LatexEditor

Release Report

TODO

ΣΑΚΚΟΣ ΣΤΑΥΡΟΣ ΑΜ:2532

VERSIONS HISTORY

Date	Version	Description	Author
24/05/2019	1.2	Second Release (Final)	Sakkos Stavros

1 Introduction

This document provides information concerning the first 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 US1

Test ID	*US-1*
User Story	*UserStory1*
Test Class	*CreateDocumentTest*
Description	*There are 5 acceptance tests : 1 for each template + 1 for an empty document.For each test,an –CreateCommand- object has been created and then executed.Finally, there is a comparison between the Object's text and document's text for each test.* (Note: This test should be run first)

2.2 Tests for User Story US2

Test ID	*US-2*
User Story	*UserStory2*
Test Class	*EditDocumentTest*
Description	*There are 2 acceptance tests : 1 for an empty document and 1 for a non empty document.For each test,an –EditDocument- Object has been created. Each Object edits the text inside the file during the execution.*

2.3 Tests for User Story US3

Test ID	*US-3*
User Story	*UserStory3*
Test Class	*AddLatexCommandTest*
Description	There are 8 acceptance tests : 1 for each given Latex command. The idea for implementing the tests is to create an AddLatexCommand that changes the contents of a document, execute it and subsequently get the new contents of the document (getContents()) and compare them against the contents that have been set.

2.4 Tests for User Story US5

Test ID	*US-5*
User Story	*UserStory5*
Test Class	*ChangeStrategyTest*
Description	To test this story we need at least 2 acceptance tests, one for each different versions strategy. An idea for implementing the tests is to create a ChangeStrategyCommand, execute it, and check whether the strategy indeed changed.

2.5 Tests for User Story US6

Test ID	*US-6*
User Story	*UserStory6*
Test Class	*TrackingTest*
Description	To test this story we will edit a Document 2 times.The first time tracking will be enabled but the second time it will be disabled.Then we will check whether the document has changed after each edit or not. (In this case,tracking is boolean instead of a Command object which changes with a toggle method – JRadioButton)

2.6 Tests for User Story US7

Test ID	*US-7*
User Story	*UserStory7*
Test Class	*RollBackTest*
Description	An idea for this test is to get the contents of the previous version of a document, create a RollbackCommand, execute it, and check whether the contents of the current document match with the contents of the previous version.

2.7 Tests for User Story US8

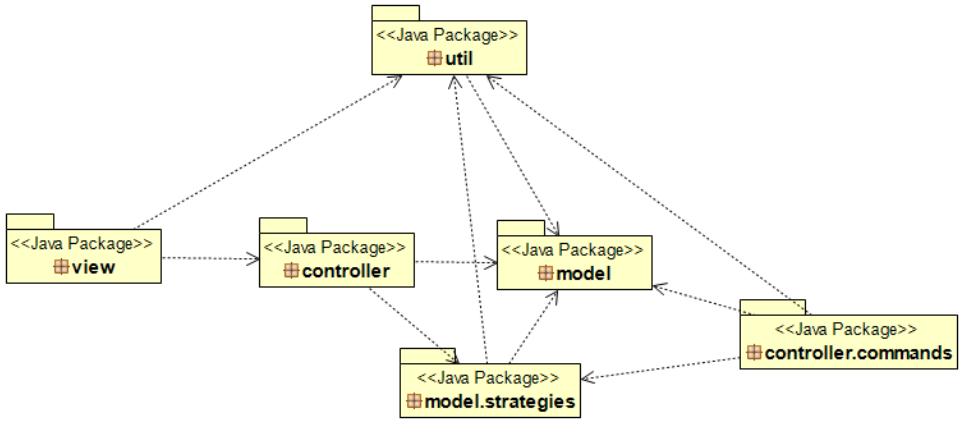
Test ID	*US-8*
User Story	*UserStory8*
Test Class	*SaveDocumentTest*
Description	The idea for implementing the tests is to create an AddLatexCommandthat changes the contents of a document, execute it and subsequently get the new contents of the document (getContents()) and compare them against the contents that have been set.

2.8 Tests for User Story US9

Test ID	*US-9*
User Story	*UserStory9*
Test Class	*LoadDocumentTest*
Description	The idea for this test is to create 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 loaded from disk .(Note: CreateDocument Test has to run first in order to Load the files properly)

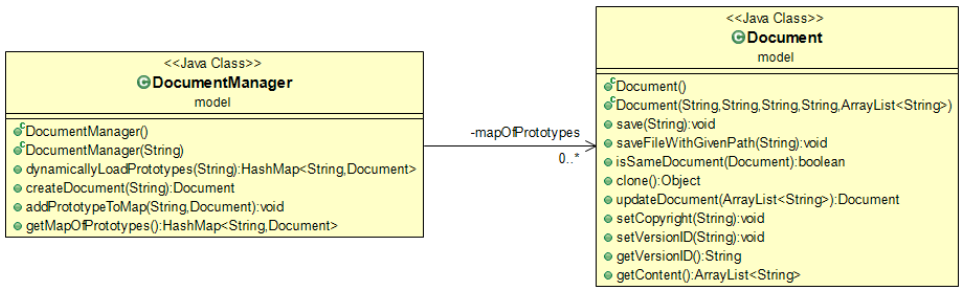
3 Design

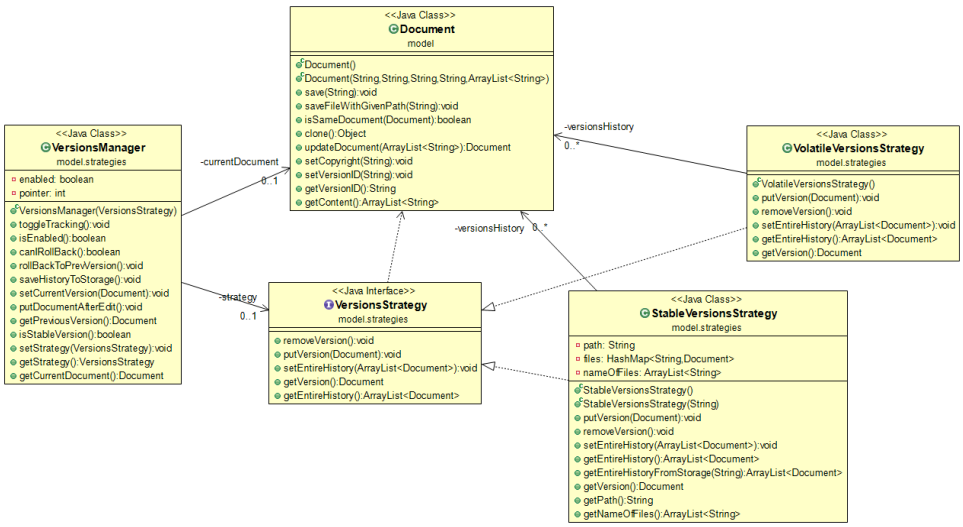
3.1 Architecture

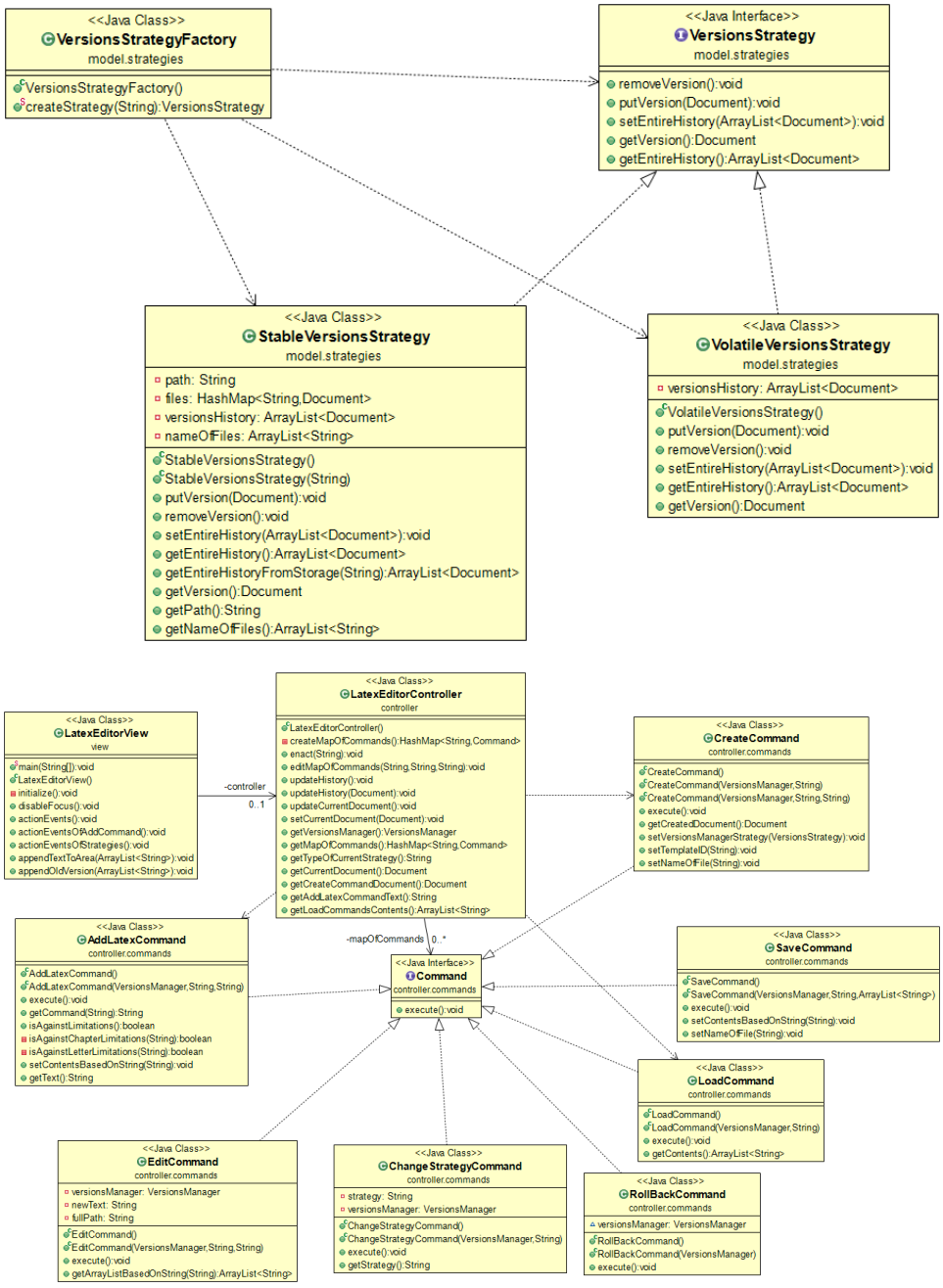


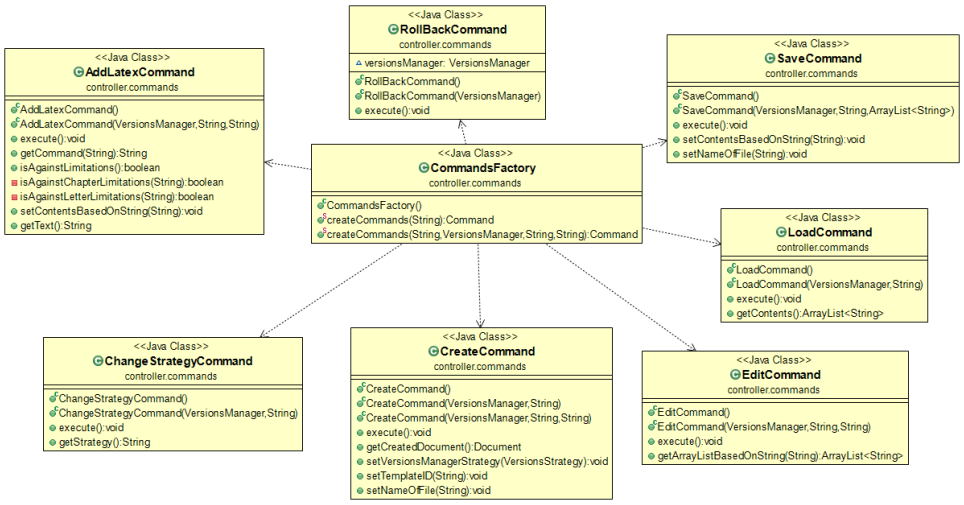
(Note: Update is done with the controller instead of the viewer.)

3.2 Design


<<Java Class>>
DocumentManager
model
DocumentManager()
DocumentManager(String)
dynamicallyLoadPrototypes(String):HashMap<String,Document>
createDocument(String):Document
addPrototypeToMap(String,Document):void
getMapOfPrototypes():HashMap<String,Document>
-mapOfPrototypes
0..*
<<Java Class>>
Document
model
Document()
Document(String,String,String,String,ArrayList<String>)
save(String):void
saveFileWithGivenPath(String):void
isSameDocument(Document):boolean
clone():Object
updateDocument(ArrayList<String>):Document
setCopyright(String):void
setVersionID(String):void
getVersionID():String
getContent():ArrayList<String>


<<Java Class>>
VersionsManager
model.strategies
enabled: boolean
pointer: int
VersionsManager(VersionsStrategy)
toggleTracking():void
isEnabled():boolean
canIRollBack():boolean
rollBackToPrevVersion():void
saveHistoryToStorage():void
setCurrentVersion(Document):void
putDocumentAfterEdit():void
getPreviousVersion():Document
isStableVersion():boolean
setStrategy(VersionsStrategy):void
getStrategy():VersionsStrategy
getCurrentDocument():Document
-currentDocument
0..1
<<Java Class>>
Document
model
Document()
Document(String,String,String,String,ArrayList<String>)
save(String):void
saveFileWithGivenPath(String):void
isSameDocument(Document):boolean
clone():Object
updateDocument(ArrayList<String>):Document
setCopyright(String):void
setVersionID(String):void
getVersionID():String
getContent():ArrayList<String>
-versionsHistory
0..*
<<Java Class>>
VolatileVersionsStrategy
model.strategies
VolatileVersionsStrategy()
putVersion(Document):void
removeVersion():void
setEntireHistory(ArrayList<Document>):void
getEntireHistory():ArrayList<Document>
getVersion():Document
-versionsHistory
0..*
-strategy
0..1
<<Java Interface>>
VersionsStrategy
model.strategies
removeVersion():void
putVersion(Document):void
setEntireHistory(ArrayList<Document>):void
getVersion():Document
getEntireHistory():ArrayList<Document>
<<Java Class>>
StableVersionsStrategy
model.strategies
path: String
files: HashMap<String,Document>
nameOfFiles: ArrayList<String>
StableVersionsStrategy()
StableVersionsStrategy(String)
putVersion(Document):void
removeVersion():void
setEntireHistory(ArrayList<Document>):void
getEntireHistory():ArrayList<Document>
getEntireHistoryFromStorage(String):ArrayList<Document>
getVersion():Document
getPath():String
getNameOfFiles():ArrayList<String>


<<Java Class>>
VersionsStrategyFactory
model.strategies
VersionsStrategyFactory()
createStrategy(String):VersionsStrategy
<<Java Interface>>
VersionsStrategy
model.strategies
removeVersion():void
putVersion(Document):void
setEntireHistory(ArrayList<Document>):void
getVersion():Document
getEntireHistory():ArrayList<Document>
<<Java Class>>
StableVersionsStrategy
model.strategies
path: String
files: HashMap<String,Document>
versionsHistory: ArrayList<Document>
nameOfFiles: ArrayList<String>
StableVersionsStrategy()
StableVersionsStrategy(String)
putVersion(Document):void
removeVersion():void
setEntireHistory(ArrayList<Document>):void
getEntireHistory():ArrayList<Document>
getEntireHistoryFromStorage(String):ArrayList<Document>
getVersion():Document
getPath():String
getNameOfFiles():ArrayList<String>
<<Java Class>>
VolatileVersionsStrategy
model.strategies
versionsHistory: ArrayList<Document>
VolatileVersionsStrategy()
putVersion(Document):void
removeVersion():void
setEntireHistory(ArrayList<Document>):void
getEntireHistory():ArrayList<Document>
getVersion():Document
<<Java Class>>
LatexEditorView
view
main(String[]):void
LatexEditorView()
initialize():void
disableFocus():void
actionEvents():void
actionEventsOfAddCommand():void
actionEventsOfStrategies():void
appendTextToArea(ArrayList<String>):void
appendOldVersion(ArrayList<String>):void
-controller
0..1
<<Java Class>>
LatexEditorController
controller
LatexEditorController()
createMapOfCommands():HashMap<String,Command>
enact(String):void
editMapOfCommands(String,String,String):void
updateHistory():void
updateHistory(Document):void
updateCurrentDocument():void
setCurrentDocument(Document):void
getVersionsManager():VersionsManager
getMapOfCommands():HashMap<String,Command>
getTypeOfCurrentStrategy():String
getCurrentDocument():Document
getCreateCommandDocument():Document
getAddLatexCommandText():String
getLoadCommandsContents():ArrayList<String>
<<Java Class>>
CreateCommand
controller.commands
CreateCommand()
CreateCommand(VersionsManager,String)
CreateCommand(VersionsManager,String,String)
execute():void
getCreatedDocument():Document
setVersionsManagerStrategy(VersionsStrategy):void
setTemplateID(String):void
setNameOfFile(String):void
<<Java Class>>
AddLatexCommand
controller.commands
AddLatexCommand()
AddLatexCommand(VersionsManager,String,String)
execute():void
getCommand(String):String
isAgainstLimitations():boolean
isAgainstChapterLimitations(String):boolean
isAgainstLetterLimitations(String):boolean
setContentsBasedOnString(String):void
getText():String
-mapOfCommands
0..*
<<Java Interface>>
Command
controller.commands
execute():void
<<Java Class>>
SaveCommand
controller.commands
SaveCommand()
SaveCommand(VersionsManager,String,ArrayList<String>)
execute():void
setContentsBasedOnString(String):void
setNameOfFile(String):void
<<Java Class>>
LoadCommand
controller.commands
LoadCommand()
LoadCommand(VersionsManager,String)
execute():void
getContents():ArrayList<String>
<<Java Class>>
EditCommand
controller.commands
versionsManager: VersionsManager
newText: String
fullPath: String
EditCommand()
EditCommand(VersionsManager,String,String)
execute():void
getArrayListBasedOnString(String):ArrayList<String>
<<Java Class>>
ChangeStrategyCommand
controller.commands
strategy: String
versionsManager: VersionsManager
ChangeStrategyCommand()
ChangeStrategyCommand(VersionsManager,String)
execute():void
getStrategy():String
<<Java Class>>
RollBackCommand
controller.commands
versionsManager: VersionsManager
RollBackCommand()
RollBackCommand(VersionsManager)
execute():void



3.3 CRC cards for the most important classes

Class Name: DocumentManager	
Responsibilities: ▪ Dynamic loading of prototypes ▪ Document-Map creation ▪ ...	**Collaborations:** ▪ Document ▪ ….

Class Name: Document	
Responsibilities: ▪ Document creation ▪ Exports Document ▪ ….	**Collaborations:** ▪ FileParser ▪ DocumentManager ▪ VersionsStrategyManager ▪ ….

Class Name: FileParser	
Responsibilities: ▪ File parser ▪ Extracts text from files ▪ Given X produces Y (String to ArrayList)	**Collaborations:** ▪ Document ▪ ….

Class Name: VersionsManager	
Responsibilities: ▪ Document history management ▪ Provides previous versions of the Documents.	**Collaborations:** ▪ VolatileVersionsStrategy ▪ StableVersionsStrategy ▪ LatexEditorController ▪ ….

Class Name: LatexEditorView	
Responsibilities: ▪ Graphics ▪ Provides data to controller ▪ ….	**Collaborations:** ▪ LatexEditorController ▪ ….

Class Name: LatexEditorController	
Responsibilities: ▪ Updates viewer's data ▪ Provides data to controller ▪ ….	**Collaborations:** ▪ LatexEditorView ▪ ….