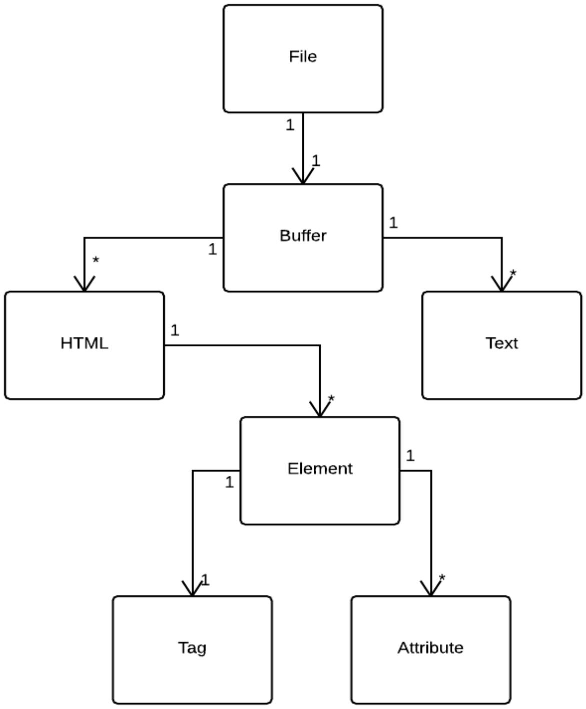
HTML Editor
Release 2

Team C
Braxton Frederick, Andrew Vogler,
Arron Reed, Adam Audycki, Dylan Hogue

Introduction

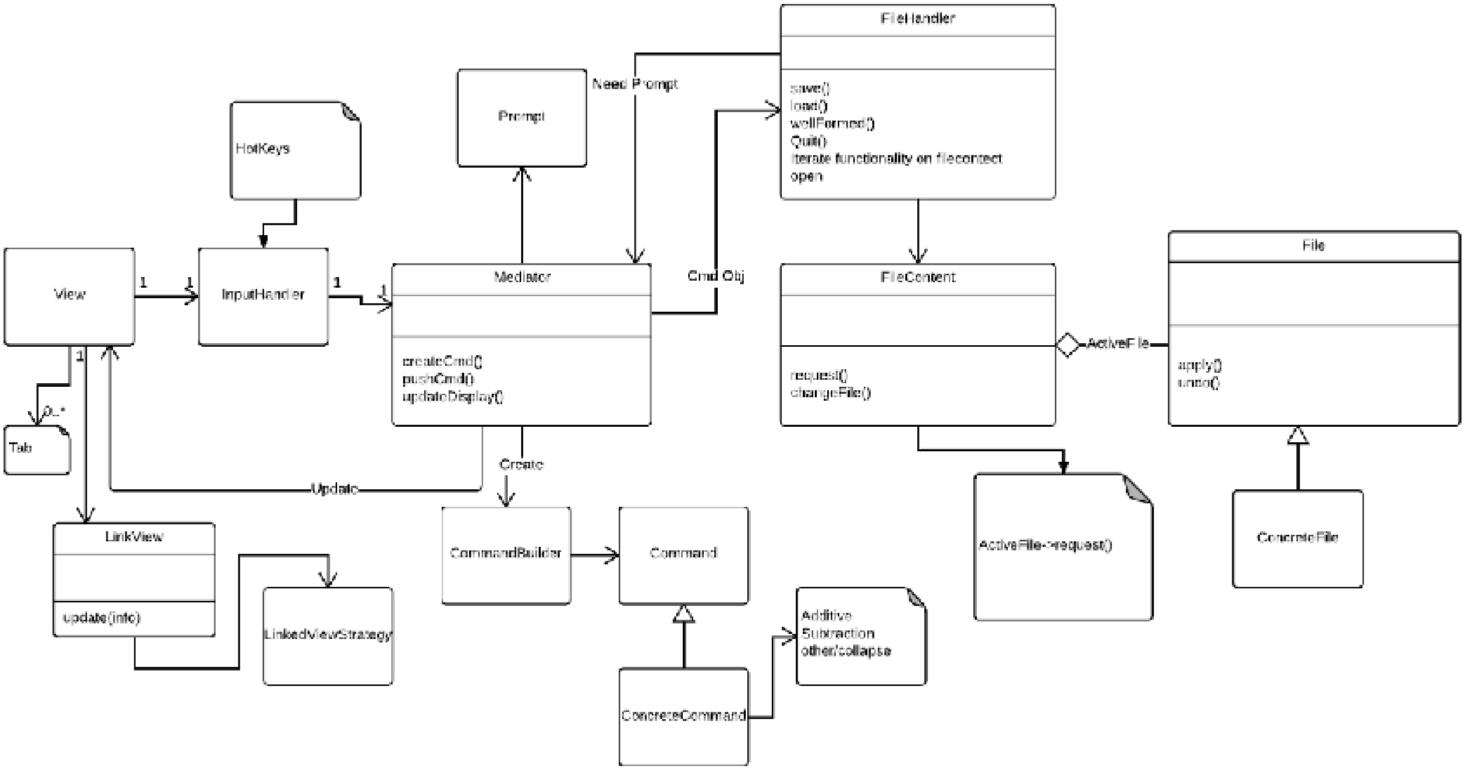
Our team was assigned to design a text editing program for HTML coding, able to open a File and have text or HTML constructs inserted, as well as making said text/constructs editable via keyboard entry or through menu selections that automatically add in changes to the page.

High Level Design



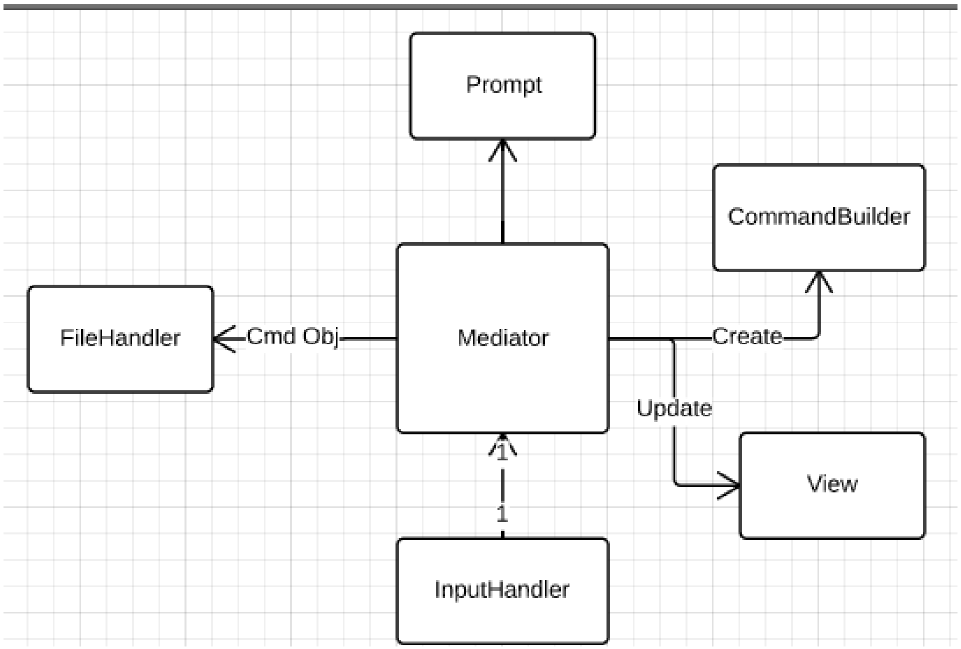
The domain for our project revolves around the actual HTML file, and everything builds off that, starting with the text buffer where all of the changes are displayed. From the buffer, we have the HTML and the text as two separate facets of the domain, with the HTML being further broken down into tags and attributes. This design exemplifies the fact that we have to focus equally on text and HTML in the surface-level functions of our program.

Complete UML System Diagram



Class Interactions

Our first major class is the Mediator, which is used to interpret requests from other classes and receive information for them by telling other classes what methods to run and getting the results. The Mediator has knowledge of five different classes; PromptManager, FileHandler, InputHandler, MainView, and CommandBuilder.



PromptManager is a class that creates page dialogs based on given information from the Mediator.

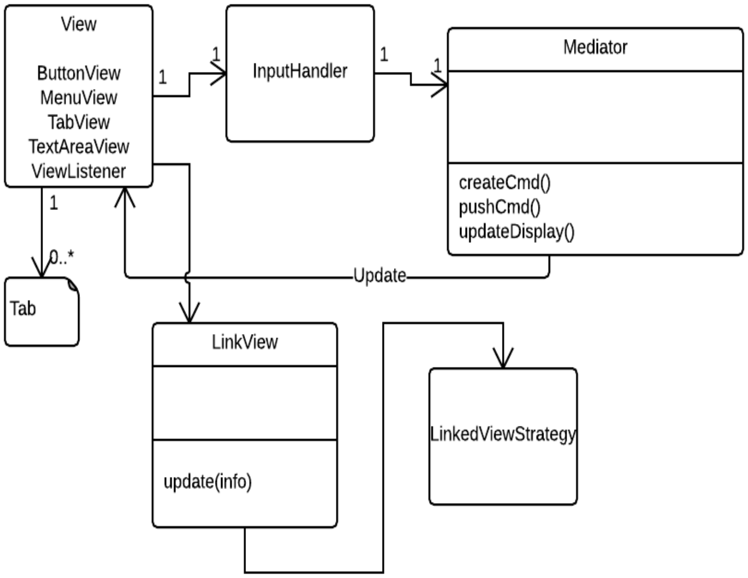
FileHandler is a class which handles the saving and loading of files, along with performing well-formed checks.

InputHandler is another Mediator class, with knowledge of Mediator and our Observable objects. The main purpose of this class is to pass information between Observable and Mediator, in order to reduce the number of things that Mediator is directly tied into.

MainView is a class which contains all of our Observable objects, and is used to display all of our content.

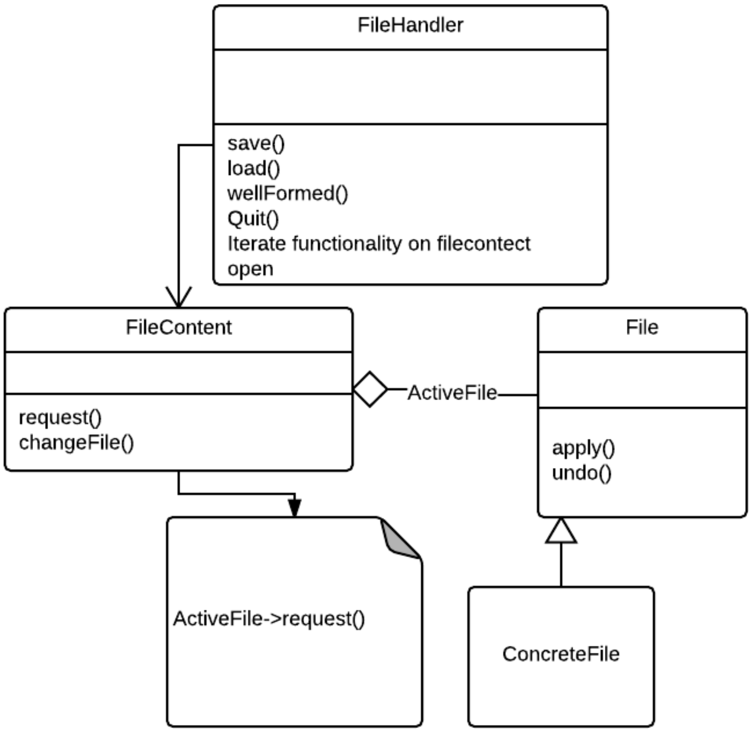
CommandBuilder is a Builder class, with knowledge of Commands. CommandBuilder takes in orders from Mediator, uses the information given to decide what kind of command needs to be run, and tells Command to make it, finally passing the result back to the Mediator.

Command is a Command class, with knowledge of the file class. The purpose of this class is to support Undo/Redo operations, using commands to control the undo/redo content stack, and sending its commands through CommandBuilder and Mediator to be used in the File.



ViewListener is an Observer class, which waits for user input and then sends it to the InputHandler, which acts as the Mediator.

LinkedView is a Strategy class, with knowledge of LinkedViewStrategy and two concrete subclasses. The purpose of these classes is to sort all of the HTML links on the page in a new dialog, and use given information to determine if it should run the SortByAlpha subclass or the SortByAppear subclass to sort the links.

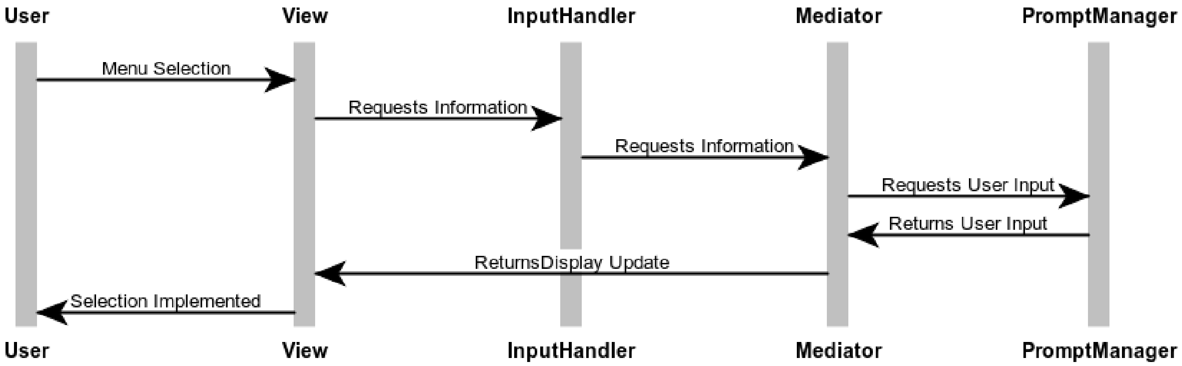


The FileHandler class manages all changes to the File, using FileContent as a dummy file which interprets all of the change requests to File.

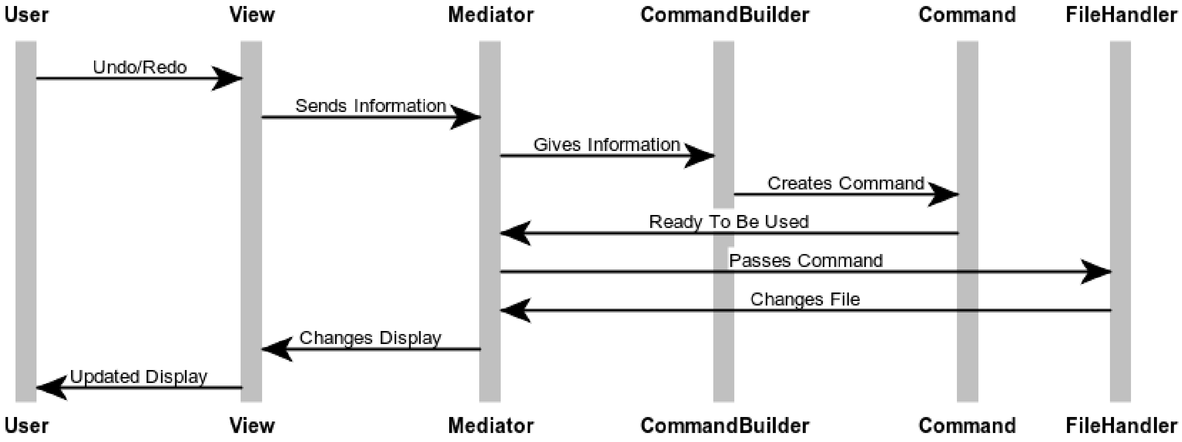
FileContent acts as the current file and stores changes, and contains a switch for changing tabs, upon which the current file is saved before loading up the other tab.

Sequence Diagrams

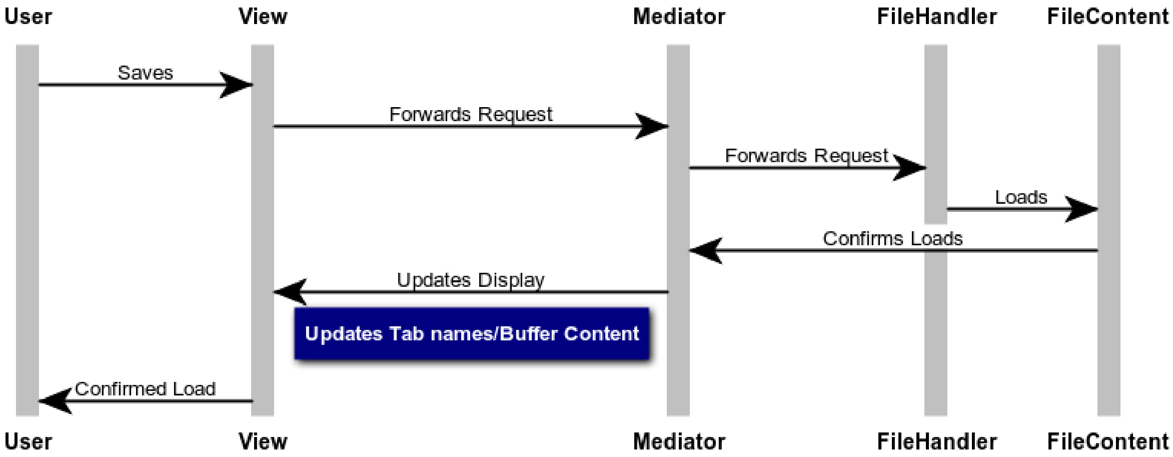
User Presses an Editor Button



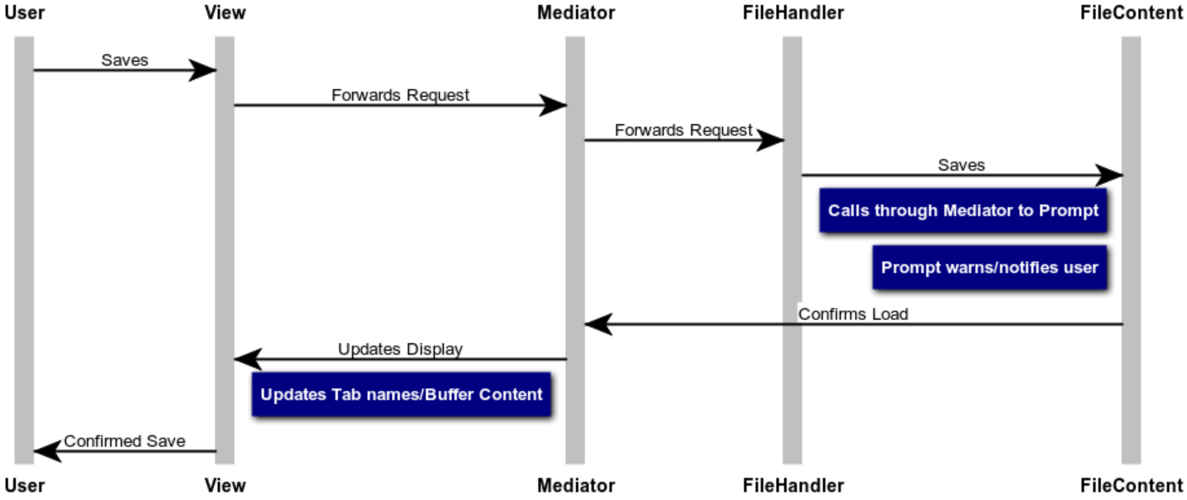
User Uses A Command



User Loads A File



User Saves A File



Implementation State

We feel our implementation completes the given requirements to a satisfactory level. Having spent several meetings on the design alone, we were able to use the following meetings to ensure that we properly implemented the patterns in our code. The only pattern that might not have been used to its full potential was the Strategy pattern, since we only used it to choose between two strategies.

In the final stretch of coding, we were able to work in some of the requirements that were removed from our instructions for R1 and R2 due to our language switch. We were able to implement the following features beyond our revised requirements.

- Reduced functionality if buffer is not well-formed
- Command-line argument for file to open

Conclusion

With this second--final--release of the HTML Editor project, we feel that the implementation of our patterns into our project, and the functionality of the project itself, are in a good place. We spent almost as much time this release on the details of our design and our usage of patterns as we did on the coding. Our system diagram is a testament to this, much more detailed than the R1 diagram, representing every major class interaction in our underlying system. For this release, we implemented five different design patterns. The ViewListener subsystem is an Observer, which we use to update the View classes and trade input information with the rest of the program. The CommandBuilder subsystem is a Builder, which we use to take information on what types of commands to create, and work with Command (a Command class) to create and “distribute” commands throughout the project. The LinkedView subsystem uses Strategy to choose how to sort the URLs present in the file and display them to the user. The Mediator and InputHandler are Mediator classes, used as information hubs, with almost all classes functioning through Mediator, so that they are less dependant on each other and more on just Mediator.

Looking back at the entire project, we feel that we’d have had a much easier time if we had used Java from the start. While Javascript is slightly simpler to implement the functionality in, it was almost impossible to properly use patterns in, and using it for our first release set us back a lot of work for the start of R2, even considering the removed requirements we negotiated. We also agreed that if it was possible, we’d have started this release sooner, and done more research into built in functionalities before diving into code. We had meetings the majority of days between R1 and R2, for several hours each time, and it still felt like a rush to the finish. This wasn’t helped by a couple instances of us trying to implement something and then shortly after realizing there was a simpler built in way to do it, leading us to scrapping code and implementing the simpler way.

Pattern Usage Tables

Name: Mediator	**GoF:** Mediator	
Class	**Role In Pattern**	**Description**
Mediator	Mediator	Breaks up subsystems and interprets information for them
FileHandler	Colleague	Gets information from Mediator for updating File
FileContent	Colleague	Gets information from Mediator for updating File
InputHandler	Colleague	Gets information from Mediator for updating View
CommandBuilder	Colleague	Gets information from Mediator for creating commands

Name: InputHandler	**GoF:** Mediator	
Class	**Role In Pattern**	**Description**
InputHandler	Mediator	Takes information from Observers, sends to Mediator
Observable	Colleague	Has information for InputHandler to use and trade with Mediator
Mediator	Colleague	Interprets information from Observable and sends it back to update the View

Name: Command	**GoF:** Command	
Class	**Role In Pattern**	**Description**
Command	Command	Used to edit the File after being sent to FileContent through Mediator
CommandBuilder	Client	Creates the commands
Mediator	Invoker	Gives instructions for commands to be created
FileContent	Reciever	Commands are sent to FileContent to be used to update the File

Name: ViewListener	**GoF:** Observer	
Class	**Role In Pattern**	**Description**
ViewListener	Observer	Gets updates from View, sends them through InputHandler and Mediator, and pushes updates to View
View	Subject	Encapsulates all of the View classes
MenuView	ConcreteSubject	Handles selection from menu bar
ButtonView	ConcreteSubject	Handles clicking on buttons
BufferView	ConcreteSubject	Handles the text area
TabView	ConcreteSubject	Handles switching file tabs

Name: CommandBuilder	**GoF:** Builder	
Class	**Role In Pattern**	**Description**
CommandBuilder	Builder	Takes information and creates a Command to send through Mediator
Command	Product	Used for Undo/Redo operations and other File edit operations
Mediator	Director	Tells the Builder what Command to create, sends Command to where it’s needed

Name: LinkedView	**GoF:** Strategy	
Class	**Role In Pattern**	**Description**
LinkedView	Context	Displays all URLs in the file, uses Strategy to decide how to display
LinkedViewStrategy	Strategy	Forwards information from the Context and chooses the correct ConcreteStrategy
SortByAlpha	ConcreteStrategy	Sorts URLs alphabetically
SortByAppear	ConcreteStrategy	Sorts URLs in order of appearance

CRC Cards

Class: AdditiveCommand
Responsibilities: Applies or undoes the addition of text into the buffer.
Uses: Command
Used By: CommandBuilder, Mediator, File, FileContent, FileHandler
Authors: Braxton Frederick, Andrew Vogler

Class: BtnView
Responsibilities: Contains methods to display buttons and run attached functions.
Uses: ViewListener
Used By: MainView, ViewListener
Authors: Dylan Hogue, Andrew Vogler

Class: Command
Responsibilities: An abstract class that provides structure for undo/redo operations.
Uses: N/A
Used By: File, FileContent, FileHandler, Mediator, ConcreteCommands
Authors: Braxton Frederick

Class: CommandBuilder
Responsibilities: Handles the creation of commands.
Uses: Command, PromptManager
Used By: Mediator
Authors: Braxton Frederick

Class: ErrorCommand
Responsibilities: Overrides Command if something is not Undoable.
Uses: Command
Used By: CommandBuilder, Mediator, File, FileContent, FileHandler
Authors: Andrew Vogler

Class: File
Responsibilities: Contains getters and setters for File information, and applies Commands to the File object.
Uses: Command
Used By: FileContent, FileHandler, Command
Authors: Andrew Vogler, Braxton Frederick

Class: FileContent
Responsibilities: Runs Undo commands and switches File being viewed.
Uses: File
Used By: FileHandler
Authors: Andrew Vogler, Braxton Frederick

Class: FileHandler
Responsibilities: Runs save, load, and open commands, and the well-formed check.
Uses: FileContent, File, PromptManager
Used By: Mediator
Authors: Braxton Frederick, Adam Audycki

Class: InputHandler
Responsibilities: Handles events fired from button presses or menu selections.
Uses: Mediator
Used By: View subsystems
Authors: Braxton Frederick, Andrew Vogler

Class: InsertImageCommand
Responsibilities: Overrides undo/redo to insert/remove an Image URL.
Uses: Command
Used By: CommandBuilder, Mediator, File, FileContent, FileHandler
Authors: Adam Audycki

Class: InsertLinkCommand
Responsibilities: Overrides undo/redo to insert/remove a URL.
Uses: Command
Used By: CommandBuilder, Mediator, File, FileContent, FileHandler
Authors: Adam Audycki, Braxton Frederick

Class: InsertListCommand
Responsibilities: Overrides undo/redo to insert/remove a List structure.
Uses: Command
Used By: CommandBuilder, Mediator, File, FileContent, FileHandler
Authors: Adam Audycki, Braxton Frederick

Class: InsertTableCommand
Responsibilities: Overrides undo/redo to insert/remove Table structures.
Uses: Command
Used By: CommandBuilder, Mediator, File, FileContent, FileHandler
Authors: Braxton Frederick

Class: InsertTagCommand
Responsibilities: Overrides undo/redo to insert/remove an HTML tag.
Uses: Command
Used By: CommandBuilder, Mediator, File, FileContent, FileHandler
Authors: Adam Audycki, Braxton

Class: LinkedView
Responsibilities: Creates a View for the LinkView display.
Uses: LinkedViewStrategy, ViewListener
Used By: ViewListener, MainView
Authors: Adam Audycki, Andrew Vogler

Class: LinkedViewStrategy
Responsibilities: Parses all of the URLs in the file to be displayed.
Uses: SortByAlpha, SortByAppear
Used By: LinkedView
Authors: Adam Audycki

Class: MainFile
Responsibilities: Starts the main program.
Uses: MainView, MenuView, Mediator, InputHandler
Used By: N/A
Authors: Andrew Vogler, Dylan Hogue, Braxton Frederick

Class: MainView
Responsibilities: Creates the main window, buttons, menus, and buffer.
Uses: InputHandler, View subsystems
Used By: MainFile, Mediator, ViewListener, InputHandler
Authors: Andrew Vogler, Braxton Frederick

Class: Mediator
Responsibilities: Sends commands and information between classes.
Uses: CommandBuilder, FileHandler, InputHandler, MainView, PromptManager
Used By: InputHandler, FileHandler
Authors: Adam Audycki, Braxton Frederick, Andrew Vogler

Class: MenuView
Responsibilities: Methods to display and run drop down menu functions.
Uses: ViewListener
Used By: MainView, ViewListener
Authors: Dylan Hogue, Andrew Vogler

Class: PromptManager
Responsibilities: Creates pop up prompts for getting user input or giving warnings.
Uses: N/A
Used By: Mediator, CommandBuilder, FileHandler
Authors: Andrew Vogler, Braxton Frederick, Arron Reed

Class: SortByAlpha
Responsibilities: ConcreteStrategy for LinkView that sorts URLs alphabetically.
Uses: N/A
Used By: LinkedViewStrategy
Authors: Adam Audycki, Andrew Vogler

Class: SortByAppear
Responsibilities: ConcreteStrategy for LinkView that sorts URLs by appearance.
Uses: N/A
Used By: LinkedViewStrategy
Authors: Adam Audycki

Class: SubtractiveCommand
Responsibilities: Applies or undoes the subtraction of text from the buffer.
Uses: Command
Used By: CommandBuilder, Mediator, File, FileContent, FileHandler
Authors: Braxton Frederick

Class: TabView

Responsibilities: Creates tabs in main window used to switch between open files.

Uses: ViewListener, MainView, TextAreaView

Used By: MainView, ViewListener

Authors: Andrew Vogler, Braxton Frederick

Class: TextAreaView

Responsibilities: Handles changes to the buffer.

Uses: ViewListener

Used By: MainView, ViewListener

Authors: Andrew Vogler

Class: ViewListener

Responsibilities: Receives updates from the View and sends them to Mediator.

Uses: InputHandler

Used By: View subsystems, Mediator

Authors: Dylan Hogue, Andrew Vogler

Class: ImagePreviewer

Responsibilities: Displays user selected image from list of all in file.

Uses: N/A

Used By: MainView, InputHandler

Authors: Dylan Hogue, Andrew Vogler

Class: FormatHelper
Responsibilities: Format HTML to follow indentation standards on button press.
Uses: MainView, TextAreaView
Used By: MainView
Authors: Dylan Hogue, Andrew Vogler

Note: Classes which have other classes passed through them are not counted as using said classes.