SOFTWARE DEVELOPMENT II PROJECT

OVERALL REPORT

DIMITROPOULOS DIMITRIOS

POULOS GRIGORIOS

TABLE OF CONTENTS

INTRODUCTION

The main objective of this project was to understand the given legacy code and then refactor it to solve various code related problems. This process helped us understand the value of clean code and how such code can be easily extended.

REFACTORED DESIGN

ARCHITECTURE

- **Package Diagram**

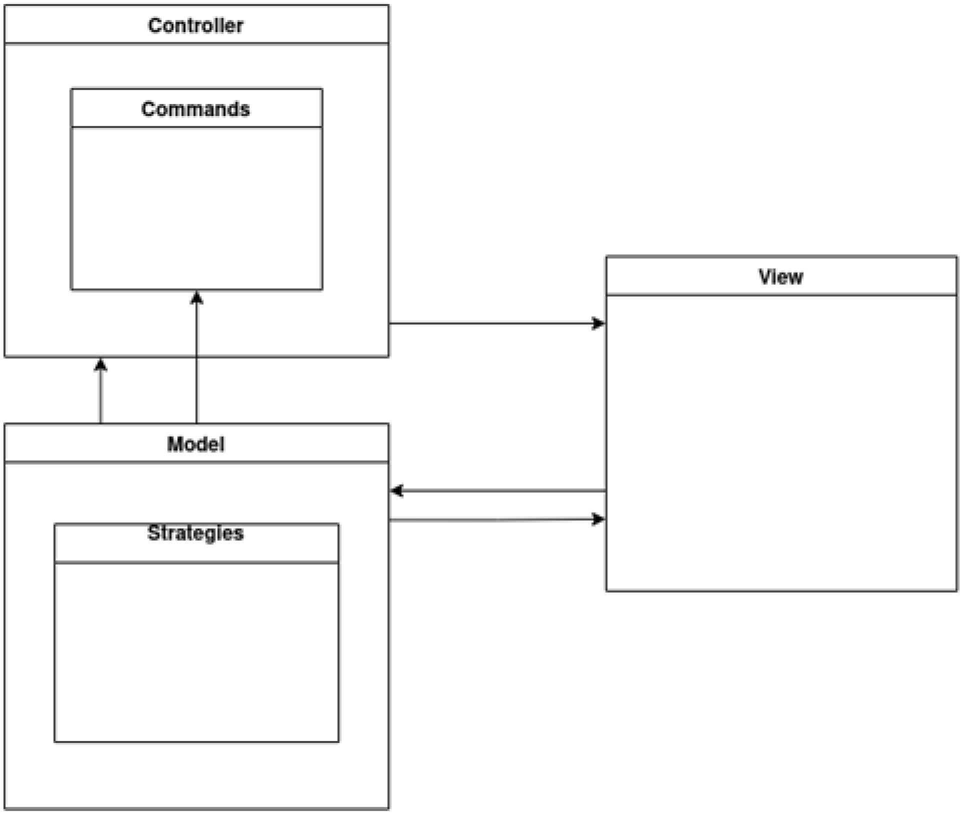
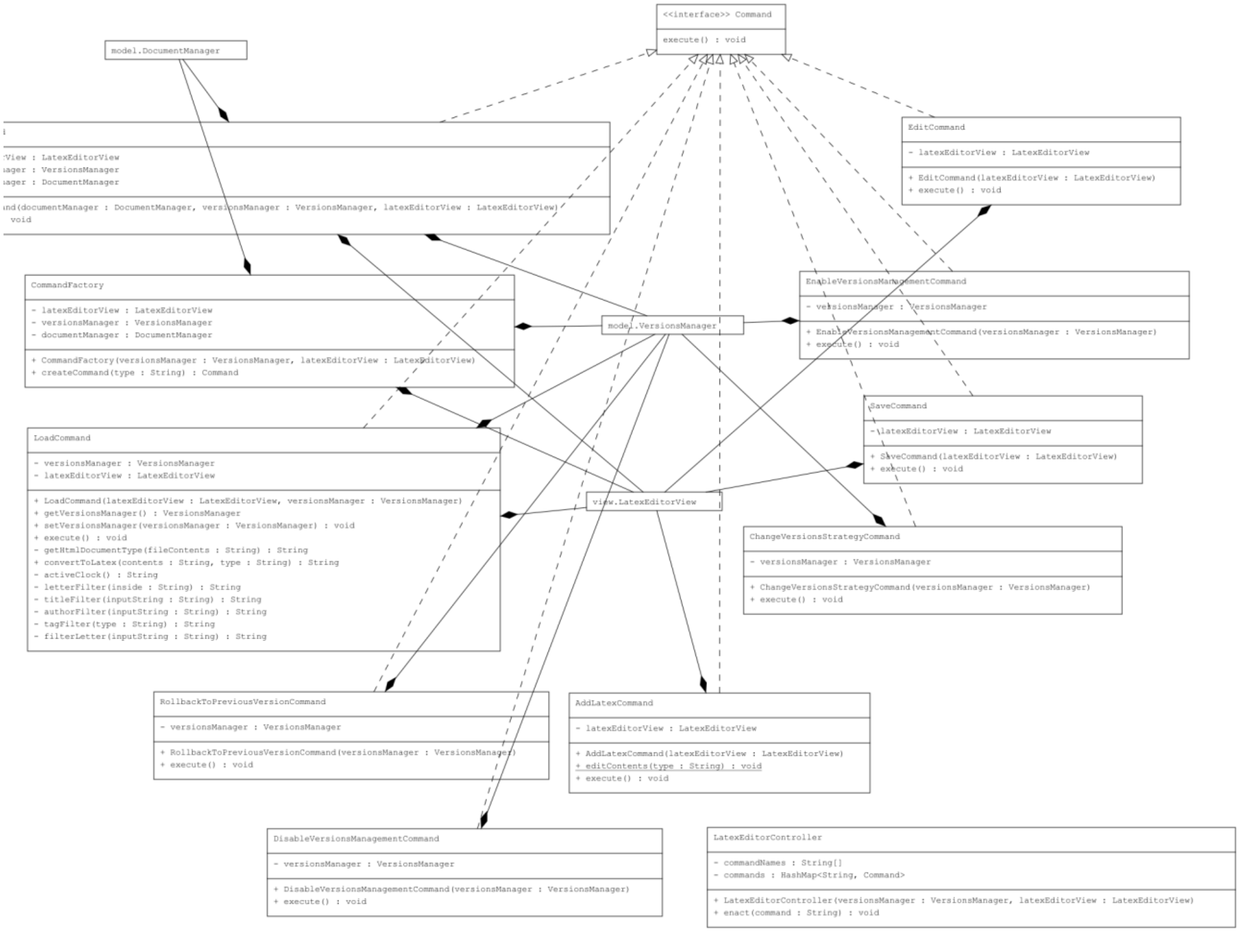


Figure 1

DETAILED DESIGN

- **Controller Package Class Diagram**



<<interface>> Command
execute() : void
model.DocumentManager
:View : LatexEditorView
ager : VersionsManager
ager : DocumentManager
nd(documentManager : DocumentManager, versionsManager : VersionsManager, latexEditorView : LatexEditorView)
void
EditCommand
- latexEditorView : LatexEditorView
+ EditCommand(latexEditorView : LatexEditorView)
+ execute() : void
CommandFactory
- latexEditorView : LatexEditorView
- versionsManager : VersionsManager
- documentManager : DocumentManager
+ CommandFactory(versionsManager : VersionsManager, latexEditorView : LatexEditorView)
+ createCommand(type : String) : Command
model.VersionsManager
EnableVersionsManagementCommand
- versionsManager : VersionsManager
+ EnableVersionsManagementCommand(versionsManager : VersionsManager)
+ execute() : void
SaveCommand
- latexEditorView : LatexEditorView
+ SaveCommand(latexEditorView : LatexEditorView)
+ execute() : void
LoadCommand
- versionsManager : VersionsManager
- latexEditorView : LatexEditorView
+ LoadCommand(latexEditorView : LatexEditorView, versionsManager : VersionsManager)
+ getVersionsManager() : VersionsManager
+ setVersionsManager(versionsManager : VersionsManager) : void
+ execute() : void
- getHtmlDocumentType(fileContents : String) : String
+ convertToLatex(contents : String, type : String) : String
- activeClock() : String
- letterFilter(inside : String) : String
- titleFilter(inputString : String) : String
- authorFilter(inputString : String) : String
- tagFilter(type : String) : String
- filterLetter(inputString : String) : String
view.LatexEditorView
ChangeVersionsStrategyCommand
- versionsManager : VersionsManager
+ ChangeVersionsStrategyCommand(versionsManager : VersionsManager)
+ execute() : void
RollbackToPreviousVersionCommand
- versionsManager : VersionsManager
+ RollbackToPreviousVersionCommand(versionsManager : VersionsManager)
+ execute() : void
AddLatexCommand
- latexEditorView : LatexEditorView
+ AddLatexCommand(latexEditorView : LatexEditorView)
+ editContents(type : String) : void
+ execute() : void
DisableVersionsManagementCommand
- versionsManager : VersionsManager
+ DisableVersionsManagementCommand(versionsManager : VersionsManager)
+ execute() : void
LatexEditorController
- commandNames : String[]
- commands : HashMap<String, Command>
+ LatexEditorController(versionsManager : VersionsManager, latexEditorView : LatexEditorView)
+ enact(command : String) : void

Figure 2

- **Model Package Class Diagram**

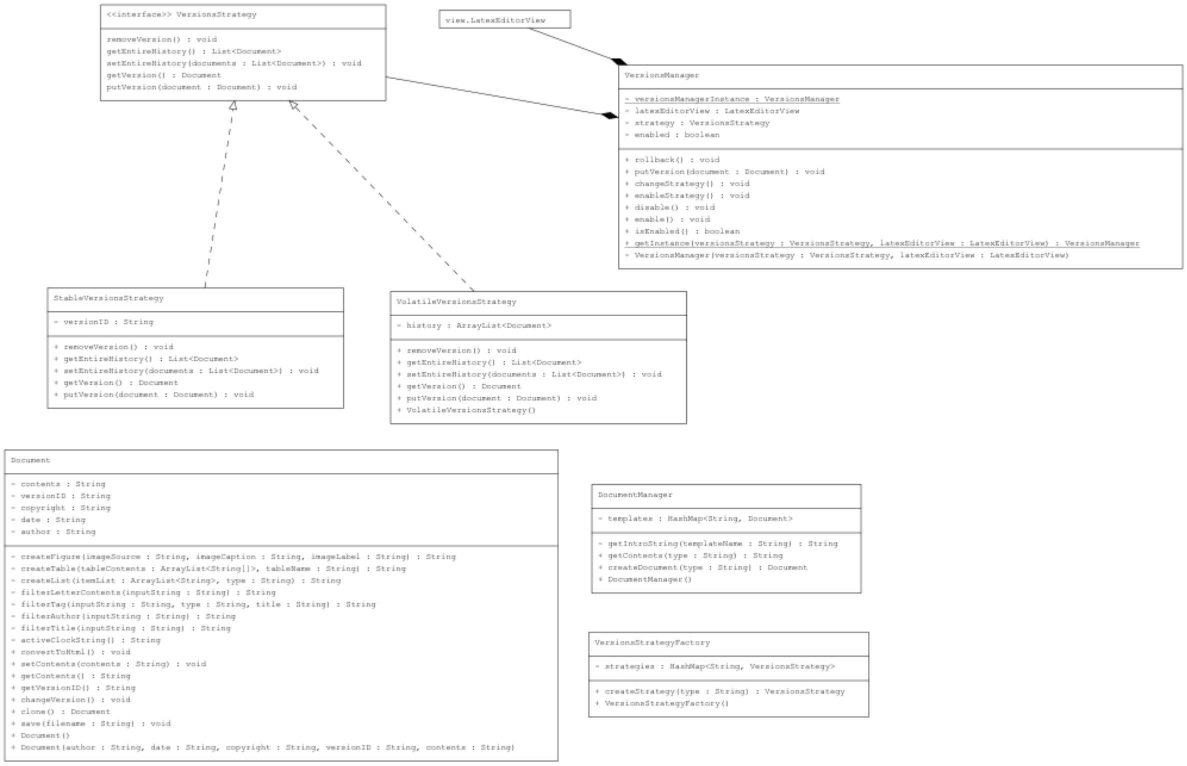

<<interface>> VersionsStrategy
removeVersion() : void
getEntireHistory() : List<Document>
setEntireHistory(documents : List<Document>) : void
getVersion() : Document
putVersion(document : Document) : void
view.LatexEditorView
VersionsManager
- versionsManagerInstance : VersionsManager
- latexEditorView : LatexEditorView
- strategy : VersionsStrategy
- enabled : boolean
+ rollback() : void
+ putVersion(document : Document) : void
+ changeStrategy() : void
+ enableStrategy() : void
+ disable() : void
+ enable() : void
+ isEnabled() : boolean
+ getInstance(versionsStrategy : VersionsStrategy, latexEditorView : LatexEditorView) : VersionsManager
- VersionsManager(versionsStrategy : VersionsStrategy, latexEditorView : LatexEditorView)
StableVersionsStrategy
- versionID : String
+ removeVersion() : void
+ getEntireHistory() : List<Document>
+ setEntireHistory(documents : List<Document>) : void
+ getVersion() : Document
+ putVersion(document : Document) : void
VolatileVersionsStrategy
- history : ArrayList<Document>
+ removeVersion() : void
+ getEntireHistory() : List<Document>
+ setEntireHistory(documents : List<Document>) : void
+ getVersion() : Document
+ putVersion(document : Document) : void
+ VolatileVersionsStrategy()
Document
- contents : String
- versionID : String
- copyright : String
- date : String
- author : String
- createFigure(imageSource : String, imageCaption : String, imageLabel : String) : String
- createTable(tableContents : ArrayList<String[]>, tableName : String) : String
- createList(itemList : ArrayList<String>, type : String) : String
- filterLetterContents(inputString : String) : String
- filterTag(inputString : String, type : String, title : String) : String
- filterAuthor(inputString : String) : String
- filterTitle(inputString : String) : String
- activeClockString() : String
+ convertToHtml() : void
+ setContents(contents : String) : void
+ getContents() : String
+ getVersionID() : String
+ changeVersion() : void
+ clone() : Document
+ save(filename : String) : void
+ Document()
+ Document(author : String, date : String, copyright : String, versionID : String, contents : String)
DocumentManager
- templates : HashMap<String, Document>
- getIntroString(templateName : String) : String
+ getContents(type : String) : String
+ createDocument(type : String) : Document
+ DocumentManager()
VersionsStrategyFactory
- strategies : HashMap<String, VersionsStrategy>
+ createStrategy(type : String) : VersionsStrategy
+ VersionsStrategyFactory()

Figure 3

• View Package Class Diagram

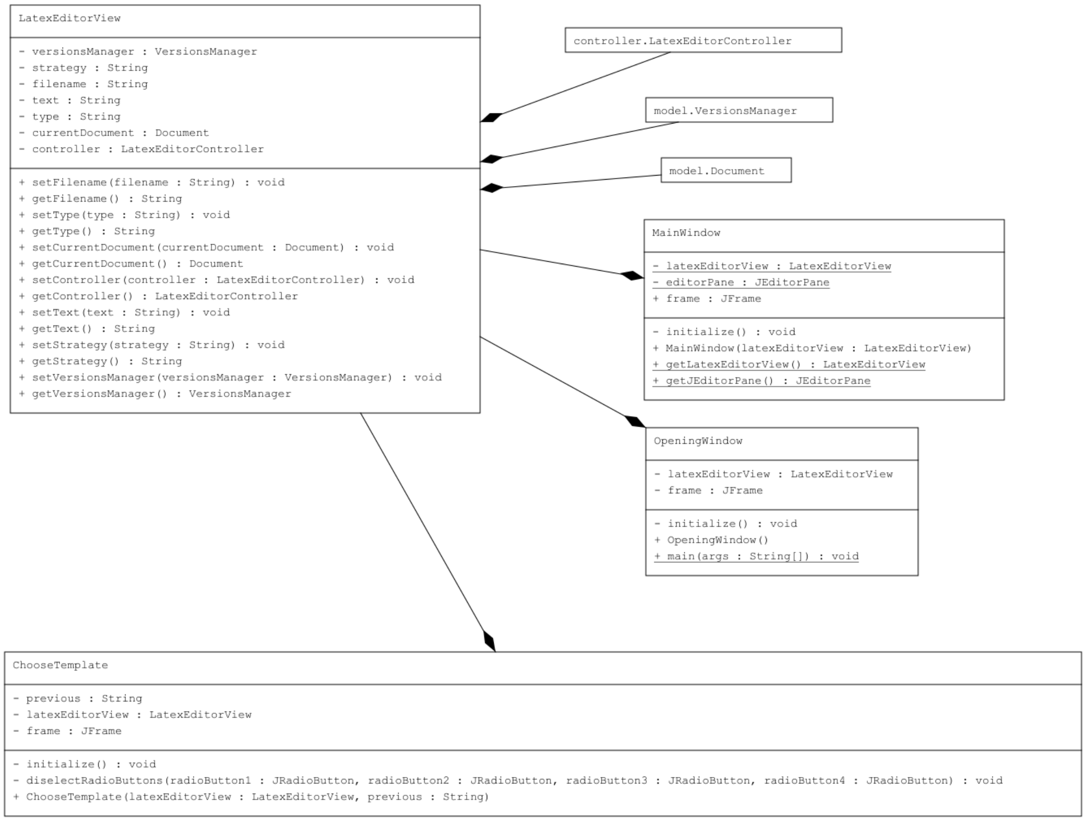

LatexEditorView
- versionsManager : VersionsManager
- strategy : String
- filename : String
- text : String
- type : String
- currentDocument : Document
- controller : LatexEditorController
+ setFilename(filename : String) : void
+ getFilename() : String
+ setType(type : String) : void
+ getType() : String
+ setCurrentDocument(currentDocument : Document) : void
+ getCurrentDocument() : Document
+ setController(controller : LatexEditorController) : void
+ getController() : LatexEditorController
+ setText(text : String) : void
+ getText() : String
+ setStrategy(strategy : String) : void
+ getStrategy() : String
+ setVersionsManager(versionsManager : VersionsManager) : void
+ getVersionsManager() : VersionsManager
controller.LatexEditorController
model.VersionsManager
model.Document
MainWindow
- latexEditorView : LatexEditorView
- editorPane : JEditorPane
+ frame : JFrame
- initialize() : void
+ MainWindow(latexEditorView : LatexEditorView)
+ getLatexEditorView() : LatexEditorView
+ getJEditorPane() : JEditorPane
OpeningWindow
- latexEditorView : LatexEditorView
- frame : JFrame
- initialize() : void
+ OpeningWindow()
+ main(args : String[]) : void
ChooseTemplate
- previous : String
- latexEditorView : LatexEditorView
- frame : JFrame
- initialize() : void
- diselectRadioButtons(radioButton1 : JRadioButton, radioButton2 : JRadioButton, radioButton3 : JRadioButton, radioButton4 : JRadioButton) : void
+ ChooseTemplate(latexEditorView : LatexEditorView, previous : String)

Figure 4

Refactoring Task 1

For this task a *HashMap* was created in the *latexEditorController* class in order to store and load the commands without code duplication.

Refactoring Task 2

For this task the *Singleton pattern* was used to the *VersionsManager* class. The constructor of this class is now private and the getInstance() method was added along with a static private field of the object VersionsManager. This method returns one VersionsManager object on its first call and then it returns the same object for every call.

Refactoring Task 3

For this task a HashMap was created in the *DocumentManeger* class in order to store and load the Latex templates. In addition, for the duplication problem of the code, variables and a method were used to simplify the code in the return values of the *getContents* method .

Refactoring Task 4

For this task the following methods were removed from the *VersionsManager* class and can be used directly from the *latexEditorView* as a result of the Remove the Middle Man refactoring. For this reason *CommandFactory* now uses a *latexEditorView* object as argument in order for the methods to functions properly. To improve readability dead code was also removed.

- getType()
- saveToFile()
- saveContents()
- loadFromFile()
- setCurrentVersion()

Refactoring Task 5

For this task to simplify the latexEditorView class the following methods were moved:

- The saveTofile() method is now implemented in the *SaveCommand* class
- The loadFromFile() method is now implemented in the *LoadCommand* class
- The saveContents() is now implemented in the *EditCommand* class and *AddLatexCommand* class

Refactoring Task 6

For this task the editContents() method was moved to the *AddLatexCommand*. For this reason a *JEditorPane* and a *LatexEditorView* getter method was created in the MainView class for the editContents() method to function properly. Also some String variables were created inside the editContents() method to solve the code duplication problem.

Extension Task 1

In order for the user to convert a Latex document to HTML the "Load File" function was changed. Now the user is able to select to save the current document to either .tex or .html. In case of .html the document is converted to an HTML document using the convertToHtml() method which was created in the *Document* class along with some private methods to simplify it.

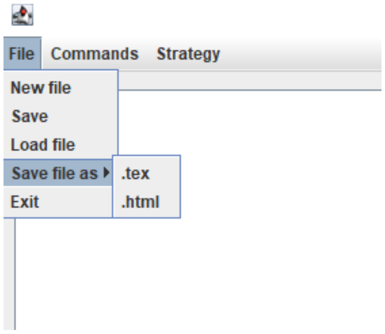


Figure 5

Extension Task 2

In order for the user to be able to load HTML documents the application was extended. Now then loading an .html document it is converted to .tex using the convertToLatex() method in the LoadFile() class. In addition some private methods were added to help simplify the convertToLatex() method.

Extra

- The exit button on the main window was is now functional

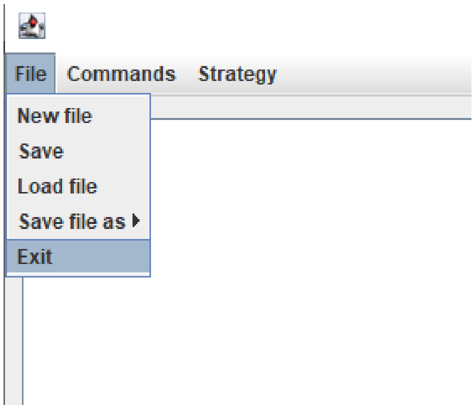


Figure 6

- The "X" button on the top right of the opening window is now functional.

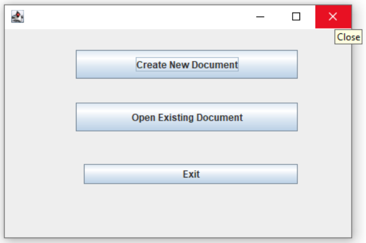


Figure 7

IMPLEMENTATION

- **CRC Cards**

Class Name: AddLatexCommand	
Responsibilities	**Collaborations**
▪ This class is responsible for adding the contents of the latex commands to the current document.	▪ Needs to import the following classes: model.Document, versionManager, view.LatexEditorView, viewWindow.

Class Name: ChangeVersionsStrategyCommand	
Responsibilities	**Collaborations**
▪ This class is responsible for changing the version storage strategy.	▪ Needs to import the model.VersionsManager class.

Class Name: CommandFactory	
Responsibilities	**Collaborations**
▪ This class is responsible for creating the command objects.	▪ Needs to import the model.DocumentManager, model.VersionsManager and the view.LatexEditorView classes.

Class Name: CreateCommand	
Responsibilities	**Collaborations**
▪ This class is responsible for initializing the chosen document template.	▪ Needs to import the following classes: model.Document, model.DocumentManager, model.VersionsManager and the view.LatexEditorView.

Class Name: DisableVersionsManagementCommand	
Responsibilities	**Collaborations**
▪ This class is responsible for disabling the version storage strategy.	▪ Need to import the VersionsManager class.

Class Name: EditCommand	
Responsibilities	**Collaborations**
▪ This class is responsible for saving the contents of the document.	▪ Needs to import the model.Document and the model.VersionsManager classes.

Class Name: EnableVersionsManagementCommand	
Responsibilities	**Collaborations**
▪ This class is responsible for enabling the storage strategy.	▪ Needs to import the model.VersionsManager class.

Class Name: LoadCommand	
Responsibilities	**Collaborations**
▪ This class is responsible for loading a document from a file. ▪ This class is responsible for converting a html file to Latex file.	▪ Needs to import the model.Document, model.VersionsManager and the view.LatexEditorView classes.

Class Name: RollbackToPreviousVersionCommand	
Responsibilities	**Collaborations**
▪ This class is responsible for rolling back to a previous version.	▪ Needs to import the model.VersionsManager class.

Class Name: SaveCommand	
Responsibilities	**Collaborations**
▪ This class is responsible for saving the contents of the current document to a file.	▪ Needs to import the model.Document and the view.LatexEditorView.

Class Name: LatexEditorController	
Responsibilities	**Collaborations**
▪ This class is responsible for initializing the latex commands of the application.	▪ Needs to import the controller.commands.Command, controller.commands.CommandFactory, model.VersionsManager and the view.LatexEditorView classes

Class Name: StableVersionsStrategy	
Responsibilities	**Collaborations**
▪ This class is responsible for the implementation of the stable storage strategy.	▪ Needs to import the model.Document class.

Class Name: VolatileVersionsStrategy	
Responsibilities	**Collaborations**
▪ This class is responsible for the implementation of the volatile storage strategy.	▪ Needs to import the model.Document class.

Class Name: VersionsStrategyFactory	
Responsibilities	**Collaborations**
▪ This class initializes the commands.	▪ It does not need to import any class.

Class Name: Document	
Responsibilities	**Collaborations**
▪ This class is responsible for the implementation of the document functions (save,clone etc) and the conversion of the Latex file to HTML file.	▪ It does not need to import any class.

Class Name: DocumentManager	
Responsibilities	**Collaborations**
▪ This class creates the document template.	▪ It does not need to import any class.

Class Name: VersionsManager	
Responsibilities	**Collaborations**
▪ It is responsible for the versions strategy functionality.	▪ It needs to import the following classes: model.strategies.StableVersionsStrategy, model.strategies.VersionsStrategy, model.strategies.VolatileVersionsStrategy, view.LatexEditorView.

Class Name: ChooseTemplate	
Responsibilities	**Collaborations**
▪ This class creates the GUI that the user selects document template.	▪ Needs the LatexEditorView to operate.

Class Name: LatexEditorView	
Responsibilities	**Collaborations**
▪ This class is responsible for the communication of the front-end and the back-end.	▪ Needs to import the following classes: controller.LatexEditorView, model.Document and model.VersionsManager.

Class Name: MainWindow	
Responsibilities	**Collaborations**
▪ This class creates the main window GUI.	▪ Needs the Document class and the LatexEditorView class to operate.

Class Name: OpeningWindow	
Responsibilities	**Collaborations**
▪ This class creates the opening window GUI.	▪ Needs the following classes to operate: LatexEditorController, VersionsManager, VersionsStrategy,, VolatileVersionsStrategy.