LATEX EDITOR

DESIGN RECOVERY AND QUALITY ASSESSMENT REPORT

VERSION <1.0>

Βασιλειάδης Μιλτιάδης 2944

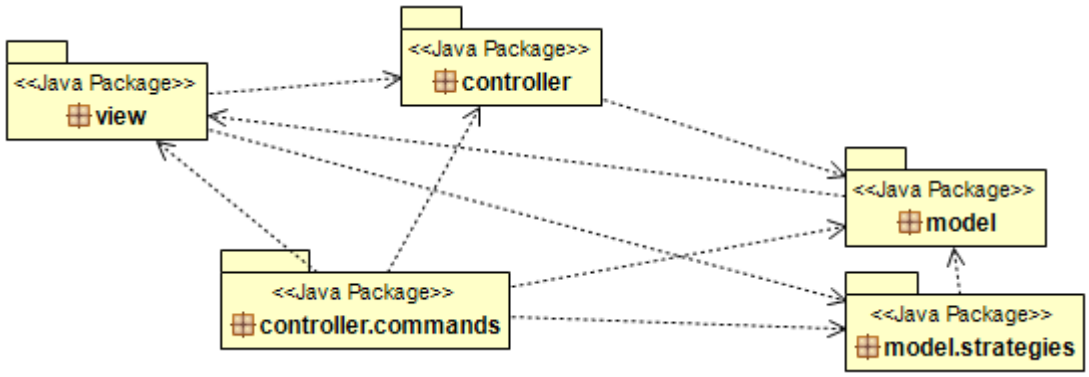
Θωμά Αθανάσιος 2979

INTRODUCTION

The goal of this project is to reengineer a Java application. At a glance, the objective of this project is to develop a simple Latex editor for inexperienced Latex users. 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 like a programming process as it involves the proper usage of Latex commands which are embedded in the document contents. The goal of the Latex 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.

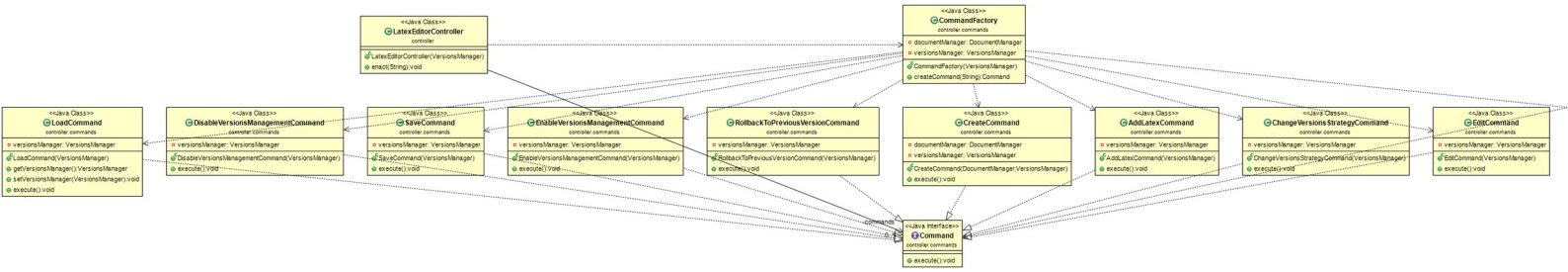
DESIGN RECOVERY

ARCHITECTURE

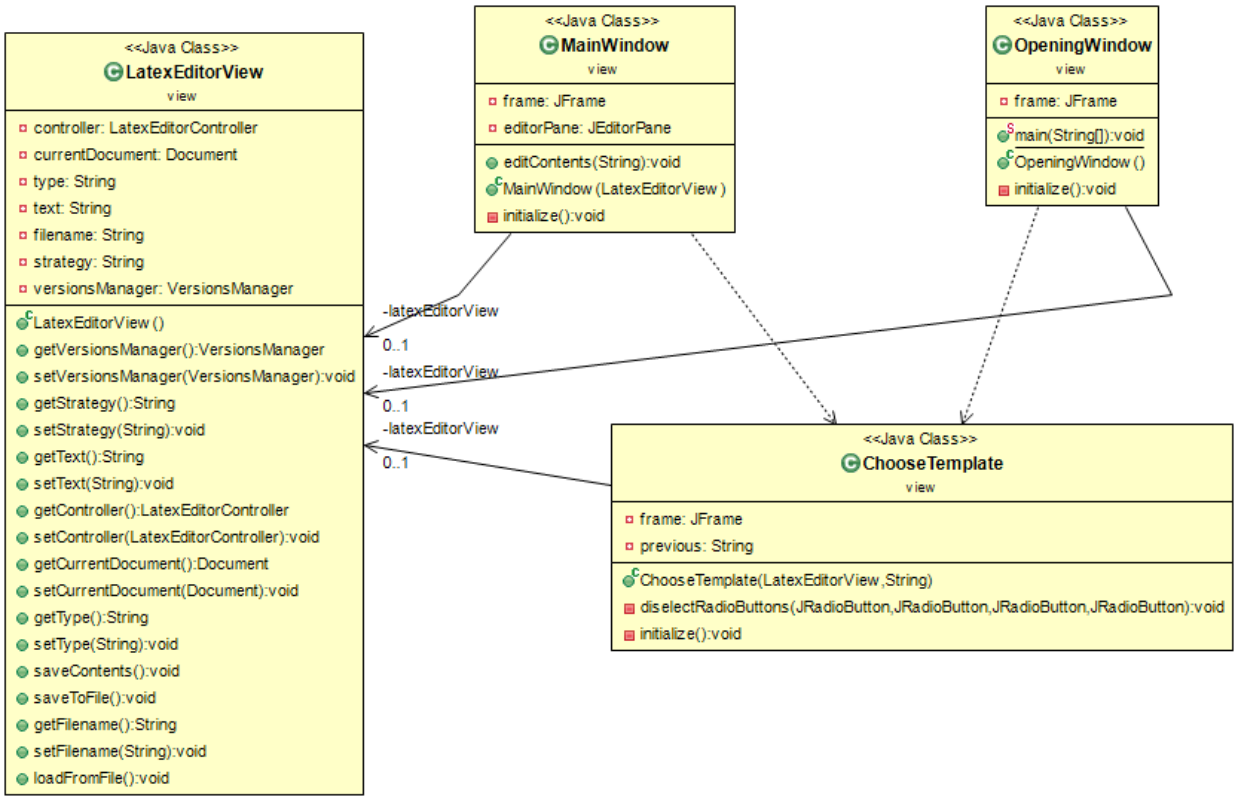


DETAILED DESIGN

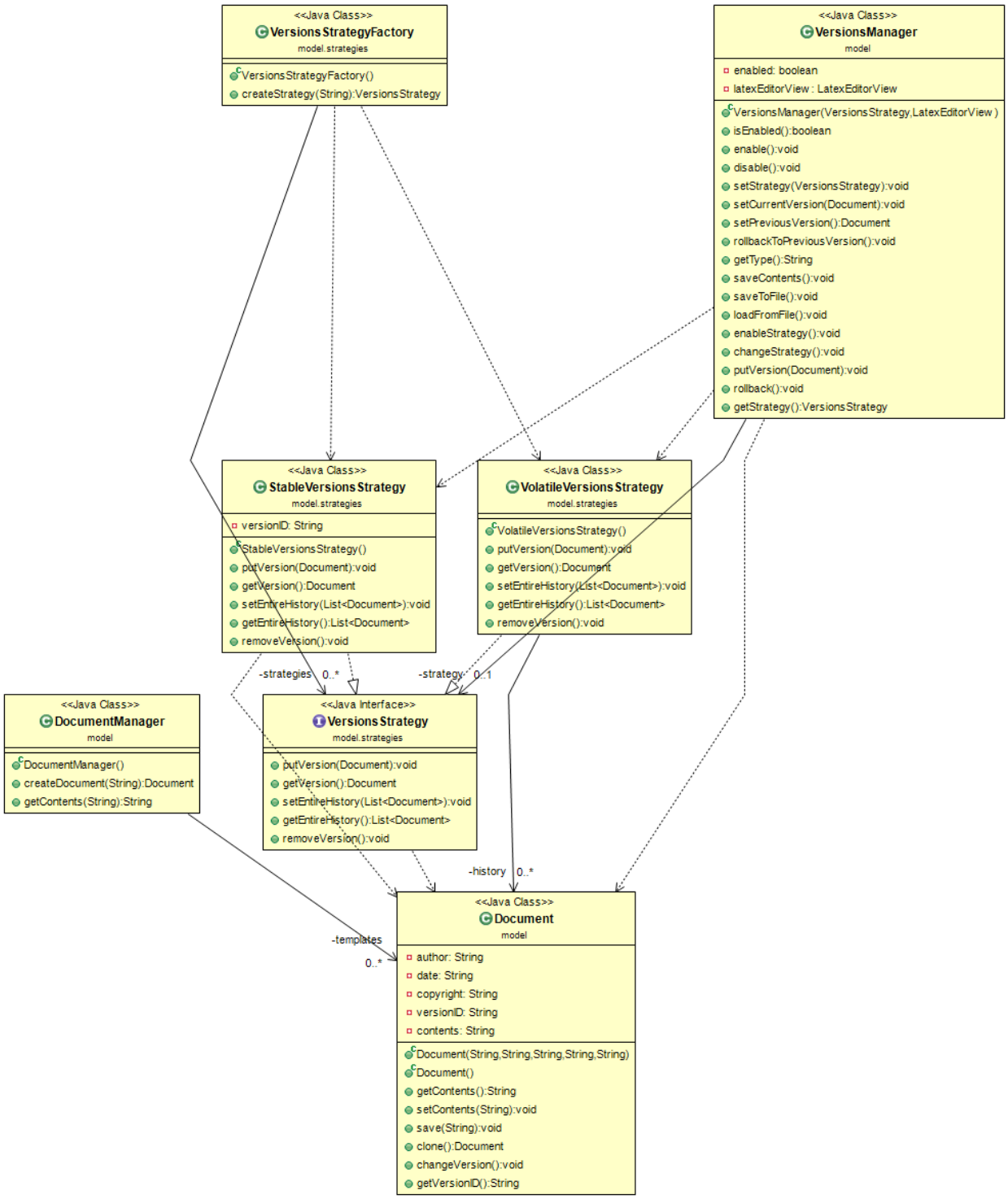
- controller package



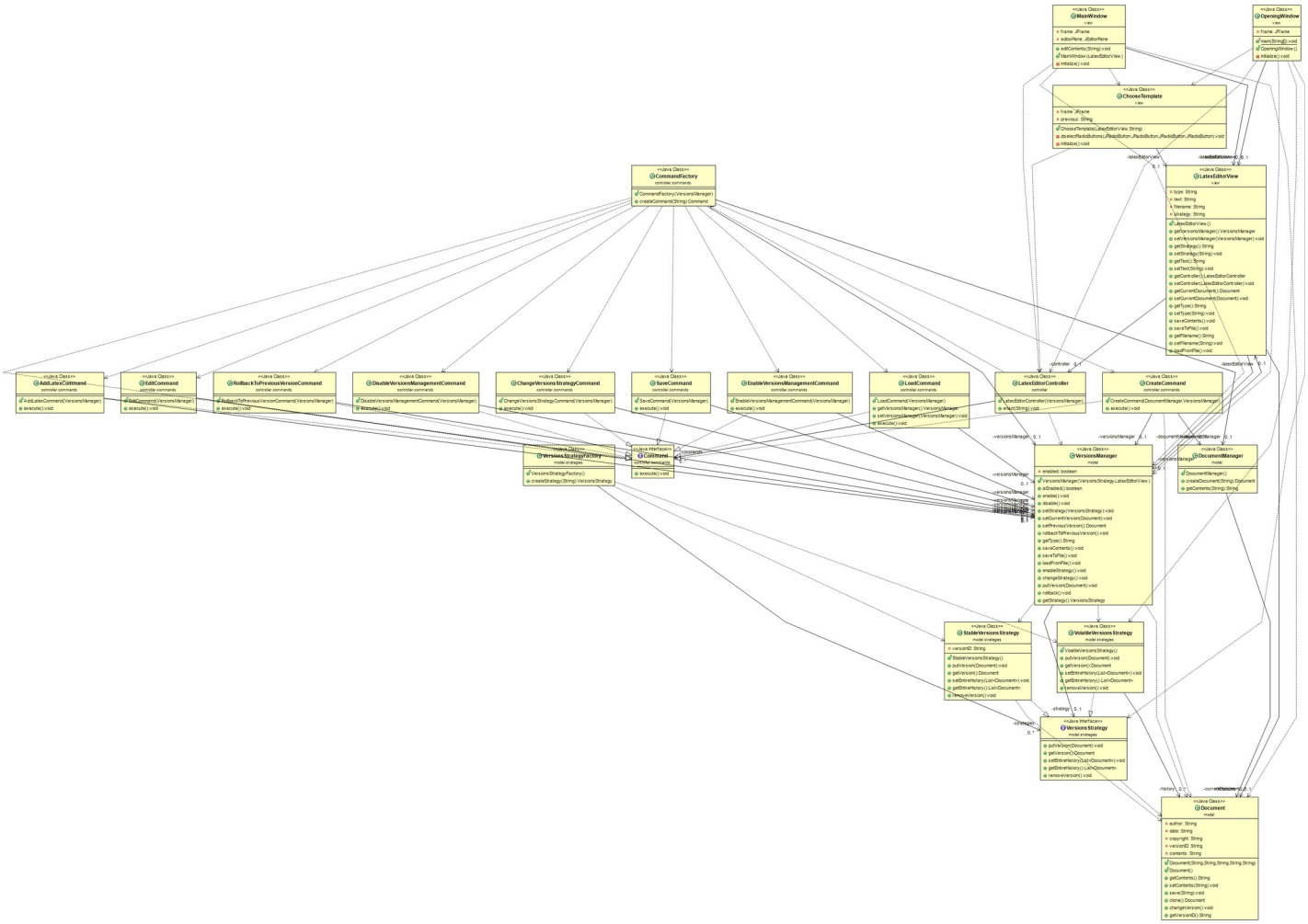
- view package



- model package



- overall graph



IMPLEMENTATION

Document	
• Represents the main model of the data of this software • Handles writing files to the filesystem • Creates clone instances	• None

Document Manager	
• Handles creation of new objects of type Document • Handles template creation	• Document

VersionsManager	
• Controls the versioning system. • Enables/disables the versioning system • Sets the mode / strategy of operation (Volatile/Stable) • Communicates with the front end to accomplish most functional requirements	• VersionsStrategy • LatexEditorView

Interface VersionsStrategy	
• Puts new version of document in the versioning system • gets previous version of document • sets entire history of documents • returns entire history of documents • removes latest version from the versioning system	• Document

VolatileVersionsStrategy	VersionsStrategy
• Stores document versions in Memory (ArrayList) • Returns most recent version stored. • Returns entire history of document versions (ArrayList) • Can "be fed" entire document history in form of ArrayList	• Document • VersionsManager

StableVersionsStrategy	VersionsStrategy
• Saves the document versions on the filesystem in .tex format • Returns previous version of Document • Returns entire document history from filesystem • Can "be fed" entire Document version history to be saved on the filesystem	• Document • VersionsManager

VersionsStrategyFactory	
• Creates objects of VolatileVersionStrategy or StableVersionStrategy based on Input	• VersionStrategy

LatexEditorController	
• Creates and stores command-type objects • Triggers commands according to input from the GUI	• LatexEditorView • Command (Interface) • CommandFactory

Interface Command	
• executes command	

CommandFactory	
• Creates instances of command classes that implement command interface	• VersionsManager • DocumentManager • EditCommand • AddLatexCommand • rollbacktoPreviousVersionCommand • SaveCommand • LoadCommand • CreateCommand • enableVersionsCommand • DisableVersionsCommand • ChangeVersionsStrategy

EnableVersionsManager	
• Enables versioning system	• VersionsManager

DisableVersionsManager	
• Disables versioning system	• VersionManager

AddLatexCommand	
• Calls VersionsManager to update contents from the text area to the current Document	• VersionsManager

CreateCommand	
• Gets Document manager to create a new document based on input (type) • Sets document created as current in VersionsManager	• DocumentManager • VersionsManager

EditCommand	
• Calls VersionsManager to update contents from the text area to the current Document	• VersionsManager

ChangeVersionsStrategyCommand	
• Calls VersionsManager to change versioning Strategy	• VersionsManager

LoadCommand	
• Calls VersionsManager to load document from file	• VersionsManager

SaveCommand	
• Calls Versions manager to save document to file	• VersionsManager

RollbacktoPreviousVersionCommand	
• calls VersionsManager to roll back to previous version of the document	• VersionsManager

OpeningWindow	
• Has the Main() method • Initializes the manager classes of the system • Sets volatile versioning strategy as default • Creates the first window of the GUI	• VersionsManager • VersionsStartegy • LatexEditorController • LatexEditorView

MainWindow	
• Adds Latex Commands (Strings) to the caret position • Asks LatexEditorView to update the current document • Updates the text area • Creates the main window of the GUI with the text area • Creates the menu items and calls LatexEditorView to trigger actions accordingly	• LatexEditorView • Document

ChooseTemplate	
• Creates the GUI window responsible for choosing a starting Latex Template (Book, Article etc)	• LatexEditorView

LatexEditorView	
• Has the current document • Is the mediator between front end and back end • Loads document from disk • Save document in disk • Decides what is the type of loaded document	• LatexEditorController • Document • VersionsManager

QUALITY ASSESSMENT

Problematic Methods:

- Document.clone() ~ creates shallow instead of deep copy.
- DocumentManager.constructor() ~ Document constructor is not used correctly, instead the contents of each document template is filled through setter method.
- VersionsManager.rollbackToPreviousVersion() ~ does nothing.
- StableVersionsStrategy.putVersion() ~ calls Document.save() method instead of implementing its own mechanism to write to the disk.
- LoadCommand ~ has setter and getter methods. Dead code.
- ChooseTemplate.initialize() ~ has duplicate code inside.
- LatexEditorView ~ has setter and getter methods. Dead code.

Classes with many responsibilities:

~Model Package

- Document: also has save() method.
- DocumentManager: getContents() method should be in Document class.
- VersionsManager: is a God class. It is not only responsible for the versioning system. This is the class that all the command classes send their job to. It also handles some of the communication with the GUI.

~View Package

- MainWindow: method editContents() should be in AddLatexCommand class.
- LatexEditorView: this is the “controller” class of the GUI package. Responsibilities such as loadFromFile() and saveContents() are shoved in here. Has some duplicate code.

Classes with very few responsibilities:

~Controller Package

- LatexEditorController: this class should be the one communicating with the GUI.

~Controller.Commands Package

- AddLatexCommand: the job of this class is implemented elsewhere. The whole class is a duplicate of EditCommand.
- EditCommand: doesn't do anything, just calls the VersionsManager.
- ChangeVersionsStrategyCommand: doesn't do anything, just passes calls VersionsManager.
- LoadCommand: doesn't do anything, just calls the VersionsManager.
- RollbackToPreviousVersionCommand: doesn't do anything, just calls the VersionsManager.
- SaveCommand: doesn't do anything, just calls the VersionsManager.