LatexEditor

Release Report

Power Rangers

Kaloudis Spyridwn 2447

Gkitsakis Dimos 2425

Mwlos Iwannis 2500

VERSIONS HISTORY

Date	Version	Description	Author
17/3/2019	<1.0>	First release of the Power Rangers LaTeX editor.	S. Kaloudis D. Gkitsakis I. Mwlos
22/5/2019	<2.0>	Second release of the Power Rangers LaTeX editor.	S. Kaloudis D. Gkitsakis I. Mwlos

Introduction

This document provides information concerning the **<2.0>** release of the project.

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.

1.1 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

2.1 Tests for User Story 1

Test ID	*createDocumentTest*
User Story	*[US-1]*
Test Class	*DocumentFactory*
Description	*We set up a Controller object and set the type to "article" then call the factoryMethod and we compare our document's contents with the article template contents*

2.2 Tests for User Story 2

Test ID	*editContentsViaGuiTest*
User Story	*[US-2]*
Test Class	*Gui, Controller*
Description	*We simulated our Gui by initializing a String as our textfield and we called the same methods that Gui class uses and we compared the new contents with the supposed textfield.*

2.3 Tests for User Story 3

Test ID	*editContentsViaCommandsViaGuiTest*
User Story	*[US-3]*
Test Class	*Gui, Controller*
Description	*We set up a Controller object and set the type to "article" then we call the factory method and we use the exact same methods that the Gui uses and then we compare the contents with the updated contents.*

2.4 Tests for User Story 4

Test ID	*04*
User Story	*[US-4]*
Test Class	*Controller,VersionsStrategyFactory*
Description	*We set up a Controller object, set the strategy type to "volatile" and call the enableVersions() method, then we set up a VersionsStrategyFactory object with the "volatile" constructor agrument and call the createStrategy() method. We then compare the controller's strategy class with the factory's strategy class.*

2.5 Tests for User Story 5

Test ID	*05*
User Story	*[US-5]*
Test Class	*Controller*
Description	*We set up a Controller object, set the strategy type to "volatile" and call the enableVersions() method and the changeVersions() method, then we set up a VersionsStrategyFactory object with the "stable" constructor agrument and call the createStrategy() method. We then compare the controller's strategy class with the factory's strategy class.*

2.6 Tests for User Story 6

Test ID	*06*
User Story	*[US-6]*
Test Class	*Controller*
Description	*We set up a Controller object, set the strategy type to "volatile" and call the enableVersions() method and we call the disableVersions() method. We compare the Controller's versionID with 0 since we turned off the automatic save so there are no versions to show.*

2.7 Tests for User Story 7

Test ID	*07*
User Story	*[US-7]*
Test Class	*Controller, Document*
Description	*We set up a new Document object and a Controller object, set the strategy type to "volatile" and call the enableVersions() method then set the contents of the Document to "test", add that as the newest version, then set the contents to "test 2", add that again, then do the same thing 2 more times ("test 3", "test 4"). Then we call the setRollback() method with 1 as the argument and we call the rollback() method. We compare the Controller's contents with the "test 2" text as this is the contents of the versionID=1 Document*

2.8 Tests for User Story 8

Test ID	*08*
User Story	*[US-8]*
Test Class	*Controller, Save*
Description	*We set up a Controller object and we use the setContents method to set the document's contents to "testing123". We then set up a Save object and we call the saveFile() method with the name "test", a path on our computer and the controller's method getContents(). We then read from the file created and we compare the file's contents with "testing123".*

2.9 Tests for User Story 9

Test ID	*09*
User Story	*[US-9]*
Test Class	*Controller, Load*
Description	*We set up a Controller object and we use the setContents method to set the document's contents to " ", the setFilePath() method with the path of the previous test's file and the setFileName() method similarly. We then set up a Load object and we call the loadFile() method . We compare contents (that were now loaded) with the actual contents of the file("testing123").*

3 Design

Class Name: Document	
Responsibilities: ▪ Store a Document's fields and information	**Collaborations:** ▪ DocumentFactory ▪ Controller

Class Name: DocumentFactory	
Responsibilities: ▪ Return a new deep copy of a Document based on a specific protype	**Collaborations:** ▪ Article ▪ Book ▪ Report ▪ Letter

Class Name: Article	
Responsibilities:	**Collaborations:**
▪ Set up the basic fields of an Article Document ▪ Return that Document to the DocumentFactory	▪ Document ▪ DocumentFactory

Class Name: Book	
Responsibilities:	**Collaborations:**
▪ Set up the basic fields of a Book Document ▪ Return that Document to the DocumentFactory	▪ Document ▪ DocumentFactory

Class Name: Report	
Responsibilities:	**Collaborations:**
▪ Set up the basic fields of a Report Document ▪ Return that Document to the DocumentFactory	▪ Document ▪ DocumentFactory

Class Name: Letter	
Responsibilities:	**Collaborations:**
▪ Set up the basic fields of a Letter Document ▪ Return that Document to the DocumentFactory	▪ Document ▪ DocumentFactory

Class Name: Empty	
Responsibilities:	**Collaborations:**
▪ Set up the basic fields of an Empty Document ▪ Return that Document to the DocumentFactory	▪ Document ▪ DocumentFactory

Class Name: Controller	
Responsibilities:	**Collaborations:**
▪ Keep the Document we’re working on ▪ Keep an object of the DocumentFactory class to create new Documents at any time ▪ Keep a HashMap with all available commands ▪ Be able to add new commands ▪ Issue execute on any command	▪ Document ▪ DocumentFactory ▪ Gui ▪ VersionsStrategy ▪ Save ▪ Load ▪ EnableVersionsManagement ▪ ChangeVersionsStrategy ▪ DisableVersionsManagement ▪ RollbackToPreviousVersion

Class Name: Command	
Responsibilities:	**Collaborations:**
▪ Let all commands inherit it’s basic execute( ) method	▪ SaveCommand ▪ LoadCommand ▪ EnableVersionsManagementCommand ▪ ChangeVersionsStrategyCommand ▪ DisableVersionsManagementCommand ▪ RollbackToPreviousVersionCommand

Class Name: SaveCommand	
Responsibilities:	**Collaborations:**
▪ Call the methods of the class Save	▪ Save

Class Name: Save	
Responsibilities:	**Collaborations:**
▪ Save a Document on the disk containing it’s contents ▪ Save a hidden file on the disk containing all other useful information (author, versionID, date, copyright)	▪ SaveCommand

Class Name: LoadCommand	
Responsibilities:	**Collaborations:**
▪ Keep a copy of the current controller ▪ Call the methods of the class Load	▪ Load ▪ Controller

Class Name: Load	
Responsibilities:	**Collaborations:**
▪ Keep a copy of the current controller ▪ Load a .tex Document’s contents from the disk ▪ Load all other info from the hidden .txt file	▪ LoadCommand ▪ Controller

Class Name: EnableVersionsManagementCommand	
Responsibilities:	**Collaborations:**
▪ Call the methods of the class EnableVersionsManagement ▪ Can return a set up VersionsStrategy object	▪ EnableVersionsManagement

Class Name: EnableVersionsManagement	
Responsibilities:	**Collaborations:**
▪ Calls the VersionsStrategyFactory’s createStrategy() and returns a set up VersionsStrategy object	▪ VersionsStrategyFactory

Class Name: ChangeVersionsStrategyCommand	
Responsibilities:	**Collaborations:**
▪ Call the methods of the class ChangeVersionsStrategy ▪ Can return a set up VersionsStrategy object	▪ ChangeVersionsStrategy

Class Name: ChangeVersionsStrategy	
Responsibilities:	**Collaborations:**
▪ Swaps the strategy from volatile to stable or the opposite ▪ If it changes from stable to volatile, then also deletes the tempSaves folder	▪ ChangeVersionsStrategyCommand

Class Name: DisableVersionsManagementCommand	
Responsibilities:	**Collaborations:**
▪ Call the methods of the class DisableVersionsManagement	▪ DisableVersionsManagement

Class Name: DisableVersionsManagement	
Responsibilities:	**Collaborations:**
▪ Disables the automatic save ▪ Deletes all tempSaves	▪ DisableVersionsManagement Command

Class Name: RollbackToPreviousVersionCommand	
Responsibilities:	**Collaborations:**
▪ Call the methods of the class RollbackToPreviousVersion	▪ RollbackToPreviousVersion

Class Name: RollbackToPreviousVersion	
Responsibilities:	**Collaborations:**
▪ Deletes the unwanted versions so that the last version in the arrayList is the version we want to roll back to	▪ RollbackToPreviousVersionCommand

Class Name: VersionsStrategy	
Responsibilities:	**Collaborations:**
▪ Inherits it’s methods to the StableVersionsStrategy and VolatileVersionsStrategy classes	▪ StableVersionsStrategy ▪ VolatileVersionsStrategy

Class Name: VolatileVersionsStrategy	
Responsibilities:	**Collaborations:**
▪ Add a copy of the current Document in an ArrayList ▪ Get the latest Document version ▪ Get the entire history of Documents (while the autosave was on) ▪ Set the entire history of Documents ▪ Remove the latest version or all versions	▪ VersionsStrategy

Class Name: StableVersionsStrategy	
Responsibilities:	**Collaborations:**
▪ Add a copy of the current Document in an ArrayList and create a corresponding .tex document in the tempSaves folder ▪ Get the latest Document version ▪ Get the entire history of Documents (while the autosave was on) ▪ Set the entire history of Documents ▪ Remove the latest version or all versions	▪ VersionsStrategy

Class Name: VersionsStrategyFactory	
Responsibilities:	**Collaborations:**
▪ Create a VersionsStrategy object based on the strategy type given as an argument ▪ Return the VersionsStrategy object	▪ VersionsStrategy

Class Name: Gui	
Responsibilities:	**Collaborations:**
▪ Create the Graphic User Interface ▪ Call the Controller's methods corresponding to the user's actions ▪ Constantly update the contents on the screen and the contents of the Document	▪ Controller