The *Course Site Generator* ™
Software Design Description

Author: Richard McKenna

Haolin Yu

Debugging Enterprises ™

Based on IEEE Std 830TM-1998 (R2009) document format

1 Introduction

This is the software Design Description (SDD) for the ***Course Site Generator***. 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 blue print for the construction of the ***Course Site Generator***. 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 for instructors to easily make and update course Web sites. There will be a common structure to the pages and so there are limitations on customization, but the site should be usable for instructors teaching courses in any department at any University.

1.3 Definitions, acronyms, and abbreviations

Document Object Model (DOM) – a tree data structure maintained by the browser that contains all content for the currently loaded Web page.

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.

HyperText Markup Language – a markup language used to describe Web pages. Web pages are text files encoded in HTML that can employ JavaScript and Stylesheets to build and style content.

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 Diagram – A UML document format that specifies how a user will interact with a system.

1.4 References

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

1.5 Overview

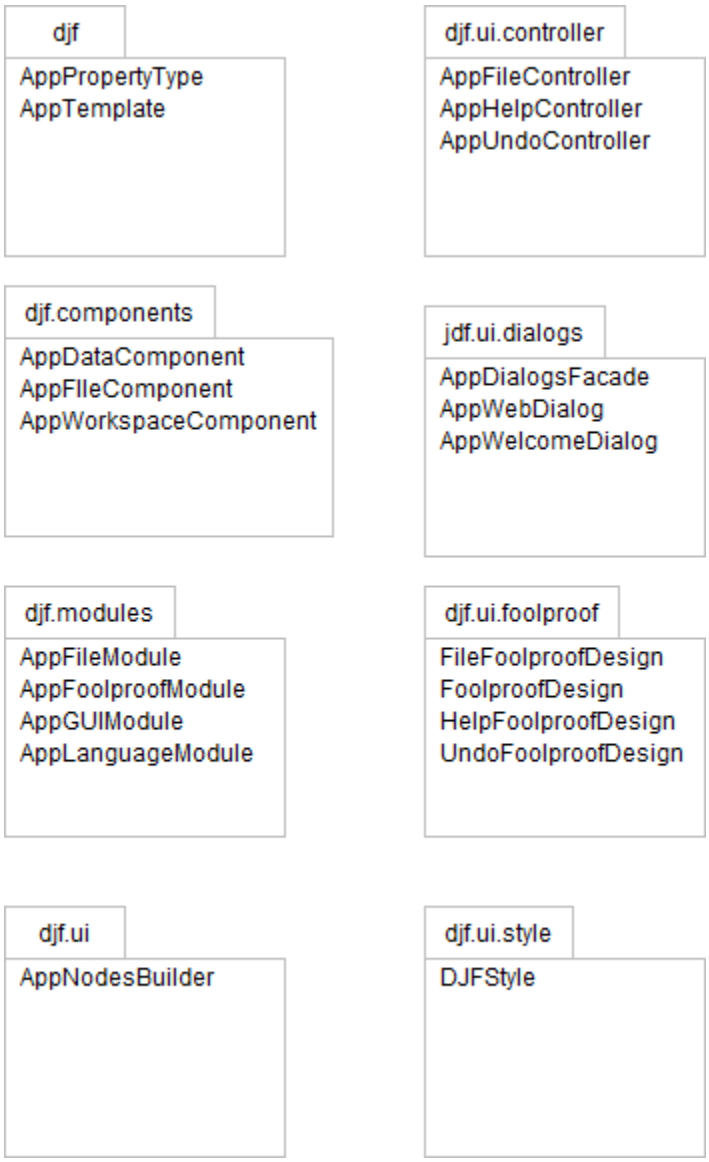
This Software Design Description document provides a working design for the ***Course Site Generator*** as described in the ***Course Site Generator*** Software Requirements Specification. Note that all parties in the implementation stage must agree upon all connections between components before proceeding 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

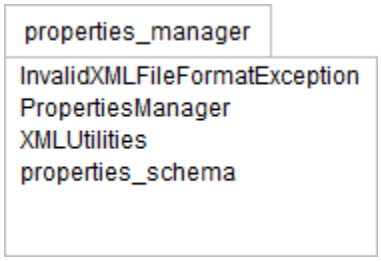
The design will encompass the ***Course Site Generator*** application, the Desktop Java Framework, JTPS transaction, and Properties Manager to be used in its construction. Following are descriptions of the components to be built, as well as how the Java API will be used to build them.

2.1 Application overview

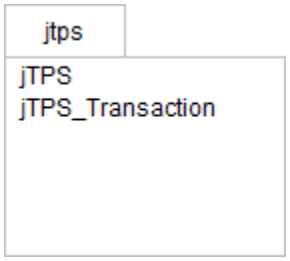
Desktop Java Framework



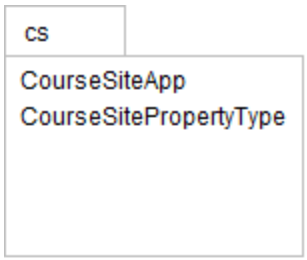
Properties Manager

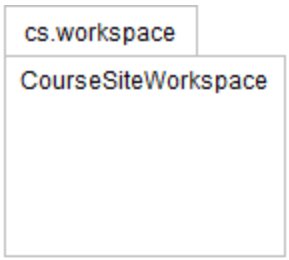


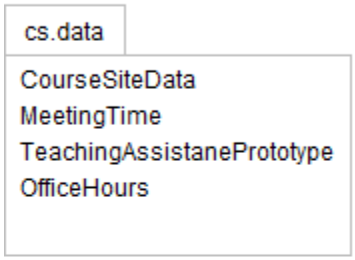
jTPS

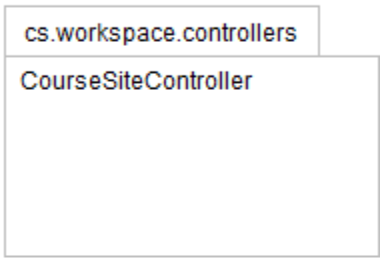


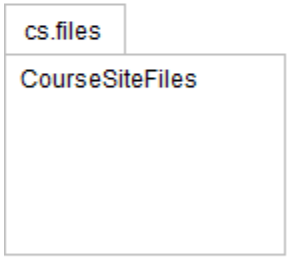
Course Site Generator

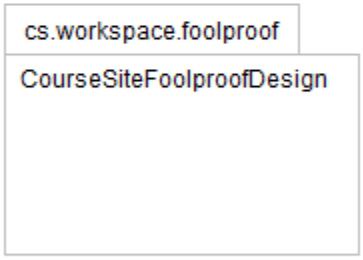


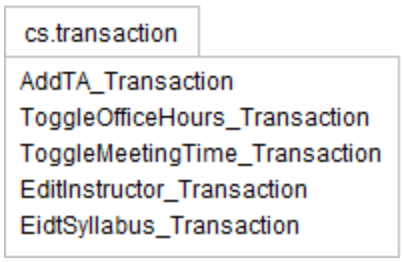


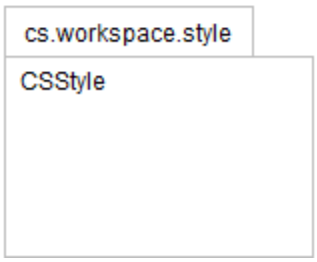






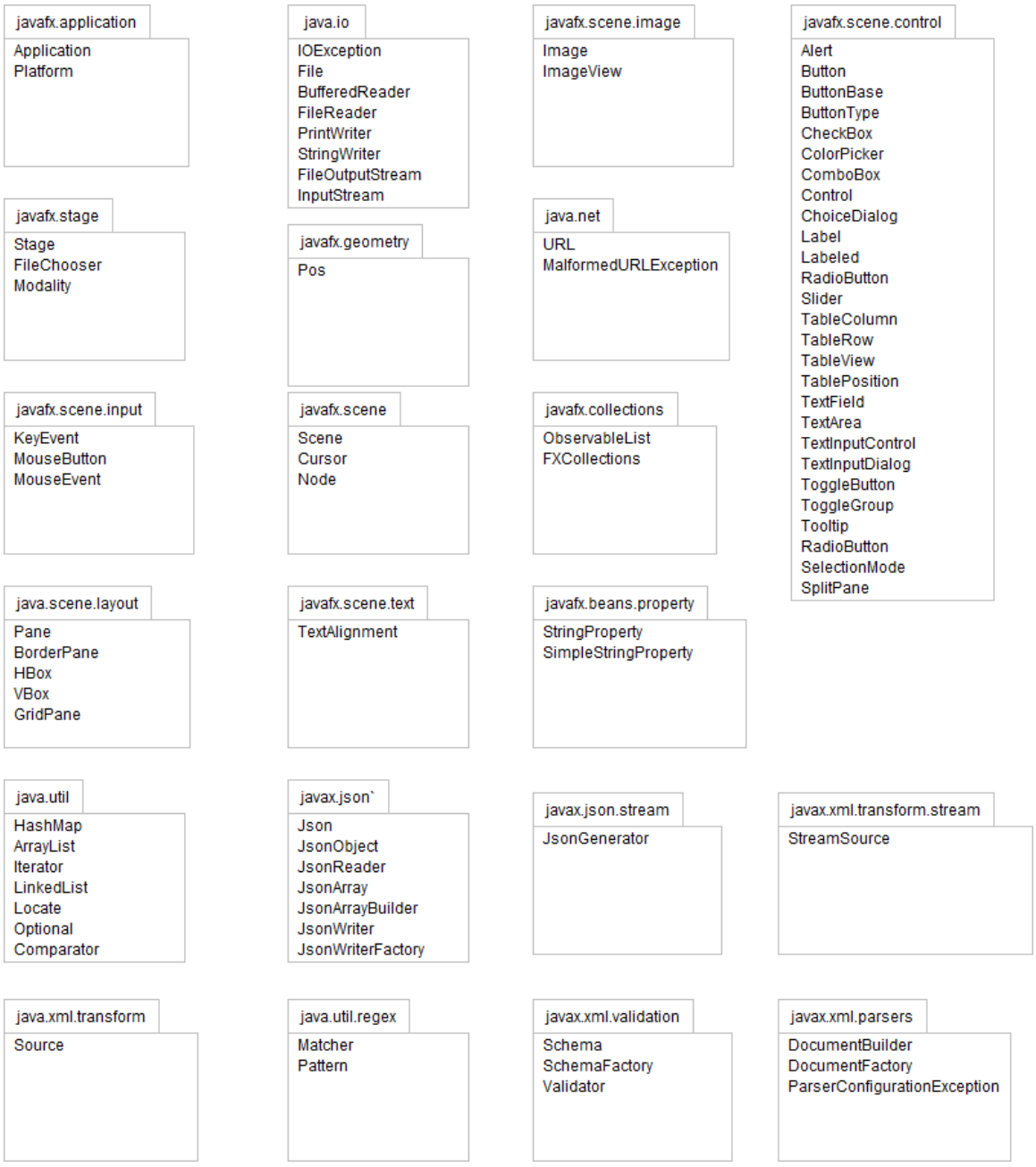






2.2 Java API Usage

The whole application will be developed using the Java programming languages. As such, this design will make use of the classes specified below.



2.3 Java API Usage Descriptions

Use for classes in the Java API's javafx.application package

Class/Interface	Use
Application	For launching a JavaFX application
Platform	For run the pane in the application

Use for classes in the Java API's javafx.stage package

Class/Interface	Use
Stage	For setting the window for the application
FileChooser	For choosing one file from directory
Modality	For setting the stage

Use for classes in the Java API's java.io package

Class/Interface	Use
IOException	For catching exception in IO
File	A data type
BufferedReader	For reading text from a input stream
FileReader	For reading file
PrintWriter	For printing formatted representations of objects to a text-output stream
StringWriter	A character stream that collects its output in a string buffer
FileOutputStream	For writing streams of raw bytes
InputStream	For getting an input stream of bytes

Use for classes in the Java API's javafx.scene.layout package

Class/Interface	Use
BorderPane	For setting the node in top, left, center, right, and bottom regions
Pane	Base class for layout panes
HBox	For setting the nodes in a single horizontal row
VBox	For setting the nodes in a single vertical column
GridPane	For setting the nodes within a flexible grid of rows and columns

Use for classes in the Java API's javafx.geometry package

Class/Interface	Use
Pos	For setting the position of nodes

Use for classes in the Java API's java.scene.image package

Class/Interface	Use
Image	For loading images from a path
ImageView	For painting image

Use for classes in the Java API's javafx.scene.control package

Class/Interface	Use
Alert	A kind of pre-build dialog
Button	A simple button control
ButtonBase	Base class for button-like UI Controls
ButtonType	Specify which buttons should be shown to users in the dialog
CheckBox	A tri-state selection Control typically skinned as a box with a checkmark or tick mark when checked
ColorPicker	ColorPicker control allows the user to select a color from either a standard palette of colors with a simple one click selection OR define their own custom color
ComboBox	An implementation of the ComboBoxBase abstract class for the most common form of ComboBox, where a popup list is shown to users providing them with a choice that they may select from
Control	Base class for all user interface controls
ChoiceDialog	A pre-bulit dialog for choices
Label	Label is a non-editable text control
Labeled	A labeled control is one which has as part of its user interface a textual content associated with it
RadioButton	RadioButtons create a series of items where only one item can be selected
Slider	The Slider Control is used to display a continuous or discrete range of valid numeric choices and allows the user to interact with the control

TableColumn	A TableView is made up of a number of TableColumn instances
TableRow	TableRow is an IndexedCell, but rarely needs to be used by developers creating TableView instances
TableView	The TableView control is designed to visualize an unlimited number of rows of data, broken out into columns
TablePosition	This class is used to represent a single row/column/cell in a TableView
TextField	Text input component that allows a user to enter a single line of unformatted text
TextArea	Text input component that allows a user to enter multiple lines of plain text
TextInputControl	Abstract base class for text input controls
TextInputDialog	A pre-built dialog for text input
ToggleButton	A ToggleButton is a specialized control which has the ability to be selected
ToggleGroup	A class which contains a reference to all Toggles whose selected variables should be managed such that only a single Toggle within the ToggleGroup may be selected at any one time
Tooltip	Tooltips are common UI elements which are typically used for showing additional information about a Control when the Control is hovered over by the mouse
RadioButton	Radio buttons are combined into a group where only one button at a time can be selected
SelectionMode	For setting the selection mode
SplitPane	A control that has two or more sides, each separated by a divider, which can be dragged by the user to give more space to one of the sides, resulting in the other side shrinking by an equal amount

Use for classes in the Java API's java.net package

Class/Interface	Use
URL	For getting path of files
MalformedURLException	For catching the errors

Use for classes in the Java API's javafx.scene package

Use/Interface	Use
Scene	For displaying the root node
Cursor	A class to encapsulate the bitmap representation of the mouse cursor
Node	Builder class for javafx.scene.ImageCursor

Use for classes in the Java API's java.util package

Use/Interface	Use
HashMap	For sorting key pairs
ArrayList	Resizable-array implementation of the List interface
Iterator	An iteractor over a collection
LinkedList	Linked nodes as a sequence
Locate	A Locale object represents a specific geographical, political, or cultural region
Opitional	A container object which may or may not contain a non-null value
Comparator	A comparison function, which imposes a total ordering on some collection of objects

Use for classes in the Java API's java.json package

Use/Interface	Use
Json	Factory class for creating json processing objects
JsonObject	For representing the json object
JsonReader	For reading json object
JsonArray	For representing a json array
JsonArrayBuilder	A builder for creating JsonArray models from scratch
JsonWriter	For writing a json object
JsonWriterFactory	For creating JsonWriter instances

Use for classes in the Java API's javafx.beans.property package

Use/Interface	Use
StringProperty	This class provides a full implementation of a Property wrapping a String value
SimpleStringProperty	This class provides a full implementation of a Property wrapping a String value

Use for classes in the Java API's javax.xml.parsers package

Class/Interface	Use
DocumentBuilder	For building a document according to xml File
DocumentBuilderFactory	For creating DocumentBuilder instances
ParserConfigurationException	For catching the parser configuration exception

Use for classes in the Java API's javax.xml.validation package

Class/Interface	Use
Schema	This object represents a set of constraints that can be checked/ enforced against an XML document
SchemaFactory	Factory that creates Schema objects
Validator	Validates bean instances

Use for classes in the Java API's javafx.scene.input package

Class/Interface	Use
KeyEvent	An event which indicates that a keystroke occurred in a Node
MouseButton	Mapping for Button Names
MouseEvent	When mouse event occurs, the top-most node under cursor is picked and the event is delivered to it through capturing and bubbling phases described at EventDispatcher

Use for classes in the Java API's javafx.collections package

Class/Interface	Use
ObservableList	A list that allows listeners to track changes when they occur
FXCollections	Utility class that consists of static methods that are 1:1 copies of java.util.Collections methods

Use for classes in the Java API's javafx.scene.text package

Class/Interface	Use
TextAlignment	The TextAlignment enum represents the horizontal text alignment

Use for classes in the Java API's java.xml.transform package

Class/Interface	Use
Source	An object that implements this interface contains the information needed to act as source input (XML source or transformation instructions)

Use for classes in the Java API's java.util.regex package

Class/Interface	Use
Matcher	An engine that performs match operations on a character sequence by interpreting a Pattern
Pattern	A compiled representation of a regular expression

Use for classes in the Java API's javax.json.stream package

Class/Interface	Use
JsonGenerator	Writes JSON data to an output source in a streaming way

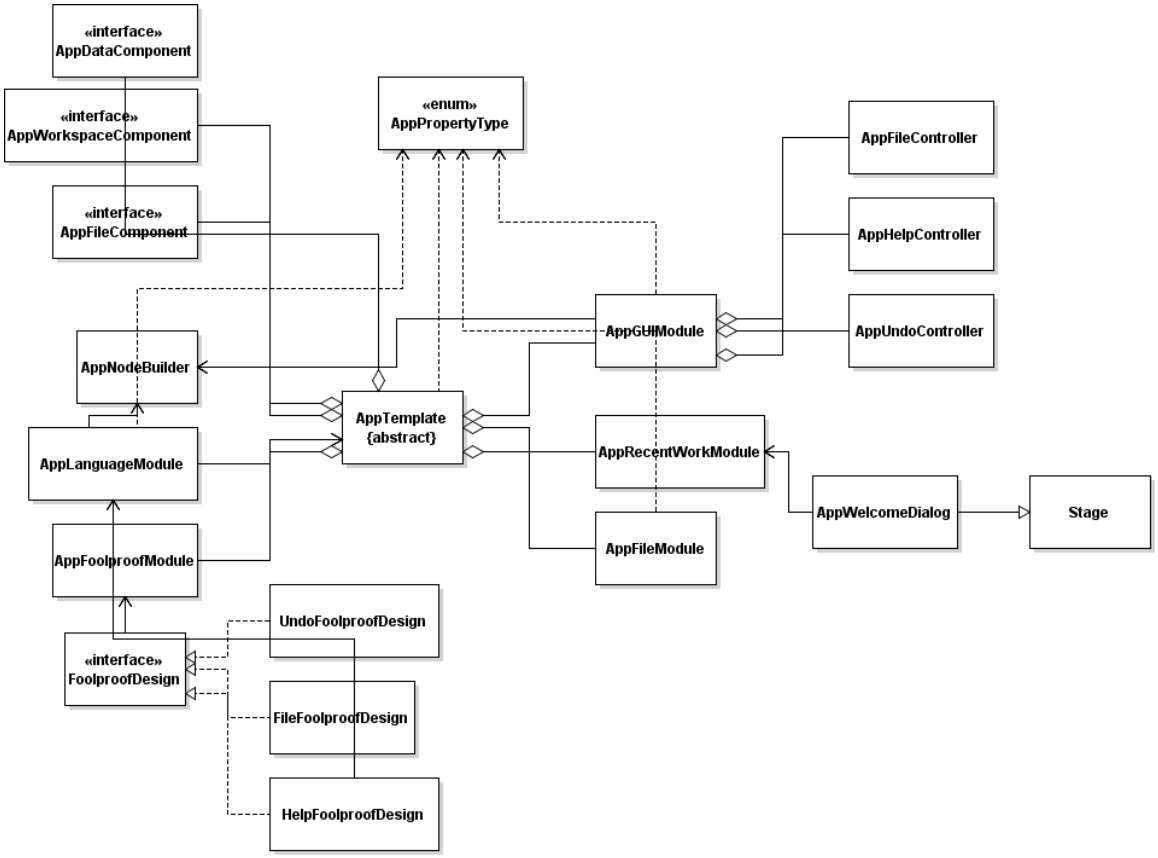
Use for classes in the java API's javax.xml.transform.stream package

Class/Interface	Use
StreamSource	Acts as an holder for a transformation Source in the form of a stream of XML markup

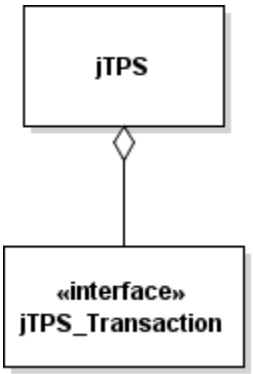
3 Class – level Design Viewpoint

As mentioned, this design will encompass the ***Course Site Generator*** application, the Desktop Java Framework, JTPS transaction, and Properties Manager. The following UML Class Diagrams reflect this. Note that due to the complexity of the project, we present the class designs using a series of diagrams going from overview diagrams down to detail ones.

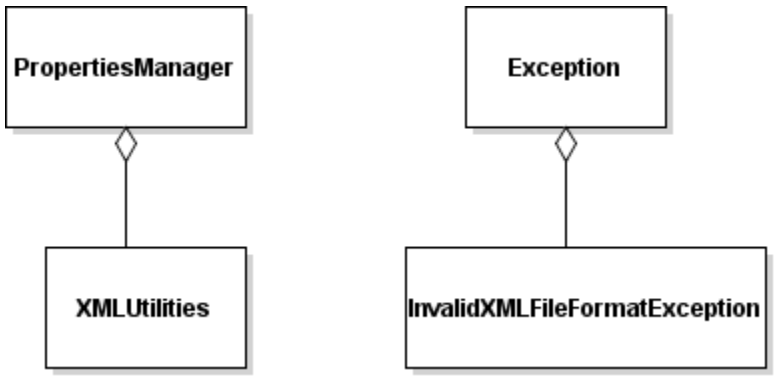
Desktop Java Framework Overview UML Class Diagram



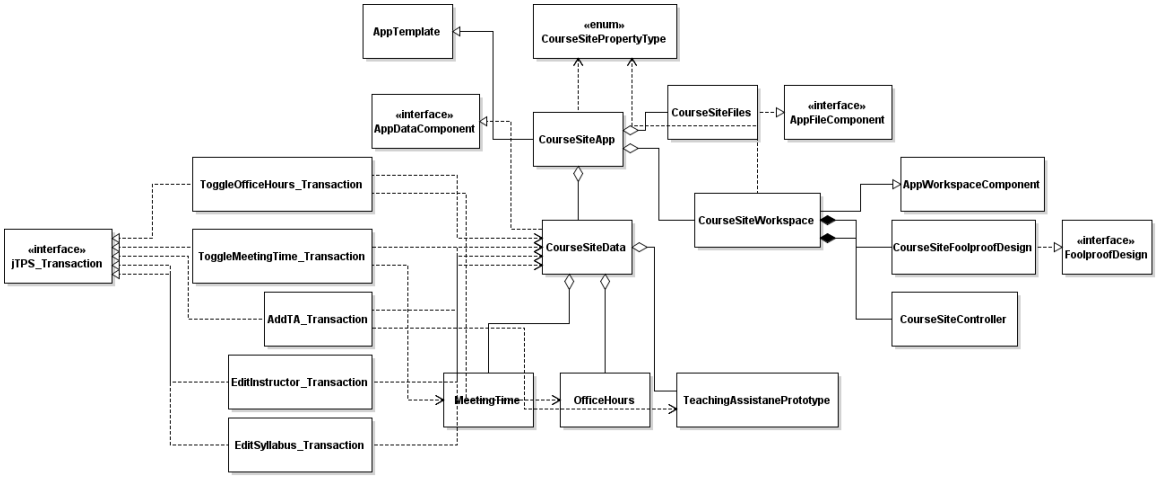
jTPS transaction Overview UML Class Diagram



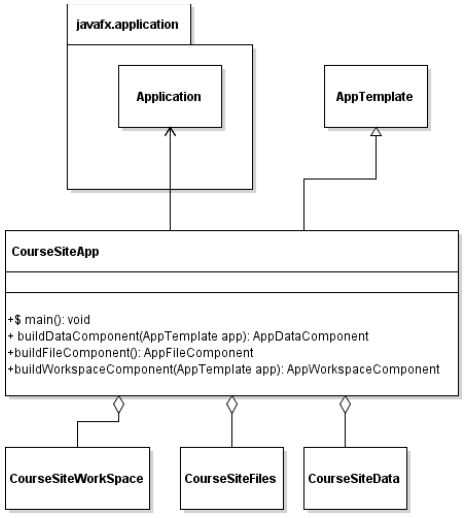
Properties Manager Framework Overview UML Class Diagram



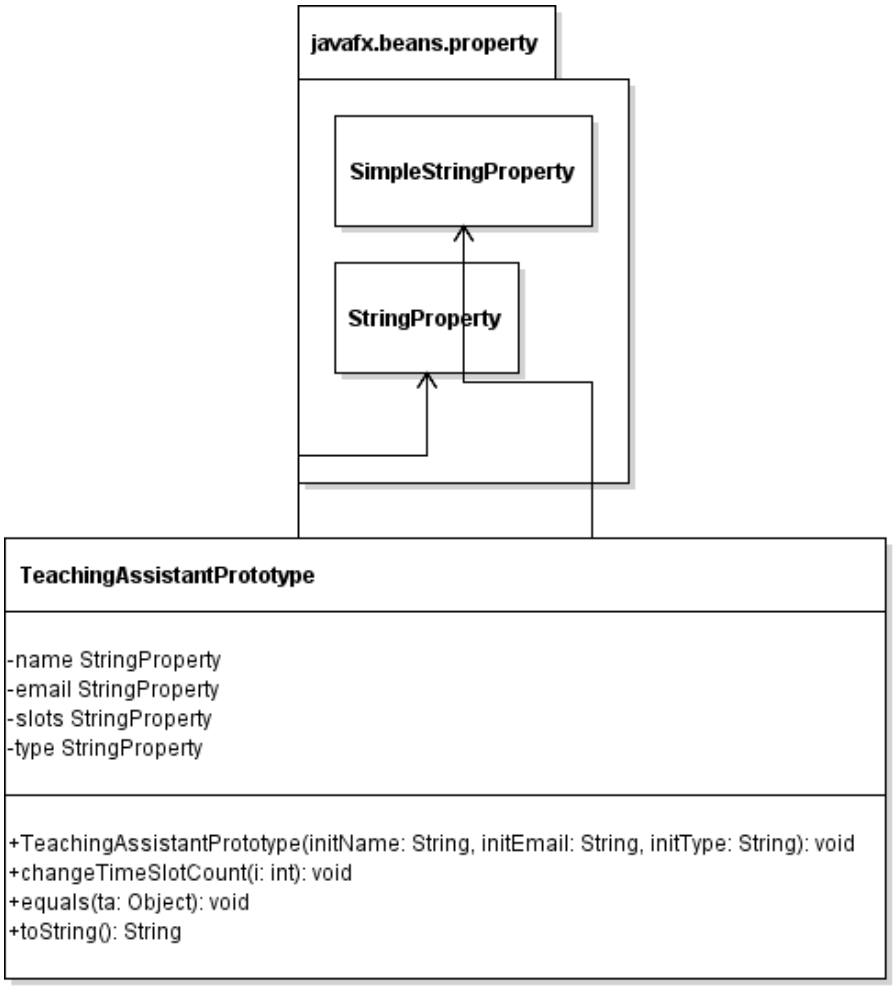
CourseSiteGenerator Overview UML Class Diagram



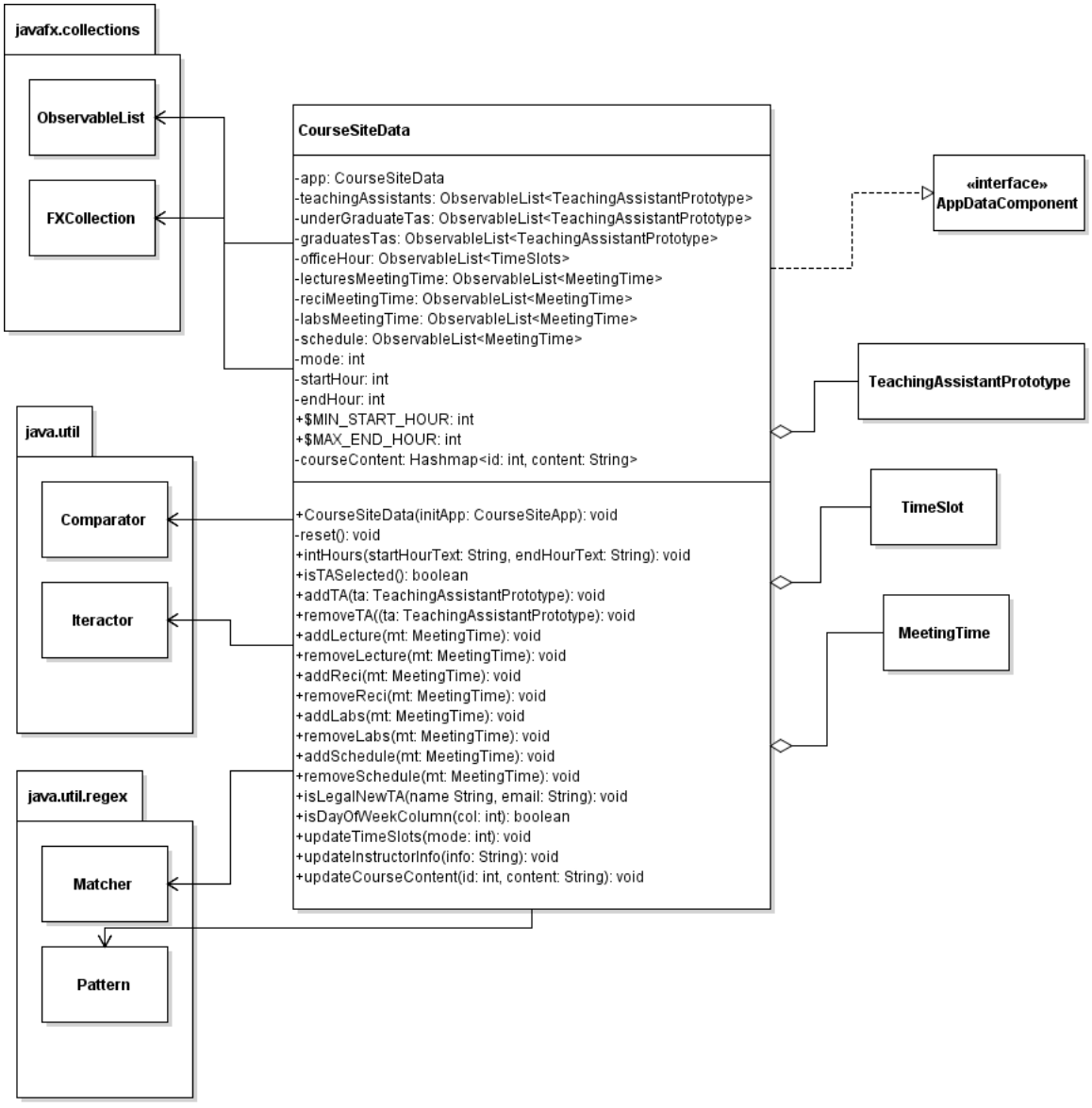
Detailed CourseSiteApp UML Class Diagram



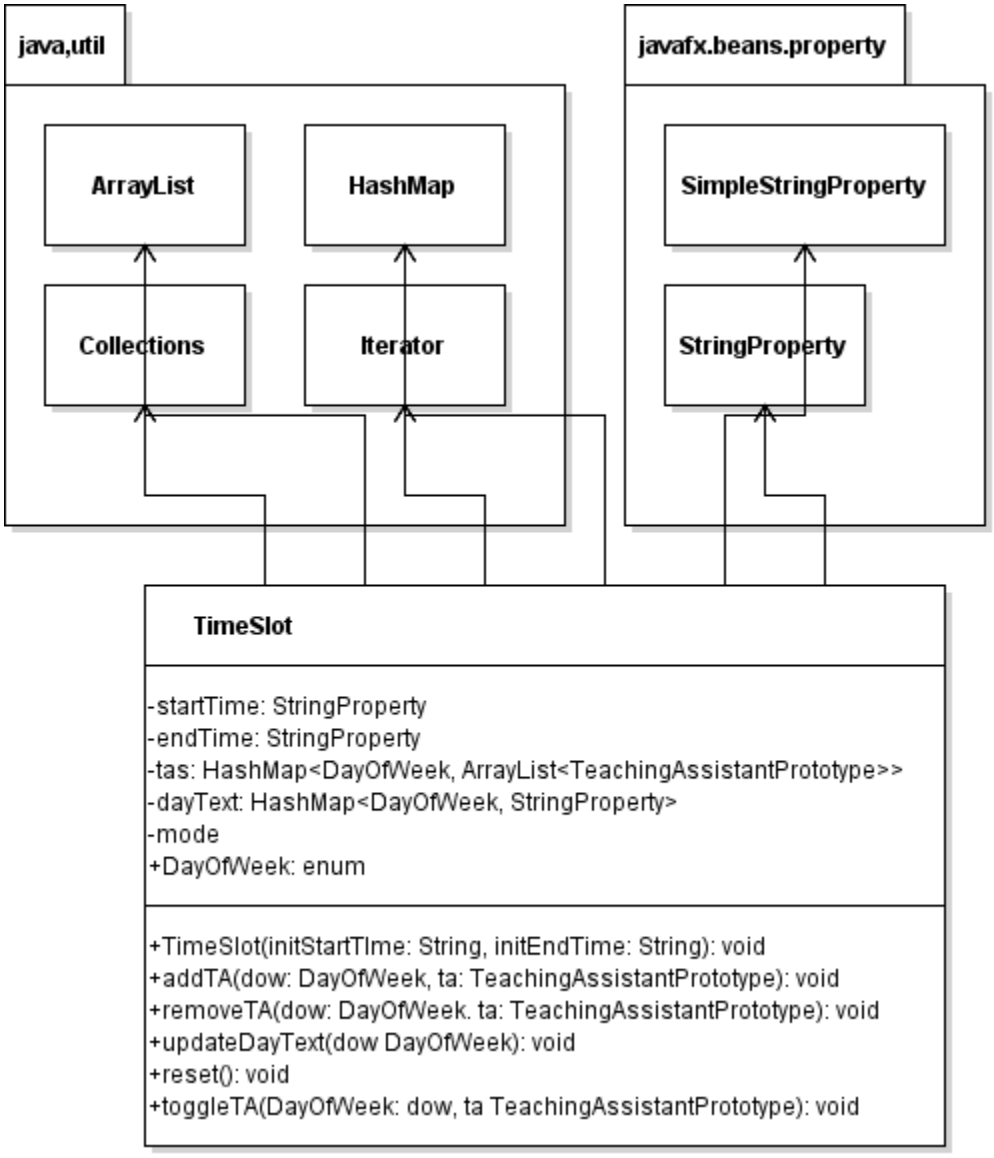
Detailed TeachingAssistantPrototype



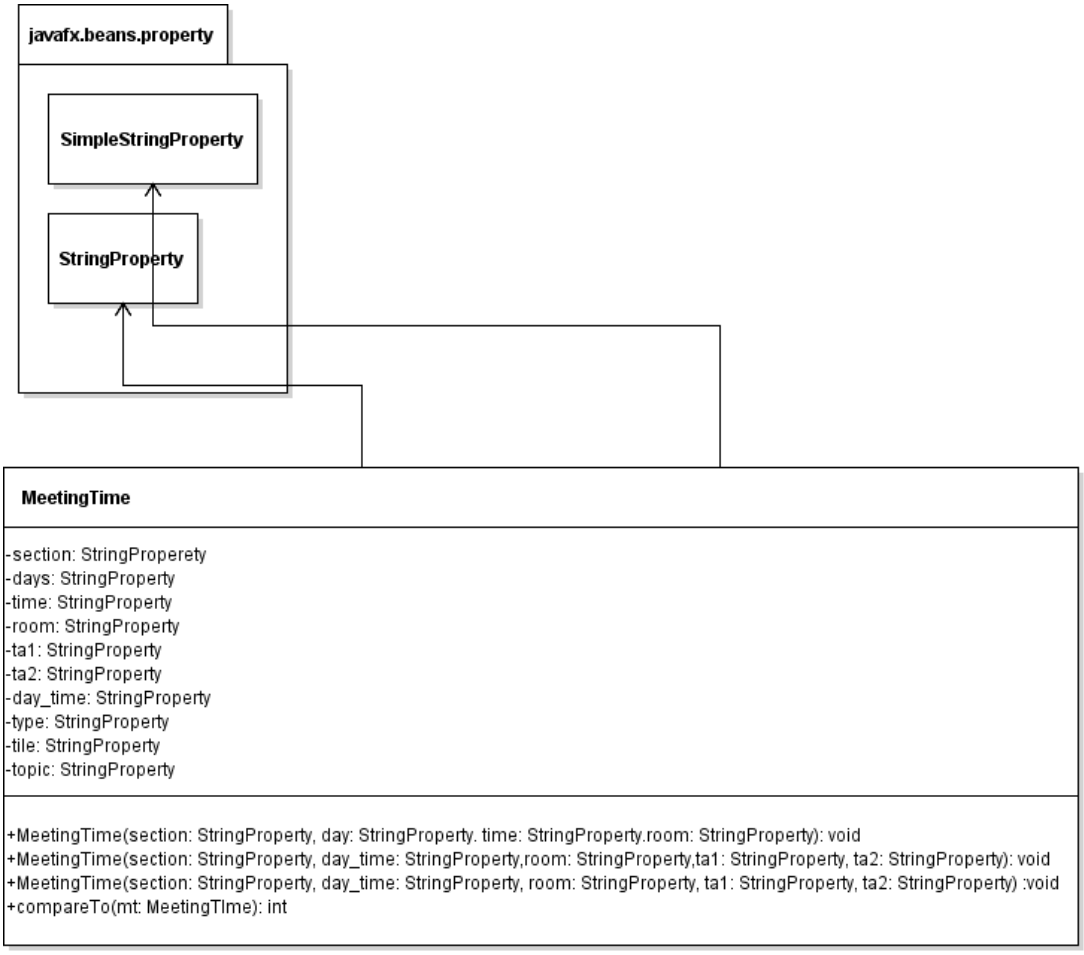
Detailed CourseSiteData UML Class Diagram



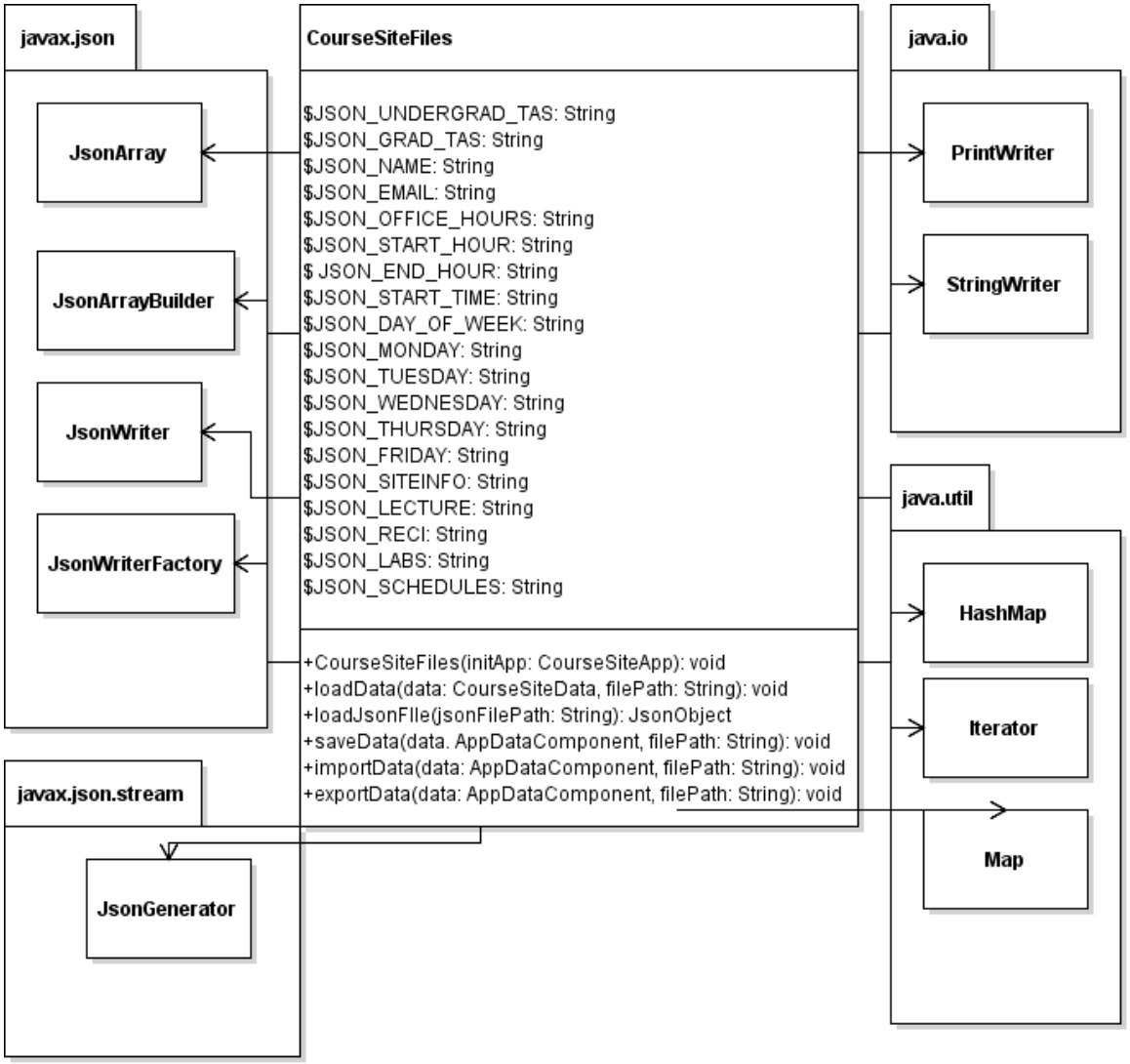
Detailed TimeSlot UML Class Design Diagram



Detailed MeetingTime UML Class Design Diagram

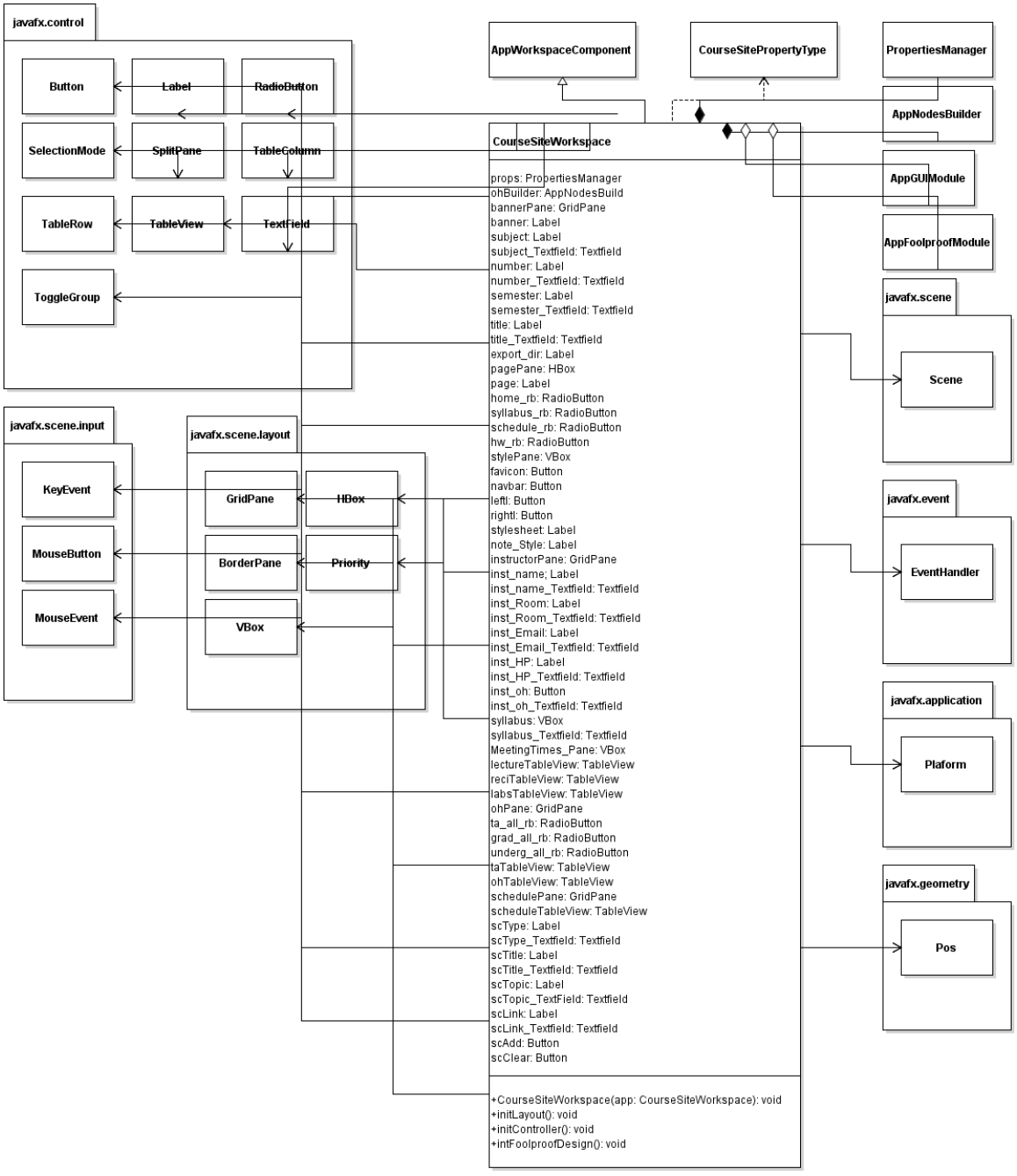


Detailed CourseSiteFiles UML Class Design Diagram

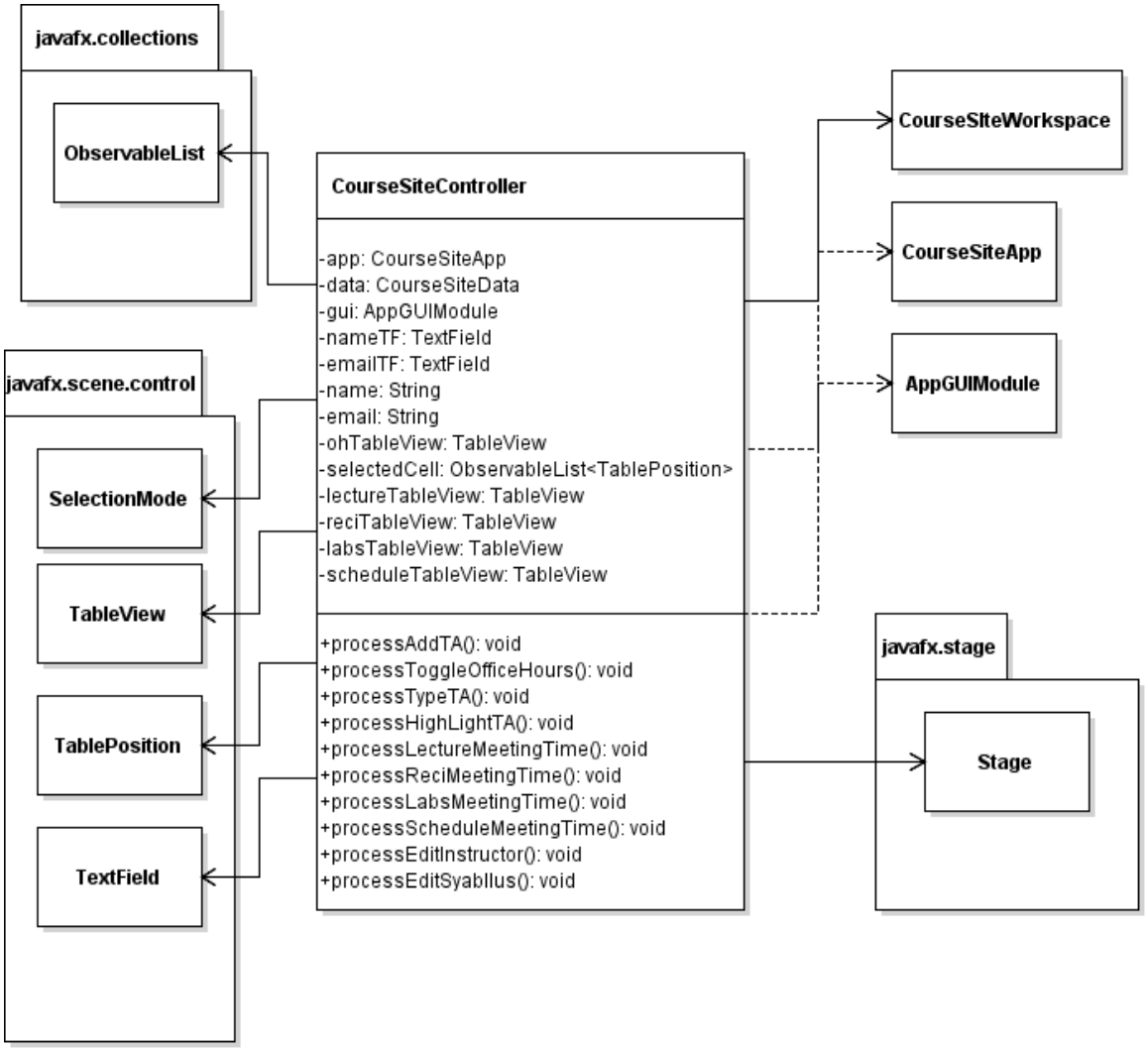


Detailed Transaction UML Class Design Diagram

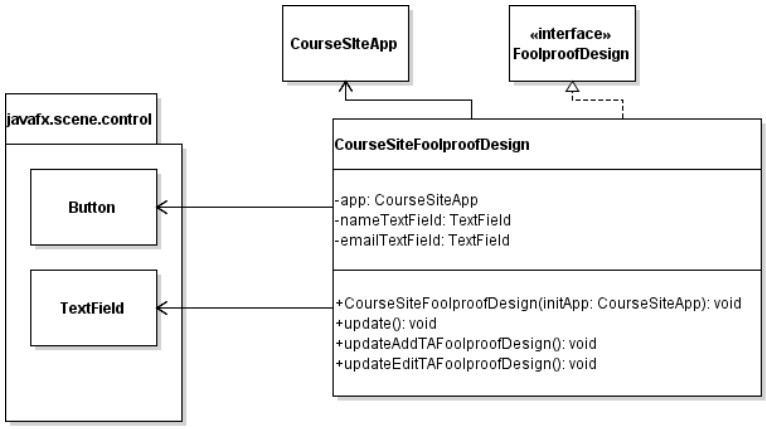
Detailed CourseSiteWorkspace UML Class Design Diagram



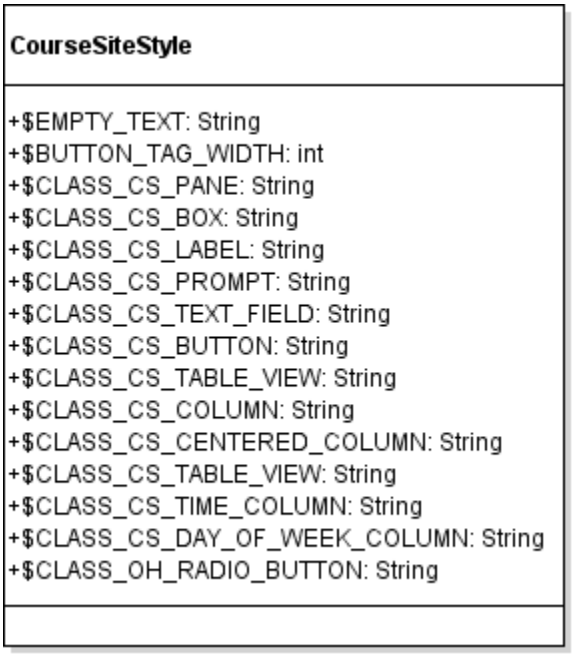
Detailed CourseSiteController UML Class Design Diagram



CourseSiteFoolproofDesign UML Class Design Diagram



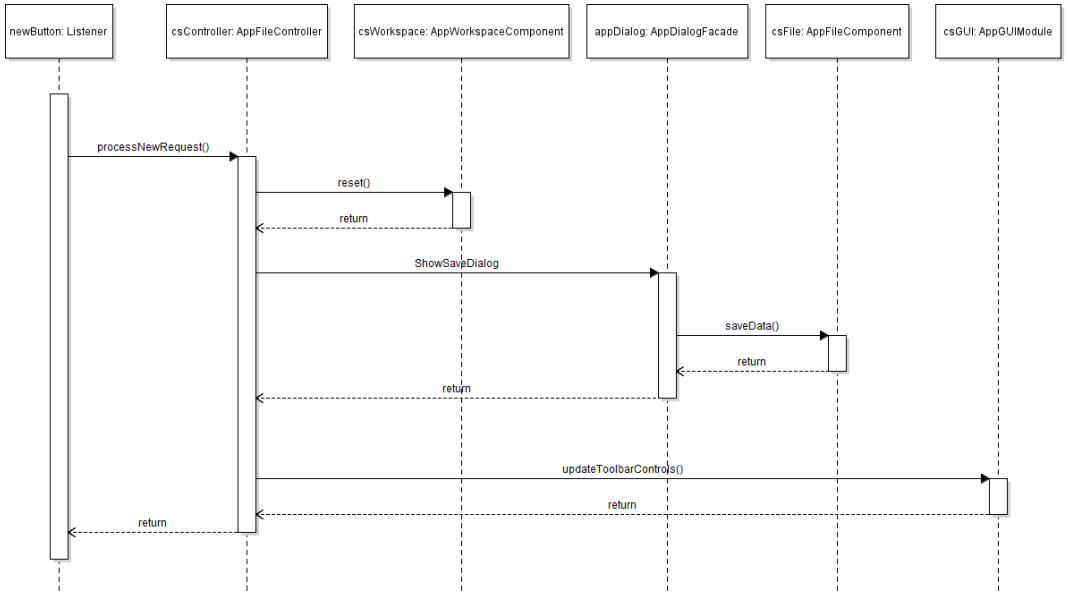
CourseSiteStyle UML Class Design Diagram



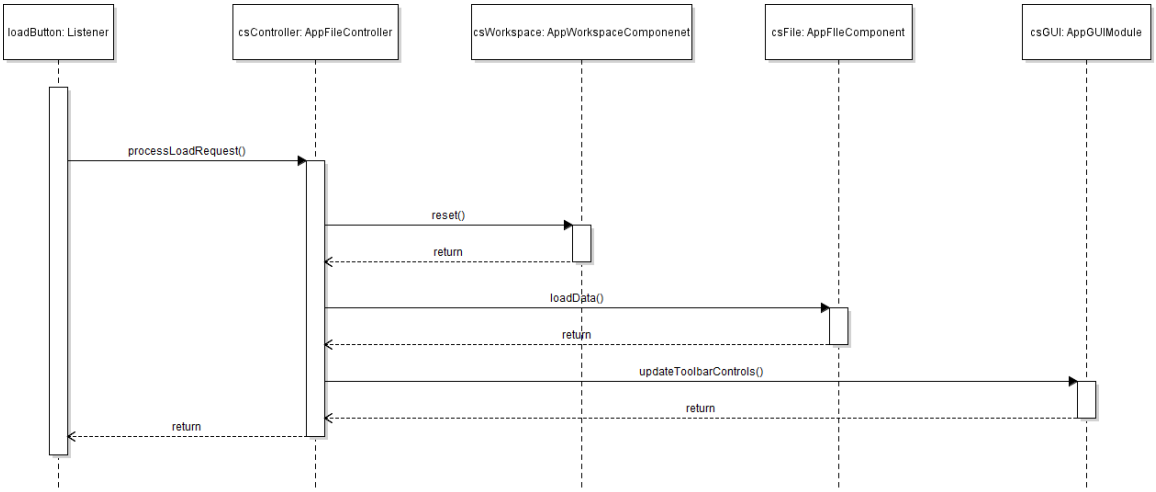
4. Method – Level Design Viewpoint

Now that the general architecture of the classes has been determined, it is time to specify how data will flow through the system. The following UML sequence Diagrams describe the methods called within the code to be developed in order to provide the appropriate event responses.

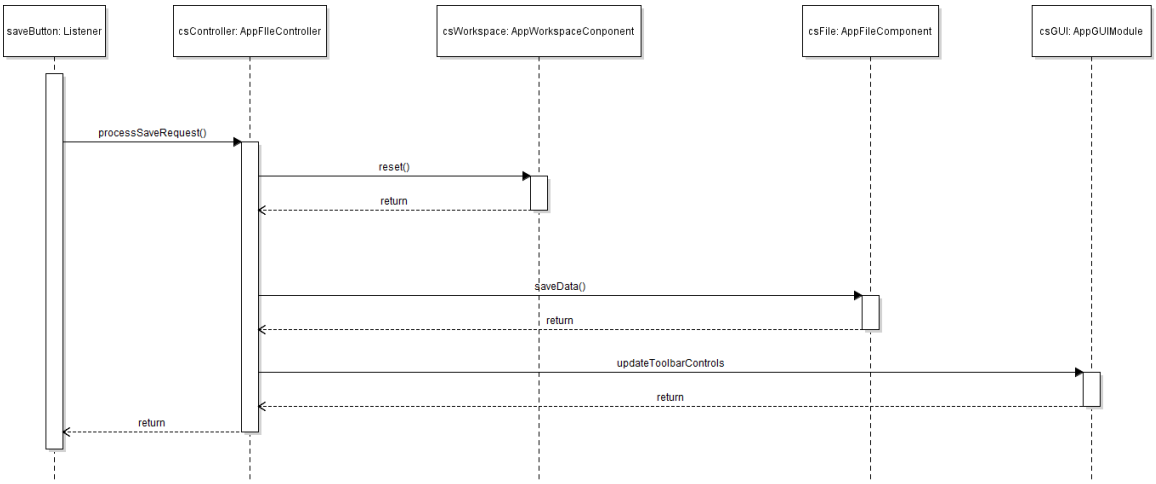
Create New Course Site UML Sequence Diagram



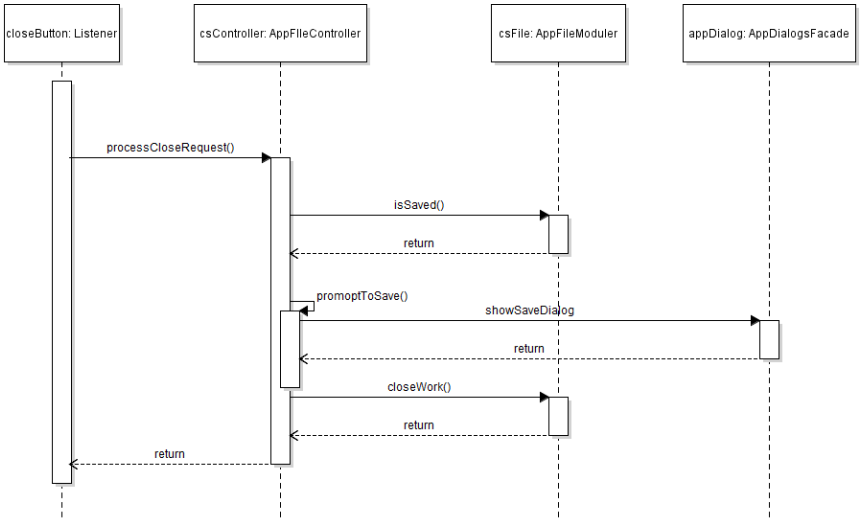
Load Course Site UML Sequence Diagram



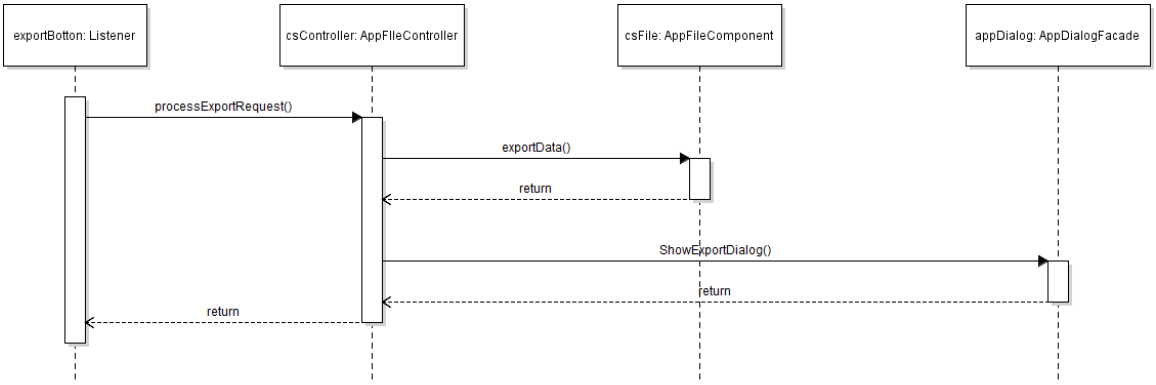
Save Course Site UML Sequence Diagram



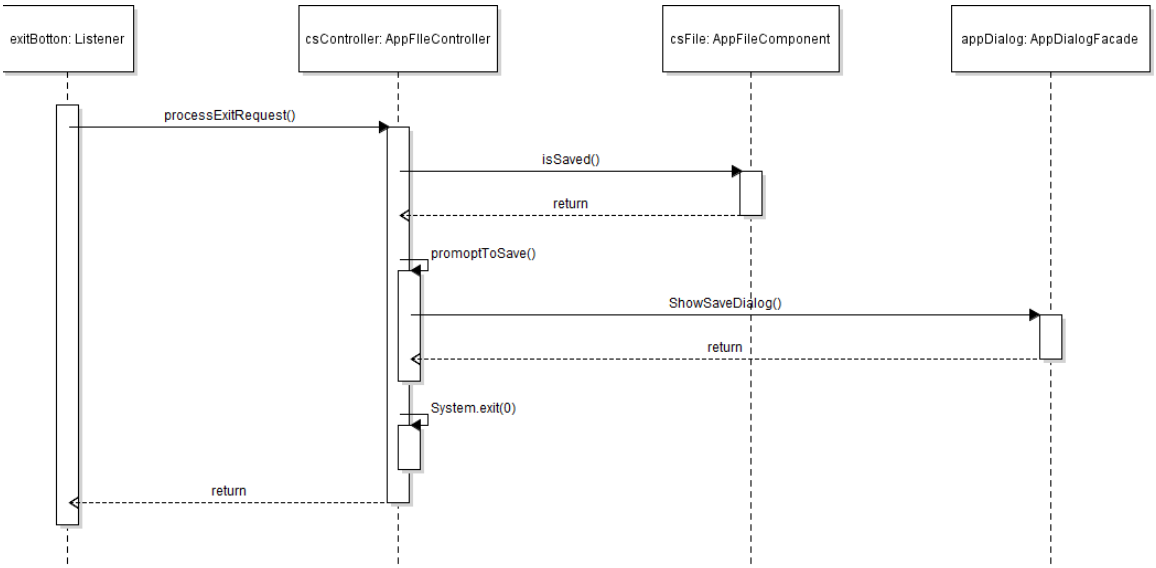
Close Course Site UML Sequence Diagram



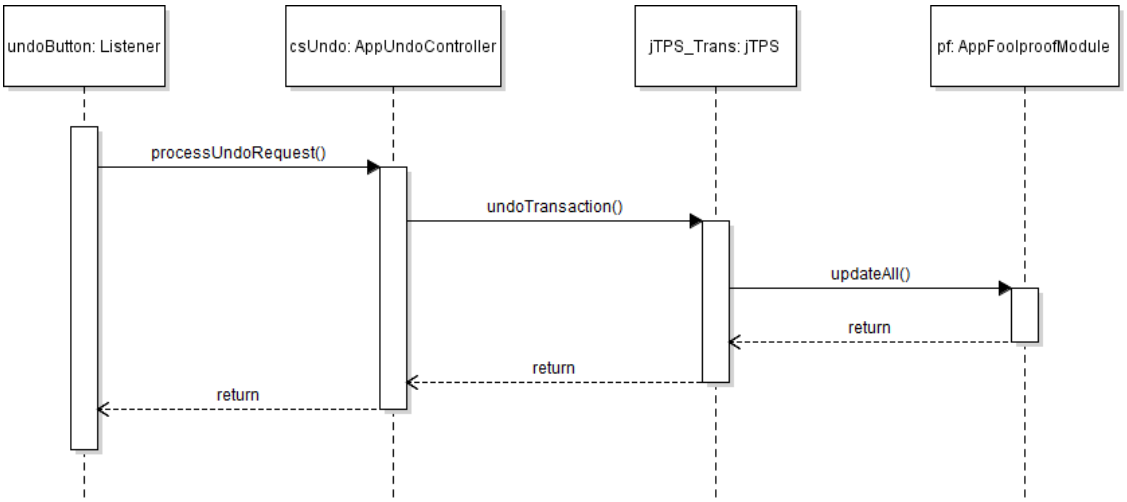
Export & View Course Site UML Sequence Diagram



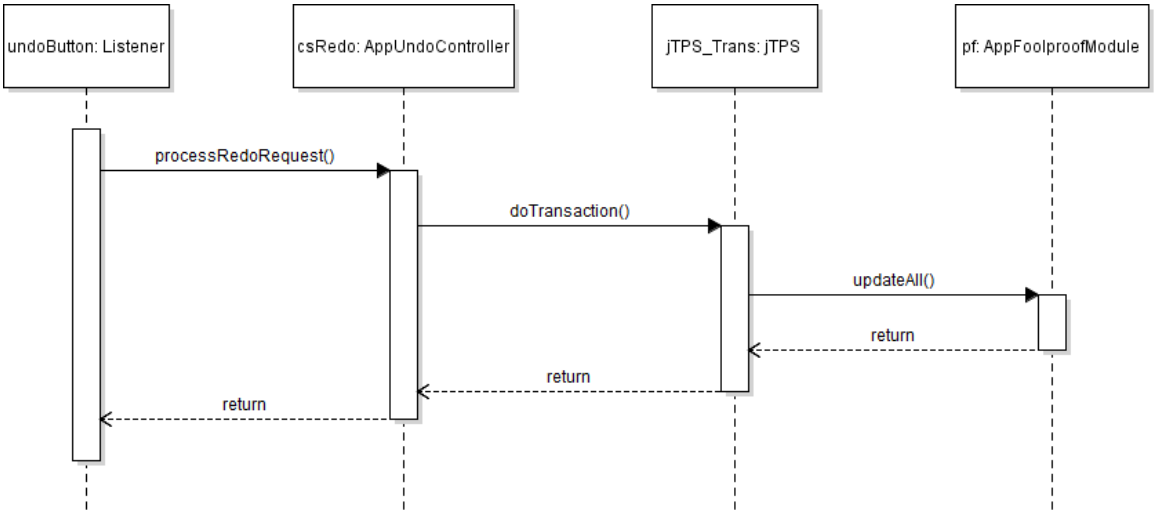
Exit Application UML Sequence Diagram



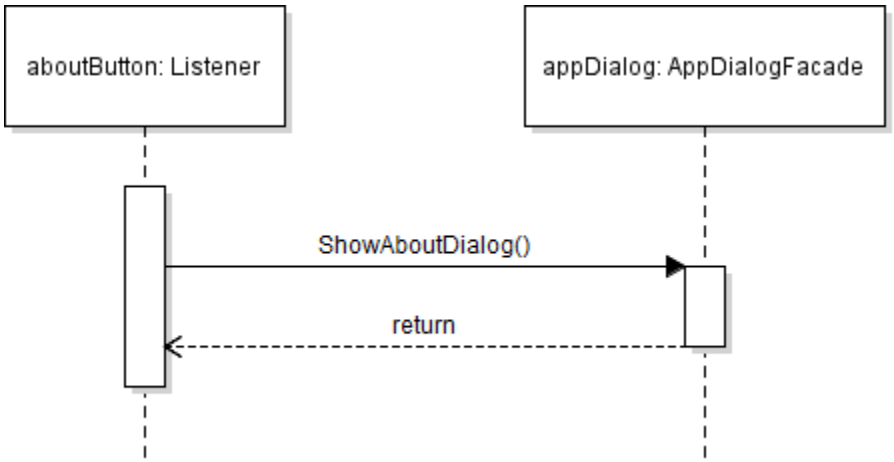
Undo UML Sequence Diagram



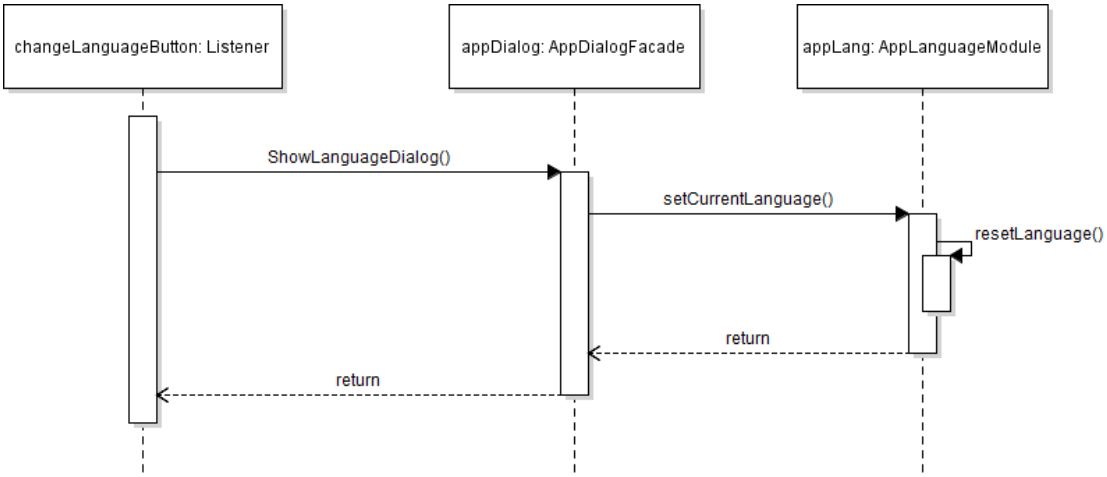
Redo UML Sequence Diagram



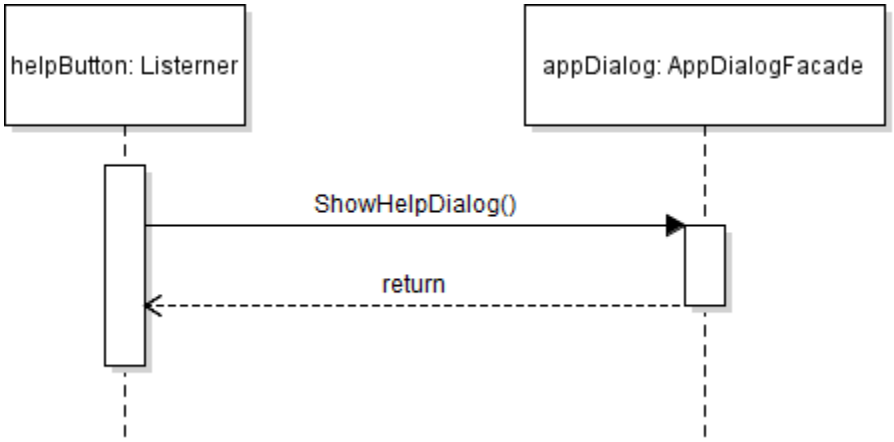
About UML Sequence Diagram



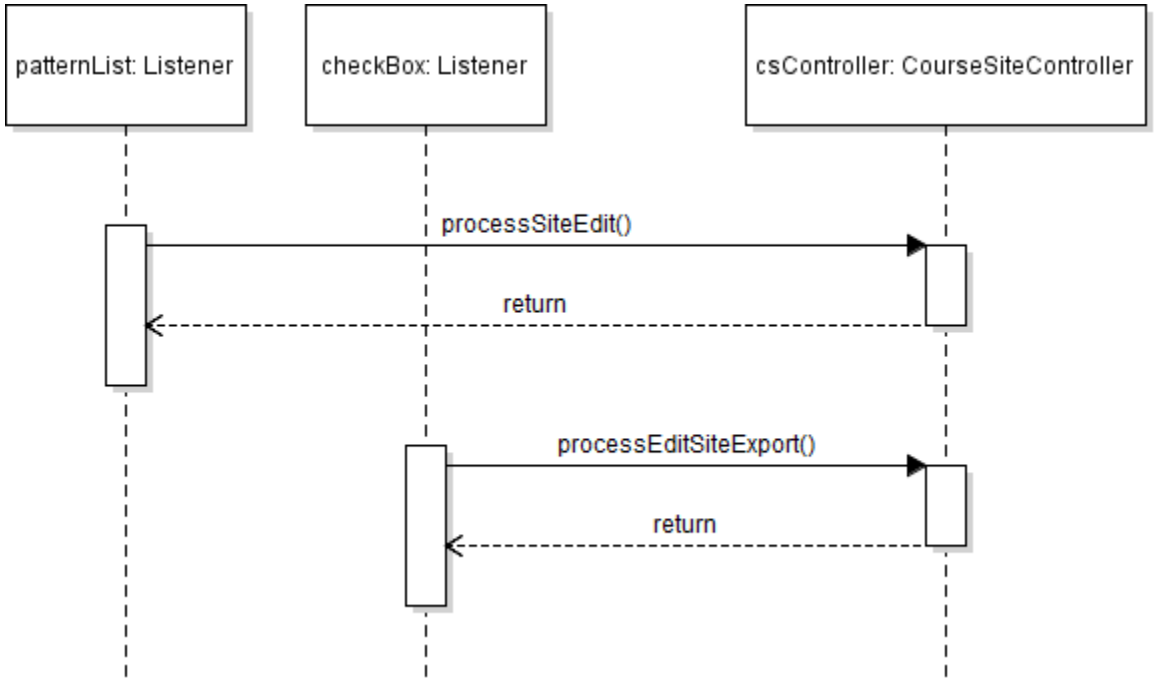
Change Language UML Sequence Diagram



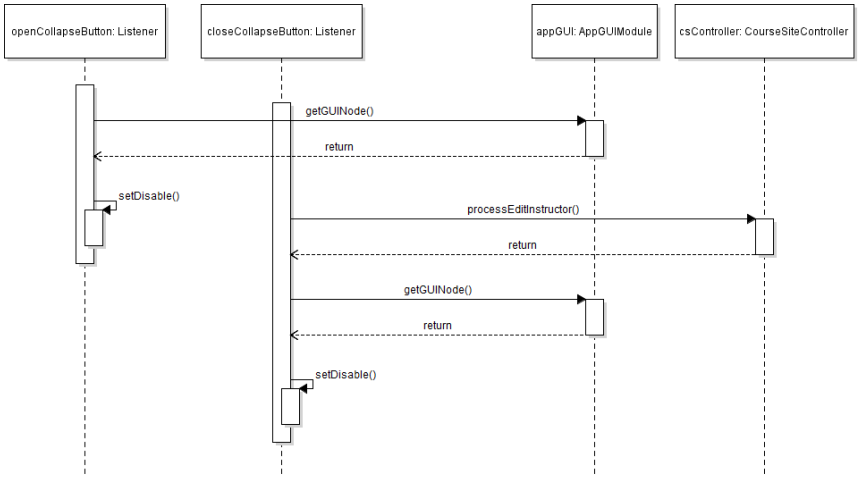
Help UML Sequence Diagram



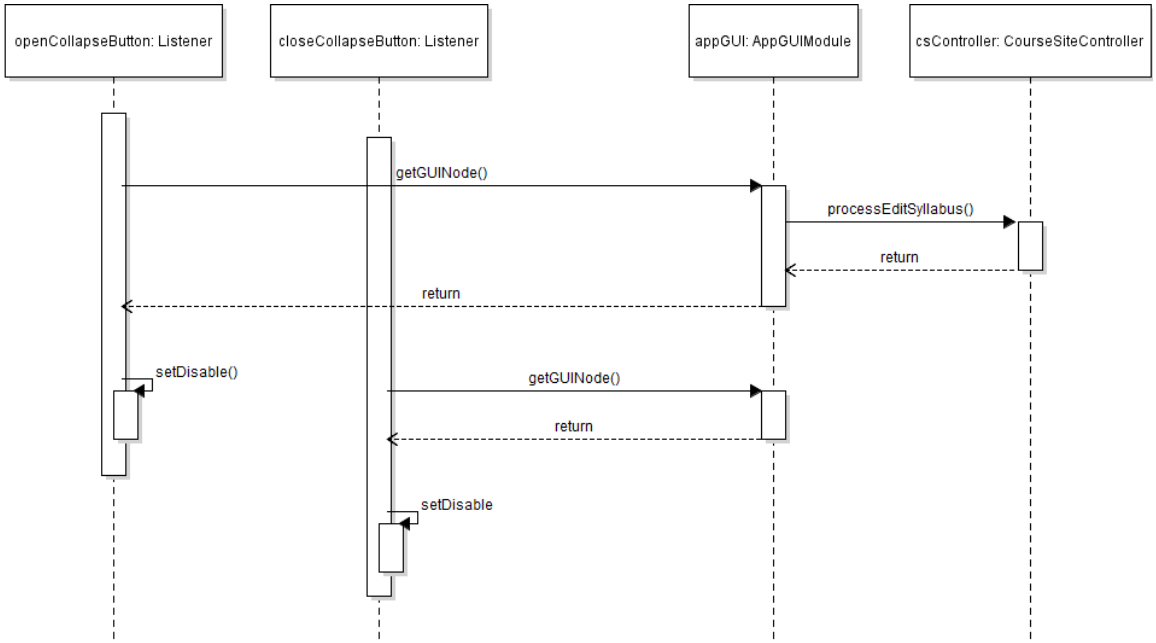
Edit Page Detail UML Sequence Diagram



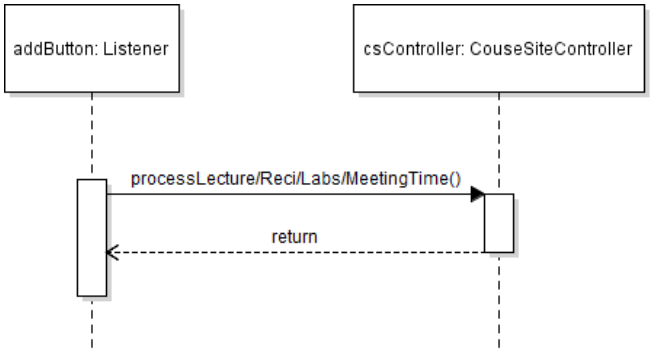
Edit Instructor UML Sequence Diagram



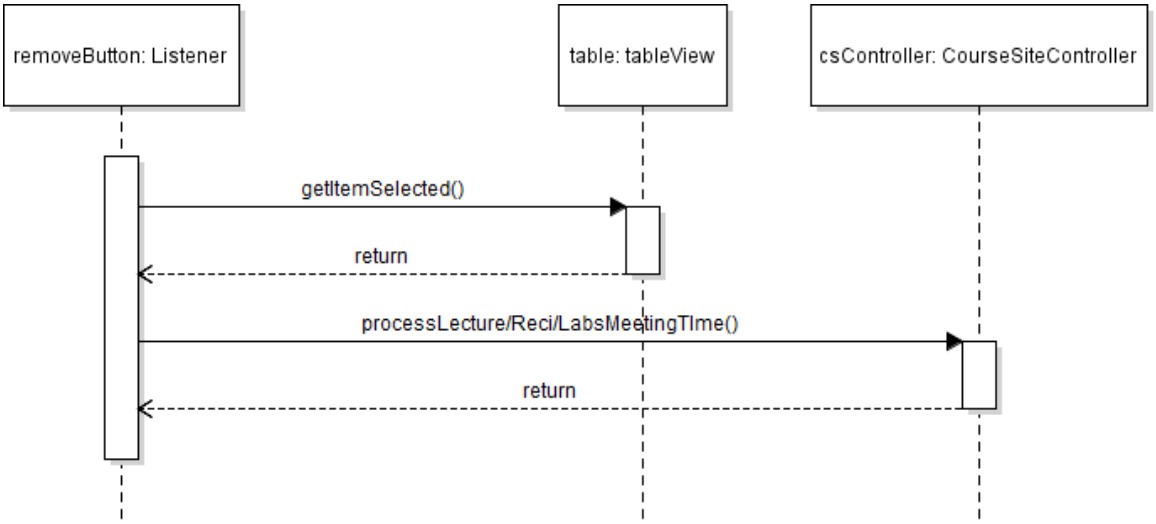
Edit Syllabus Details UML Sequence Diagram



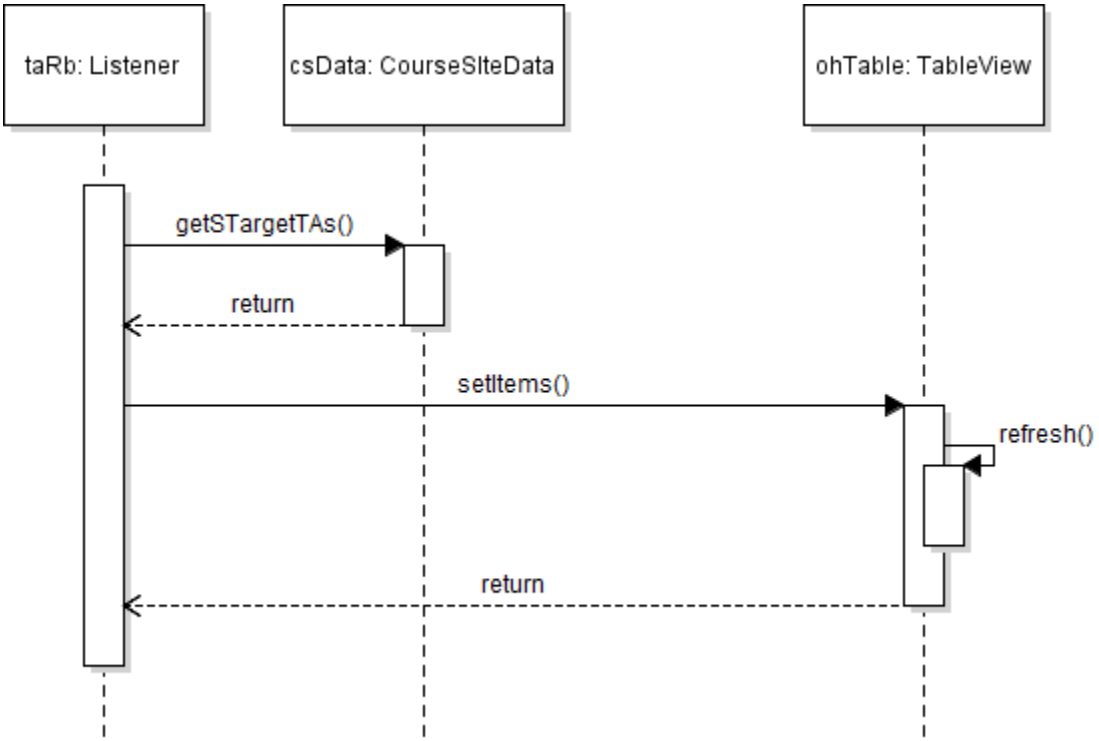
Add Lecture/Recitation/Lab UML Sequence Diagram



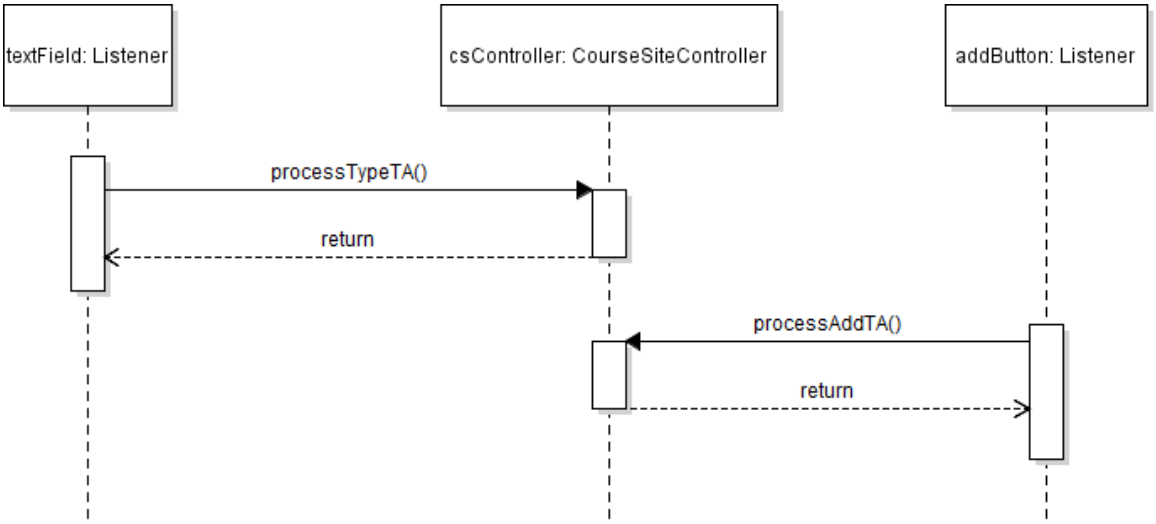
Remove Lecture/Recitation/Lab UML Sequence Diagram



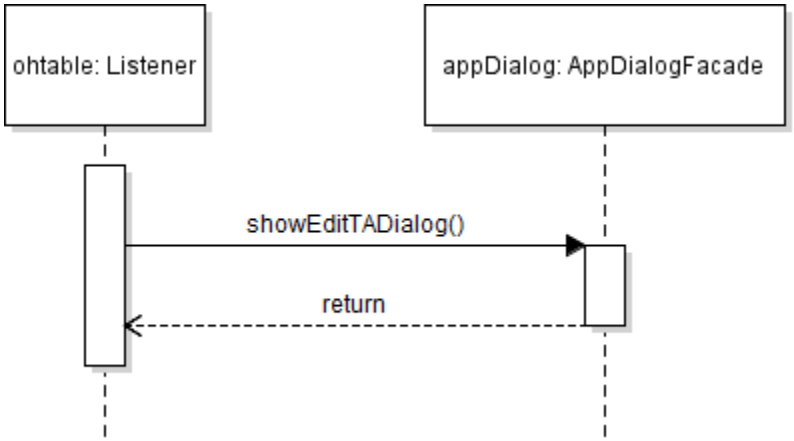
Filter Tas UML Sequence Diagram



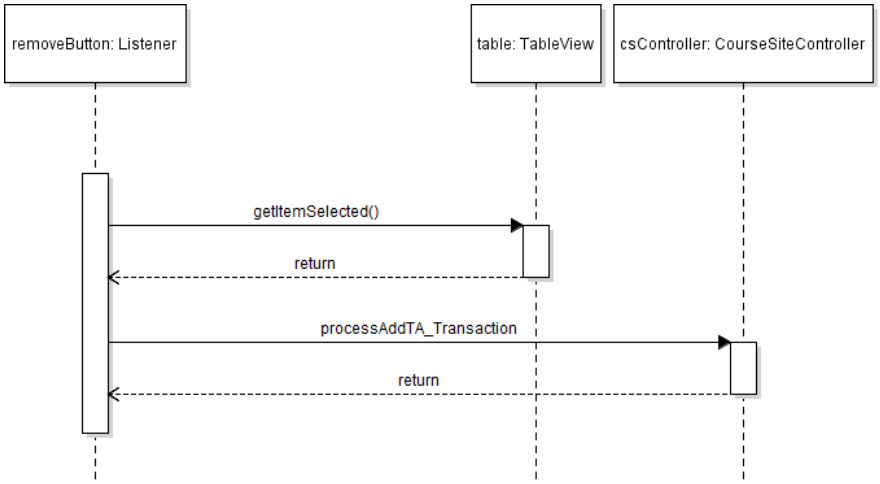
Add TA UML Sequence Diagram



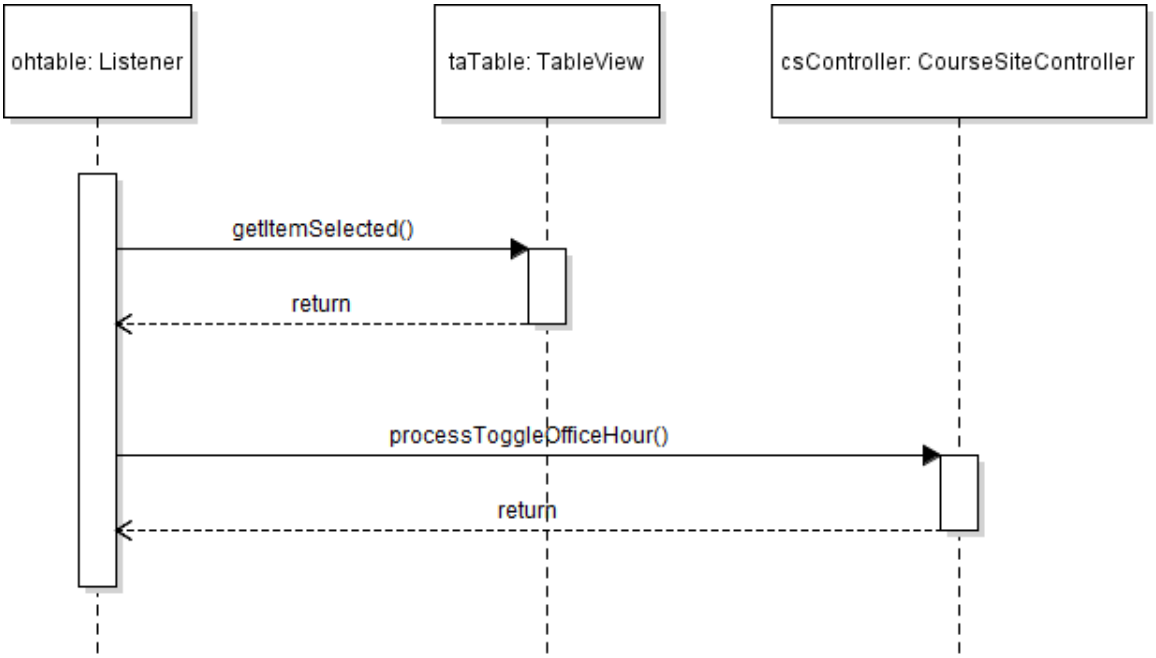
Edit TA UML Sequence Diagram



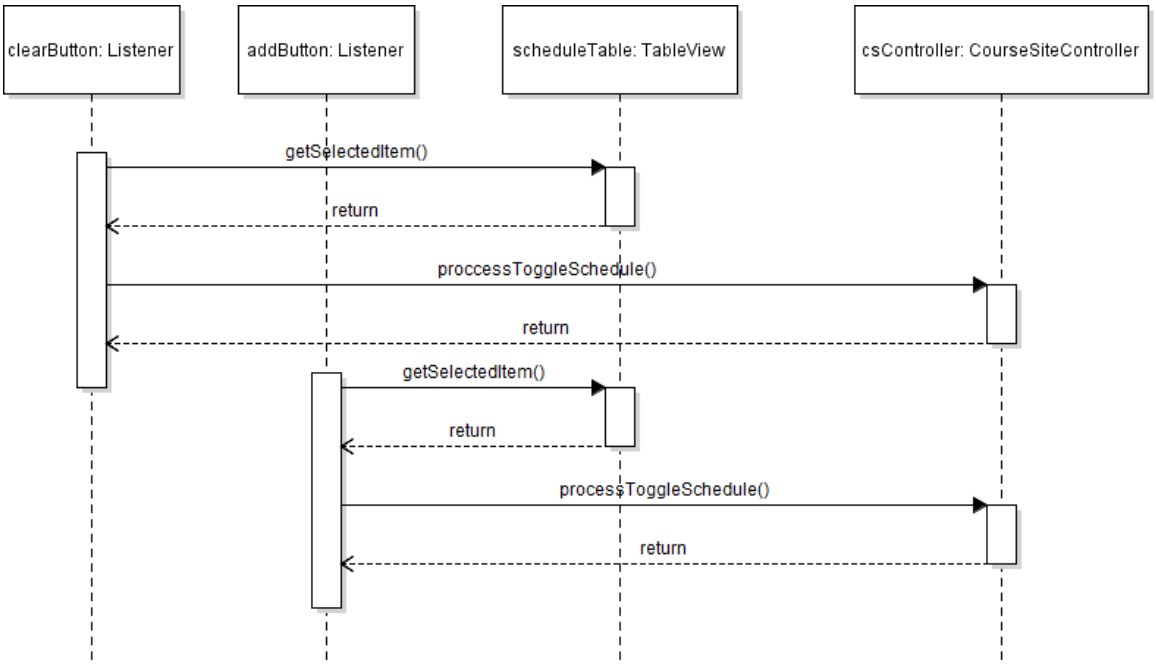
Remove TA UML Sequence Diagram



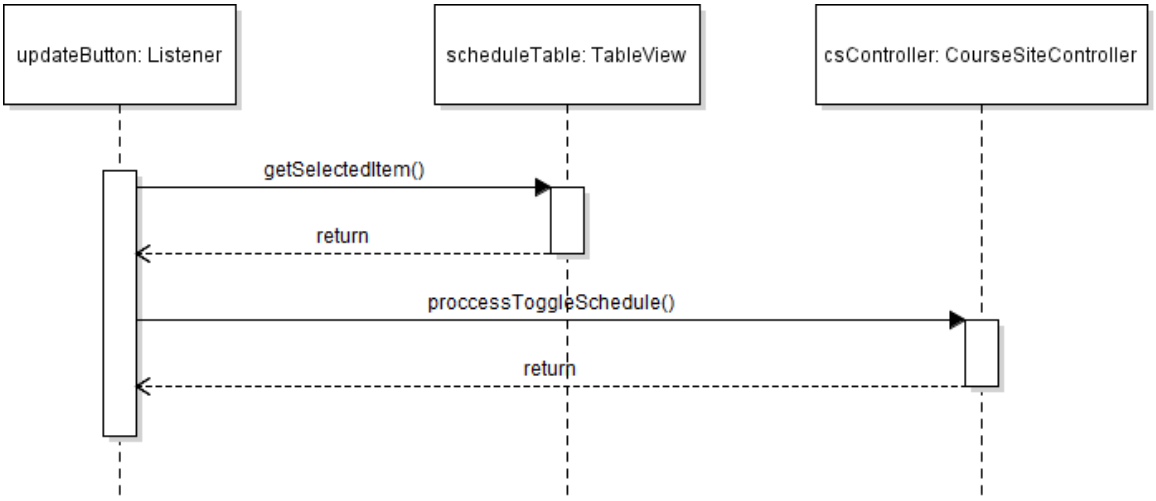
Toggle TA Office Hours UML Sequence Diagram



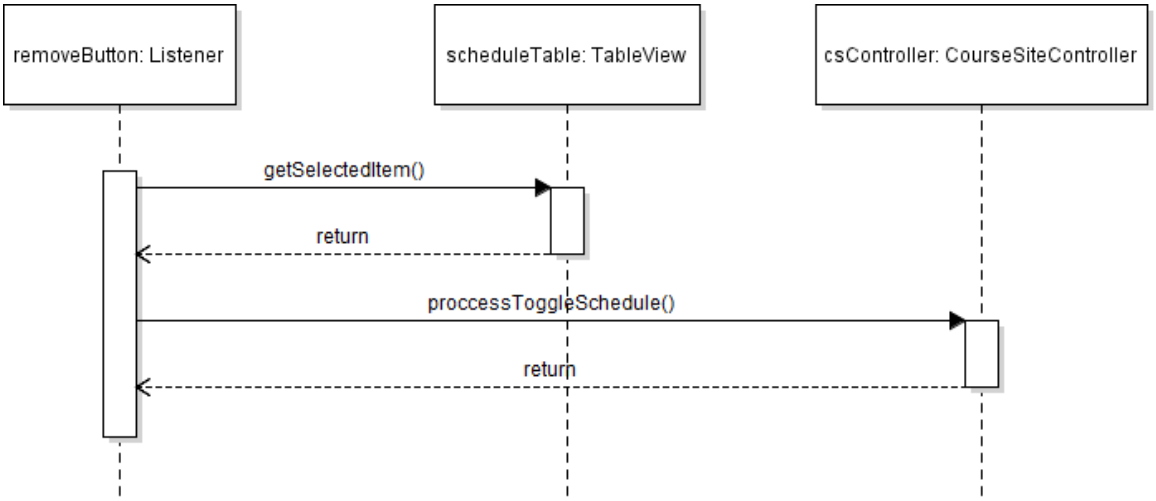
Add Schedule Item UML Sequence Diagram



Edit Schedule Item UML Sequence Diagram



Remove Schedule Item UML Sequence Diagram



5. File Structure and Formats

Note that the Desk Top Framework will be provided inside DesktopFramework.jar, a Java Archive file that will encapsulate the entire framework. This should be imported in to the necessary project for the ***Course Site Generator*** application and will be included in the deployment of a single, executable JAR file titled CourseSiteGenerator.jar. Note that all necessary data and files must accompany this program. Specifies the necessary file structure the launched application should use. Note that all necessary images should of course go in the image directory.

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