LATEX EDITOR PROJECT

DESIGN RECOVERY AND QUALITY ASSESSMENT REPORT

VERSION <2.0>

<Άγγελος Κατσαλήρος 2997

Δημήτριος Μπακάλης 3033>

TABLE OF CONTENTS

Introduction 4

Design Recovery 4

- Architecture 4
- Detailed Design 4
- Implementation 9

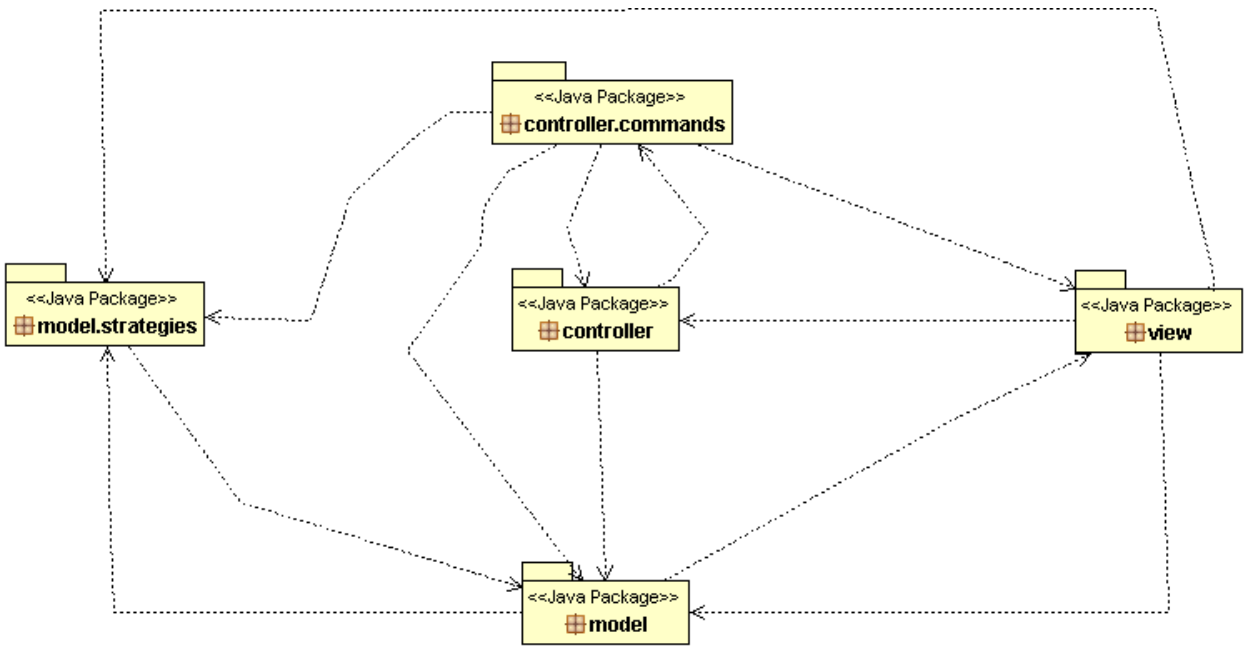
Quality Assessment 30

INTRODUCTION

The purpose of this phase is to understand the function of each individual class from the 5 packages, of the project, and how they interact with each other. We also going to demonstrate the uml package diagram as well as the uml class diagrams of each package.

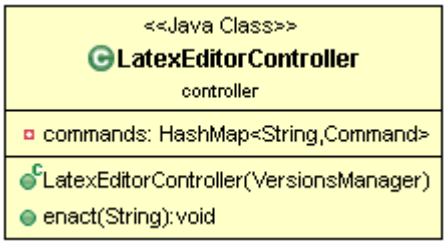
DESIGN RECOVERY

ARCHITECTURE

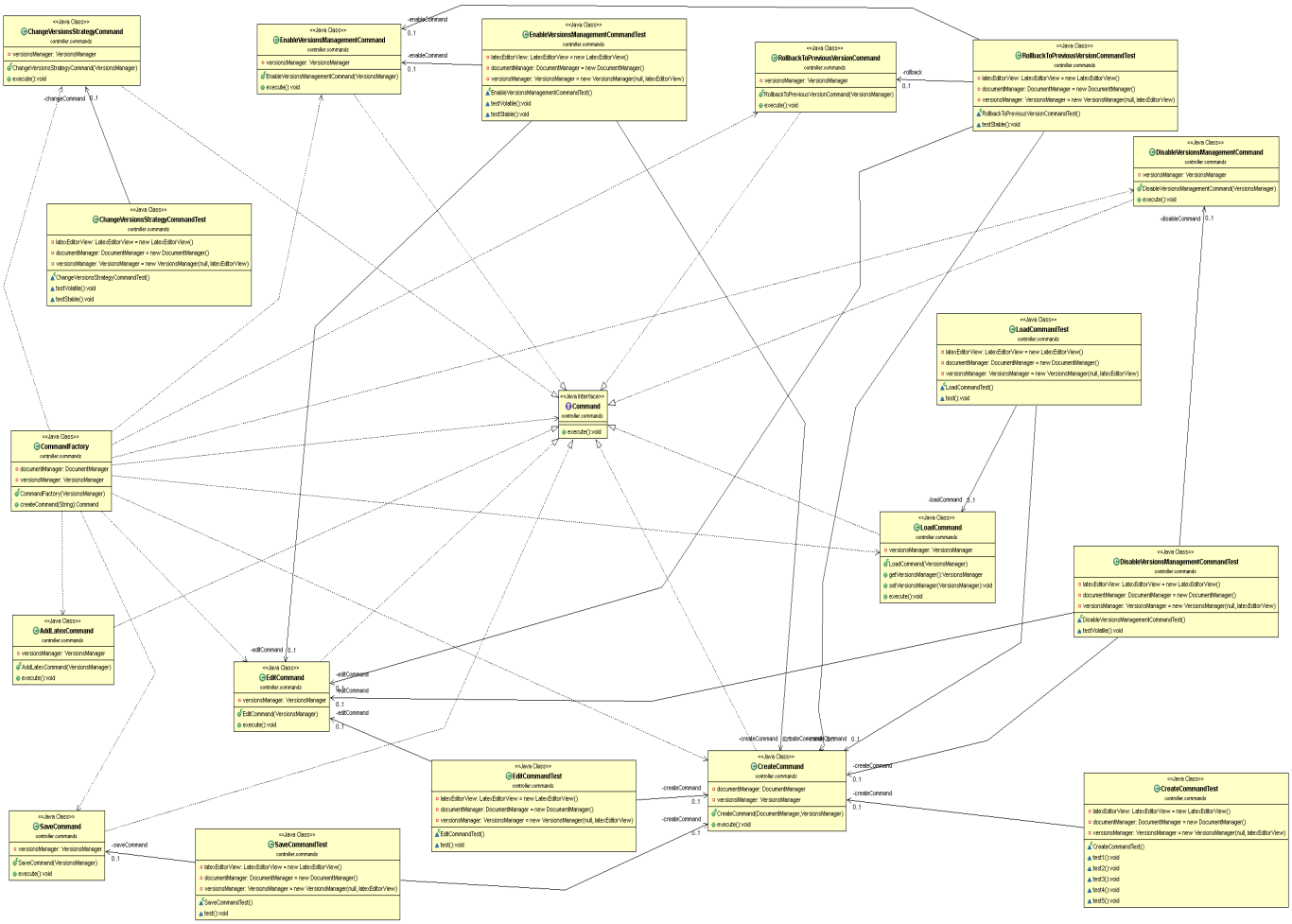


DETAILED DESIGN

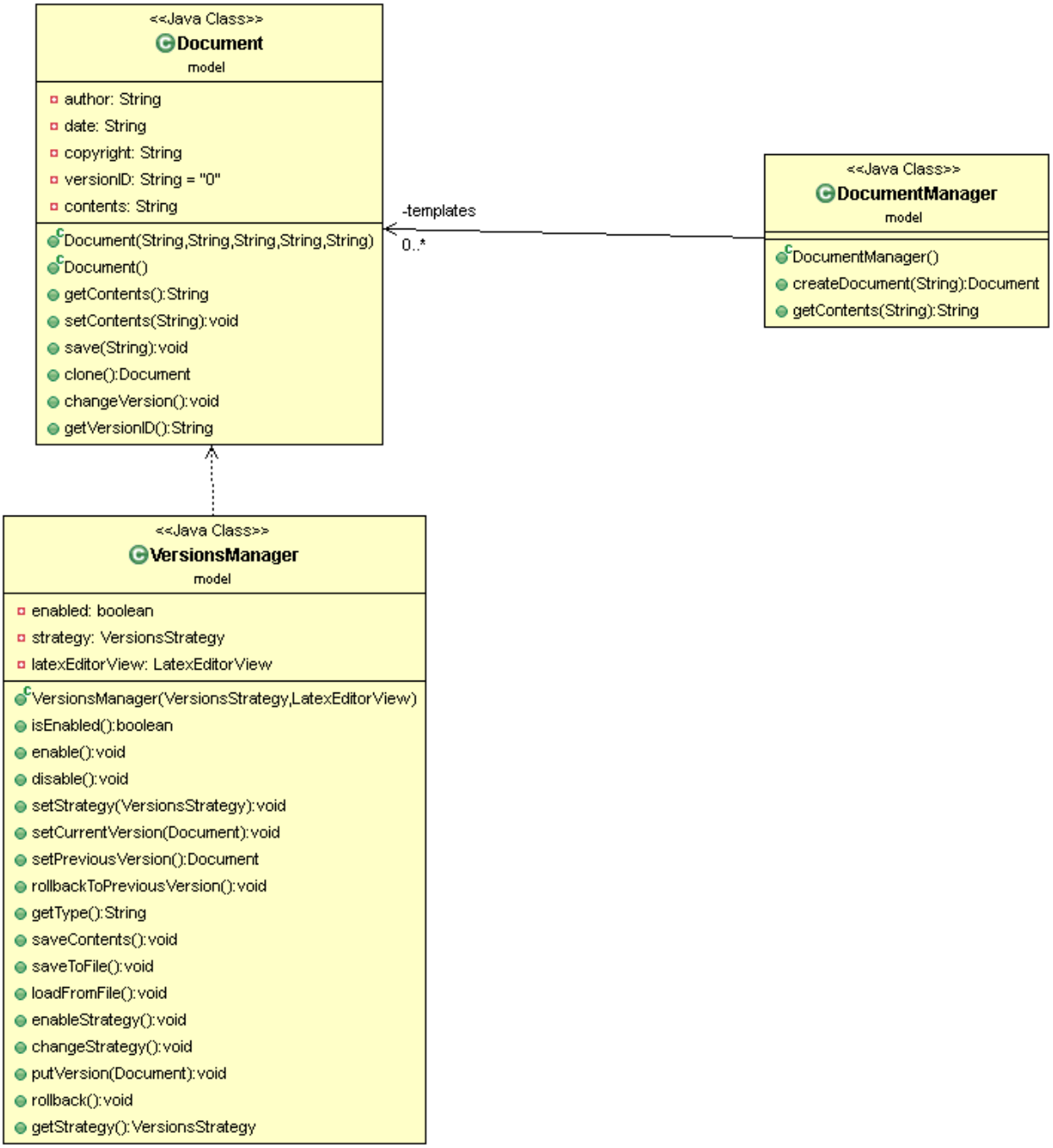
UML class diagram for the package controller:



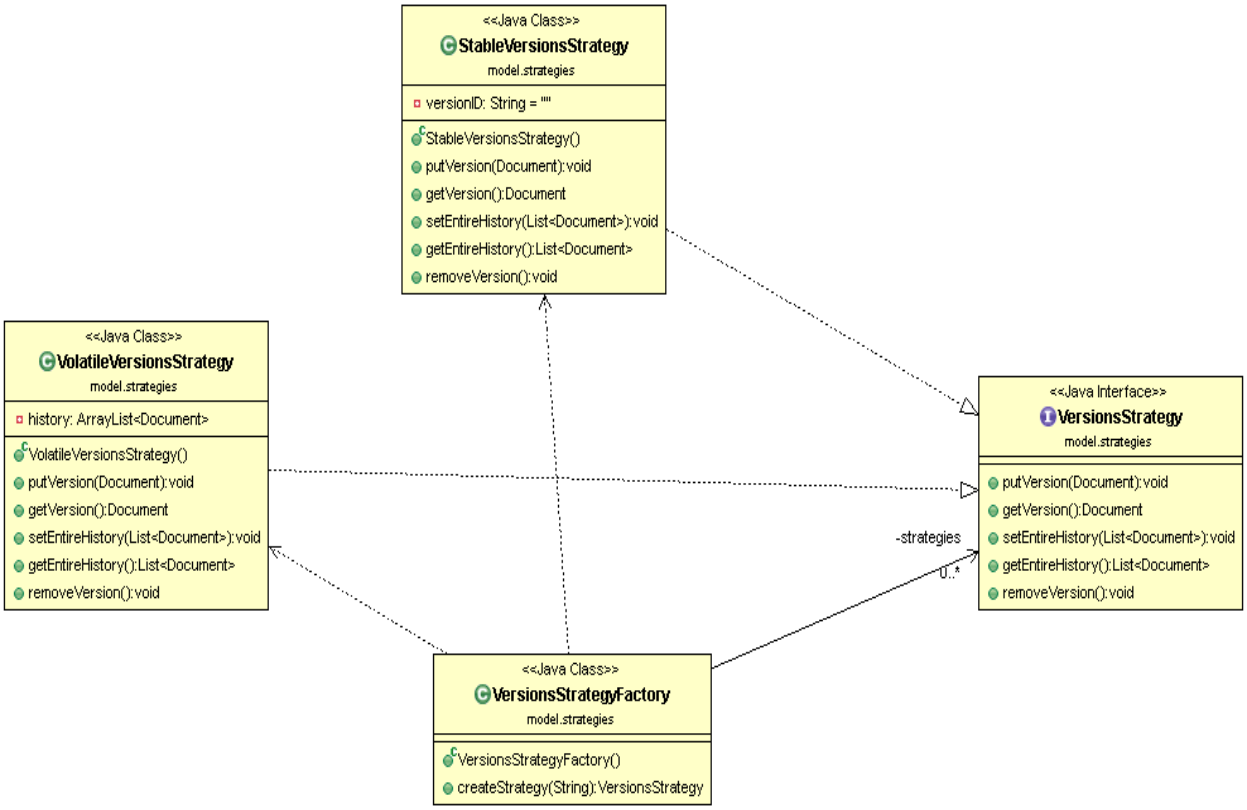
UML class diagram for the package controller.commands:



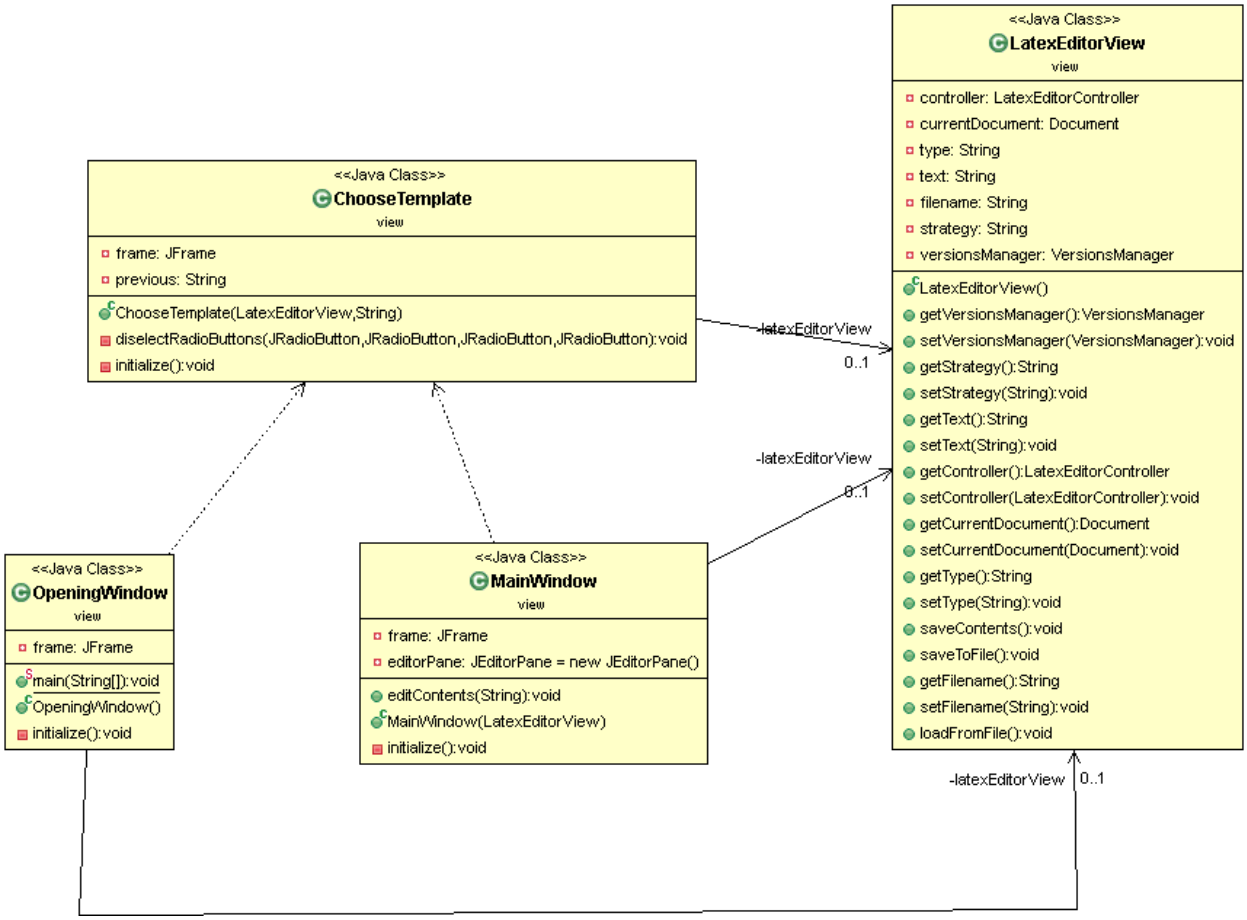
UML class diagram for the package model:



UML class diagram for the package model.strategies:



UML class diagram for the package view:


<<Java Class>>
LatexEditorView
view
controller: LatexEditorController
currentDocument: Document
type: String
text: String
filename: String
strategy: String
versionsManager: VersionsManager
LatexEditorView()
getVersionsManager():VersionsManager
setVersionsManager(VersionsManager):void
getStrategy():String
setStrategy(String):void
getText():String
setText(String):void
getController():LatexEditorController
setController(LatexEditorController):void
getCurrentDocument():Document
setCurrentDocument(Document):void
getType():String
setType(String):void
saveContents():void
saveToFile():void
getFilename():String
setFilename(String):void
loadFromFile():void
<<Java Class>>
ChooseTemplate
view
frame: JFrame
previous: String
ChooseTemplate(LatexEditorView,String)
diselectRadioButtons(JRadioButton,JRadioButton,JRadioButton,JRadioButton):void
initialize():void
-latexEditorView
0..1
-latexEditorView
0..1
<<Java Class>>
OpeningWindow
view
frame: JFrame
main(String[]):void
OpeningWindow()
initialize():void
<<Java Class>>
MainWindow
view
frame: JFrame
editorPane: JEditorPane = new JEditorPane()
editContents(String):void
MainWindow(LatexEditorView)
initialize():void
-latexEditorView
0..1

IMPLEMENTATION

Class Name: LatexEditorController	
Responsibilities	**Collaborations**
○ This class is responsible for executing the required action by calling the "execute" method of the Command object. ○ This class creates a HashMap and stores the various Commands in it.	▪ CommandsFactory ▪ Command

Class Name: Command	
Responsibilities	**Collaborations**
○ This class is responsible for creating an interface, that implements the Command Pattern.	▪ AddLatexCommand ▪ ChangeVersionStrategyCommand ▪ CommandFactory ▪ CreateCommand ▪ DisableVersionsManagementCommand

	▪ EditCommand ▪ EnableVersionsManagmentCommand ▪ LoadCommand ▪ RollBackToPreviousVersionCommand ▪ SaveCommand

Class Name: CommandFactory	
Responsibilities	**Collaborations**
○ This class creates a method “createCommand” that creates and returns a specific Command object.	▪ AddLatexCommand ▪ ChangeVersionStrategyCommand ▪ CommandFactory ▪ CreateCommand ▪ DisableVersionsManagementCommand ▪ EditCommand ▪ EnableVersionsManagmentCommand ▪ LoadCommand

	▪ RollBackToPreviousVersionCommand ▪ SaveCommand ▪ DocumentManager ▪ VersionsManager

Class Name: CreateCommand	
Responsibilities	**Collaborations**
○ This class creates a method "createCommand" that creates and returns a specific Command object.	▪ CommandFactory ▪ Command ▪ CreateCommandTest ▪ DisableVersionsManagementCommandTest ▪ EditCommandTest ▪ EnableVersionsManagmentCommandTest ▪ LoadCommandTest ▪ RollBackToPreviousVersionCommandTest ▪ SaveCommandTest

	▪ Document ▪ DocumentManager ▪ VersionsManager

Class Name: CreateCommandTest	
Responsibilities	**Collaborations**
○ This class is responsible for testing the "CreateCommand" class, to ensure that it works properly.	▪ LatexEditorView (form controller package) ▪ DocumentManager (from model package) ▪ CreateCommand ▪ VersionsManager (from model.strategies package)

Class Name: EditCommand	
Responsibilities	**Collaborations**
○ This class implements from the Command interface. ○ This class is responsible for executing the method "saveContents" for a specific VersionsManager object.	▪ CommandFactory ▪ Command ▪ DisableVersionsManagementCommandTest ▪ EditCommandTest ▪ EnableVersionsManagmentCommandTest ▪ RollBackToPreviousVersionCommandTest ▪ VersionsManager (from model.strategies package)

Class Name: EditCommandTest	
Responsibilities	**Collaborations**
○ This class is responsible for testing the "EditCommand" class, to ensure that it works properly.	▪ LatexEditorView (form controller package) ▪ DocumentManager (from model package) ▪ EditCommand

	▪ VersionsManager (from model.strategies package) ▪ CreateCommand

Class Name: ChangeVersionStrategyCommand	
Responsibilities	**Collaborations**
○ This class implements from the Command interface. ○ This class is responsible for executing the method "changeStrategy" for a specific VersionsManager object.	▪ CommandFactory ▪ Command ▪ ChangeVersionStrategyCommand ▪ VersionsManager (from model.strategies package)

Class Name: ChangeVersionStrategyCommandTest	
Responsibilities	**Collaborations**
	▪ LatexEditorView (form controller package) ▪ DocumentManager (from model package)

○ This class is responsible for testing the "ChangeVersionStrategyCommand" class, to ensure that it works properly.	▪ ChangeVersionStrategyCommand ▪ VersionsManager (from model.strategies package) ▪ VersionsStrategy ▪ VolatileVersionsStrategy ▪ StableVersionsStrategy

Class Name: AddLatexCommand	
Responsibilities	**Collaborations**
○ This class implements from the Command interface. ○ This class is responsible for executing the method "saveContents" for a specific VersionsManager object.	▪ CommandFactory ▪ Command ▪ VersionsManager (from model.strategies package)

Class Name: SaveCommand	
Responsibilities	**Collaborations**
○ This class implements from the Command interface. ○ This class is responsible for executing the method "saveToFile" for a specific VersionsManager object.	▪ CommandFactory ▪ Command ▪ SaveCommandTest ▪ VersionsManager (from model.strategies package)

Class Name: SaveCommandTest	
Responsibilities	**Collaborations**
○ This class is responsible for testing the "SaveCommand" class, to ensure that it works properly.	▪ LatexEditorView (form controller package) ▪ DocumentManager (from model package) ▪ SaveCommand

	▪ VersionsManager (from model.strategies package) ▪ CreateCommand

Class Name: LoadCommand	
Responsibilities	**Collaborations**
○ This class implements from the Command interface. ○ This class is responsible for executing the method "loadFromFile" for a specific VersionsManager object. ○ This class is creating a "getVersionManger" method that is responsible for returning the created VersionsManager object. ○ This class is creating a "setVersionManger" method that is responsible for changing the value of the created VersionsManager object to a "new" one.	▪ CommandFactory ▪ Command ▪ LoadCommandTest ▪ VersionsManager (from model.strategies package)

Class Name: LoadCommandTest	
Responsibilities	**Collaborations**
○ This class is responsible for testing the "LoadCommand" class, to ensure that it works properly.	▪ LatexEditorView (form controller package) ▪ DocumentManager (from model package) ▪ LoadCommand ▪ VersionsManager (from model.strategies package) ▪ CreateCommand

Class Name: DisableVersionsManagementCommand	
Responsibilities	**Collaborations**
○ This class implements from the Command interface.	▪ CommandFactory ▪ Command ▪ DisableVersionsManagementCommandTest

○ This class is responsible for executing the method “loadFromFile” for a specific VersionsManager object. ○ This class is creating a “disable” method that is responsible for returning the created VersionsManager object.	▪ VersionsManager (from model.strategies package)

Class Name: DisableVersionsManagementCommandTest	
Responsibilities	**Collaborations**
○ This class is responsible for testing the “DisableVersionsManagemetCommand” class, to ensure that it works properly.	▪ LatexEditorView (form controller package) ▪ DocumentManager (from model package) ▪ DisableVersionsManagementCommand

	▪ VersionsManager (from model.strategies package) ▪ CreateCommand ▪ EditCommand ▪ VersionsStrategy ▪ VolatileVersionsStrategy ▪ StableVersionsStrategy

Class Name: RollBackToPreviousVersionCommand	
Responsibilities	**Collaborations**
○ This class implements from the Command interface. ○ This class is responsible for executing the method "rollback" for a specific VersionsManager object.	▪ CommandFactory ▪ Command ▪ RollBackToPreviousVersionCommandTest ▪ VersionsManager (from model.strategies package)

Class Name: RollBackToPreviousVersionCommandTest	
Responsibilities	**Collaborations**
○ This class is responsible for testing the "RollBackToPreviousVersionCommand" class, to ensure that it works properly.	▪ LatexEditorView (form controller package) ▪ DocumentManager (from model package) ▪ RollBackToPreviousVersionCommand ▪ VersionsManager (from model.strategies package) ▪ CreateCommand ▪ EditCommand ▪ EnableVersionsManagerCommand ▪ VersionsStartegy ▪ StableVersionsStrategy

Class Name: EnableVersionsManagementCommand	
Responsibilities	**Collaborations**
○ This class implements from the Command interface. ○ This class is responsible for executing the method "enableStrategy" for a specific VersionsManager object.	▪ CommandFactory ▪ Command ▪ EnableVersionsManagementCommandTest ▪ VersionsManager (from model.strategies package) ▪ RollBackToPreviousCommandTest

Class Name: EnableVersionsManagementCommandTest	
Responsibilities	**Collaborations**
○ This class is responsible for testing the "EnableVersionsManagemetCommand" class, to ensure that it works properly.	▪ LatexEditorView (form controller package) ▪ DocumentManager (from model package) ▪ EnableVersionsManagementCommand

	▪ VersionsManager (from model.strategies package) ▪ CreateCommand ▪ EditCommand ▪ VersionsStrategy ▪ VolatileVersionsStartegy ▪ StableVersionsStrategy

Class Name: Document	
Responsibilities	**Collaborations**
○ This class is responsible for creating the Prototype Pattern using the “clone” method, that returns a clone Document.	▪ DocumentManager ▪ VersionsManager

○ This class can also “use” the contents of a specific Document through out the methods “getContents”, “setContents”, “save”, “changeVersion” and “getVersion”.	

Class Name: DocumentManager	
Responsibilities	**Collaborations**
○ This class creates a HashMap and stores the various Template Prototypes in it. ○ This class is responsible for creating a method “createDocument”, that retrieves a Template Prototype from the HashMap and clones it, by calling the “clone” method.	▪ Document

Class Name: VersionsManager	
Responsibilities	**Collaborations**
○ This class is responsible for changing the “status” and the “contets” of the current Version through out the methods “isEnabled”, “enable”, “disable”, “setStrategy”, “setCurrentVersion”, “setPreviousVersion”, “rollbackToPreviousVersion”, “getType”, “saveContents”, “saveToFile”, “loadFromFile”, “enableStrategy”, “changeStrategy”, “putVersion”, “rollback”, “getStrategy”	▪ Document ▪ VersionsStrategy (from model.strategies package) ▪ LatexEditorView (from view package)

Class Name: VersionsStrategy	
Responsibilities	**Collaborations**
○ This class is responsible for creating an interface, that implements the Strategy Pattern.	▪ Document ▪ VersionsStrategyFactory ▪ VolatileVersionsStrategy ▪ StableVersionsStrategy

Class Name: VersionsStrategyFactory	
Responsibilities	**Collaborations**
○ This class creates a HashMap and stores the two type of Strategies in it. ○ This class is responsible for returning a specific strategy form the Hash map through the “createStrategy” method.	▪ StableVersionsStrategy ▪ VersionsStrategy ▪ VolatileVersionsStrategy

Class Name: VolatileVersionsStrategy	
Responsibilities	**Collaborations**
○ This class implements from the VersionsStrategy interface. ○ This class is responsible for changing and returning the contents of the specific Strategy Pattern through out the methods “putVersion”, “getVersion”, “setEntireHistory”, “getEntireHistrory” and “removeVersion”.	▪ VersionsStrategy ▪ VersionsStrategyFactory ▪ Document
Class Name: StableVersionsStrategy	
Responsibilities	**Collaborations**
○ This class implements from the VersionsStrategy interface.	▪ VersionsStrategy ▪ VersionsStrategyFactory ▪ Document

○ This class is responsible for changing and returning the contents of the specific Strategy Pattern through out the methods “putVersion”, “getVersion”, “setEntireHistory”, “getEntireHistrory” and “removeVersion”.	

Class Name: LatexEditorView	
Responsibilities	**Collaborations**
○ This class is responsible for connecting the “view” package with the “controller” and “model” packages in order to use the various methods.	▪ LatexEditorController ▪ VersionsManager ▪ MainWindow ▪ OpeningWindow ▪ ChooseTemplate ▪ Document

Class Name: MainWindow	
Responsibilities	**Collaborations**
○ This class is responsible for creating the main UI window.	▪ LatexEditorView ▪ ChooseTemplate

Class Name: OpeningWindow	
Responsibilities	**Collaborations**
○ This class is responsible for creating the first UI window, when the program is launched.	▪ LatexEditorView ▪ VersionsStrategy ▪ VolatileVersionsStrategy ▪ VersionsManager ▪ LatexEditorController ▪ ChooseTemplate

Class Name: ChooseTemplate	
Responsibilities	**Collaborations**
○ This class is responsible for creating the second UI window, to choose from the various Templates.	▪ LatexEditorView ▪ MainWindow ▪ OpeningWindow

QUALITY ASSESSMENT

- At the constructor of the class LatexEditorController we detect duplicated code for inserting the different type of Commands into the HashMap named “commands”.
- We detect that, at the Command classes, we have the same fields such as “VersionsManager” and “DocumentManager”.
- At the constructor of DocumentManager class, we detect duplicated code for inserting the different type of Templates into the “templates” HashMap.
- We detect that the method “getContents” of the DocumentManager class has really long code.

- In every test, for the classes ---Command, we detect that the main ---Command object, instead of being created by the CommandFactory class, it is created as stand alone.

- We detect that in the MainWindow class we have a method, “editContents” that is shouldn’t be there because of the nature of the MainWindow class that creates the main GUI Window. The method “editContets” has to do with the contents of a Document.

- We can see that the VersionsManager class is really long, in terms of functions, and does some functions through it’s methods such as “saveToFile” that is shouldn’t be able to do.

- In the VersionsManager class, we detect that the methods “changeStrategy” and “enableStrategy” have very complicated if statements.

- We detect that the VersionsManager class has some methods such as “getType”, “saveToFile”, “loadFromFile” etc that are used also in the LatexEditorView class.

- We detect that, in the VersionsManager class, we have some methods that do nothing. Like “rollbackToPreviousVersion”.

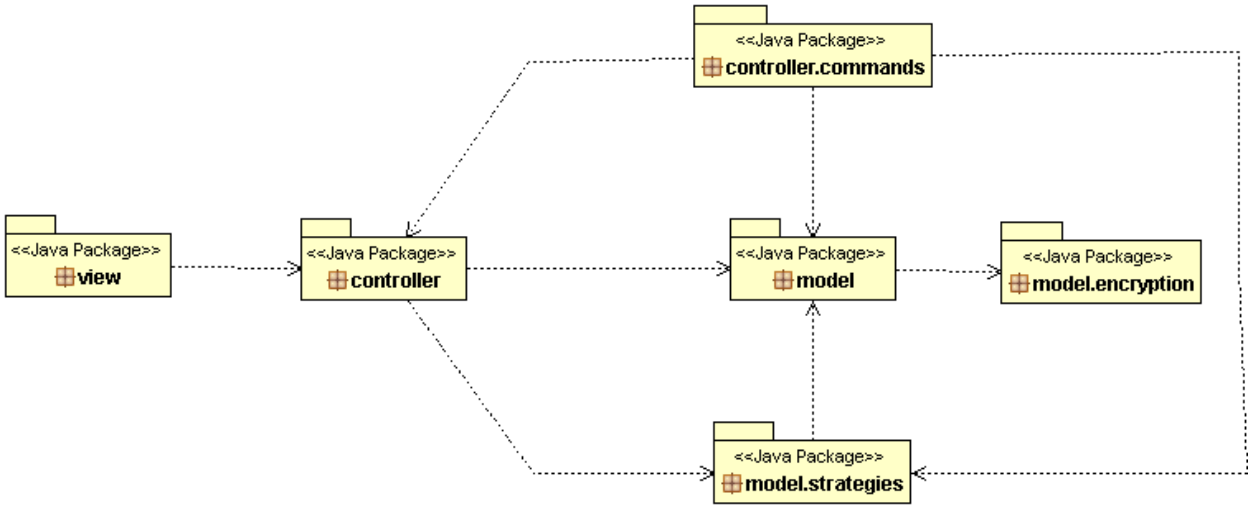
- We can see that the class LatexEditorView has some methods like “saveToFile”, “loadFromFile” etc that are not supposed to be there, due to the fact that this class is repsonisble for managing the GUI and not managing the contents of the current document.

- In the StableVersionsStrategy class, we detect that the method “setEntireHistroy” uses a variable with the name “i”. The same happens with the “getEntireHistory” and additionally we detect that an ArrayList is initialized as “List”.

- We detect that we have unnecessary comments at various classes of the project with the name “`TODO Auto-generated method stub”.`

RE-ENGINEERED DESIGN

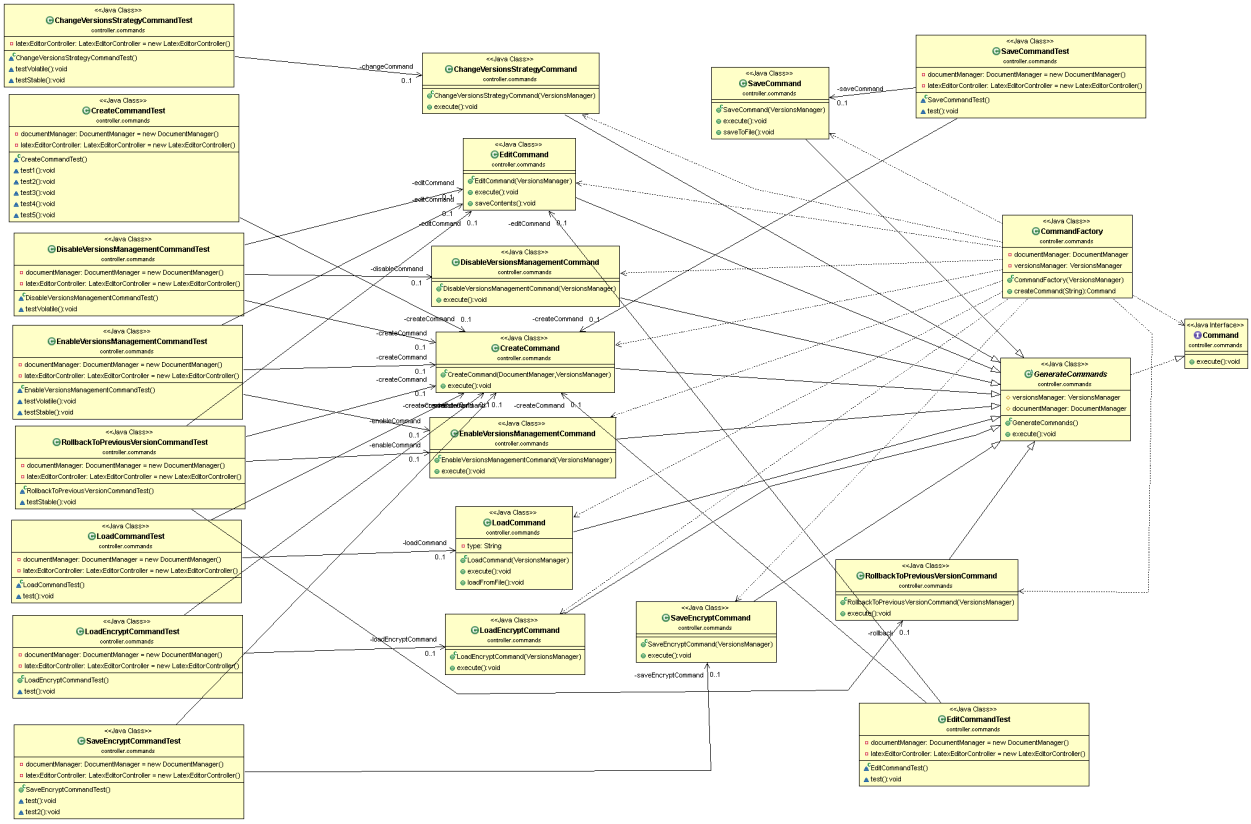
ARCHITECTURE



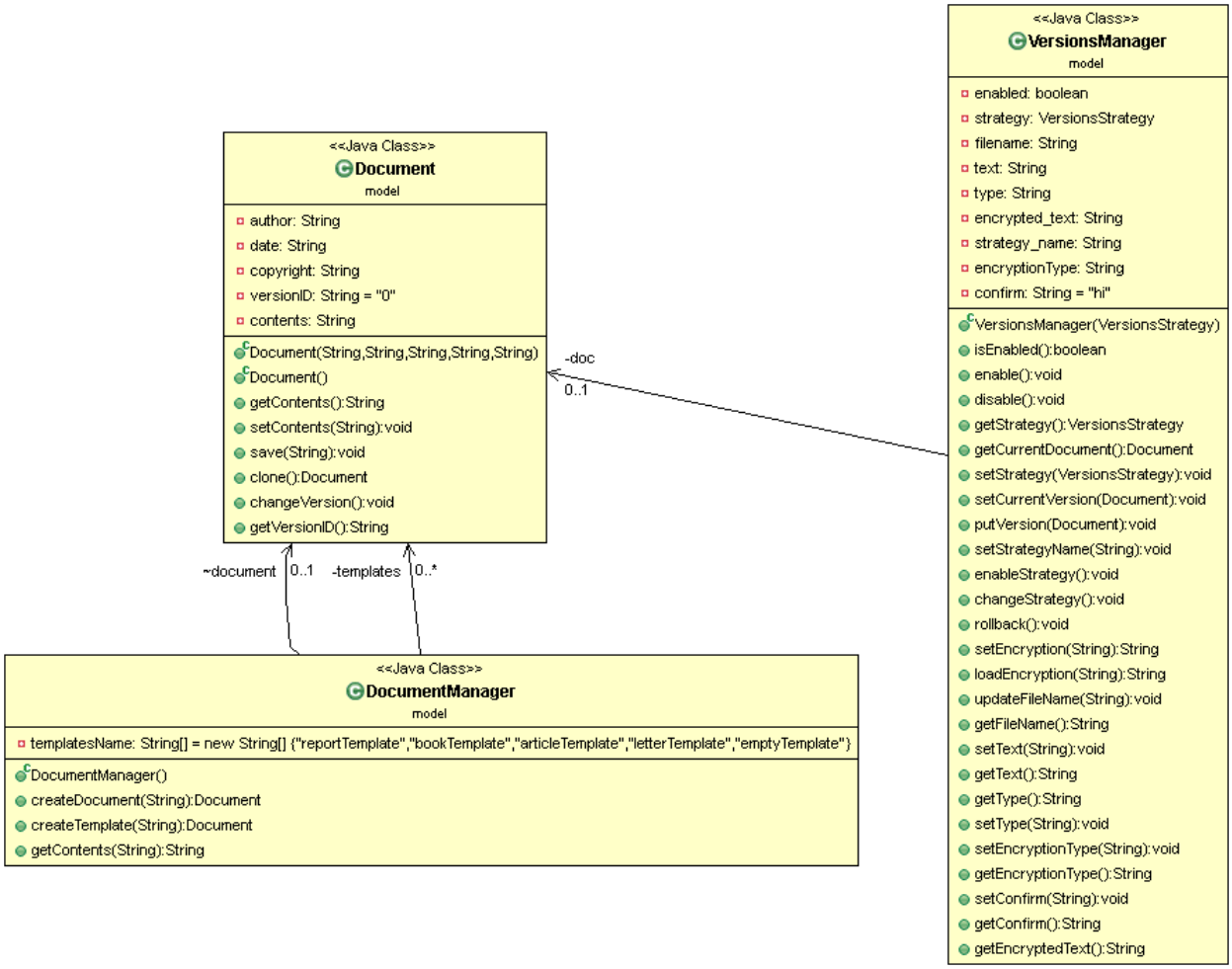
DETAILED DESIGN

UML class diagram for the package controller:

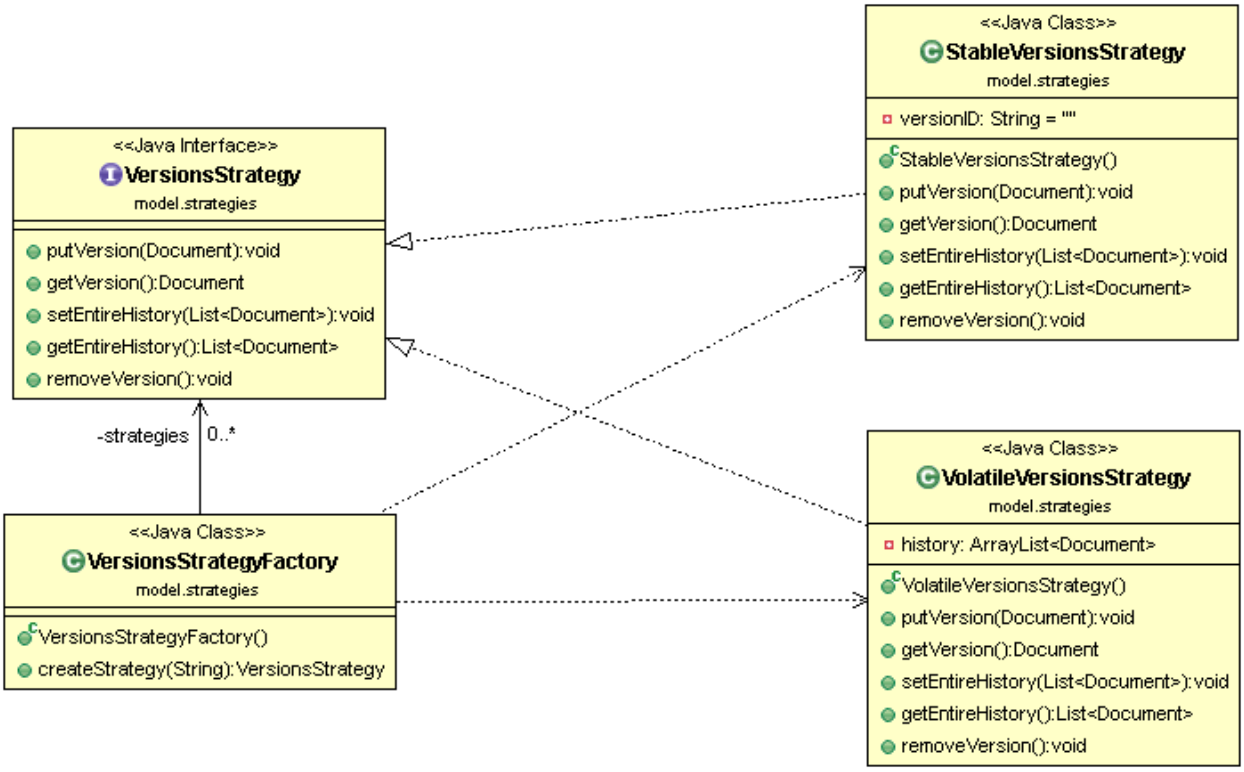
UML class diagram for the package controller.commands:



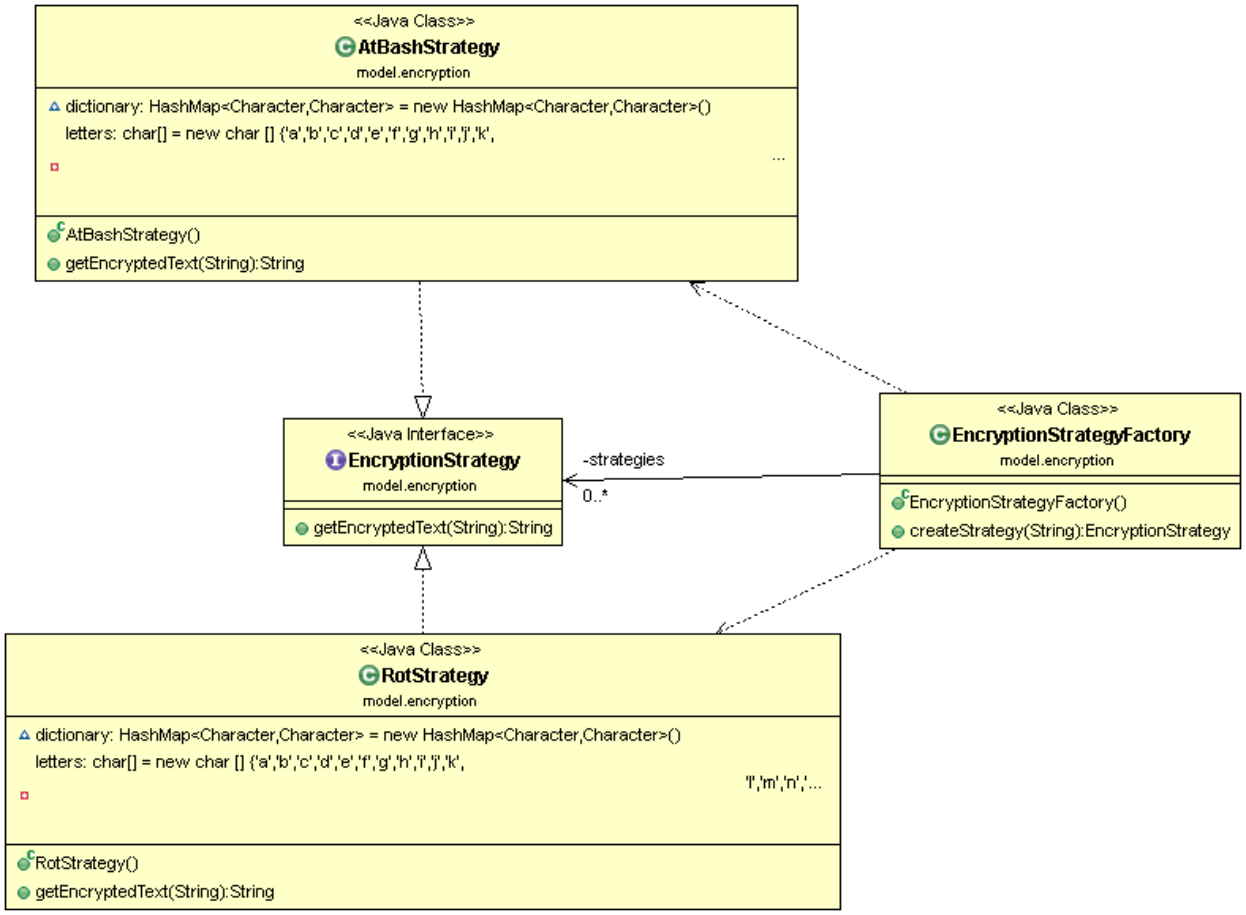
UML class diagram for the package model:



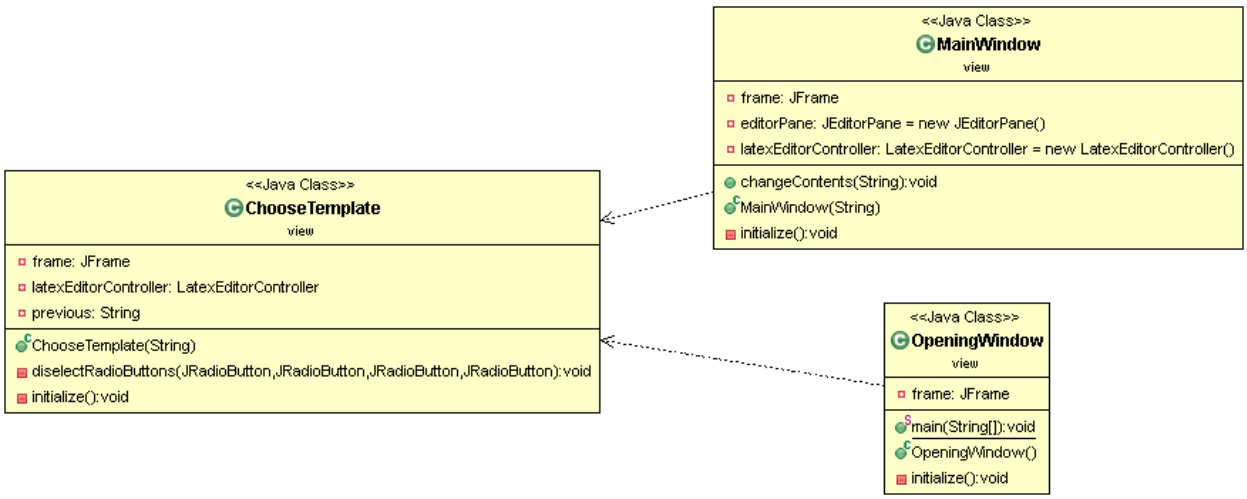
UML class diagram for the package model.strategies:



UML class diagram for the package model.encryption:



UML class diagram for the package view:



- *DocumentManager:* We have eliminated the duplicated code problem with the use of a HashMap. In the method “getContents” we moved the long strings at individual .txt files and used them from there.

- *VersionsManager:* We moved the necessary methods to the correct classes and eliminated the problem of the MiddleMan. We also removed the methods with DeadCode.

- *Commands*: We created an abstract class named GenerateCommands that implements the Command interface. Then we extended the Command classes from the GenerateCommand in order to use the fields of the GenerateCommand. With that, we have eliminated the DuplicateCode problem.

- *LatexEditorController:* We have moved every usefull class from the LatexEditorView to this class, in orded to eliminate the MiddleMan problem. We eliminated the problem of DuplicatedCode from the constructor with the use of a HashMap.

- *LatexEditorView:* This class was a general problem to the design of the program, so we removed it and moved all the methods and the fields to the classes: LatexEditorController, EditCommand, SaveCommand and LoadCommand.

- *MainWindow:* We moved the method “editContents” to the AddLatexCommand.

- *OpeningWindow:* We do not use anymore the LatexEditorController and we communicate directly with the MainWindow.

- *Encryption:* We have created the two new Classes SaveEncryptCommand and LoadEncryptCommand

- *CommandTests:* We updated the Test Commands to the new design pattern. Tests no longer use the LatexEditorView.

Class Name: LatexEditorController	
Responsibilities	**Collaborations**
○ This class is responsible for executing the required action by calling the "execute" method of the Command object. ○ This class creates a HashMap and stores the various Commands in it.	▪ CommandsFactory ▪ Command ▪ MainWindow ▪ AddLatexCommand

Class Name: Command	
Responsibilities	**Collaborations**
○ This class is responsible for creating an interface, that implements the Command Pattern.	▪ AddLatexCommand ▪ ChangeVersionStrategyCommand ▪ CommandFactory ▪ CreateCommand ▪ DisableVersionsManagementCommand

	▪ EditCommand ▪ EnableVersionsManagmentCommand ▪ LoadCommand ▪ RollBackToPreviousVersionCommand ▪ SaveCommand ▪ SaveEncryptCommand ▪ LoadEncryptCommand

Class Name: CommandFactory	
Responsibilities	**Collaborations**
○ This class creates a method "createCommand" that creates and returns a specific Command object.	▪ AddLatexCommand ▪ ChangeVersionStrategyCommand ▪ CommandFactory ▪ CreateCommand ▪ DisableVersionsManagementCommand ▪ EditCommand

	▪ EnableVersionsManagmentCommand ▪ LoadCommand ▪ RollBackToPreviousVersionCommand ▪ SaveCommand ▪ SaveEncryptCommand ▪ LoadEncryptCommand ▪ DocumentManager ▪ VersionsManager

Class Name: CreateCommand	
Responsibilities	**Collaborations**
○ This class creates a method "createCommand" that creates and returns a specific Command object.	▪ CommandFactory ▪ Command ▪ CreateCommandTest ▪ DisableVersionsManagementCommandTest

	▪ EditCommandTest ▪ EnableVersionsManagmentCommandTest ▪ LoadCommandTest ▪ RollBackToPreviousVersionCommandTest ▪ SaveCommandTest ▪ SaveEncryptCommandTest ▪ LoadEncryptCommandTest ▪ Document ▪ DocumentManager ▪ VersionsManager

Class Name: CreateCommandTest	
Responsibilities	**Collaborations**
	▪ LatexEditorController (form controller package)

○ This class is responsible for testing the "CreateCommand" class, to ensure that it works properly.	▪ DocumentManager (from model package) ▪ CreateCommand

Class Name: EditCommand	
Responsibilities	**Collaborations**
○ This class implements from the Command interface. ○ This class is responsible for executing the method "saveContents" for a specific VersionsManager object.	▪ CommandFactory ▪ Command ▪ DisableVersionsManagementCommandTest ▪ EditCommandTest ▪ EnableVersionsManagmentCommandTest ▪ RollBackToPreviousVersionCommandTest

Class Name: EditCommandTest	
Responsibilities	**Collaborations**
○ This class is responsible for testing the "EditCommand" class, to ensure that it works properly.	▪ LatexEditorController (form controller package) ▪ DocumentManager (from model package) ▪ EditCommand ▪ CreateCommand

Class Name: ChangeVersionStrategyCommand	
Responsibilities	**Collaborations**
○ This class implements from the Command interface. ○ This class is responsible for executing the method "changeStrategy" for a specific VersionsManager object.	▪ CommandFactory ▪ Command ▪ ChangeVersionStrategyCommand ▪ VersionsManager (from model.strategies package)

Class Name: ChangeVersionStrategyCommandTest

Responsibilities	Collaborations
○ This class is responsible for testing the "ChangeVersionStrategyCommand" class, to ensure that it works properly.	▪ LatexEditorController (form controller package) ▪ DocumentManager (from model package) ▪ ChangeVersionStrategyCommand ▪ VersionsStrategy ▪ VolatileVersionsStrategy ▪ StableVersionsStrategy

Class Name: AddLatexCommand	
Responsibilities	**Collaborations**
○ This class implements from the Command interface.	▪ CommandFactory ▪ Command

○ This class is responsible for executing the method "saveContents" for a specific VersionsManager object.	▪ VersionsManager (from model.strategies package) ▪ EditCommand

Class Name: SaveCommand	
Responsibilities	**Collaborations**
○ This class implements from the Command interface. ○ This class is responsible for executing the method "saveToFile" for a specific VersionsManager object.	▪ CommandFactory ▪ Command ▪ SaveCommandTest ▪ VersionsManager (from model.strategies package)

Class Name: SaveCommandTest	
Responsibilities	**Collaborations**
	▪ LatexEditorController (form controller package)

○ This class is responsible for testing the “SaveCommand” class, to ensure that it works properly.	▪ DocumentManager (from model package) ▪ SaveCommand ▪ CreateCommand

Class Name: LoadCommand	
Responsibilities	**Collaborations**
○ This class implements from the Command interface. ○ This class is responsible for executing the method “loadFromFile” for a specific VersionsManager object. ○ This class is creating a “getVersionManger” method that is responsible for returning the created VersionsManager object.	▪ CommandFactory ▪ Command ▪ LoadCommandTest ▪ VersionsManager (from model.strategies package)

○ This class is creating a "setVersionManger" method that is responsible for changing the value of the created VersionsManager object to a "new" one.	

Class Name: LoadCommandTest	
Responsibilities	**Collaborations**
○ This class is responsible for testing the "LoadCommand" class, to ensure that it works properly.	▪ LatexEditorController (form controller package) ▪ DocumentManager (from model package) ▪ LoadCommand ▪ CreateCommand

Class Name: DisableVersionsManagementCommand	
Responsibilities	**Collaborations**

○ This class implements from the Command interface. ○ This class is responsible for executing the method “loadFromFile” for a specific VersionsManager object. ○ This class is creating a “disable” method that is responsible for returning the created VersionsManager object.	▪ CommandFactory ▪ Command ▪ DisableVersionsManagementCommandTest ▪ VersionsManager (from model.strategies package)

Class Name: DisableVersionsManagementCommandTest	
Responsibilities	**Collaborations**
	▪ LatexEditorController (form controller package) ▪ DocumentManager (from model package)

○ This class is responsible for testing the "DisableVersionsManagemetCommand" class, to ensure that it works properly.	▪ DisableVersionsManagementCommand ▪ CreateCommand ▪ EditCommand ▪ VersionsStrategy ▪ VolatileVersionsStrategy ▪ StableVersionsStrategy

Class Name: RollBackToPreviousVersionCommand	
Responsibilities	**Collaborations**
○ This class implements from the Command interface. ○ This class is responsible for executing the method "rollback" for a specific VersionsManager object.	▪ CommandFactory ▪ Command ▪ RollBackToPreviousVersionCommandTest ▪ VersionsManager (from model.strategies package)

Class Name: RollBackToPreviousVersionCommandTest	
Responsibilities	**Collaborations**
○ This class is responsible for testing the "RollBackToPreviousVersionCommand" class, to ensure that it works properly.	▪ LatexEditorController (form controller package) ▪ DocumentManager (from model package) ▪ RollBackToPreviousVersionCommand ▪ CreateCommand ▪ EditCommand ▪ EnableVersionsManagerCommand ▪ VersionsStartegy ▪ StableVersionsStrategy

Class Name: EnableVersionsManagementCommand	
Responsibilities	**Collaborations**
○ This class implements from the Command interface. ○ This class is responsible for executing the method "enableStrategy" for a specific VersionsManager object.	▪ CommandFactory ▪ Command ▪ EnableVersionsManagementCommandTest ▪ VersionsManager (from model.strategies package) ▪ RollBackToPreviousCommandTest

Class Name: EnableVersionsManagementCommandTest	
Responsibilities	**Collaborations**
○ This class is responsible for testing the "EnableVersionsManagemetCommand" class, to ensure that it works properly.	▪ LatexEditorControlller (form controller package) ▪ DocumentManager (from model package) ▪ EnableVersionsManagementCommand

	▪ CreateCommand ▪ EditCommand ▪ VersionsStrategy ▪ VolatileVersionsStartegy ▪ StableVersionsStrategy

Class Name: Document	
Responsibilities	**Collaborations**
○ This class is responsible for creating the Prototype Pattern using the “clone” method, that returns a clone Document. ○ This class can also “use” the contents of a specific Document through out the methods “getContents”, “setContents”, “save”, “changeVersion” and “getVersion”.	▪ DocumentManager ▪ VersionsManager

Class Name: DocumentManager	
Responsibilities	**Collaborations**
○ This class creates a HashMap and stores the various Template Prototypes in it. ○ This class is responsible for creating a method "createDocument", that retrieves a Template Prototype from the HashMap and clones it, by calling the "clone" method.	▪ Document ▪ controller.command package

Class Name: VersionsManager	
Responsibilities	**Collaborations**
	▪ Document ▪ VersionsStrategy (from model.strategies package)

○ This class is responsible for changing the “status” and the “contets” of the current Version .	▪ LatexEditorController

Class Name: VersionsStrategy	
Responsibilities	**Collaborations**
○ This class is responsible for creating an interface, that implements the Strategy Pattern.	▪ Document ▪ VersionsStrategyFactory ▪ VolatileVersionsStrategy ▪ StableVersionsStrategy

Class Name: VersionsStrategyFactory	
Responsibilities	**Collaborations**
○ This class creates a HashMap and stores the two type of Strategies in it. ○ This class is responsible for returning a specific strategy form the Hash map through the “createStrategy” method.	▪ StableVersionsStrategy ▪ VersionsStrategy ▪ VolatileVersionsStrategy

Class Name: VolatileVersionsStrategy	
Responsibilities	**Collaborations**
○ This class implements from the VersionsStrategy interface.	▪ VersionsStrategy ▪ VersionsStrategyFactory ▪ Document ▪ ChangeVersionsStrategyCommandTest ▪ DisableVersionsManagementCommandTest

○ This class is responsible for changing and returning the contents of the specific Strategy Pattern through out the methods “putVersion”, “getVersion”, “setEntireHistory”, “getEntireHistrory” and “removeVersion”.	▪ EnableVersionsManagementCommandTest ▪ RollbackToPreviousVersionCommandTest
Class Name: StableVersionsStrategy	
Responsibilities	**Collaborations**
○ This class implements from the VersionsStrategy interface.	▪ VersionsStrategy ▪ VersionsStrategyFactory ▪ Document ▪ ChangeVersionsStrategyCommandTest ▪ DisableVersionsManagementCommandTest ▪ EnableVersionsManagementCommandTest ▪ RollbackToPreviousVersionCommandTest

○ This class is responsible for changing and returning the contents of the specific Strategy Pattern through out the methods “putVersion”, “getVersion”, “setEntireHistory”, “getEntireHistrory” and “removeVersion”.	

Class Name: MainWindow	
Responsibilities	**Collaborations**
○ This class is responsible for creating the main UI window.	▪ LatexEditorController ▪ ChooseTemplate

Class Name: OpeningWindow

Responsibilities	Collaborations
○ This class is responsible for creating the first UI window, when the program is launched.	▪ ChooseTemplate

Class Name: ChooseTemplate	
Responsibilities	**Collaborations**
○ This class is responsible for creating the second UI window, to choose from the various Templates.	▪ MainWindow ▪ OpeningWindow

Class Name: EncryptionStrategy	
Responsibilities	**Collaborations**
○ This class is responsible for creating an interface,which is responsible for the encryption strategy.	▪ AtBashStrategy ▪ RotStrategy

Class Name:AtBashStategy	
Responsibilities	**Collaborations**
○ This class is responsible for implementing the atbash encryption algorithm.	▪ EncryptionStrategy ▪ EncryptionStrategyFactory

Class Name: RotStrategy

Responsibilities	Collaborations
○ This class is responsible implementing the rot-13 algorithm.	▪ EncryptionStartegy ▪ EncryptionStrategyFactory

Class Name: EncryptionStrategyFactory	
Responsibilities	**Collaborations**
○ This class is responsible for creating encryptionstartegy objects for example atbash object	▪ AtBashStrategy ▪ RotStrategy

Class Name: SaveEncryptCommand	
Responsibilities	**Collaborations**
	▪ VersionsManager

○ This class is responsible for executing the encrypt method . ○ This class implements the Command Interface	▪ Command

Class Name: SaveEncryptCommandTest	
Responsibilities	**Collaborations**
○ This class is responsible for testing the SaveEncryptCommand class with the different encrypt types	▪ LatexEditorController ▪ DocumentManager ▪ CreateCommand ▪ SaveEncryptCommand

Class Name: LoadEncryptCommand	
Responsibilities	**Collaborations**
○ This class is responsible for loading the specified encrypted tex file and it decrypts it. ○ This class implements the Command Interface.	▪ VersionsManager ▪ Command

Class Name: LoadEncryptCommandTest	
Responsibilities	**Collaborations**
○ This class is responsible for testing the LoadEncryptCommand.	▪ LatexEditorController ▪ DocumentManager ▪ CreateCommand ▪ LoadEncryptCommand