AdvancedText2SpeechEditor

Sprint Report

TEAM MEMBERS:

- ΣΤΕΡΓΙΟΥ ΒΑΣΙΛΕΙΟΣ ΑΜ:4300
- ΠΕΤΡΙΔΗΣ ΕΥΣΤΑΘΙΟΣ ΑΜ 4157

VERSIONS HISTORY

Date	Version	Description	Author
26 – 05 – 2021	1.0	This version of the project is final version of the of the application utilising the freeTTS library for the purpose of converting text to speech in order to help its user.	

. Introduction

This document provides information concerning the **<X>** sprint of the project.

1. Purpose

The purpose of this application is to assist people who suffer from vision problems, by converting a text

from a .docx or .xlsx file ,that is given as an input , into speech, so that they can hear its contents instead of using effort to read them.

2. Document Structure

The rest of this document is structured as follows. Section 2 describes out Scrum team and specifies the this Sprint's backlog. Section 3 specifies the main design concepts for this release of the project.

. Scrum team and Sprint Backlog

<For the user stories included in this release specify below corresponding tests using a typical tabular form.>

For the testing of .docx and .xlsx files, we use pre-defined files and and their respective paths in the code of the tests.

In the testing of Editing and Saving the contents of a Excel (.docx) file, the changes are successfully saved in the current file or in a new file, depending on the user's choice.

The saved contents are also written in a .txt file which is used to test if the contents were successfully written.

However, despite the fact that the changes in the .xlsx file are successfully done, without the system crashing, and the contents of the .txt file are the same as the .xlsx file, this test is marked as a fail in the case of Editing (User Story 2) and Saving (User Story 3) in a .xlsx file.

The problem above does not happen in the case of .docx files, in which all the tests are successful.

USE STORY TEST 1:

The purpose of the class testUC1 in testing package is to test the successful opening of the file that the user wants to open. This task is implemented with the use of method createTestFile(String filename), which writes the contents of the file that has been opened, in a .txt file and compares their contents.

USE STORY TEST 3:

The purpose of the class testUC3 in testing package is to test the successful saving of the file that the user wants to save. This task is implemented with the use of method createTestFile(String filename), which writes the contents of the file that the user saved, in a .txt file and compares their contents.

USE STORY TEST 4:

The purpose of the class testUC4 in testing package is to test the successful convertion of the contents, that the text area has, into speech. This task is implemented with the use of method createTestFile(String filename,String Text), which writes in a .txt file the contents that are converted into speech and compares them with the contents of the file that the user opened.

USE STORY TEST 5:

The purpose of the class testUC5 in testing package is to test the successful convertion of the contents of the text area that the user selected, into speech. This task is implemented with the use of method createTestFile(String filename,String Text), which writes in a .txt file the contents that the user selected to be converted into speech and compares them with the contents of the file that the user selected.

USE STORY TEST 6:

The purpose of the class testUC4 in testing package is to test the successful convertion of the contents, that the text area has, into speech, along with volume,pitch,rate. This testing is performed in the exact same way as testUC5.

USE STORY TEST 7:

The purpose of the class testUC7 in testing package is to test the successful recording of the actions that the user performed. This task is implemented with the use of method createTestFile(String filename,String Text), which writes in a .txt file the indicator 1 when the recording is enabled and checks the contents of the .txt file to be the value 1.

USE STORY TEST 8:

The purpose of the class testUC8 in testing package is to test the successful replaying of the text that the user last heard. This task is implemented with the use of method createTestFile(String filename,String Text), which writes in a .txt file the contents of the file that the user replayed and checks if the replayed contents are contained in the file that the user opened.

USE STORY TEST 9:

The purpose of the class testUC9 in testing package is to test the successful disable of the actions that the user performed. This task is implemented with the use of method createTestFile(String filename,String Text), which writes in a .txt file the indicator 1 when the recording is disabled and checks the contents of the .txt file to be the value 1.

1. Scrum team

Product Owner	Apostolos Zarras
Scrum Master	....
Development Team	1. Στεργίου Βασίλειος 2. Πετρίδης Ευστάθιος

2. Sprints

<List below the sprints that you performed and the user stories that have been realized in each Sprint>

Sprint No	Begin Date	End Date	Number of weeks	User stories

. Use Cases

<Specify the concrete Use Cases that describe the interaction of the user with the applications, as derived from the abstract user stories. Give a **UML Use Case diagram** and the **detailed use case descriptions**.>

1. <Use Case 1>

Use case ID	UC1
Actors	The user of the application.
Pre conditions	The application must be running before << Open File >> is pressed.
Main flow of events	1. The use case starts when the user presses the << Open File >> button 2. A window is opened to guide the user into choosing the desired file. 3. The user chooses the file. 4. The file is opened.
Alternative flow 1	If the user does not choose a .docx or a .xlsx file to open or does not choose a file at all, an error window appears and informs the user.
Alternative flow 2	.....
Post conditions	The contents of the file are presented in the text area, so that they can be converted into speech later on.

2. <Use Case 2>

Use case ID	UC2
Actors	The user of the application.
Pre conditions	The application must be running and the file that the user selected must be already open.
Main flow of events	1. The use case begins when the user clicks on the text area of the application. 2. The system allows the user to make changes in the contents of the text area. 3. The user makes the desired changes.
Alternative flow 1	…….
Alternative flow 2	…….
Post conditions	……

3. <Use Case 3>

Use case ID	UC3
Actors	The user of the application.
Pre conditions	1.The application must be running , 2.the file that the user selected must be already open. 3. The user must have choosen the desired encoding from the the << Available Encoding formats>> menu.
Main flow of events	1. The use case begins when the <<Save changes in new file>> or <<Save changes in current file>> button is pressed. 2. The user can save the changes of the file in the current file or create a new file to save them. 2.1 If <<Save changes in new file>> is pressed, a new file is created and the contents of the text area are saved into that file.

	2.2 If <<Save changes in current file>> is pressed, the contents of the text area are saved into the currently open file.
Alternative flow 1	If <<Save changes in new file>> is pressed and a file is not selected in order for the user to save the contents of the text area, an error window appears and informs the user properly.
Alternative flow 2	…….
Post conditions	The contents of the text area are saved in the desired file.

4. <Use Case 4>

Use case ID	UC4
Actors	The user of the application.
Pre conditions	1.The application must be running , 2.the file that the user selected must be already open. 3. The contents of the text area must not be empty
Main flow of events	1. The use case begins when the <<Play Contents>> button is pressed. 2. The system reads all the contents of the text area and converts them into speech.
Alternative flow 1	….
Alternative flow 2	…….
Post conditions	The user hears the contents of the whole file as they are transformed from text to speech by the system.

5. <Use Case 5>

Use case ID	UC5
Actors	The user of the application.
Pre conditions	1.The application must be running , 2.the file that the user selected must be already open. 3. The contents of the text area must not be empty 4. The user must have already selected a part of the contnets of the file using the mouse.
Main flow of events	1. The use case begins when the <<Play Contents>> button is pressed. 2. The system reads the selected contents of the text area and converts them into speech.
Alternative flow 1	….
Alternative flow 2	…….
Post conditions	The user hears the selected contents of the file as they are transformed from text to speech by the system.

6. <Use Case 6>

Use case ID	UC6
Actors	The user of the application.
Pre conditions	1.The application must be running , 2.the file that the user selected must be already open.
Main flow of events	1. The use case begins when either the <<Play Contents>> button or <<Replay Contents>> button is pressed. 2. The system reads the desired Volume , Pitch and Rate from the respective fields. 3. The desired contents are transformed or replayed using the values of the above three fields.
Alternative flow 1	….
Alternative flow 2	…….
Post conditions	The user hears the selected contents of the file as they are transformed from text to speech by the system with the desired setting on the volume, the pitch and the rate that they are transformed.

7. <Use Case 7>

Use case ID	UC7
Actors	The user of the application.
Pre conditions	1.The application must be running ,
Main flow of events	1. The use case begins when the <<Start Recording>> button is pressed. 2. The system records which button was presseda and the date the action happened, along side the hour.
Alternative flow 1	….
Alternative flow 2	…….
Post conditions	The recorded actions are presented in a text area.

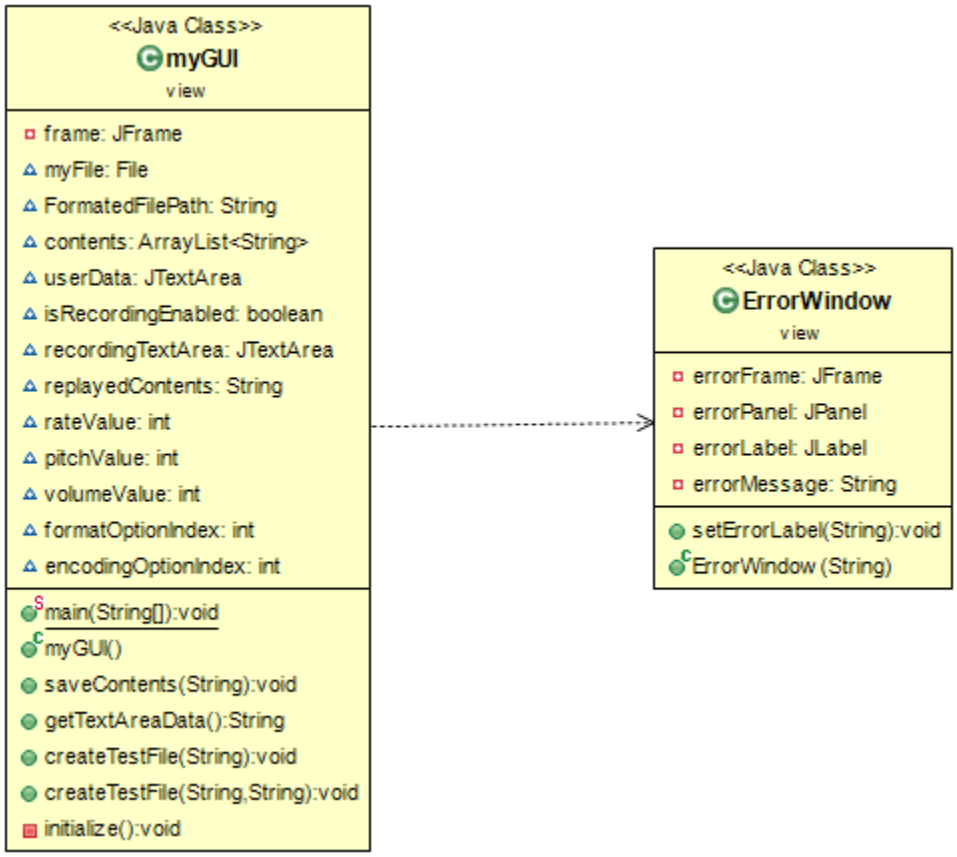
8. <Use Case 8>

Use case ID	UC8
Actors	The user of the application.
Pre conditions	1.The application must be running , 2. The contents of the text area or a part of them must have been transformed at least one time.
Main flow of events	1. The use case begins when the <<Replay Contents>> button is pressed. 2. The system converts from text to speech the last sequence of text that was executed before.
Alternative flow 1	….
Alternative flow 2	…….
Post conditions	The user hears again the text that is converted to speech by the ssytem.

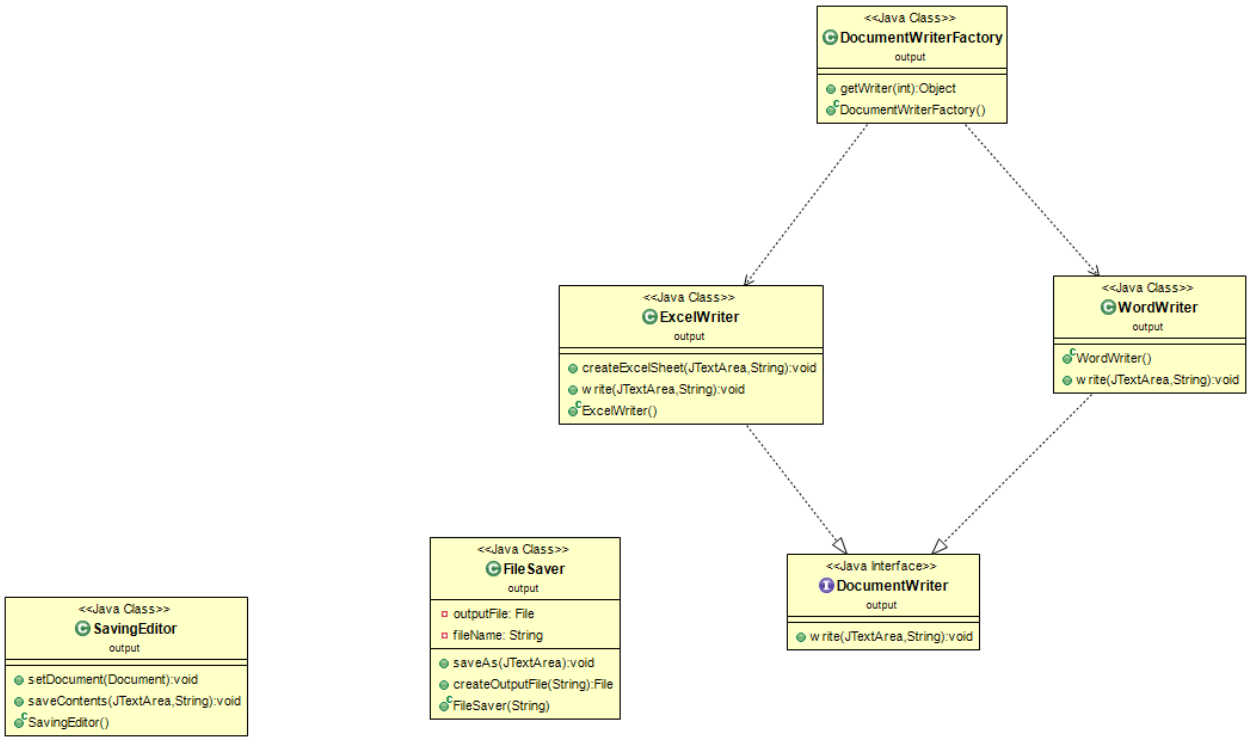
9. <Use Case 9>

Use case ID	UC9
Actors	The user of the application.
Pre conditions	1.The application must be running , 2. The <<Start Recording>> button must be pressed.
Main flow of events	1. The use case begins when the <<Start Recording>> button is pressed again. 2. The system stops the recording of the sequence of actions, so that the actions happening from this point on should not be added to the text area.
Alternative flow 1	….
Alternative flow 2	…….
Post conditions	…

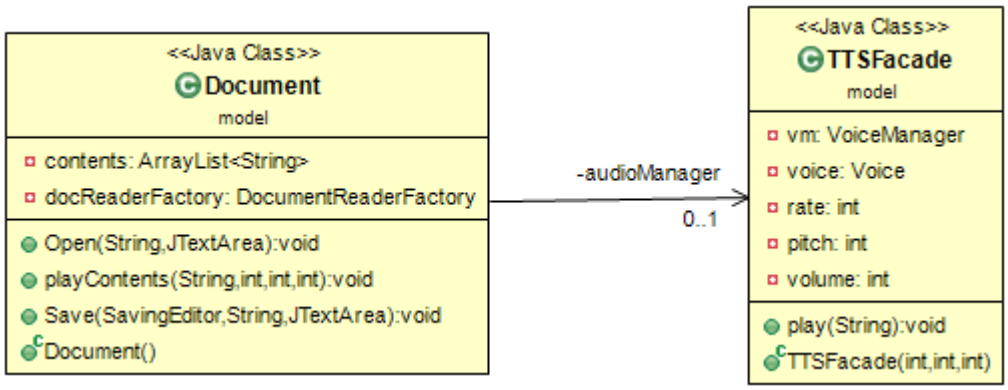
This is the UML Class Diagram of view package.



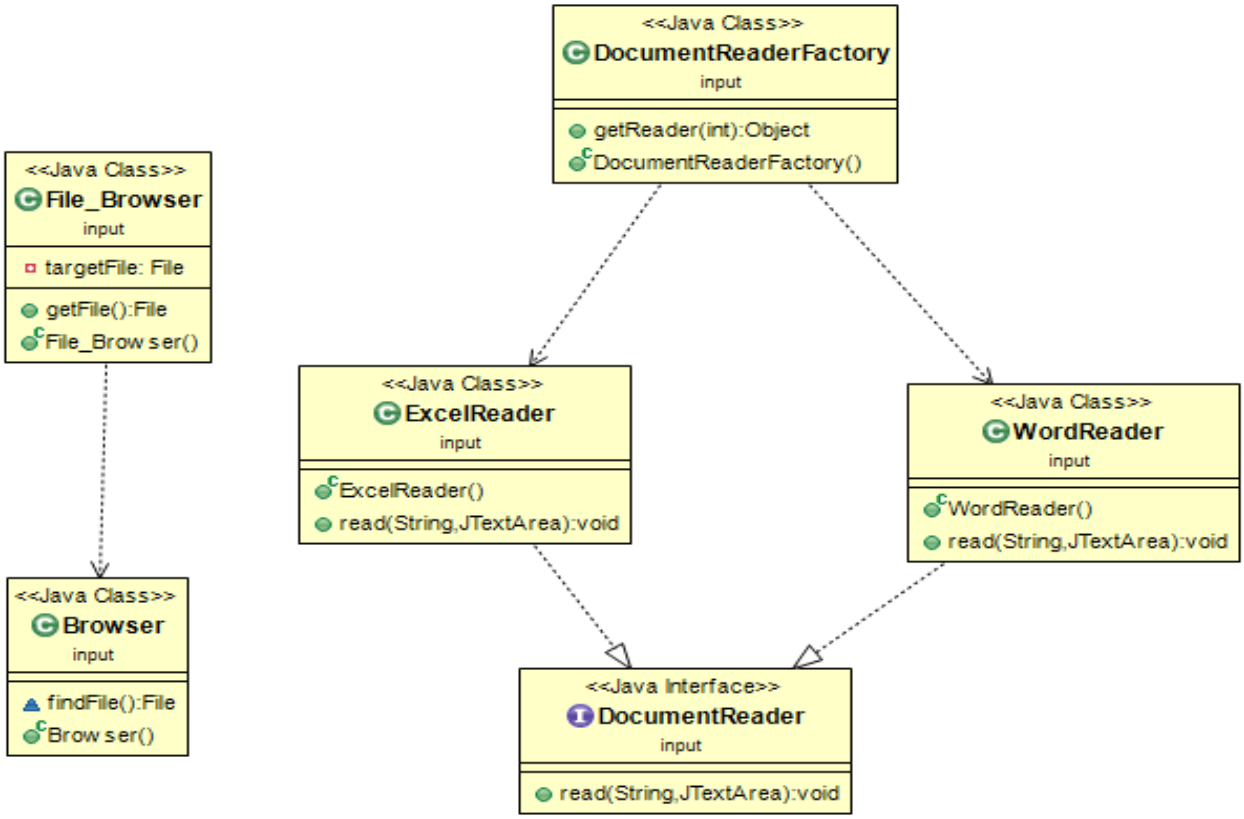
This UML Class Diagram represents the relastionships between the classes in output package.



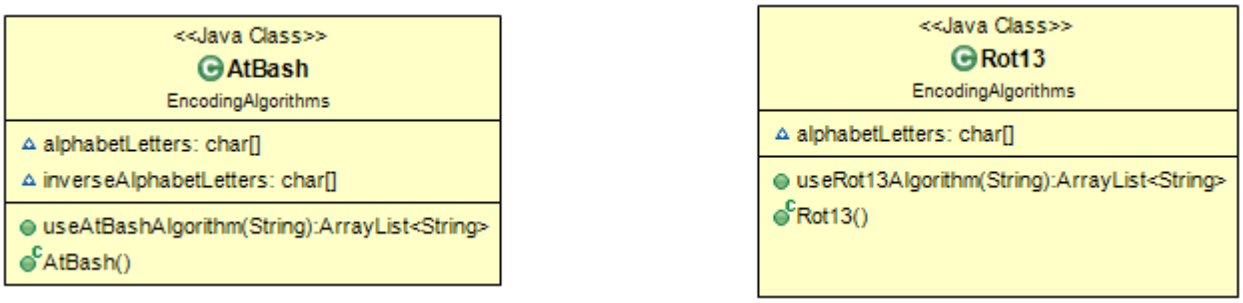
This UML Class Diagram represents the relastionships between the classes in model package.



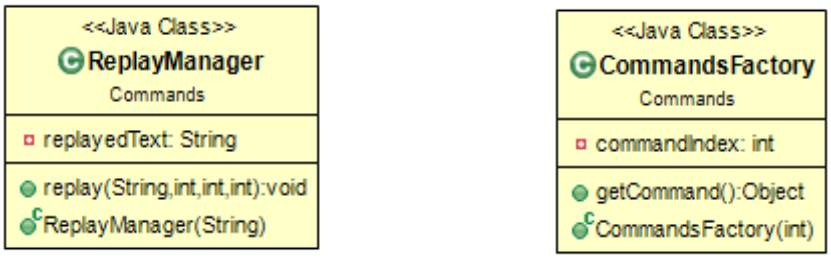
This UML Class Diagram represents the relastionships between the classes in input package.



This UML Class Diagram represents the relastionships between the classes in encodingAlgorithms package.



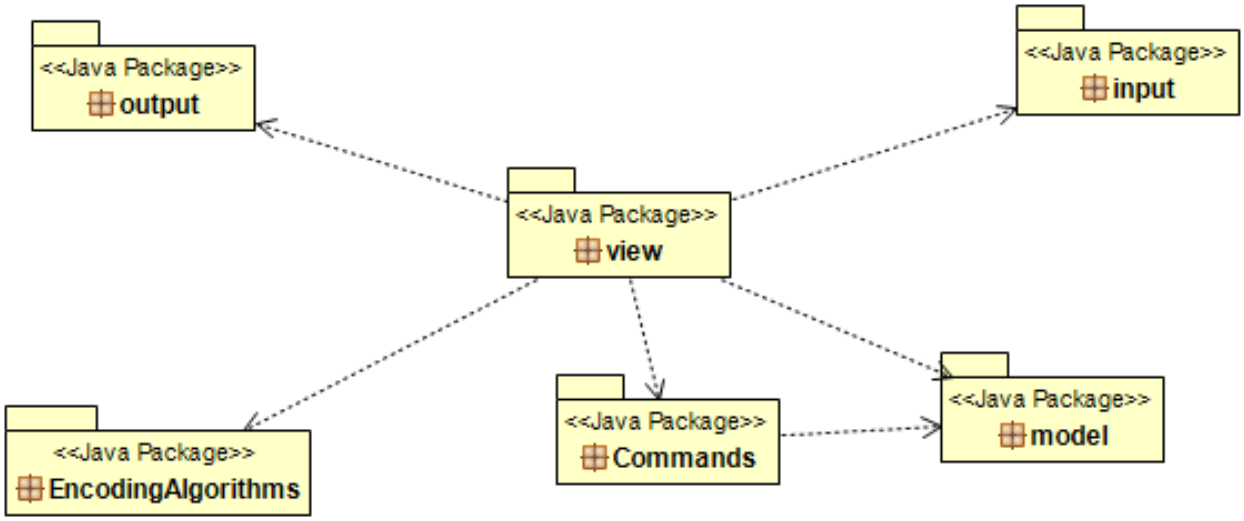
This UML Class Diagram represents the relastionships between the classes in Commands package.



. 4. Design

1. Architecture

<Specify the overall architecture for this release in terms of a **UML package diagram**.>



2. Design

<Specify the detailed design for this release in terms of **UML class diagrams**.>

<Document the classes that are included in this release in terms of CRC cards according to the template that is given below.>

Class Name: myGUI	
Responsibilities: ▪ repressents to the user the available interactions with the system ▪ Initializes the commponents of the GUI	**Collaborations:** ▪ it uses CommandsFactory() class in each case to execute the option that the user selects. ▪ It uses the Documment() class in order to use Save(), Open() and playContents() functions in the respective case

Class Name: ErrorWindow	
Responsibilities: ▪ Initializes the window that indicates the error that the user made. ▪ Displays a proper message to the user, depending on the error	**Collaborations:** ▪ It is used by myGUI() class whenever opening a file faills ▪ It is used by myGUI() class wheneversaving saving the contents of a text area in a new file also fails.

Interface Name: DocumentWriter	
Responsibilities: ▪ It initializes the write() method, so that it can be implemented by ExcelWriter() and WordWriter() classes	**Collaborations:** ▪ This interface is implemented by ExcelWriter() class, so that it can be used to save the contents of the text area in a .xlsx file. ▪ This interface is also implemented by WordWriter() class, so that it can be used to save the contents of the text area in a .docx file.

Class Name: ExcelWriter	
Responsibilities: ▪ As stated above, the class ExcelWriter() implements the write() method that is used to save the contents of the text area into a .xlsx file.	**Collaborations:** ▪ This class is returned from DocumentWriterFactory() class, which is called in Document() class. ▪ This class is used in Document() class and specifically in its Save() method, in case the user desires to save the contentsof the text area in an .xlsx file.

Class Name: WordWriter	
Responsibilities: ▪ As stated above, the class WordWriter() implements the write() method that is used to save the contents of the text area into a .docx file.	**Collaborations:** ▪ This class is returned from DocumentWriterFactory() class, which is called in Document() class. ▪ This class is used in Document() class and specifically in its Save() method, in case the user desires to save the contentsof the text area in an .docx file.

Class Name: DocumentWriterFactory	
Responsibilities: ▪ Depending on whether the user wants to save the contents of the text area in a .docx or .xlsx file, this class returns an WordWriter or ExcelWriter class respectively	**Collaborations:** ▪ this class is used by the Save() method in Document() class, depending on the format of the file (.docx , .xlsx) that the text of text area will be saved.

Class Name: Document	
Responsibilities: ▪ It provides Open(), Save() and playContents() methods in order to handle the actions that the user wants to perform	**Collaborations:** ▪ this class is used by the main GUI, which is the class myGUI () and depending on the action, one of its appropriate methods is called.

Class Name: TTSFacade	
Responsibilities: ▪ this class is used for handling the conversion of user text to speech, alongside with volume,pitch and rate that the user desires.	**Collaborations:** ▪ this class in used by the Document class and its play() method specifically, so that the desired contents can be converted to speech by the system.

Interface Name: DocumentReader	
Responsibilities: ▪ This interface initializes the read() method, so that it can be implemented() in WordReader() and ExcelReader() classes.	**Collaborations:** ▪ this interface is implemented by WordReader and ExcelReader() classes ,in order for them to be able to use the read() method for reading .docx and .xlsx files respectively.

Class Name: ExcelReader	
Responsibilities: ▪ As stated above, the class ExcelReader() implements the read() method that is used to read the contents of the a .xlsx file and place them into the text area of the GUI.	**Collaborations:** ▪ This class is returned from DocumentReaderFactory() class, which is called in Document() class. ▪ This class is used in Document() class and specifically in its Open() method, in case the user desires to open an .xlsx file.

Class Name: WordReader	
Responsibilities: ▪ As stated above, the class WordReader() implements the read() method that is used to read the contents of the a .docx file and place them into the text area of the GUI.	**Collaborations:** ▪ This class is returned from DocumentReaderFactory() class, which is called in Document() class. ▪ This class is used in Document() class and specifically in its Open() method, in case the user desires to open an .docx file.

Class Name: DocumentReaderFactory	
Responsibilities: ▪ Depending on whether the user wants to open a .docx or .xlsx file, this class returns an WordReader or ExcelReader class respectively.	**Collaborations:** ▪ this class is used by the Open() method in Document() class, depending on the format of the file (.docx , .xlsx) that the user desire to open.

Class Name: Browser	
Responsibilities: ▪ This class implements the findFile() method that locates the file that the user desires to open.	**Collaborations:** ▪ The file that the findFile () method locates is returned to the class File_Browser(), where a Browser item is created, in order for that to be used by the findFile() method as well.

Class Name: File_Browser	
Responsibilities: ▪ This class utilises an item of Browser() class and specifically its findFile() method to locate the file that the user desires to open().	**Collaborations:** ▪ This class is returned by the getCommand() method of class CommandsFactory() to the main GUI. ▪ The method getFile() of the File_Broswer() class is used by the main GUI to get the value of its private field targetFile.

Class Name: File_Browser	
Responsibilities: ▪ This class utilises an item of Browser() class and specifically its findFile() method to locate the file that the user desires to open().	**Collaborations:** ▪ This class is returned by the getCommand() method of class CommandsFactory() to the main GUI. ▪ The method getFile() of the File_Broswer() class is used by the main GUI to get the value of its private field targetFile.

Class Name: AtBash	
Responsibilities: ▪ This class is responsible for the AtBash encoding of the contents in the GUI text area. ▪ Its useAtBashAlgorithm() method implements this encoding. ▪ The contents of the text area are encoded and then the changes are saved in anew file.	**Collaborations:** ▪ This class is used in the main GUI in case the << Save changes in new file>> button is pressed and the user chooses to use AtBash encoding form the available encoding options.

Class Name: Rot13	
Responsibilities: ▪ This class is responsible for the Rot13 encoding of the contents in the GUI text area. ▪ Its useRot13Algorithm() method implements this encoding. ▪ The contents of the text area are encoded and then the changes are saved in anew file.	**Collaborations:** ▪ This class is used in the main GUI in case the << Save changes in new file>> button is pressed and the user chooses to use Rot13 encoding form the available encoding options.

Class Name: CommandsFactory	
Responsibilities: ▪ This class is used to determine the command that the user wants to perform. ▪ This task is performed by the getCommand() functions.	**Collaborations:** ▪ The class that performs the command of the user is returned by the getCommand() method to the main GUI ▪ The returned class then utilises the appropriate function for the execution of the task.

Class Name: ReplayManager	
Responsibilities: ▪ This class is responsible for replaying the contents that were last converted to speech by the application. ▪ The task described above is performed by the replay() method.	**Collaborations:** ▪ The main GUI creates an object of type ReplayManager and uses its replay() method, whenever the user decides to replay the contents.