LatexEditor

Release Report

Les Logiciels
Konstantina Vatavali 2649, Ilias Mavrianos 2753

VERSIONS HISTORY

Date	Version	Description	Author
19/4/19	1	First Release. Implementation of User Story 1 and User Story 2.	Les Logiciels
24/5/19	2	Second Release. Implementation of the rest of User Stories	Les Logiciels

1 Introduction

This document provides information concerning the **<2st>** release of the project.

1.1 Purpose

Latex is a well known high quality document preparation markup language. It provides a large variety of styles and commands that enable advanced document formatting. Typically, a Latex document is compiled with a tool like MikTex, Lyx, etc. to produce a respective formatted document in pdf, ps, etc. Formatting documents with Latex is a programming like process as it involves the proper usage of Latex commands which are embedded in the document contents. The objective of this project is to develop a simple Latex editor for inexperienced Latex users. The goal of the editor is to facilitate the usage of Latex commands for the preparation of Latex documents. One of the prominent features that distinguishes the LatexEditor from other similar applications is its multi-strategy version tracking functionalities that enable undo and redo actions.

1.2 Document Structure

The rest of this document is structured as follows. Section 2 specifies the acceptance tests that have been employed for this release of the project. Section 3 specifies the main design concepts for this release of the project.

2 Tests

2.1 Tests for User Story **<1>**

Test ID	*Test Of User Story 1 [US1]*
User Story	*Creation of a new Latex Document based on a particular template or not.*
Test Class	*CreateCommandTest.java*
Description	The purpose of the test is to create a CreateCommand object, execute it and compare the contents of the newly created document with the expected contents of the selected template. We used five assertEquals() methods, one for each possibility. We created 5 different String variables for each testing content(ex. testingArticleContents) using the correspond method (ex. testArticleTemplate). We compared them with the expected contents that we took from the txt files. The JUnitTest runned successfully.

2.2 Tests for User Story **<2>**

Test ID	*Test Of User Story 2 [US2]*
User Story	*Editing the contents of the Latex document via the application's user interface.*
Test Class	*EditCommandTest.java*
Description	The purpose of the test is to create a EditCommand object, that changes the contents of the document, execute it, and subsequently get the new contents of the document and compare them against the contents that have been set. We created a testingEditCommand object from the EditCommand class and changed the contents of the Text Area using setText(). Then, we put the edited contents at the changedContents variable. We set the contents of the document with the changedContents. We took the new contents of the documents and placed them at the variable documentContents. Finally we compared these two variables. The JUnit test runned successfully.

2.3 Tests for User Story **<3>**

Test ID	*Test Of User Story 3 [US3]*
User Story	*Editing the contents of the document, using each one of the different Latex commands.*
Test Class	*AddLatexCommandTest.java*
Description	The purpose of the test is an AddLatexCommand class that changes the contents of a document, execute it and subsequently get the new contents of the document and compare them against the contents that have been set. We created 8 different secondary methods that gives as the contents depending on the corresponding command (ex. Add Chapter) and we compared them with the expected command added contents. We used 8 different assertEquals() methods, one for each possibility. The Junıt test runned successfully.

2.4 Tests for User Story **<4>**

Test ID	*Test Of User Story 4 [US4]*
User Story	*Enabling and using the two different Version Strategies*
Test Class	*EnableVersionsManagementCommandTest.java*
Description	The purpose of the test is to enable and using the two different strategies. We keeped the edited contents at the variable expectedContents. We created an EnableVersionsManagementCommand, execute it , got the the contents of the previous version of the document at the variable testingContents , and compared them against the contents of the document before the edit action. We did this procedure for the two different strategies We used 2 different assertEquals() methods, one for each possibility. The Junıt test runned successfully.

2.5 Tests for User Story **<5>**

Test ID	*Test Of User Story 5 [US5]*
User Story	*Changing the Versions Strategy.*
Test Class	*ChangeVersionStrategyCommandTest.java*
Description	The purpose of the test is to create a ChangeVersionsStrategyCommad, execute it, and check whether the strategy indeed change. We did the procedure for changing one to another strategy. We checked if the changed strategy is type of VolatileVersionsStrategy/StableVersionsStrategy class, respectively. We used 2 different assertEquals() methods, one for each possibility.The JUnit test runned successfully.

2.6 Tests for User Story **<6>**

Test ID	*Test Of User Story 6 [US6]*
User Story	*Disabling the Versions Strategy*
Test Class	*DisableVersionStrategyCommandTest.java*
Description	The purpose of the test is to create a DisableVersionsManagementCommand, execute it, create an EditCommand that changes the contents of the document, execute it, and check if the versions history remains unchanged, and check whether the mechanism is actually disabled. We kept the size of the list before the disabling at the variable expectedSize. After, we disabled the tracking mechanism and edited the contents. We kept the new size of the list at the variable testingSize and compared the size of these two different list. We used an assertEquals() method. The JUnit test runned successfully.

2.7 Tests for User Story **<7>**

Test ID	*Test Of User Story 7 [US7]*
User Story	*Rolling back to the previous version.*
Test Class	*RollbackToPreviousVersionCommandTest.java*
Description	The purpose of the test is to get the contents of the previous version of a document, create a RollbackToPreviousVersionCommand, execute it, and check whether the contents of the current document match with the contents of the previous version. We kept the contens of the version 1 to the variable ExpectedPreviousVersionContents. We edited the document. At version 2, we rolled back to previous version and kept the contens to the variable testingPreviousVersionContents, and compared them. We used an assertEquals() method. The JUnit test runned successfully.

2.8 Tests for User Story **<8>**

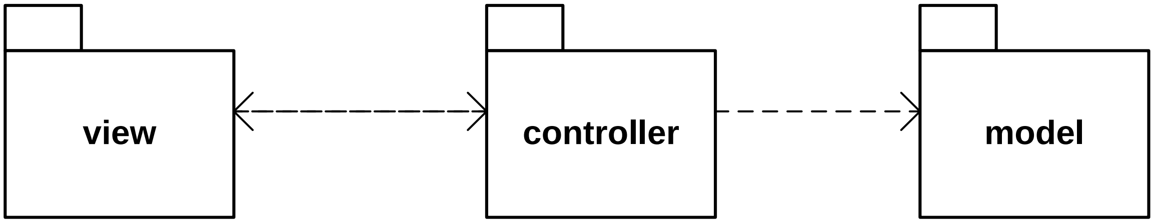
Test ID	*Test Of User Story 8 [US8]*
User Story	*Saving the document to the disk.*
Test Class	*SaveCommadTest.java*
Description	The purpose of the test is to get the contents of the current version of a document, create SaveCommand, execute it, and check whether the contents of the current document match with the contents of the file that has been saved to disk. We kept the expected contents from the document at the variable expectedContents. We took the contents from the saved file and kept them at the variable testingContents, and compared them. We used an assertEquals() method. The JUnit test runned successfully.

2.9 Tests for User Story **<9>**

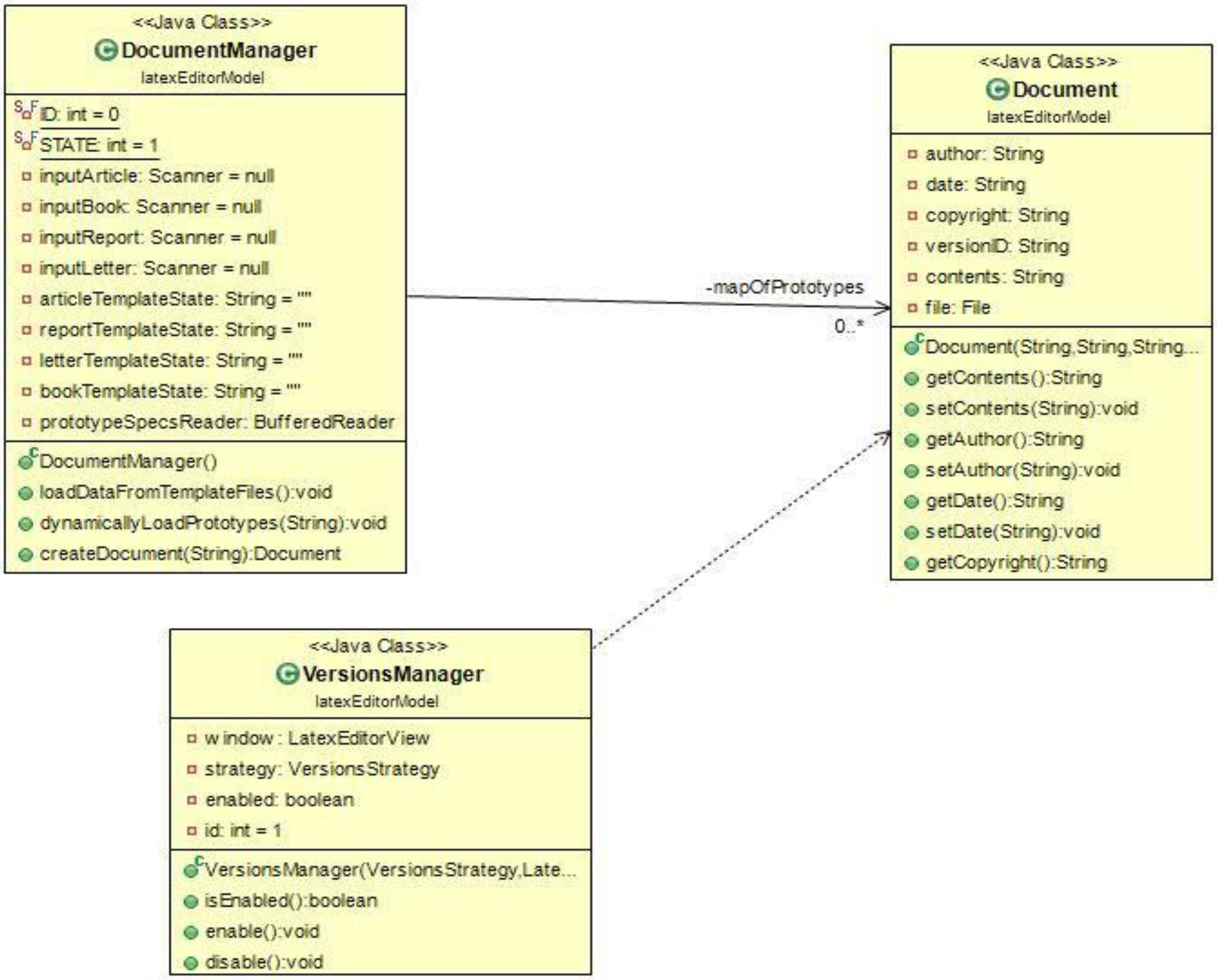
Test ID	*Test Of User Story 9 [US9]*
User Story	*Loading a file from the disk.*
Test Class	*LoadCommadTest.java*
Description	The purpose of the test is to create a LoadCommand, execute it, get the contents of the current version of a document and check whether the contents match with the contents of the file that have been laoded from the disk. We kept the expected contents from the file at the variable expectedContents and the contents from the document that has created from the loaded file at the variable testingContents. We compared them. The JUnit test runned successfully

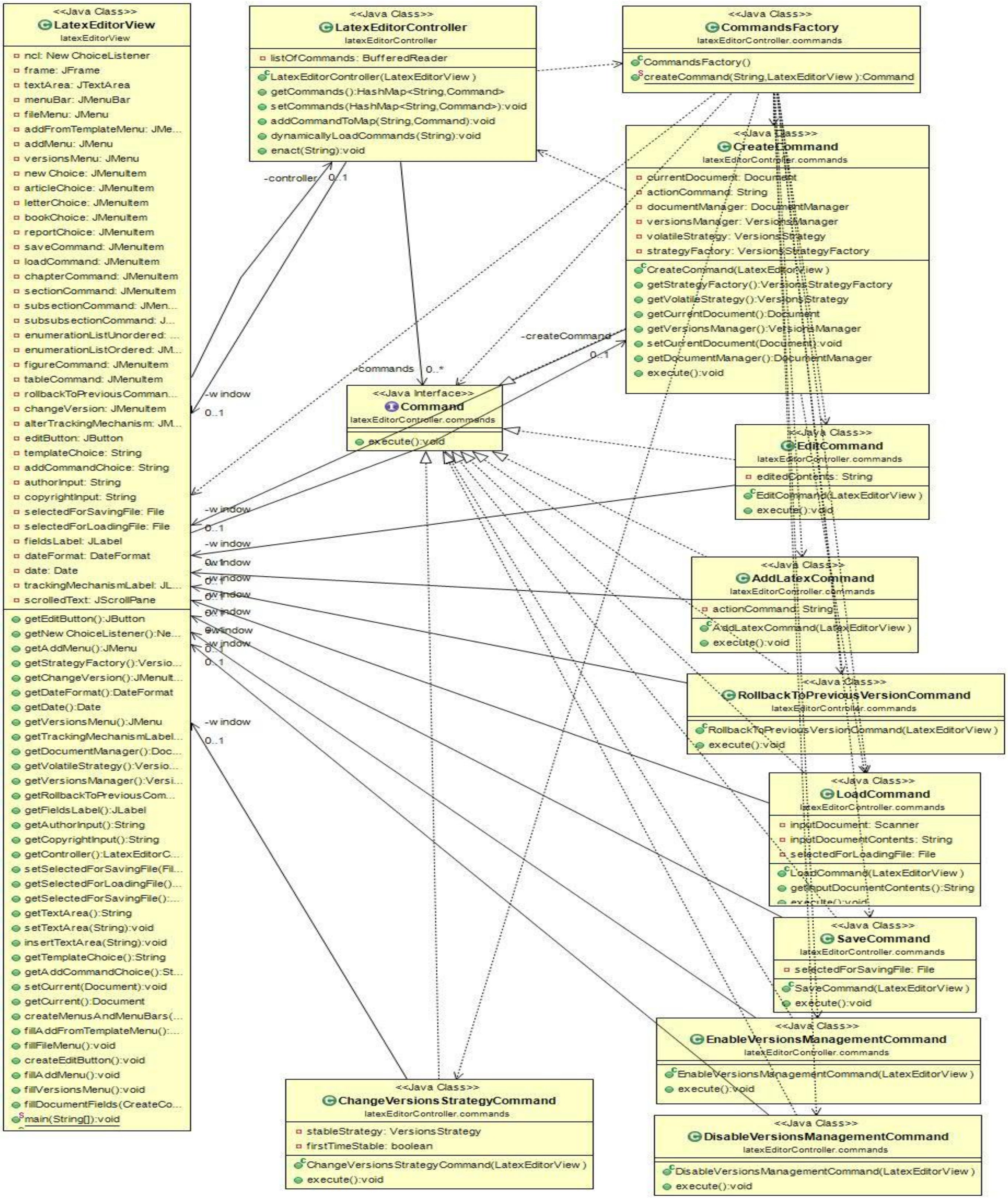
3 Design

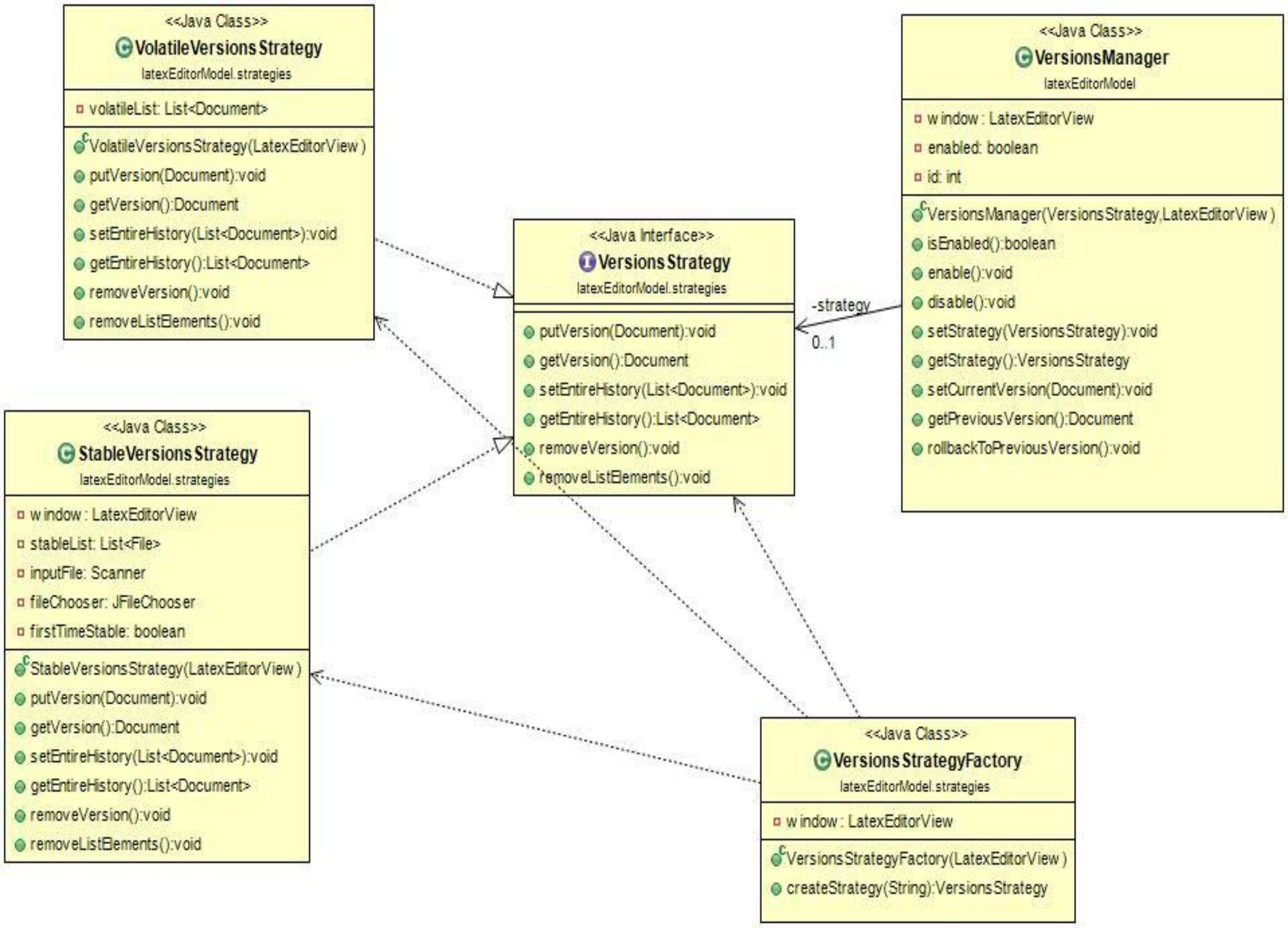
3.1 Architecture



3.2 Design




<<Java Class>>
LatexEditorView
latexEditorView
ncl: New ChoiceListener
frame: JFrame
textArea: JTextArea
menuBar: JMenuBar
fileMenu: JMenu
addFromTemplateMenu: JMe...
addMenu: JMenu
versionsMenu: JMenu
new Choice: JMenuItem
articleChoice: JMenuItem
letterChoice: JMenuItem
bookChoice: JMenuItem
reportChoice: JMenuItem
saveCommand: JMenuItem
loadCommand: JMenuItem
chapterCommand: JMenuItem
sectionCommand: JMenuItem
subsectionCommand: JMen...
subsubsectionCommand: J...
enumerationListUnordered: ...
enumerationListOrdered: JM...
figureCommand: JMenuItem
tableCommand: JMenuItem
rollbackToPreviousComman...
changeVersion: JMenuItem
alterTrackingMechanism: JM...
editButton: JButton
templateChoice: String
addCommandChoice: String
authorInput: String
copyrightInput: String
selectedForSavingFile: File
selectedForLoadingFile: File
fieldsLabel: JLabel
dateFormat: DateFormat
date: Date
trackingMechanismLabel: JL...
scrolledText: JScrollPane
getEditButton():JButton
getNew ChoiceListener():Ne...
getAddMenu():JMenu
getStrategyFactory():Versio...
getChangeVersion():JMenuIt...
getDateFormat():DateFormat
getDate():Date
getVersionsMenu():JMenu
getTrackingMechanismLabel...
getDocumentManager():Doc...
getVolatileStrategy():Versio...
getVersionsManager():Versi...
getRollbackToPreviousCom...
getFieldsLabel():JLabel
getAuthorInput():String
getCopyrightInput():String
getController():LatexEditorC...
setSelectedForSavingFile(Fil...
getSelectedForLoadingFile()...
getSelectedForSavingFile():...
getTextArea():String
setTextArea(String):void
insertTextArea(String):void
getTemplateChoice():String
getAddCommandChoice():St...
setCurrent(Document):void
getCurrent():Document
createMenusAndMenuBars(...
fillAddFromTemplateMenu():...
fillFileMenu():void
createEditButton():void
fillAddMenu():void
fillVersionsMenu():void
fillDocumentFields(CreateCo...
main(String[]):void
<<Java Class>>
LatexEditorController
latexEditorController
listOfCommands: BufferedReader
LatexEditorController(LatexEditorView)
getCommands():HashMap<String,Command>
setCommands(HashMap<String,Command>):void
addCommandToMap(String,Command):void
dynamicallyLoadCommands(String):void
enact(String):void
-controller
0..1
-commands
0..*
<<Java Class>>
CommandsFactory
latexEditorController.commands
CommandsFactory()
createCommand(String,LatexEditorView):Command
<<Java Class>>
CreateCommand
latexEditorController.commands
currentDocument: Document
actionCommand: String
documentManager: DocumentManager
versionsManager: VersionsManager
volatileStrategy: VersionsStrategy
strategyFactory: VersionsStrategyFactory
CreateCommand(LatexEditorView)
getStrategyFactory():VersionsStrategyFactory
getVolatileStrategy():VersionsStrategy
getCurrentDocument():Document
getVersionsManager():VersionsManager
setCurrentDocument(Document):void
getDocumentManager():DocumentManager
execute():void
-createCommand
0..1
<<Java Interface>>
Command
latexEditorController.commands
execute():void
-window
0..1
<<Java Class>>
EditCommand
latexEditorController.commands
editedContents: String
EditCommand(LatexEditorView)
execute():void
<<Java Class>>
AddLatexCommand
latexEditorController.commands
actionCommand: String
AddLatexCommand(LatexEditorView)
execute():void
<<Java Class>>
RollbackToPreviousVersionCommand
latexEditorController.commands
RollbackToPreviousVersionCommand(LatexEditorView)
execute():void
<<Java Class>>
LoadCommand
latexEditorController.commands
inputDocument: Scanner
inputDocumentContents: String
selectedForLoadingFile: File
LoadCommand(LatexEditorView)
getInputDocumentContents():String
execute():void
<<Java Class>>
SaveCommand
latexEditorController.commands
selectedForSavingFile: File
SaveCommand(LatexEditorView)
execute():void
<<Java Class>>
EnableVersionsManagementCommand
latexEditorController.commands
EnableVersionsManagementCommand(LatexEditorView)
execute():void
<<Java Class>>
DisableVersionsManagementCommand
latexEditorController.commands
DisableVersionsManagementCommand(LatexEditorView)
execute():void
<<Java Class>>
ChangeVersionsStrategyCommand
latexEditorController.commands
stableStrategy: VersionsStrategy
firstTimeStable: boolean
ChangeVersionsStrategyCommand(LatexEditorView)
execute():void


<<Java Class>>
VolatileVersionsStrategy
latexEditorModel.strategies
volatileList: List<Document>
VolatileVersionsStrategy(LatexEditorView)
putVersion(Document):void
getVersion():Document
setEntireHistory(List<Document>):void
getEntireHistory():List<Document>
removeVersion():void
removeListElements():void
<<Java Interface>>
VersionsStrategy
latexEditorModel.strategies
putVersion(Document):void
getVersion():Document
setEntireHistory(List<Document>):void
getEntireHistory():List<Document>
removeVersion():void
removeListElements():void
-strategy
0..1
<<Java Class>>
VersionsManager
latexEditorModel
window: LatexEditorView
enabled: boolean
id: int
VersionsManager(VersionsStrategy,LatexEditorView)
isEnabled():boolean
enable():void
disable():void
setStrategy(VersionsStrategy):void
getStrategy():VersionsStrategy
setCurrentVersion(Document):void
getPreviousVersion():Document
rollbackToPreviousVersion():void
<<Java Class>>
StableVersionsStrategy
latexEditorModel.strategies
window: LatexEditorView
stableList: List<File>
inputFile: Scanner
fileChooser: JFileChooser
firstTimeStable: boolean
StableVersionsStrategy(LatexEditorView)
putVersion(Document):void
getVersion():Document
setEntireHistory(List<Document>):void
getEntireHistory():List<Document>
removeVersion():void
removeListElements():void
<<Java Class>>
VersionsStrategyFactory
latexEditorModel.strategies
window: LatexEditorView
VersionsStrategyFactory(LatexEditorView)
createStrategy(String):VersionsStrategy

Class Name: Document	
Responsibilities: ▪ Holds the data of a document. ▪ Has the clone() method. ▪ Has the save() method that saved the document at the disk.	**Collaborations:** ▪ DocumentManager ▪ VersionsManager

Class Name: LatexEditorController	
Responsibilities:	**Collaborations:**
▪ Has a map of command objects. ▪ Add command objects to the map. ▪ Enacts the specific command.	▪ Document ▪ Command ▪ CommandsFactory ▪ LatexEditorView

Class Name: DocumentManager	
Responsibilities:	**Collaborations:**
▪ Creates the template states from file. ▪ Creates the prototypes and puts them in the map. ▪ Creates the Document.	▪ Document ▪ CreateCommand

Class Name: VersionsManager	
Responsibilities:	**Collaborations:**
▪ Has a reference to the current version of the Latex document. ▪ Has the setStrategy() method that sets the preferable strategy ▪ Has the setCurrentVersion() method that add the current version to the history and replaces it with the new document version. ▪ Has the rollbackToPreviousVersion() method that removes the current version and set the previous version to the current	▪ Document ▪ VersionsStrategy ▪ LatexEditorView

Class Name: CommandsFactory	
Responsibilities:	**Collaborations:**
▪ Creates the objects similarly to the specific command class under the interface Command. (ex a CreateCommand object)	▪ Command

Class Name: CreateCommand	
Responsibilities:	**Collaborations:**
▪ Creates a Document of the user's template choice or from the loading file. The user choices a template or none, via the application interface. ▪ Sets enabled the versionsMenu, the vddMenu, the Edit Button, the changeVersion JMenuItem, the rollbackToPreviousCommand JMenuItem and the Tracking Mechanism JLabel.	▪ Command ▪ LatexEditorView ▪ DocumentManager ▪ VersionsManager ▪ VersionsStrategyFactory

Class Name: EditCommand	
Responsibilities:	**Collaborations:**
▪ Edits the contents of the document via the application interface. If the tracking mechanism is enabled, it keeps the versions to the history.	▪ Command ▪ LatexEditorView

Class Name: LatexEditorView	
Responsibilities:	**Collaborations:**
▪ Creates the window for the application interface. Then puts, the Text Area, the Menu Bar, the Menus and the Menu Items. It appears the contents of the Document based on the selected template. It edits the contents. It keeps the buttons and menu items for all the necessary energies. ▪ Its sets the JMenu items of AddMenu accordingly what template, the user choosed.	▪ LatexEditorController ▪ CreateCommand

Class Name: AddLatexCommand	
Responsibilities: ▪ Adds to the text area the add command choice of the user. (ex. Adds a chapter)	**Collaborations:** ▪ LatexEditorView ▪ Command

Class Name: ChangeVersionsCommand	
Responsibilities: ▪ Change the strategy depending on the user's demand. ▪ Sets the Tracking Mechanism Label	**Collaborations:** ▪ LatexEditorView ▪ Command ▪ VersionsStrategy

Class Name: Command	
Responsibilities: ▪ It's the interface for the different commands. ▪ Keeps the execute() method.	**Collaborations:**

Class Name: DisableVersionsManagementCommand / EnableVersionsManagementCommand	
Responsibilities:	**Collaborations:**
▪ Disable/enables the tracking mechanism. ▪ If it disables the tracking mechanism, it disables the JMenu item for rolling back to the previous version too. ▪ It sets the Tracking Mechanism label.	▪ LatexEditorView ▪ Command

Class Name: RollbackToPreviousVersionCommand	
Responsibilities:	**Collaborations:**
▪ Rolls back to the previous version of the document using Volatile or Stable strategy.	▪ LatexEditorView ▪ Command

Class Name: SaveCommand	
Responsibilities:	**Collaborations:**
▪ Saves the document to the disk storage.	▪ LatexEditorView ▪ Command

Class Name: LoadCommand	
Responsibilities:	**Collaborations:**
▪ Loads the file from disk storage.	▪ LatexEditorView ▪ Command

Class Name: VersionsStrategy	
Responsibilities:	**Collaborations:**
▪ It’s the interface for the StableVersionsStrategyCommand and VolatileVersionsStrategyCommand	▪ Document ▪ StableVersionsStrategyCommand ▪ VolatileVersionsStrategyCommand

Class Name: VersionsStrategyFactory	
Responsibilities:	**Collaborations:**
▪ Creates the objects similarly to the specific command class under the interface VersionsStrategy.	▪ LatexEditorView

Class Name: VolatileVersionsStrategyCommand	
Responsibilities:	**Collaborations:**
▪ Puts the current version to the list of documents. ▪ Gets the last element of the list. ▪ Returns the entire list of documents. ▪ Sets anew the document list , accordingly the elements that took from the stable strategy. ▪ Removes the last element of the list. ▪ It removes all the elements from the list.	▪ VersionsManager ▪ VersionsStrategy

Class Name: StableVersionsStrategyCommand	
Responsibilities: ▪ Puts the current version to the list of files. ▪ Gets the last element of the list. ▪ Returns the entire list of files. ▪ Sets anew the file list , accordingly the elements that took from the volatile strategy. ▪ Removes the last element of the list. ▪ It removes all the elements from the list.	**Collaborations:** ▪ VersionsManager ▪ VersionsStrategy