SOFTWARE DEVELOPMENT II PROJECT 2021-2022

OVERALL REPORT

FINAL VERSION

Σωτήριος Παναγιώτου: 4456
Δημήτριος Γιαννακόπουλος: 4336

TABLE OF CONTENTS

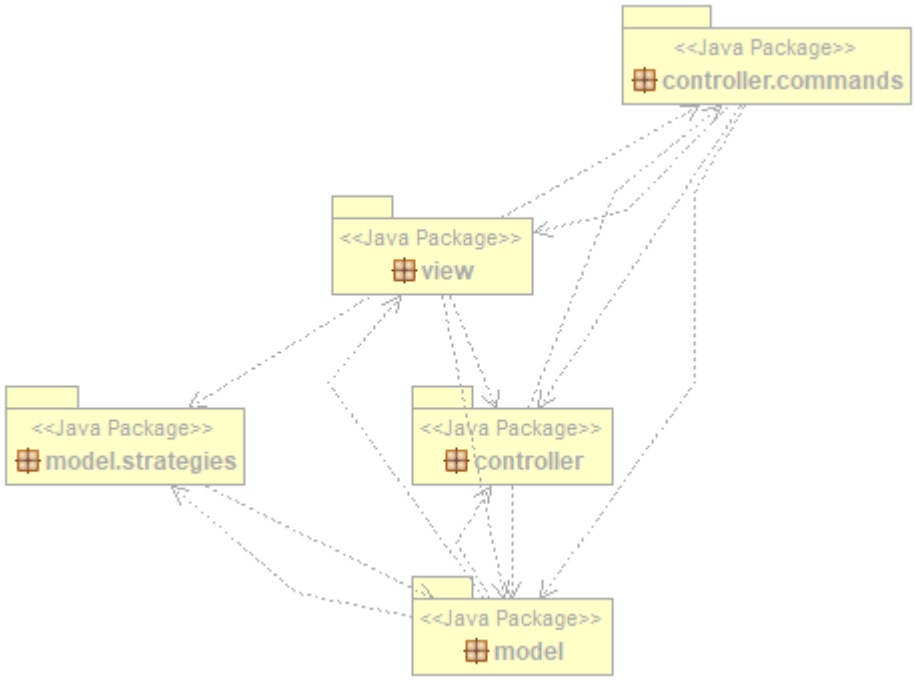
INTRODUCTION

To begin with, we studied the given project as well as the pdf files coming with it, we ran the project, understood the structure and confirmed that the user stories are working. We noticed some problems during the runtime (e.g. some functionalities were not working) and we wrote the test cases for every user story. In addition we fixed all the issues on the structure of the given project by using some refactoring techniques. Last but not least, we generated the UML class diagrams.

REFACTORED DESIGN

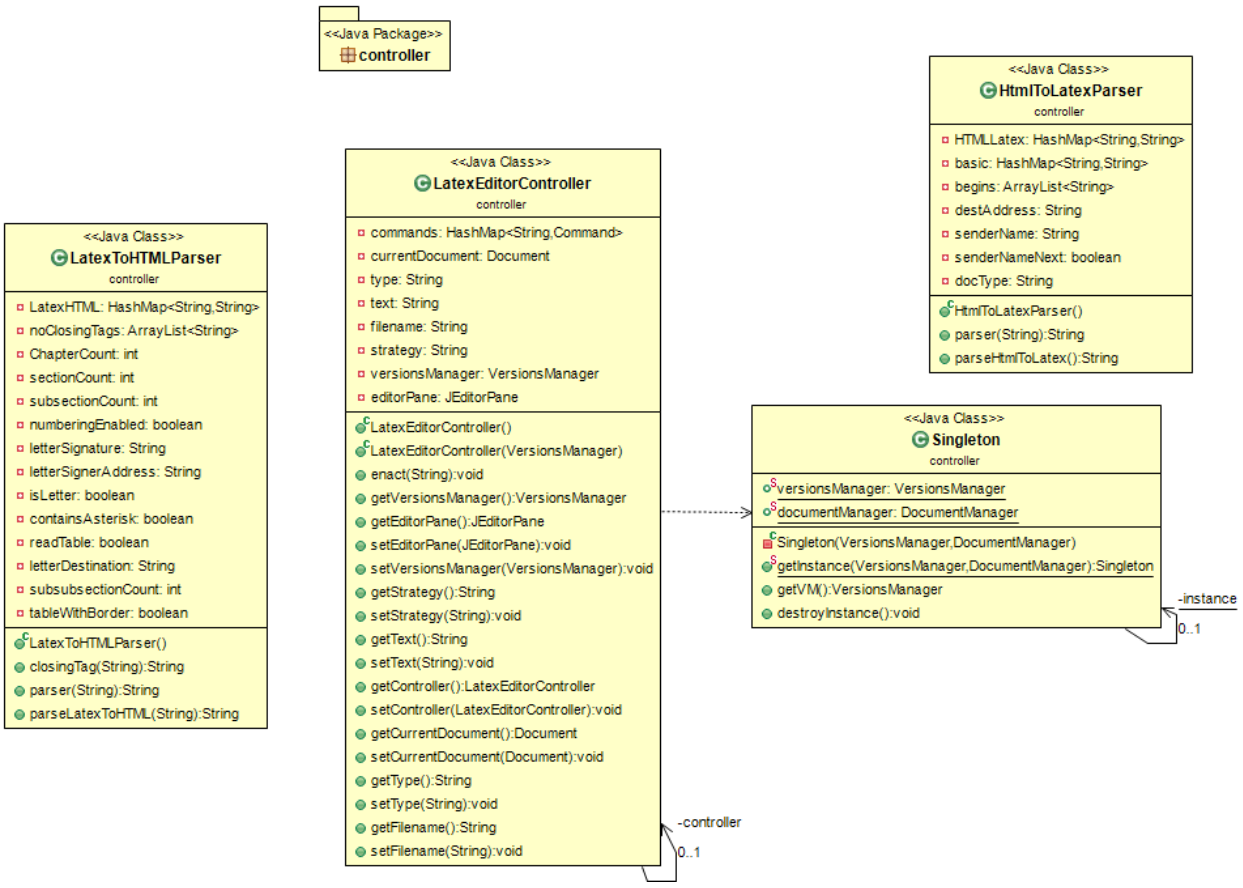
ARCHITECTURE

Package Diagram:

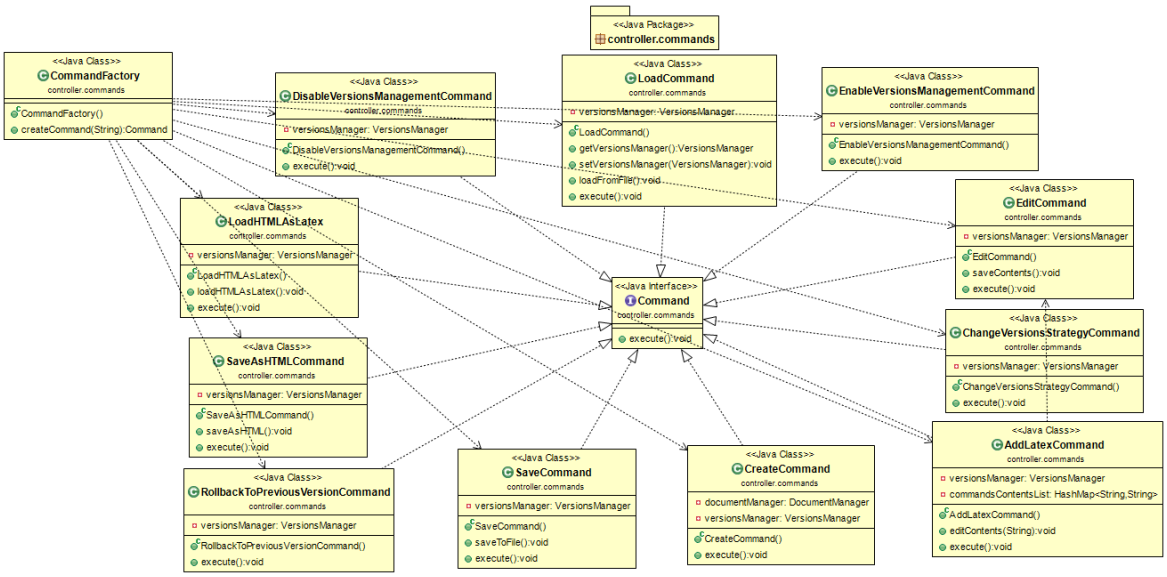


UML Class Diagrams

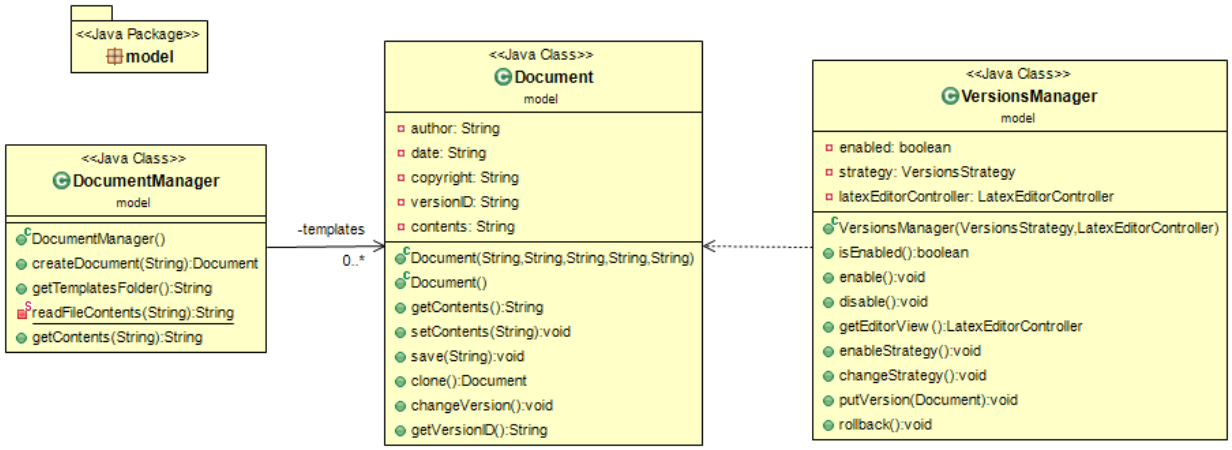
- **controller package**



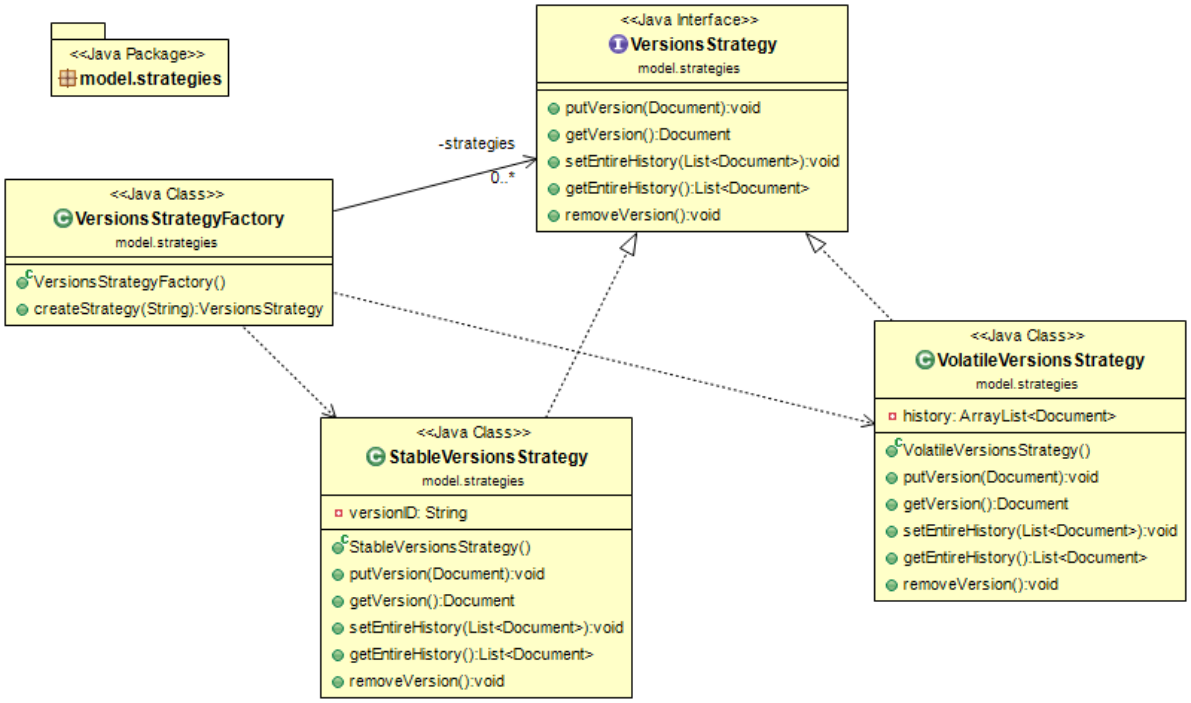
- **controller.commands package**



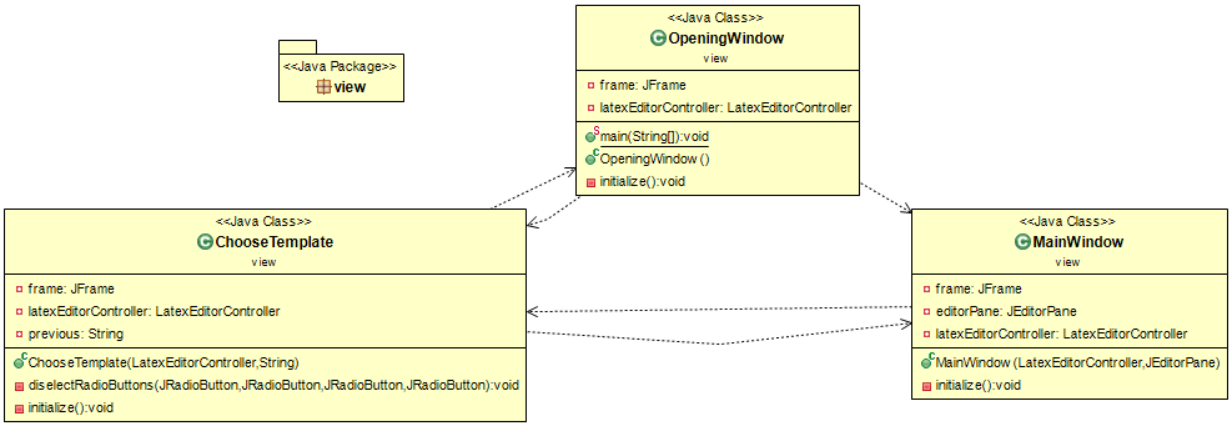
- **model package**



- **model.strategies package**



- **view package**



- We created a Singleton class to reduce the usage of the shared fields (VersionManager, DocumentManager) that were in every command class.

- For us to delete the duplicate code (e.g. multiple if conditions etc.), we put all the inputs in a HashMap and then with the help of the HashMap we populate the methods.

- Instead of including the templates of each latex file into the code, we save them into separate files and when we need them, we grab the template contents from each file.

- Each class should have a specific responsibility (e.g. a class named Save should ONLY do stuff related to saving). Methods that weren't related to the class' responsibilities are getting removed from that class and placed to the one that they should be in.

- We changed a few method calls which were leading to a chain of other calls and we made them direct from method to method.

- Instead of calling a method via a "third" (middle man) class, we make the call direct by moving the method to the correct class.

- We removed some methods that weren't being used at all.

- We removed classes that don't have a reason to exist.

IMPLEMENTATION

Below we display a brief description of the **main** responsibilities and collaborations of the most important classes.

Class Name: Document	
Responsibilities	**Collaborations**
▪ This class creates document objects and holds their properties.	▪ The document class is being managed by DocumentManager.

Class Name: DocumentManager	
Responsibilities	**Collaborations**
▪ This class creates a document and gets templates' contents from a file.	▪ Manages the Document class.

Class Name: VersionsManager	
Responsibilities	**Collaborations**
▪ This class is responsible for holding the document's editing history.	▪ Changes some of the document's object fields. ▪ It collaborates with all the strategies classes as well.

Class Name: LatexEditorController	
Responsibilities	**Collaborations**
▪ Initializes the commands for each user story.	▪ It uses the DocumentManager and VersionManager to initialize the Singleton instance. ▪ It uses the commandFactory to create a command for each user story.

Class Name: CommandFactory	
Responsibilities	**Collaborations**
▪ Creates the commands for each user story.	▪ Collaborates with all command classes. ▪ It gets called from LatexEditorController to create commands.

Class Name: Singleton	
Responsibilities	**Collaborations**
▪ Holds shared fields used by multiple other classes.	▪ Collaborates with all the command classes.

Class Name: CreateCommand	
Responsibilities	**Collaborations**
▪ Has the responsibility to create the document and to pass it to the main window.	▪ Collaborates with DocumentManager to create the document. ▪ Gets called from CommandFactory.

Class Name: EditCommand	
Responsibilities	**Collaborations**
▪ Saves each change that occurs in the document.	▪ Collaborates with Document class to make changes to the document. ▪ Collaborates with VersionsManager to check if the strategy is enabled and to process the strategy.

Class Name: LoadCommand	
Responsibilities	**Collaborations**
▪ To load and create a document from Latex file.	▪ Collaborates with Document to create the document.

Class Name: SaveCommand	
Responsibilities	**Collaborations**
▪ Saves the document in a file.	▪ Collaborates with Document.

Class Name: SaveAsHTMLCommand	
Responsibilities	**Collaborations**
▪ It is responsible to convert the Latex file to Html and then save it in a file.	▪ Collaborates with LatexToHtmlParser to parse the Latex file into Html.

Class Name: LoadHtmlAsLatex	
Responsibilities	**Collaborations**
▪ It is responsible to convert the Html file to Latex and then display it on the main window for editing.	▪ Collaborates with HtmlToLatexParser to parse the Html file to Latex.

Class Name: HtmltoLatexParser	
Responsibilities	**Collaborations**
▪ It is responsible to parse Html contents to Latex contents.	▪ Returns the result to its caller LoadHtmlAsLatex.

Class Name: LatexToHtmlParser	
Responsibilities	**Collaborations**
▪ It is responsible to parse Latex contents to Html contents.	▪ Returns the result to its caller SaveAsHtmlCommand.