LATEX EDITOR

OVERALL REPORT

VERSION 2.0

Victor Megir 3026

TABLE OF CONTENTS

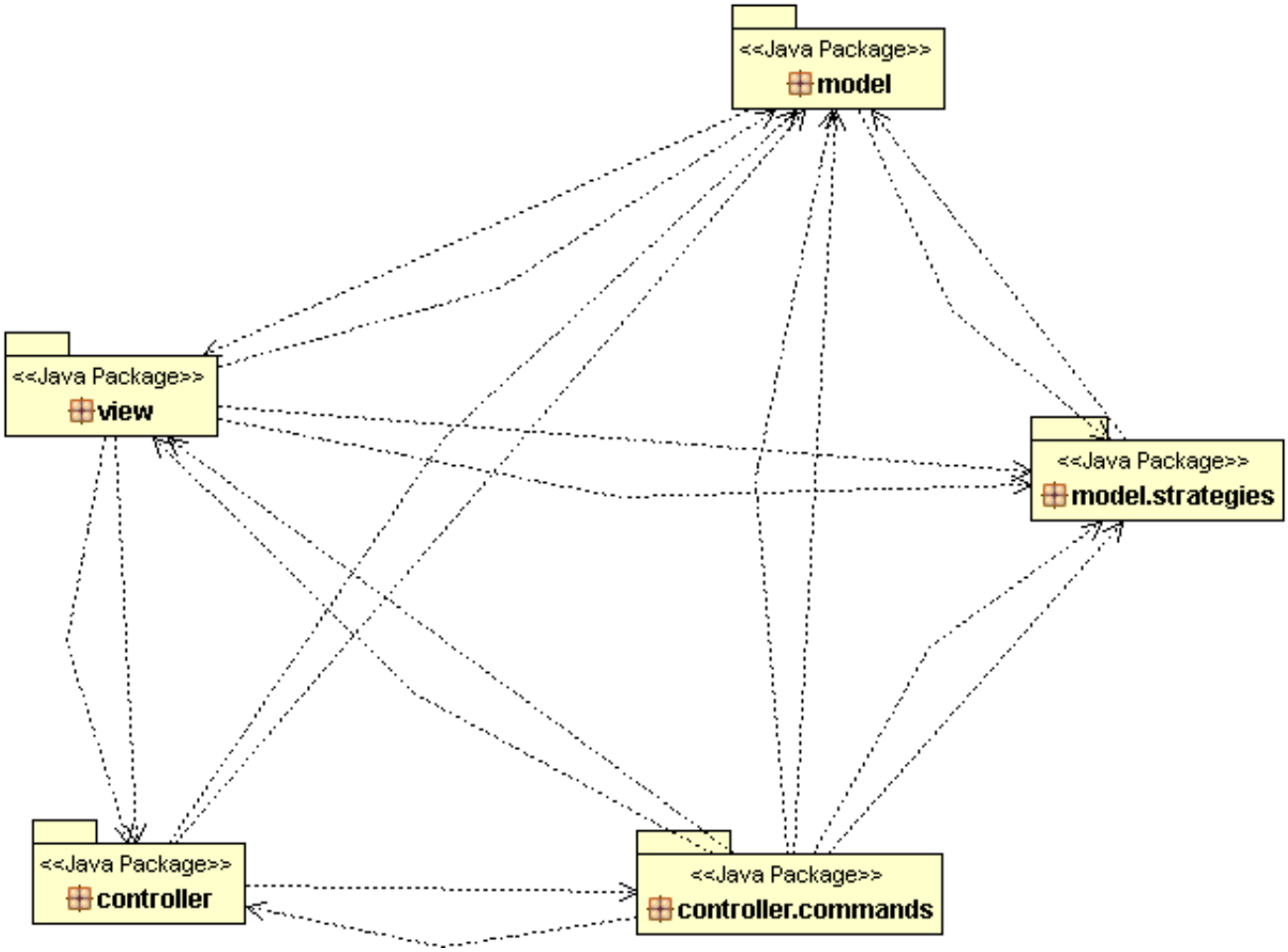
INTRODUCTION

The objectives of this phase of the project are to implement the refactorings needed to remove the code smells and to make the code easier to extend.

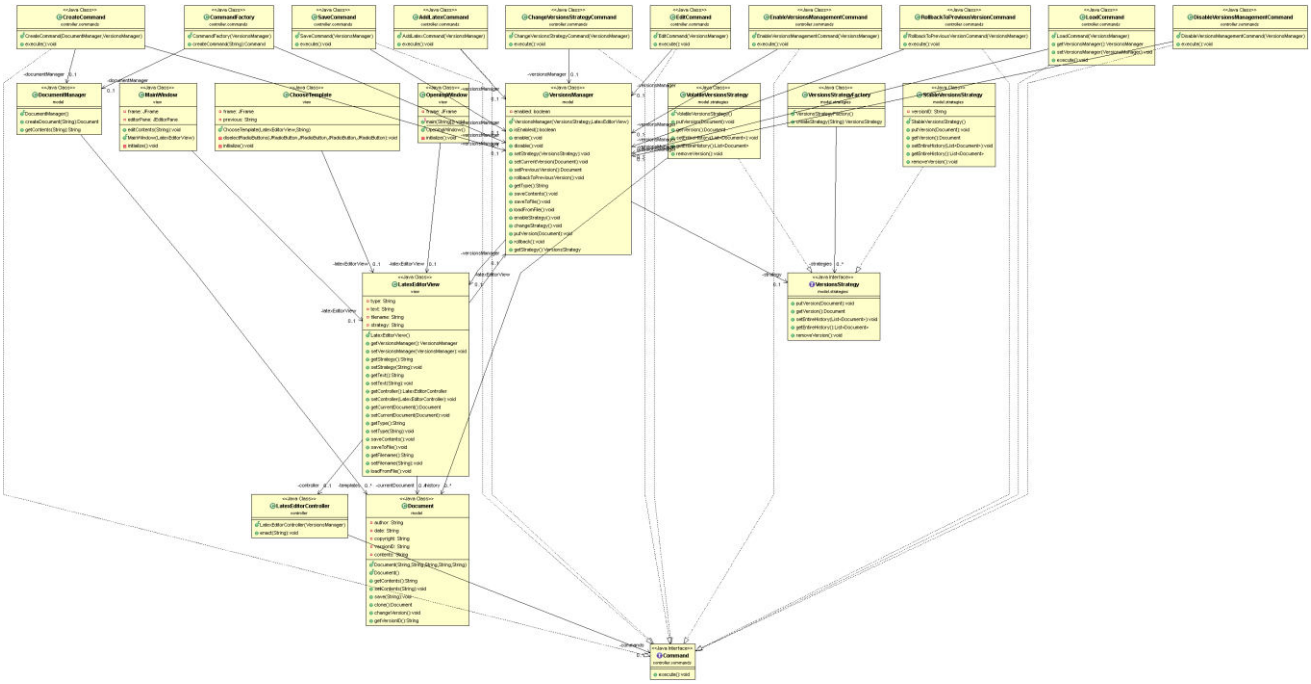
After the refactorings are finished, an additional two user stories are implemented

DESIGN RECOVERY

- ARCHITECTURE



- DETAILED DESIGN



- IMPLEMENTATION

Class Name: Command	
Responsibilities	**Collaborations**
• Interface for the method execute()	• AddLatexCommand • ChangeVersionsStrategyCommand • CreateCommand • DisableVersionsManagementCommand • EditCommand • EnableVersionsManagerCommand • LoadCommand • RollbackToPreviousVersionCommand • SaveCommand

• LatexEditorController

Class Name: AddLatexCommand	
Responsibilities	**Collaborations**
• Updates the contents of the document • Implements the Command Interface	• Command • VersionsManager

Class Name: EditCommand	
Responsibilities	**Collaborations**
• Updates the contents of the document • Implements the Command Interface	• Command • VersionsManager

Class Name: ChangeVersionsStrategyCommand	
Responsibilities	**Collaborations**
• Changes the version auto save strategy • Implements the Command Interface	• Command • VersionsManager

Class Name: CreateCommand

Responsibilities	Collaborations
• Creates the document given the chosen template • Implements the Command Interface	• Command • VersionsManager • DocumentManager

Class Name: LoadCommand	
Responsibilities	**Collaborations**
• Creates the document given the chosen file • Implements the Command Interface	• Command • VersionsManager

Class Name: DisableVersionsManagementCommand	
Responsibilities	**Collaborations**
• Disables the version tracking strategy • Implements the Command Interface	• Command • VersionsManager

Class Name: EnableVersionsManagementCommand	
Responsibilities	**Collaborations**
• Enables the version tracking strategy • Implements the Command Interface	• Command • VersionsManager

Class Name: RollbackToPreviousVersionCommand	
Responsibilities	**Collaborations**
• Load previous version of Document • Implements the Command Interface	• Command • VersionsManager

Class Name: SaveCommand	
Responsibilities	**Collaborations**
• Saves Document to file • Implements the Command Interface	• Command • VersionsManager

Class Name: CommandFactory	
Responsibilities	**Collaborations**
• Creates a command(not the Interface) given the type	• DocumentManager • VersionsManager • AddLatexCommand • ChangeVersionsStrategyCommand • CreateCommand • DisableVersionsManagementCommand • EditCommand • EnableVersionsManagementCommand • LoadCommand

• RollbackToPreviousVersionCommand • SaveCommand

Class Name: LatexEditorController	
Responsibilities	**Collaborations**
• Uses the CommandFactory to give a command to LatexEditorView	• Command • CommandFactory • LatexEditorView

Class Name: Document	
Responsibilities	**Collaborations**
• Represents the Latex Document	• DocumentManager • LatexEditorView • VolatileVersionStrategy • StableVersionStrategy

Class Name: DocumentManager	
Responsibilities	**Collaborations**
• Creates the Templates	• Document • CreateCommand • CommandFactory

Class Name: VersionsManager	
Responsibilities	**Collaborations**
• Holds a reference to the version strategy of the Document • Manages the version strategy	• Document • CreateCommand • CommandFactory • AddLatexCommand • ChangeVersionsStrategyCommand • CreateCommand • DisableVersionsManagementCommand • EditCommand • EnableVersionsManagerCommand • LoadCommand • RollbackToPreviousVersionCommand • SaveCommand • LatexEditorController • VersionsStrategy • LatexEditorView

Class Name: VersionsStrategy	
Responsibilities	**Collaborations**
• Interface for version tracking strategy related methods	• VersionStrategyFactory • StableVersionsStrategy

	• VolatileVersionsStrategy • VersionsManager

Class Name: StableVersionsStrategy	
Responsibilities	**Collaborations**
• Implements the VersionStrategy Interface • Saves version to file	• VersionStrategyFactory • Document • VersionsManager

Class Name: VolatileVersionsStrategy	
Responsibilities	**Collaborations**
• Implements the VersionStrategy Interface • Saves version to memory	• VersionStrategyFactory • Document • VersionsManager

Class Name: VersionsStrategyFactory	
Responsibilities	**Collaborations**
• Implements the VersionStrategy Interface • Saves version to memory	• VersionStrategy • StableVersionStrategy • VolatileVersionStrategy

Class Name: OpeningWindow	
Responsibilities	**Collaborations**
• Opens the starting window • If the uses chooses to, opens existing file Document	• LatexEditorView • VersionsStrategy • VersionsManager • VolatileVersionsStrategy • LatexEditorController

Class Name: ChooseTemplate	
Responsibilities	**Collaborations**
• Opens the choosing template window • Saves template choice	• LatexEditorView

Class Name: MainWindow	
Responsibilities	**Collaborations**
• Opens the main window • Creates Latex commands	• LatexEditorView

Class Name: LatexEditorView	
Responsibilities	**Collaborations**

• Stores document, auto save version strategy attributes • Gives access to it's attributes to other Classes	• ChooseTemplate • OpeningWindow • MainWIndow • LatexEditorController • VersionsManager • Document

QUALITY ASSESSMENT

This project is almost fully functional to the extent of the user stories implemented. However there are multiple problems and inefficiencies in the code. Firstly the **MVC** package architecture is not being respected. There are multiple methods with **misplaced responsibilities**. There are **long methods** and **duplicate code**. Lastly there is extensive **responsibility delegation** and **message chains**

1. The LatexEditorView class has too many responsibilities. Its name seems to imply that its responsibilities are related to the Graphic User Interface but it also has methods used to load and save documents.
   VersionsManager is supposed to manage all the data that have to do with version tracking. However it contains all of the code that is associated with the Command classes. It effectively functions as a middle man for several other classes which delegate their responsibilities to VersionsManager.

2. All of the Command classes delegate their responsibilities to VersionsManager which leaves them with too few responsibilities and very similar code.
   Seeing how in the MVC package architecture the controller classes are responsisble for implementing the communication between the view and model packages, I believe that the LatexEditorController class has too few responsibilities. It simply puts some Command Objects to a HashMap and then uses them in the enact method.

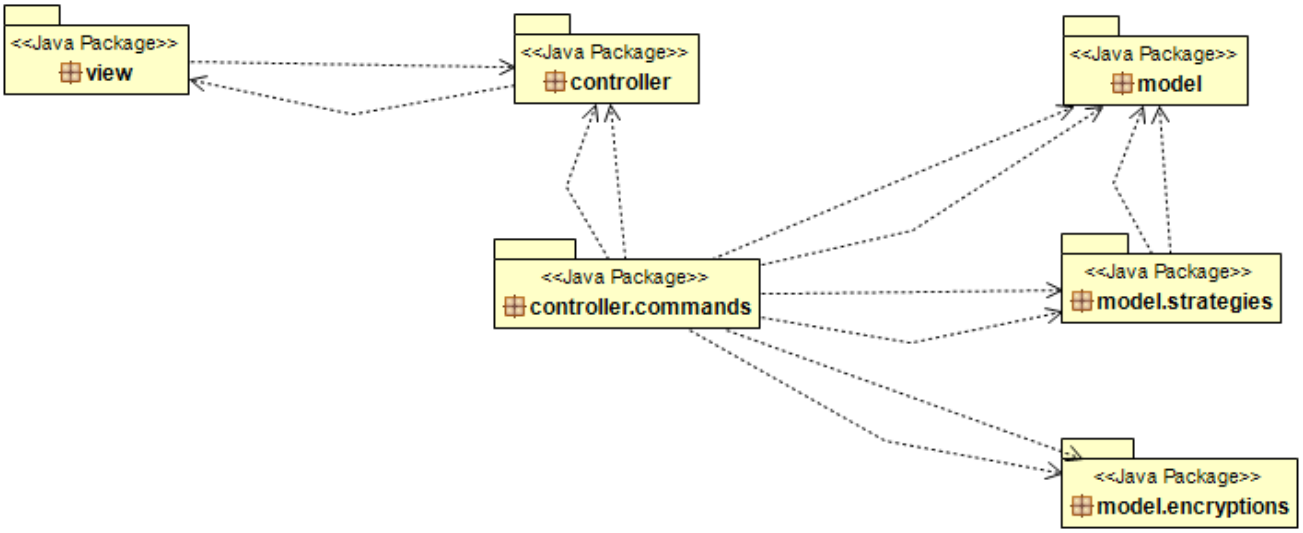
3. Most of the Command classes have similar code and in the case of AddLatexCommand and EditCommand the code is identical. In the VersionsManager class the methods enableStrategy and changeStrategy are almost the same.

4. In the VersionsManager class the enableStrategy and changeStrategy are almost similar. In the LatexEditorController class, in the constructor method all the Command objects are put into the HashMap manually, even though the code responsible for this, is essentially the same.
   In the DocumentManager class, in the method getContents the string of contents for each

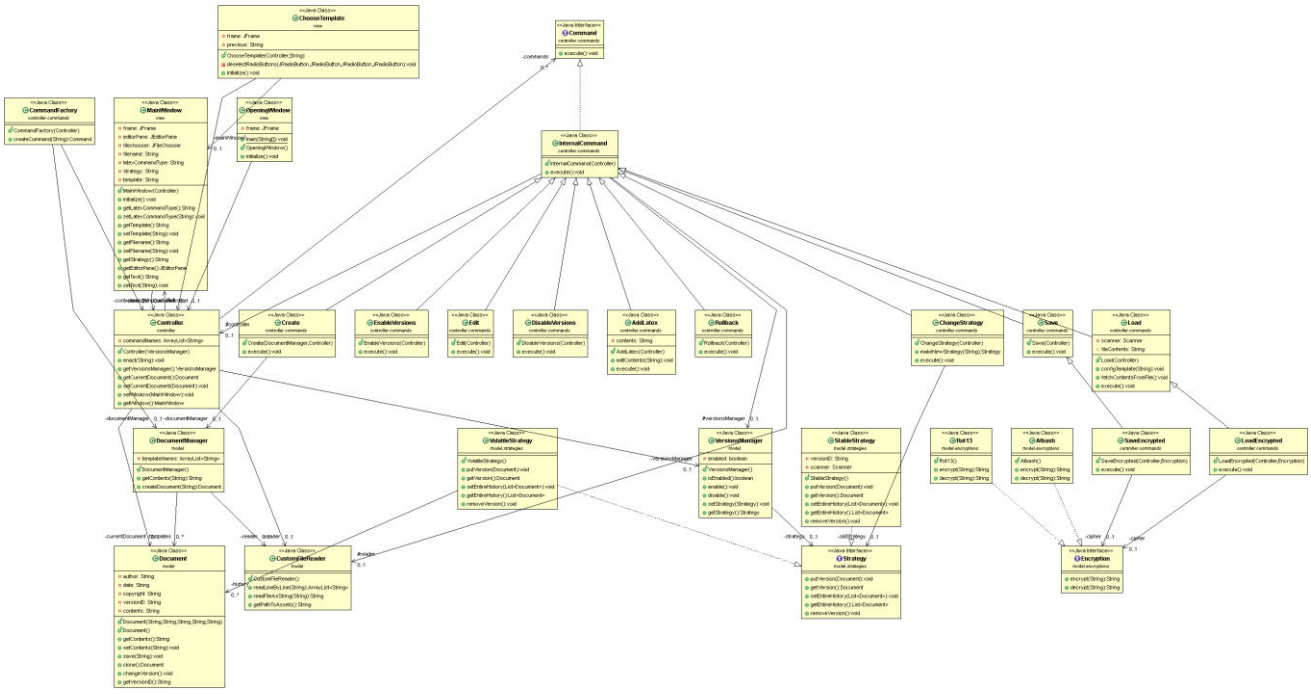
template is hard coded manually and returned after a conditional of multiple if statements. This causes the getContents method to become really long with duplicate code. Also in the DocumentManager class, in the constructor method each Document object is created and put into a HashMap manually even though the code is essentially the same. In the MainWindow class, in the editContents method the contents of the displayed document are edited using one of several possible strings, all of which are hard coded and get selected using a conditional of multiple if statements. Once again this increases the duplicate code as well as the length of the method.

RE-ENGINEERED DESIGN

- ARCHITECTURE



- DETAILED DESIGN



Images of class diagrams as well as package diagrams are provided alongside with this report.

- The first thing I changed was to implement the Substitute Algorithm refactoring to all the long methods. In most cases the huge hard coded strings were to blame for making the methods long and prone to duplication. So I created a new class the CustomFileReader class so that I can read the string from a file. I put all the strings in files and put the files to the Assets folder.

- To reduce the responsibilites of VersionsManager I used the Remove Middle Man refactoring. I moved all the code of VersionsManager to the Command classes that were delegating their responsibilites to VersionsManager. I also removed two dead code methods.

- After that I decided to work on LatexEditorView. Changing this class was tricky because the classes from the view package were intrinsically dependent on LatexEditorView. I used the Move Method as well as the Move Field refactorings and transfered all of the class's code to the LatexEditorController class.
  At this point I moved some of the responsibilites to the corresponding Command classes, namely saveToFile to SaveCommand, loadFromFile to LoadCommand and saveContents to EditCommand and AddLatexCommand, just as the phase two hint report suggested. This was by far the trickiest refactoring and left the LatexEditroController class with way too many responsibilities, most of which were getters and setters for several fields used in the classes of the view package.

- I moved some of the fields of LatexController class (formerly fields of LatexEditorView) to the MainWindow class. This allowed me to relieve LatexEditorController from some of the getters and setter and move them to the MainWindow as well.
- I removed the unused class VersionsStrategyFactory and I changed the code of the ChangeVersionsStrategyCommand by simplifying the conditionals. The new implementation takes advantage of the fact that there exist only two possible version tracking strategies. Were I to implement more strategies it would make sense to reintroduce the VersionsStrategyFactory class and use it to create new strategies with the String strategy field of MainWindow.
- I removed the duplicate code in the Command classes with the Extract Super Class refactoring. To do this I created the InternalCommand class which has all the fields needed by the majority of Command classes as protected fields while the constructor instantiated them.
- I implemented the extension user stories with respect to the project's architecture by creating the SaveEncrypted and LoadEncrypted classes which extend the SaveCommand and LoadCommand classes respectively. I also added the encryptions sub-package in the model package which contains the Encryption interface and the Rot13 and Atbash implementing classes. I used code taken from the internet to implement the two encryptions.
- Lastly I removed the LatexEditorView class, I changed the names of all the classes to be more comprehensive, wrote junit tests to test the Command classes and did some aesthetic touch ups to the display of the application.

- IMPLEMENTATION

Class Name: Controller	
Responsibilities • This class holds several fields related to version tracking. • It holds a reference of the main window and gives the	**Collaborations** • Document • VersionsManager • MainWindow

Command classes access to it.	

Class Name: MainWindow	
Responsibilities	**Collaborations**
• Implements the main Graphic User Interface of the application. • It holds several fields related to Command classes.	• Controller

Class Name: InternalCommand	
Responsibilities	**Collaborations**
• Instantiates the objects needed by the Command classes. • All the Command classes extend this class. • Implements the Command interface.	• Controller • VersionsManager • CustomFileReader

Class Name: AddLatexCommand	
Responsibilities	**Collaborations**
• Adds the contents of a Latex	• Controller

command to the display text. • Extends the InternalCommand class.	• VersionsManager

Class Name: EditCommand	
Responsibilities	**Collaborations**
• Saves a version of the Displayed document. • Extends the InternalCommand class.	• Controller • VersionsManager

Class Name: ChangeStrategy	
Responsibilities	**Collaborations**
• Changes the version tracking strategy by creating a new Strategy object. • Extends the InternalCommand class.	• Controller • VersionsManager

Class Name: EnableVerisons	
Responsibilities	**Collaborations**
• Enables version tracking mechanism.	• Controller • VersionsManager

• Extends the InternalCommand class.	

Class Name: Load	
Responsibilities	**Collaborations**
• Loads the contents of a saved latex file (.tex). • Displays the loaded file's contents on the Graphic User Interface. • Extends the InternalCommand class.	• Controller • VersionsManager • CustomFileReader • Document

Class Name: Save	
Responsibilities	**Collaborations**
• Creates a latex file (.tex) and saves the displayed Document's contents in it. • Extends the InternalCommand class.	• Controller

Class Name: Rollback	
Responsibilities	**Collaborations**

• Returns the displayed document in its previous version. • Removes the version that has been rolled back from. • Extends the InternalCommand class.	• Controller • VersionsManager

Class Name:	
Responsibilities	**Collaborations**
• Adds the contents of a Latex command to the display text. • Extends the InternalCommand class.	• Controller • VersionsManager

Class Name: SaveEncrypted	
Responsibilities	**Collaborations**
• Encrypts the contents of the displayed document using an Encryption object. • Saves the encrypted Document as a latex file (.tex). • Extends the Save class.	• Controller • Encryption

Class Name: LoadEncrypted	
Responsibilities	**Collaborations**
• Loads an ecrypted latex file (.tex). • Decrypts its contents using an Encryption object. • Extends the Load class.	• Controller • Encryption • Document

Class Name: Atbash	
Responsibilities	**Collaborations**
• Encrypts and decrypts the contents of a file using the Atbash encryption. • Implements the Encryption interface.	• Encryption • SaveEncrypted • LoadEncrypted

Class Name: Rot13	
Responsibilities	**Collaborations**
• Encrypts and decrypts the contents of a file using the Rot13 encryption. • Implements the Encryption interface.	• Encryption • SaveEncrypted • LoadEncrypted

Class Name: CustomFileReader	
Responsibilities	**Collaborations**
• Reads from a file and returns the content as a String.	• VersionsManager • Controller • InternalCommand