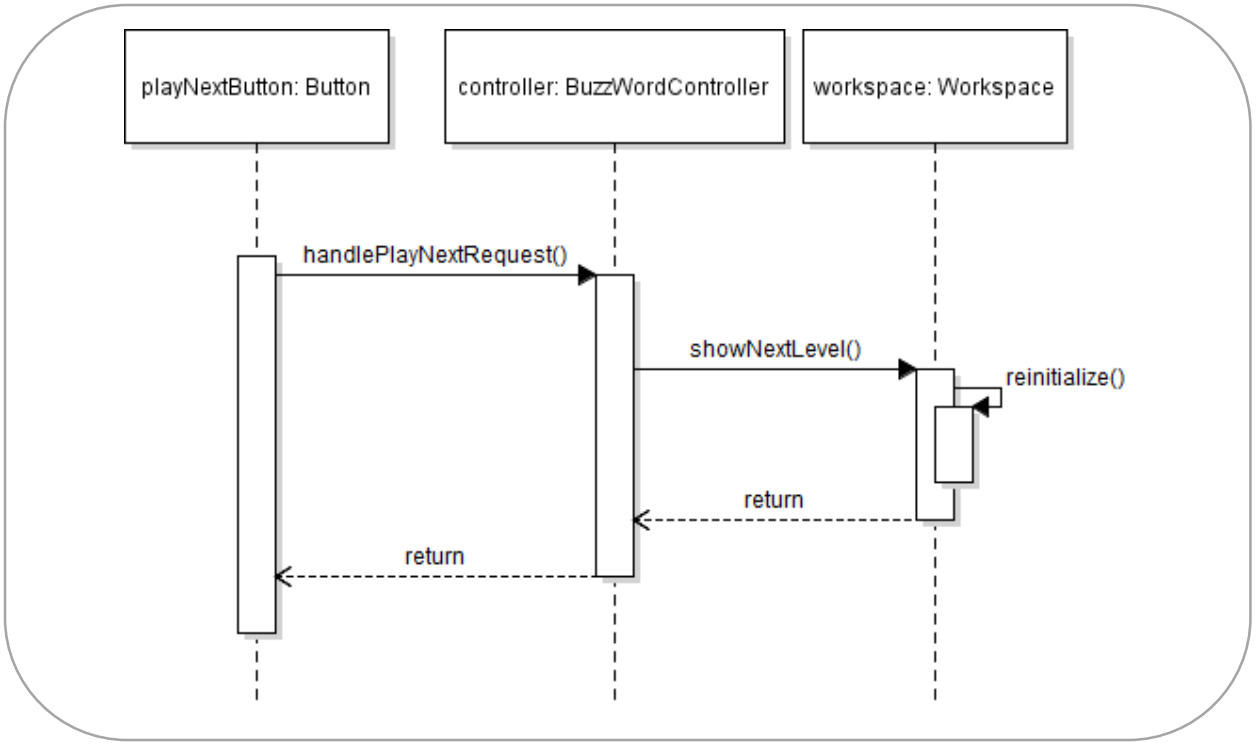

Figure 4.13: playNextButton UML Sequence Diagram

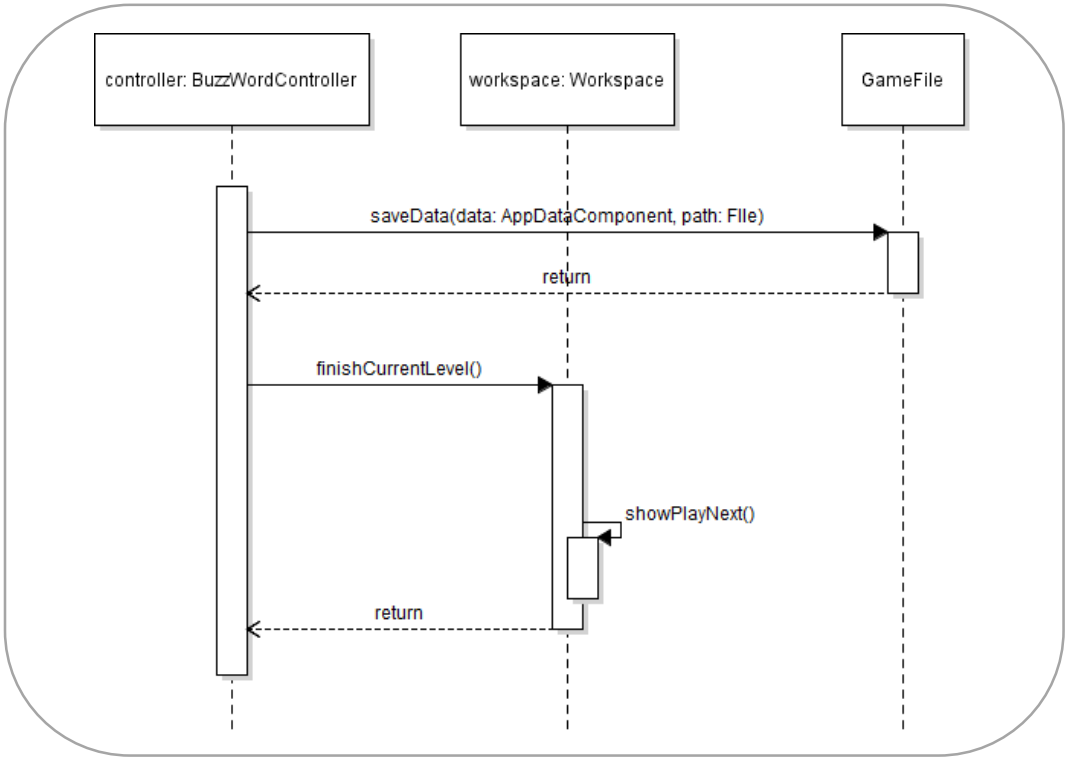

Figure 4.14: Automatically saving progress when completing a level

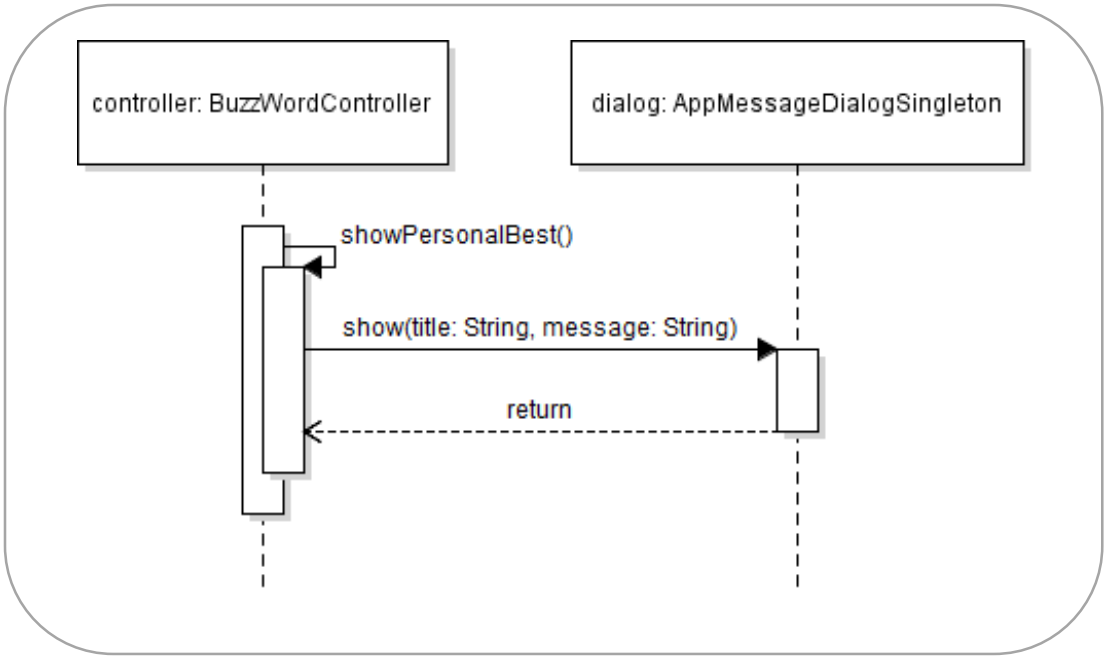

Figure 4.15: Showing a "Personal Best" Dialog

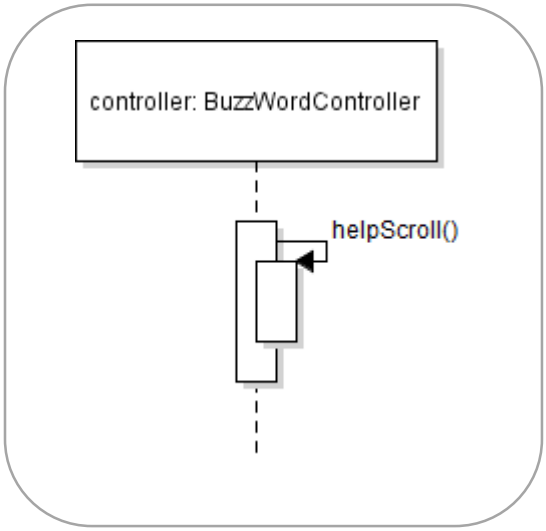

Figure 4.16: Scroll through help using key press

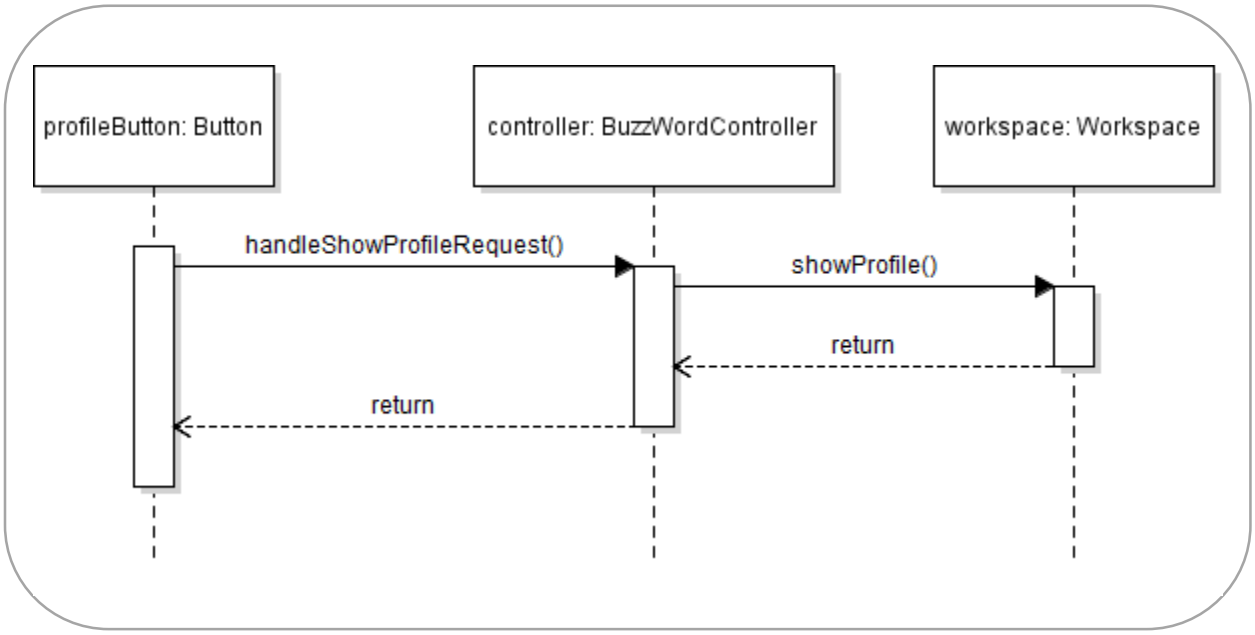

Figure 4.17: profileButton UML Sequence Diagram

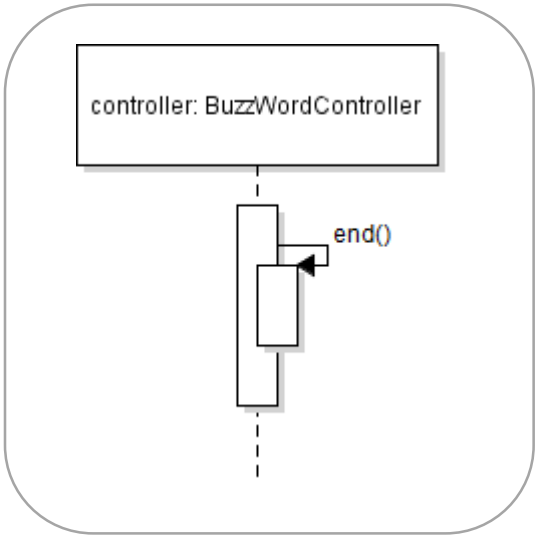

Figure 4.18: UML Sequence Diagram for when the level ends; will differentiate between whether or not the player lost or won

5 File Structure and Formats

Note that an executable JAR file XMLUtilities.jar will be provided and imported as a separate framework to this project to parse properties for the PropertyManager class. Note that all necessary data and image files must accompany the program BuzzWord, which will be deployed as an executable JAR file titled BuzzWord.jar. These data and image files must be under BuzzWord/resources.

Saved login credentials and data will be in a JSON file located in a path that the user will not know about.

6 Supporting Information

Note that this document should serve as a reference for implementing the code, and so a table of contents will be provided to help quickly find important sections.

6.1 Table of Contents

6.2 Appendixes

N/A