LatexEditor

Release Report

ZOUGIANOS GIORGOS, 2442

TZIOLAS GIORGOS, 2591

TSAKNAKI ELEFTHERIA, 2559

VERSIONS HISTORY

Date	Version	Description	Author
18/03/2019	Release Plan	The first turnin concerning the Use Cases and the programming of the implementation of the project	**2442 Z.G.** **2591 T.G.** **2559 T.E.**
17/4/2019	1.0	First project implementation regarding the following Use Cases: US-1, US-2, US-3, US-6, US-8, US-9	**2442 Z.G.** **2591 T.G.** **2559 T.E.**

1   Introduction

This document provides information concerning the **<1st>** 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

Tests concerning the above User Stories e.g. US-1, US-2, US-3, US-6, US-8, US-9

2.1   Tests for User Story **<1>**

Test ID	*1*
User Story	*1*
Test Class	*DocumentRelatedTests*
Description	*Validate that DocumentManager#createDocument method creates the document properly with the selected template. If no template(document type) is given the document is created empty. -- Two documents are created; one of them is using the constructor of the document class with predefined arguments and the other is created by document manager. The expected result is that these two documents will be equal.*

2.2 Tests for User Story **<2>**

Test ID	2
User Story	2
Test Class	-
Description	*GUI related. It is not being tested automatically.*

2.3 Tests for User Story **<3>**

Test ID	3
User Story	3
Test Class	*CommandsTest*
Description	*Validate that certain text commands can be used in certain documents only. Validate also that the text command is being inserted in the text component properly and in the proper position (caret's index).*

2.4 Design Tests for User Story **<6>**

Test ID	6
User Story	6
Test Class	-
Description	*GUI related. It is not being tested automatically*

2.5 Tests for User Story **<8, 9>**

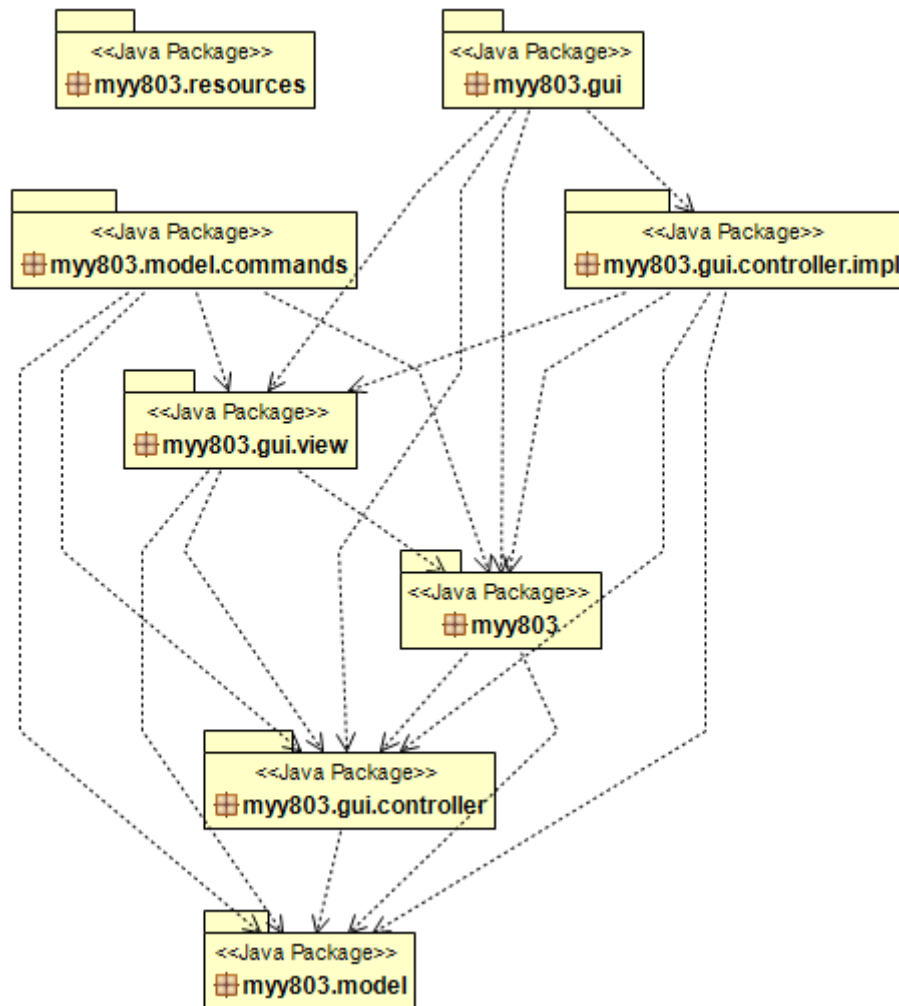| Test ID	8
User Story	8, 9
Test Class	*DocumentRelatedTests*
Description	*Validate that the DocumentManager can serialize and deserialize document objects; A new document is being created, DocumentManager saves it. Then DocumentManager loads a document with the same path and finally we expect that these documents are equal.*

3 Design

3.1 Architecture

UML package diagram

(https://raw.githubusercontent.com/gzougianos/software-engineering-class/master/uml_package_diagram.png)

<<Java Package>>
⊞ myy803.resources

<<Java Package>>
⊞ myy803.gui

<<Java Package>>
⊞ myy803.model.commands

<<Java Package>>
⊞ myy803.gui.controller.impl

<<Java Package>>
⊞ myy803.gui.view

<<Java Package>>
⊞ myy803

<<Java Package>>
⊞ myy803.gui.controller

<<Java Package>>
⊞ myy803.model

3.2 Design

UML class diagrams

(https://raw.githubusercontent.com/gzougianos/software-engineering-class/master/uml_class_diagram.png**)**

CRC cards for core classes:

Class Name: myy803.CommandFactory	
Responsibilities:	**Collaborations:**
▪ Factoring commands	▪ All implementations of CommandInterface ▪ TextCommand

Class Name: myy803.DocumentManager	
Responsibilities:	**Collaborations:**
▪ Hold all opened documents ▪ Create document ▪ Save document ▪ Load document	▪ Document ▪ Some classes java.io package

Class Name: myy803.RecentFileManager	
Responsibilities:	**Collaborations:**
▪ Hold all recently opened documents	▪ Some classes java.io package

Class Name: myy803.TextCommandManager	
Responsibilities:	**Collaborations:**
▪ Hold all predifined commands(TexCommand) that a user can add	▪ Resources (commands.xml) ▪ Some classes of org.w3c.dom package ▪ Some classes of javax.xml.parsers package

Class Name: myy803.gui.MainFrame	
Responsibilities:	**Collaborations:**
- Load the backend data - Initialize GUI	- DocumentManager - TextCommandManager - RecentFileManager - TabbedView - TabController - Web look and feel

Class Name: myy803.gui.controller.impl.AddDocumentControllerImpl	
Responsibilities:	**Collaborations:**
- Control an AddDocumentView - Load a document when the user asks it from the view - Create a document when the user wants it	- AddDocumentView - DocumentType - TabController - CommandFactory - DocumentManager - RecentFileManager

Class Name: myy803.gui.controller.impl.DocumentControllerImpl	
Responsibilities:	**Collaborations:**
- Control a DocumentView - Save, save as, load the document	- DocumentView - AddDocumentView - TabController - CommandFactory - TextCommand

Class Name: myy803.gui.controller.impl.TabControllerImpl

Responsibilities:	Collaborations:
▪ Control a TabbedView	▪ TabbedView ▪ AddDocumentController ▪ DocumentController ▪ DocumentManager

Class Name: myy803.gui.view.AddDocumentPanel

Responsibilities:	Collaborations:
▪ Implement AddDocumentView ▪ Allow the user to create a document based on a template ▪ Allow the user to load a document ▪ Allow the user to load a recently opened document	▪ AddDocumentController

Class Name: myy803.gui.view.DocumentPanel

Responsibilities:	Collaborations:
▪ Implement DocumentView ▪ Allow the user to edit the content of the document ▪ Allow the user to save, save as and load a document ▪ Allow the user to select a predefined TextCommand	▪ DocumentView ▪ DocumentController ▪ Document

Class Name: myy803.gui.view.TabbedPanel

Responsibilities:	Collaborations:
■ Implement TabbedView ■ Allow the user to edit multiple documents simultaneously in a tabbed environment ■ Close a selected tab	■ TabbedController ■ TabbedView ■ AddDocumentView

Class Name: myy803.model.Document

Responsibilities:	Collaborations:
■ Holds the last modified date, copyrights, version id, content, document type, path and if it is saved	■ DocumentType

Class Name: myy803.model.DocumentType

Responsibilities:	Collaborations:
■ ARTICLE ■ BOOK ■ EMPTY ■ LETTER ■ REPORT	■ Resources

Class Name: myy803.model.TextCommand

Responsibilities:	Collaborations:
■ Hold the name, content, description, allowed types and disallowed types of a predefined text command	■ DocumentType

Class Name: myy803.model.commands.AddTextCommand	
Responsibilities:	**Collaborations:**
▪ Implement command interface ▪ Insert the content of a text command to the TextComponent	▪ TextCommand ▪ JtextComponent

Class Name: myy803.model.commands.LoadCommand	
Responsibilities:	**Collaborations:**
▪ Load a document via a file chooser ▪ Notify the controller to update his view ▪ Implement CommandInterface	▪ TabController ▪ FileChooser ▪ Document ▪ DocumentManager

Class Name: myy803.model.commands.SaveAsCommand	
Responsibilities:	**Collaborations:**
▪ Implement CommandInterface ▪ Allow the user to save the document of the current DocumentView	▪ JfileChooser ▪ DocumentView ▪ Document ▪ DocumentManager

Class Name: myy803.model.commands.SaveCommand	
Responsibilities:	**Collaborations:**
▪ Implement CommandInterface ▪ Allow the user to save the document of the current DocumentView	▪ DocumentView ▪ Document ▪ DocumentManager