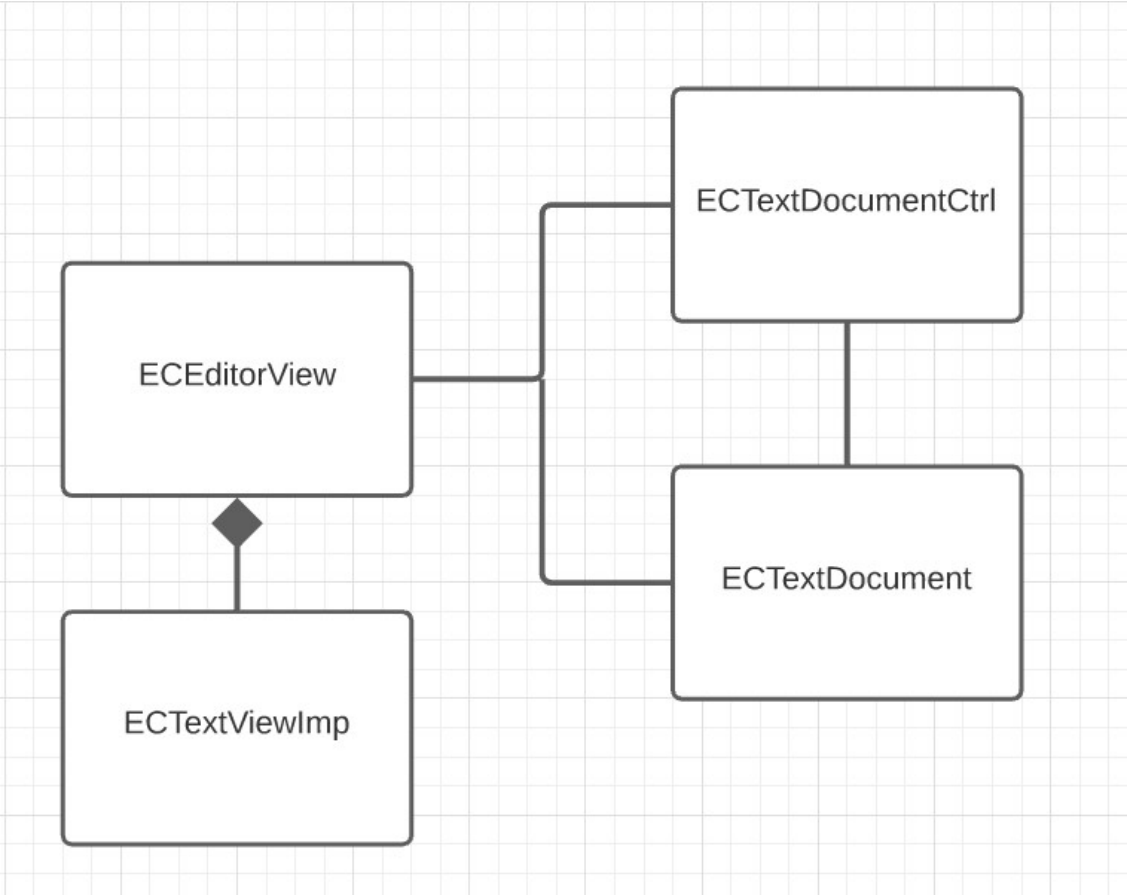
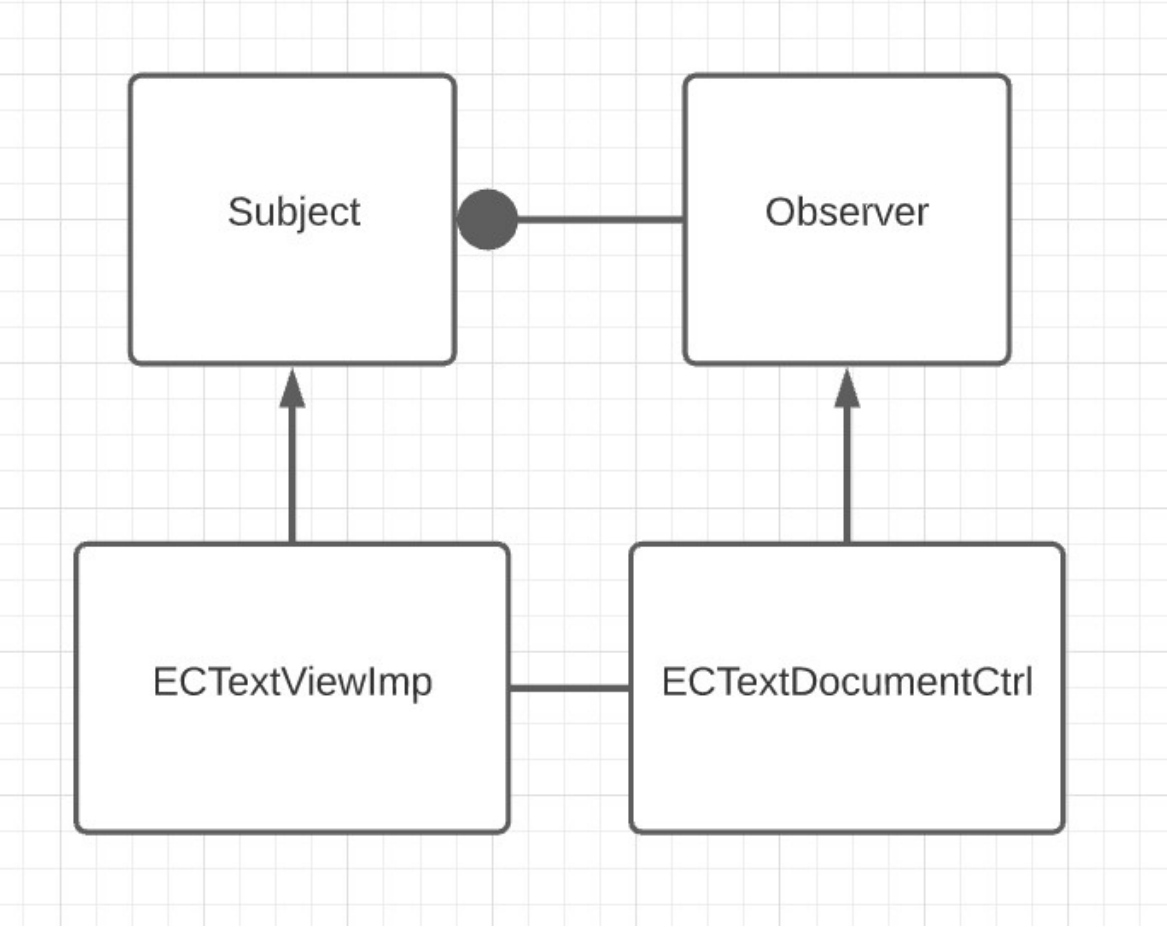
Shahawn Hussain

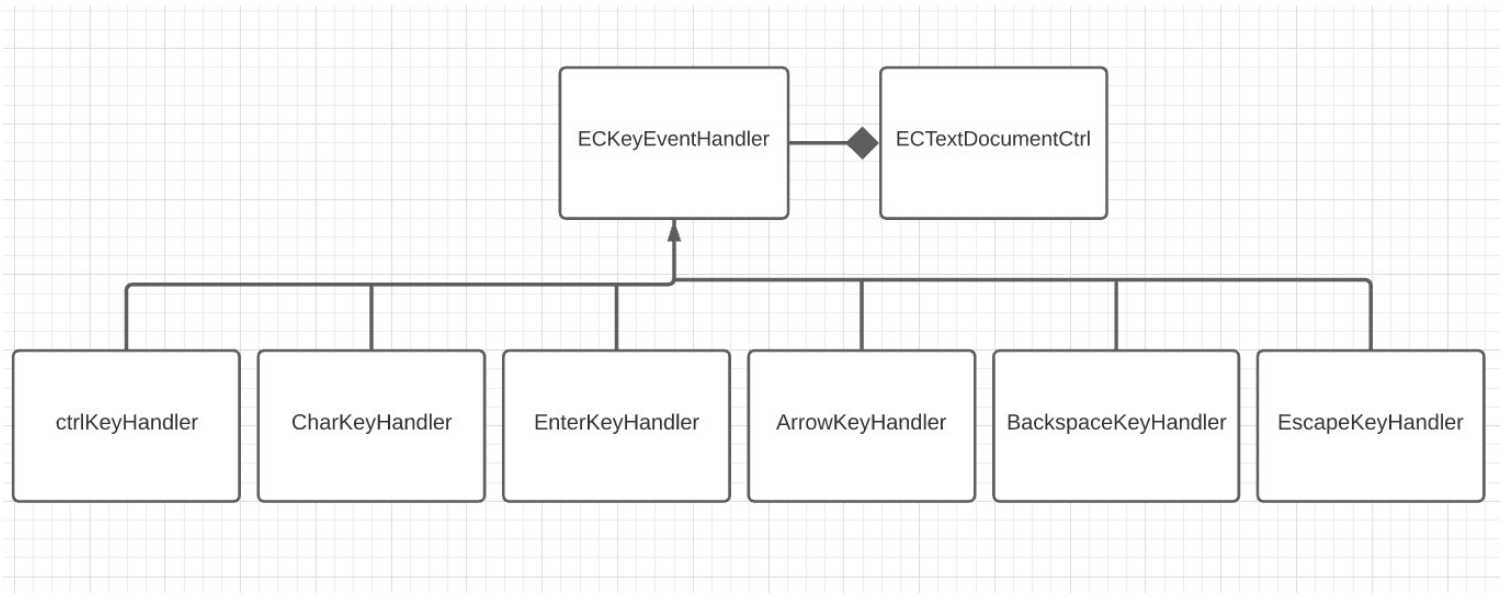
CSE 3150 Project Report



Model-View-Controller: The Model-View-Controller design pattern splits the application into three components: the model, which holds and maintains the data; the view, which displays the data to the user; and the controller, which controls interactions between the model and the view. In my code, ECTextDocument is the model and it contains a vector of strings to hold the text in the document. It also, has functions for inserting and deleting text. ECEditorView is the view, it has an instance of ECTextViewImp and is responsible for displaying the text and cursor. ECTextDocumentCtrl is the controller and it contains an instance of both ECTextDocument and ECEditorView. It contains functions that call the document to insert or remove text, as well as functions to control where the cursor should go.

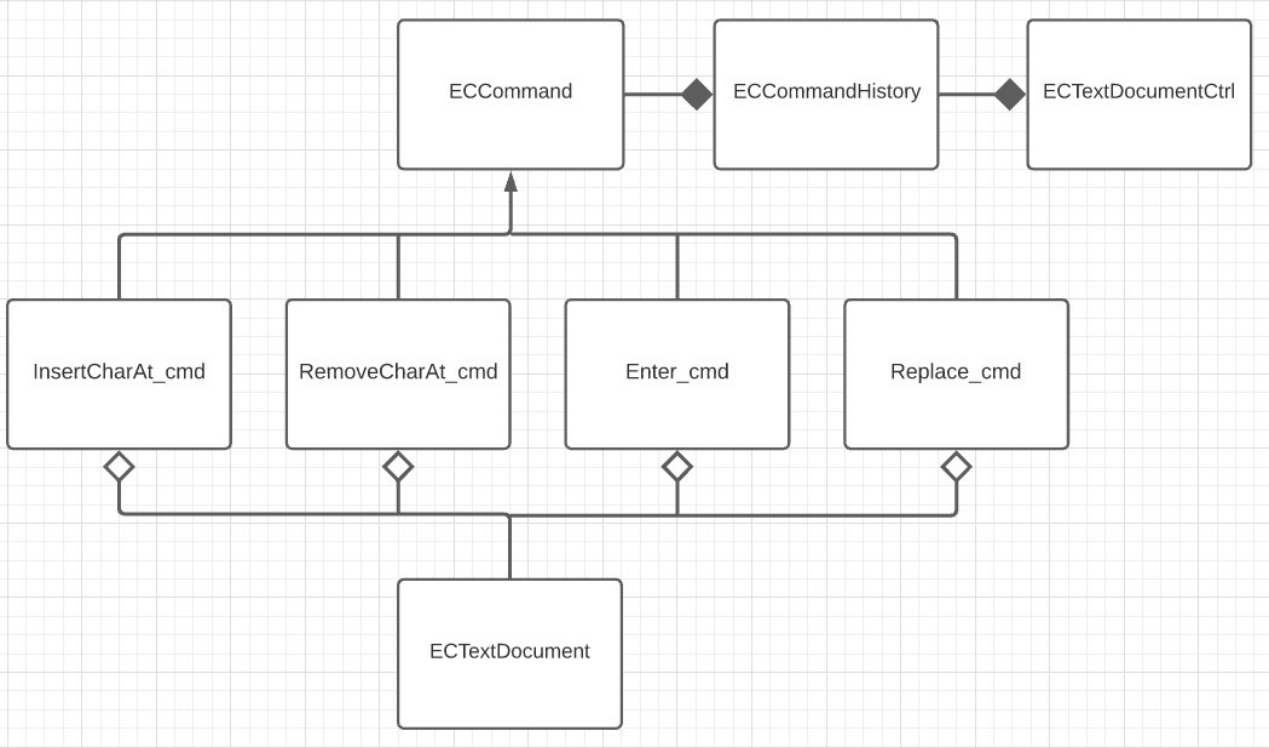


Observer: The observer pattern is a design pattern in which an object, named the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods. In my code, ECTextViewImp is the subject and ECTextDocumentCtrl is the observer. ECTextDocumentCtrl attaches itself to the list of observers in ECTextViewImp, in ECEditorTest. ECTextViewImp waits for a key to be pressed and notifies its observers, which is only ECTextDocumentCtrl in this case. Once it has been notified, ECTextDocumentCtrl calls the function Update(), where it takes in the key that was pressed and sends it over to a key handler.



Chain of Responsibility:

The chain of responsibility pattern creates a chain of receiver objects for a request. This pattern decouples sender and receiver of a request based on type of request. If one object cannot handle the request then it passes the same to the next receiver and so on. In my code, I have six different key handlers, as seen above, which can only handle certain keys. If the key that was pressed can not be handled by the current handler, it passes the key on to the next handler. I've also made it so the current handler also passes the mode that it is in for Find and Replace. If the key can be handled, the handler will call a specific function in ECTextDocumentCtrl. The controller also holds an instance of the first key handler in line, which is the BackspaceKeyHandler in my case.



Command: The command pattern is a behavioral design pattern in which an object is used to encapsulate all information needed to perform an action or trigger an event at a later time. In our case we use this pattern, so we can have undo/redo functionality. Anything that updates the document is made into a command, this includes inserting text, removing text, moving text (Enter), and replacing text. Each command has a reference to ECTextDocument, so that it can edit the document. The ECCommandHistory class contains an undoStack and a redoStack, which contains these commands. When a command is called, it is added to the undoStack and when it is undone, it goes into the redoStack. ECTextDocumentCtrl contains an instance of ECComandHistory called histCmds.