AdvancedText2SpeechEditor

Sprint Report

Thundercats

Vlaxopoulos Lampros 2948

Krommydas Georgios 3260

Tsiami Panoraia 3350

IOANNINA, 2021

VERSIONS HISTORY

Date	Version	Description	Author
29/05/2021	2.0	This is the final version of the report for the AdvancedTextToSpeech software application	L. Vlaxopoulos G. Krommydas P. Tsiami

1 Introduction

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

1.1 Purpose

FreeTTS is an open source speech synthesis system which allows texts to be transformed into sound. It is highly used in audio generation applications, which are provided with the input of documents and generate the sound. The purpose of this project is to use this library, in order to convert various documents into speech. The user can load or create a new document, which can be edited. In addition, either the whole document can be transformed into speech or some selected lines of the document. Another feature of the application is to encode and decode the document, while it can be also transformed into speech. It can also record a session of the user's work, in order to replay it after finishing is work. The record can be activated and de-activated.

1.2 Document Structure

The rest of this document is structured as follows. Section 2 describes out Scrum team and specifies this Sprint's backlog with the unit tests for the use cases. Section 3 specifies the main design concepts for this release of the project. Section 4 specifies the architecture of the system with its functional and non-functional requirements of this project.

2 Scrum team and Sprint Backlog

On this section are the necessary tests for the use cases, in order to achieve the excepted results from the user stories. The tests were described in general. In TestsJUnit package there are analytical tests for every case.

Test's User Story	*[US1] – UC1*
Test File Name	*TestOpen*
Test Description	*In this test, we test the implementation of US1. By opening a certain file from the folder with its path file as a Document object. Then we open it with the open() method. The next step is to create an array list and add some content. If the content of the array list is the same with the opened document then the results are correct and the expected.*

Test’s User Story	*[US1] – UC2*
Test File Name	*TestEncode*
Test Description	*In this test, we test the implementation of US1. By opening a certain file from the folder with its path file and an encoding type (Rot13, Atbash) as a Document object. We open it with the method open(doc.getFileType(),doc.getPath(),doc.getTyoecode()) and check if the content is encoded with a certain encoded type. If its true, then this functionality works and documents can be encoded*

Test’s User Story	*[US1] – UC3*
Test File Name	*TestDecode*
Test Description	*In this test, we test the implementation of US1. By opening a certain encoded file from the folder with its path file and an encoding type (Rot13, Atbash) as a Document object. We open it with the method open(doc.getFileType(),doc.getPath(),doc.getTyoecode()) and check if the encoded content have been decoded. If the decoded results are correct and the expected then this functionality works and encoded documents can be decoded successfully.*

Test’s User Story	*[US2] – UC4*
Test File Name	*TestEdit*
Test Description	*In this test, we test the implementation of US2. As a certain document has opened as Document object, we add new content into a string and add it to the document. If the string has been added successfully into the document, then when we re-open the document the line should be appear inside the document*

Test's User Story	*[US3] – UC5*
Test File Name	*TestSave*
Test Description	*In this test, we test the implementation of US3. We create an empty document as a Document object and add some lines into with an array list. Then we save the file with the content into the folder with its path file. If the save was successful, then we check if the content is what was placed in the file at first*

Test's User Story	*[US4] – UC6*
Test File Name	*TestPlayAllContents*
Test Description	*In this test, we test the implementation of US4. We open a certain document from the folder with its path file. Then we call the FakeTTSFacade object so we can play all the contents of the file through the audio manager. If the opening and the speech convert was successful, then the method's functionality is correct. If a document is empty, then we add some content and repeat the same process.*

Test's User Story	*[US5] – UC7*
Test File Name	*TestPlayLine*
Test Description	*In this test, we test the implementation of US5. We open a certain document from the folder with its path file. Then we call the FakeTTSFacade object so we can play a certain part of the content. On this occasion we check for the first line. If the opening and the speech convert was successful, then the method's functionality is correct.*

Test's User Story	*[US6] – UC8*
Test File Name	*TestSoundSettings*
Test Description	*In this test, we test the implementation of US6. First we create an empty document and an array list that keeps the three parameters of audio(pitch, rate, volume). We add them certain values. Then we call the FakeTTSFacade object and set the audio manager for the document. The next step is to set the parameters the same as those into the array list. The last thing is to check if the parameters have been adjusted to those above. If so, then this functionality is successfully working.*

Test's User Story	*[US8] – UC10*
Test File Name	*TestReplay*
Test Description	*In this test we check the implementation of the US7,8,9. First we open a document from the folder as a Document object. Then we create an ActionListener array list to check later the replay session. We open as an OpenDocument object the previous Document object and we call the FakeTTSFacade object, thus using the setAudioManager(doc) method to the Document object and then convert it to speech with the DocumentToSpeech object. We add the previous process on the array list . We create another array list and add some content inside it. Then we edit the previous document that we opened and add to the ActionListener array list. The last thing to do is to use the replay method on the list and check if the actions that we used are the actual ones from the application.*

2.1 Scrum team

Product Owner	Apostolos Zarras
Scrum Master	....
Development Team	Labros Vlaxopoulos, Georgios Krommydas, Panoraia Tsiami

2.2 Sprints

Sprint No	Begin Date	End Date	Number of weeks	User stories
2.0	12/03/2021	20/03/2021	1	All user stories. Beginning of project.
2.0	20/03/2021	27/03/2021	1	All user stories. End of Use Cases
2.0	27/03/2021	11/04/2021	2	All user stories. Implementation of use cases. Start of uml diagrams
2.0	11/04/2021	13/04/2021	0	All user stories. Implementation of open, save and textToSpeech functionalities
2.0	13/04/2021	14/04/2021	0	All user stories. Checking open functionality with gui and back-end interaction
2.0	14/04/2021	15/04/2021	0	All user stories. Checking tts functionality with gui and back-end interaction
2.0	15/04/2021	20/04/2021	0	All user stories. Change of gui implementation. End of use cases and use case diagrams
2.0	20/04/2021	25/04/2021	0	[US6]. Implementation of tune parameters and interaction of gui with back-end
2.0	25/04/2021	27/04/2021	0	All user stories. Fixing functionalities of tune parameters. Checking open, save, tts play all contents, tts part of contents, sound settings functionalities with gui and back-end interaction
2.0	27/04/2021	24/05/2021	4	All user stories. Final details on project. Implementation of all Junit tests
2.0	24/05/2021	28/05/2021	0	All user stories. End of project. End of Junit tests. End of Sprint Report

2.3 Sprint Backlog

- [US1] As a user I want to open a file that is stored on disk and view its contents. The application should allow me to open different kinds of files. The application should support at least Microsoft Word (.docx) and Excel (.xlsx) documents. The application should also allow me to open files with encoded contents. The application should support different encodings, including Atbash and Rot-13. So that I can edit the contents of the file and transform them to audio.
- [US2] As a user I want to be able to edit the contents of the file. So that I can produce a new version of the file that I opened.
- [US3] As a user I want to save the contents of the file that I opened on disk. The application should allow me to specify the format of the file, the encoding (if any) and the filename. So that I can store a new version of the file that I opened.
- [US4] As a user I want to transform the contents of the file which I opened, to audio. So that I can listen what is in the file instead of having to read it.
- [US5] As a user I want to select a part of the contents of the file(e.g from line X to line Y) that I opened and transform them to audio. So that I can listen only a part of the contents of the file instead of all.
- [US6] As a user I want to tune the audio parameters, i.e., the volume, the speech rate and the pitch. So that I can customize the audio to my needs.
- [US7] As a user I want to activate a recording operation that keeps track of a sequence of text to audio transformation actions/commands. So that ia can re-execute them multiple times.
- [US8] As a user I want to replay the recorded sequence of actions. So that I can listen again the contents of the file I opened.
- [US9] As a user I want to de-activate the recording operation. So that I can clean up the sequence of actions that have been recorded.

3 Use Cases

On this section we realize the user stories of the project and describe the use cases. As well the uml use case diagram, on how the use cases interact with the user and with its other.

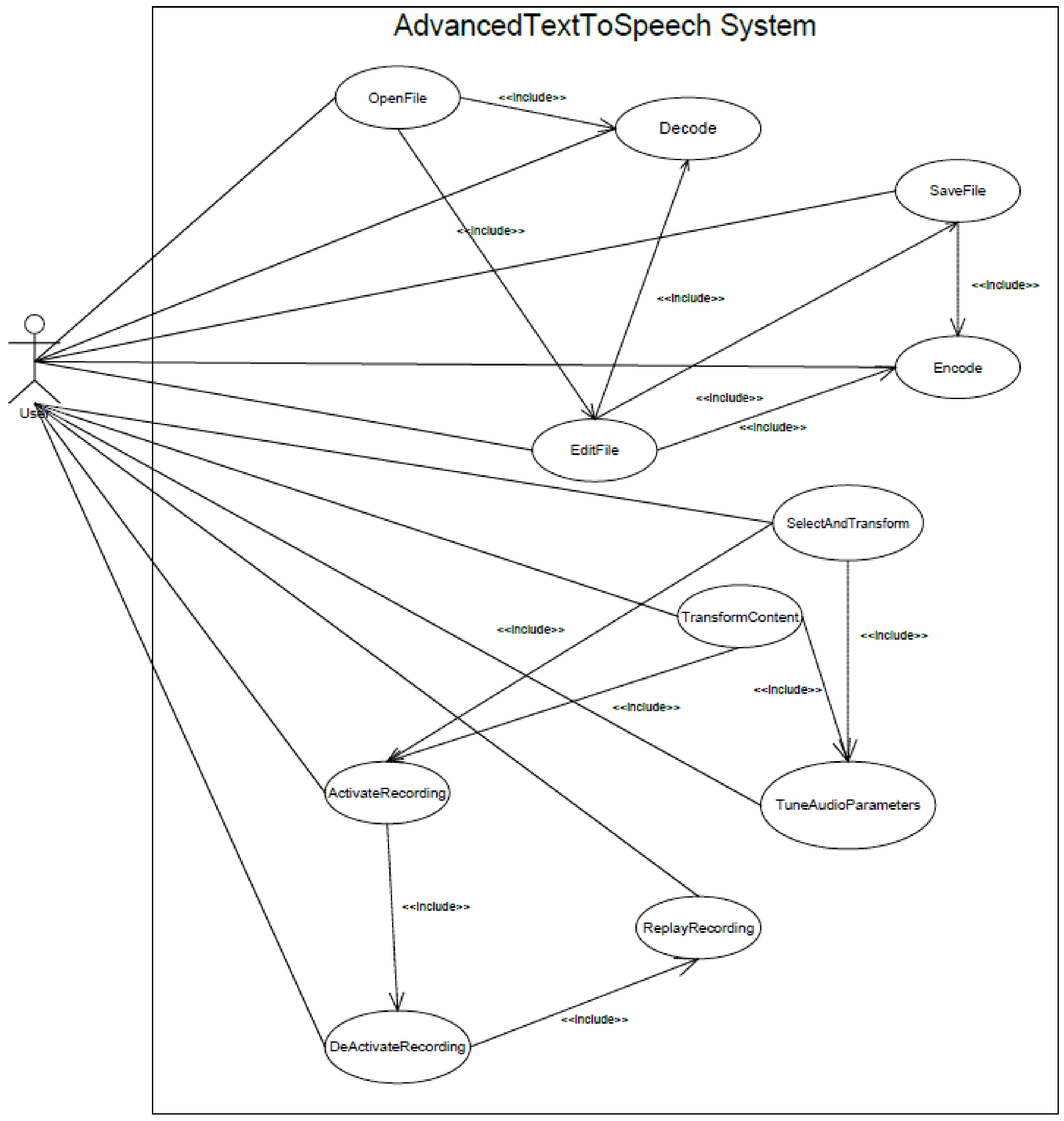


FIGURE 1: USE CASE DIAGRAM

3.1 OpenFile

Use case ID	UC1
Actors	User
Description & Goals	The use case is responsible for loading the file from the user
Pre conditions	1. The application is up and running
Main flow of events	1. The use case starts when the user presses the button “File/Open” from the application 2. The system receives the path of the file from the user, in order to open it 3. The system projects on the screen the selected file and reads the content of the file
Alternative flow	-
Post conditions	The file has been loaded into the application, so the user can edit it
Exceptions	1. If the file does not exist, then the system will project an error message

3.2 Encode

Use case ID	US2
Actors	User
Descritpion & Goals	The use case is responsible for encoding the content of the file
Pre conditions	The file must exist and opened from the system
Main flow of events	1. The use case starts when the user selects an encoding type 2. The user must select between the two types of encoding (Atbash or Rot-13) 3. The system starts the encoding of the file with the selected encoding method
Alternative flow	-
Post conditions	1. The file has been encoded with the selected method from the user
Exceptions	1. Projects an error message when the document is empty

3.3 Decode

Use case ID	UC3
Actors	User
Description & Goals	The use case is responsible for decoding the content of the document
Pre conditions	The file must exist, with its content encoded. It must be opened from the system
Main flow of events	1. The use case starts when the user selects a decoded type, while opening a document 2. The system decodes the document
Alternative flow	-
Post conditions	1. The content of the document has been decoded
Exceptions	-

3.4 EditFile

Use case ID	UC4
Actors	User
Description & Goals	The use case is responsible for the edit of the loaded file from the user
Pre conditions	The file should be loaded in the environment of the application. Its content should be decoded
Main flow of events	1. The use case starts when the user presses the button “File/Edit”
Alternative flow	-
Post conditions	1. The file has been edited
Exceptions	-

3.5 SaveFile

Use case ID	UC5
Actors	User
Description & Goals	The use case is responsible for saving the document in the disk
Pre conditions	The file must exist and be opened by the system.
Main flow of events	1. The use case starts when the user presses the button "File/Save" 2. The system asks the user the type of the document, the method of the encoding (if the user wants to encode) and the name of the file 3. The user gives the requested information 4. The system saves the file with the changes to disk
Alternative flow	-
Post conditions	1. The file is saved to the disk with the given information
Exceptions	1. If there is a problem saving the file to the disk, then the system will project an error message 2. If the user does not give the requested information, then the system projects an error message

3.6 TransformContents

Use case ID	UC6
Actors	User
Description & Goals	The use case is responsible for the transformation of the file content from text to speech
Pre conditions	The file must be loaded from the disk. It should not be empty and must be decoded
Main flow of events	1. The use case starts when the user presses the button “Text To Speech/All Contents” 2. The system converts the content of the file to speech
Alternative flow	-
Post conditions	1. The content of the file has been converted to speech
Exceptions	1. If the file is empty, then the system projects an error message 2. If the system fails during the transformation of the contents to speech, then it projects an error message

3.7 SelectAndTransform

Use case ID	UC7
Actors	User
Description & Goals	The use case selects a part from the loaded file to transform it into speech
Pre conditions	The file has been loaded with its content
Main flow of events	1. The use case starts when the user pushes the button “Text To Speech/Part of Contents” 2. The user chooses from the file the lines he wants to transform into speech
Alternative flow	-
Post conditions	1. The line space for conversion into speech have been selected
Exceptions	1. The user has to give a valid line space. For example if the document has 10 lines and he chooses the space 5-7, then the space is valid, else if he choosesthe space 11-20, then is not valid. If the user selects invalid spce line, then the system will project an error message

3.8 TuneAudioParameters

Use case ID	UC8
Actors	User
Description & Goals	The use case is responsible for tuning the parameters
Pre conditions	The application is up and running
Main flow of events	1. The use case starts when the user press the button “Sound Settings” 2. The user gives the parameters desired values to the system for the volume the speech rate and the pitch 3. The system tunes the parameters
Alternative flow	-
Post conditions	1. The parameters have been tuned from the desired values of the user
Exceptions	-

3.9 ActivateRecording

Use case ID	UC9
Actors	User
Description & Goals	The use case activates the recording session.
Pre conditions	The application is up and running
Main flow of events	1. The use case starts when the user press the button “Record Settings/ Activate Recording” 2. The system starts to record the actions, the commands and the operations of the user.
Alternative flow	-
Post conditions	1. The operations/actions of the user related to the system’s functionality have been saved
Exceptions	1. If there is no memory storage for the recording, then project an error message 2. If an operation, which is chosen from the user is displayed incorrectly, then display an error message and terminate the procedure

3.10 ReplayRecording

Use case ID	UC10
Actors	User
Description & Goals	The use case is responsible for replaying the recording from the user
Pre conditions	Has to exist a recording in the disk
Main flow of events	1. The use case starts when the user press the button "Record Settings/Replay recording" 2. The system play's the recording which is been selected from the user
Alternative flow	-
Post conditions	1. The replay has been displayed by the system
Exceptions	1. Displays an error message if there is not a recorded file in the disk

3.11 DeActivateRecording

Use case ID	UC11
Actors	User
Pre conditions	The recording session should be enabled
Main flow of events	1. The use case starts when the user press the button "Record Settings/ De-activate recording" 2. The system stops the recording procedure
Alternative flow	-
Post conditions	1. The recording session is disabled
Exceptions	1. Displays an error message if the user did not start the recording

4 Design

4.1 Architecture

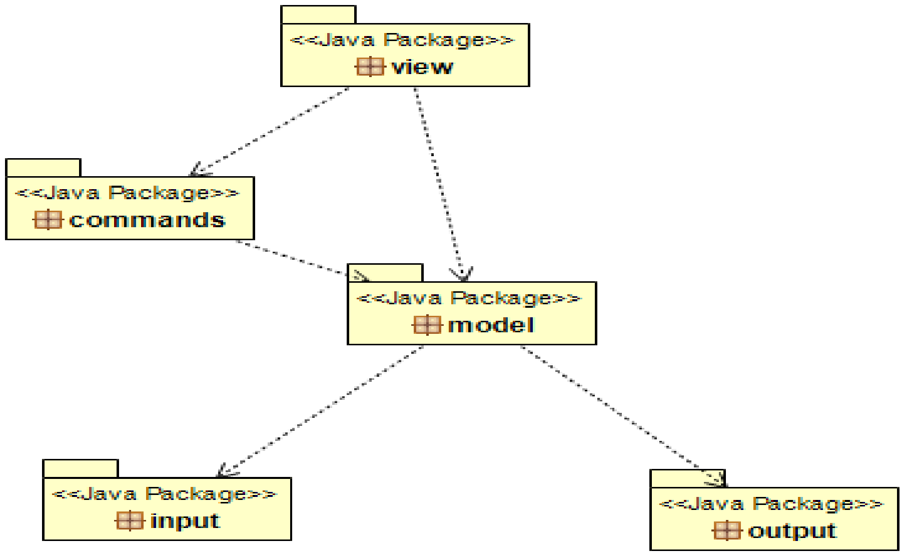
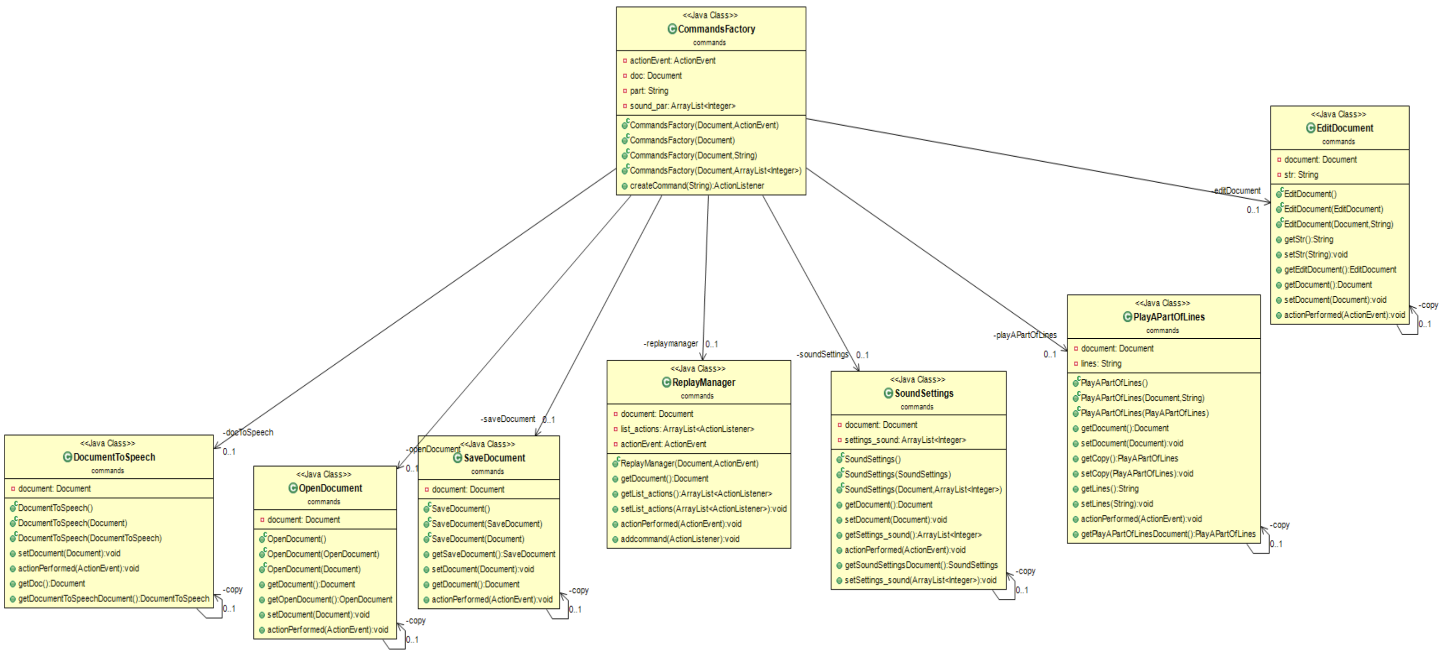


FIGURE 2: UML PACKAGE DIAGRAM

4.2 Design



<<Java Class>>
CommandsFactory
commands
actionEvent: ActionEvent
doc: Document
part: String
sound_par: ArrayList<Integer>
CommandsFactory(Document,ActionEvent)
CommandsFactory(Document)
CommandsFactory(Document,String)
CommandsFactory(Document,ArrayList<Integer>)
createCommand(String):ActionListener
-editDocument
0..1
<<Java Class>>
EditDocument
commands
document: Document
str: String
EditDocument()
EditDocument(EditDocument)
EditDocument(Document,String)
getStr():String
setStr(String):void
getEditDocument():EditDocument
getDocument():Document
setDocument(Document):void
actionPerformed(ActionEvent):void
-copy
0..1
-playAPartOfLines
0..1
<<Java Class>>
PlayAPartOfLines
commands
document: Document
lines: String
PlayAPartOfLines()
PlayAPartOfLines(Document,String)
PlayAPartOfLines(PlayAPartOfLines)
getDocument():Document
setDocument(Document):void
getCopy():PlayAPartOfLines
setCopy(PlayAPartOfLines):void
getLines():String
setLines(String):void
actionPerformed(ActionEvent):void
getPlayAPartOfLinesDocument():PlayAPartOfLines
-copy
0..1
-replaymanager
0..1
<<Java Class>>
ReplayManager
commands
document: Document
list_actions: ArrayList<ActionListener>
actionEvent: ActionEvent
ReplayManager(Document,ActionEvent)
getDocument():Document
getList_actions():ArrayList<ActionListener>
setList_actions(ArrayList<ActionListener>):void
actionPerformed(ActionEvent):void
addcommand(ActionListener):void
-soundSettings
0..1
<<Java Class>>
SoundSettings
commands
document: Document
settings_sound: ArrayList<Integer>
SoundSettings()
SoundSettings(SoundSettings)
SoundSettings(Document,ArrayList<Integer>)
getDocument():Document
setDocument(Document):void
getSettings_sound():ArrayList<Integer>
actionPerformed(ActionEvent):void
getSoundSettingsDocument():SoundSettings
setSettings_sound(ArrayList<Integer>):void
-copy
0..1
-docToSpeech
0..1
<<Java Class>>
DocumentToSpeech
commands
document: Document
DocumentToSpeech()
DocumentToSpeech(Document)
DocumentToSpeech(DocumentToSpeech)
setDocument(Document):void
actionPerformed(ActionEvent):void
getDoc():Document
getDocumentToSpeechDocument():DocumentToSpeech
-copy
0..1
-openDocument
0..1
<<Java Class>>
OpenDocument
commands
document: Document
OpenDocument()
OpenDocument(OpenDocument)
OpenDocument(Document)
getDocument():Document
getOpenDocument():OpenDocument
setDocument(Document):void
actionPerformed(ActionEvent):void
-copy
0..1
-saveDocument
0..1
<<Java Class>>
SaveDocument
commands
document: Document
SaveDocument()
SaveDocument(SaveDocument)
SaveDocument(Document)
getSaveDocument():SaveDocument
setDocument(Document):void
getDocument():Document
actionPerformed(ActionEvent):void
-copy
0..1

FIGURE 3: UML COMMANDS PACKAGE CLASS DIAGRAM

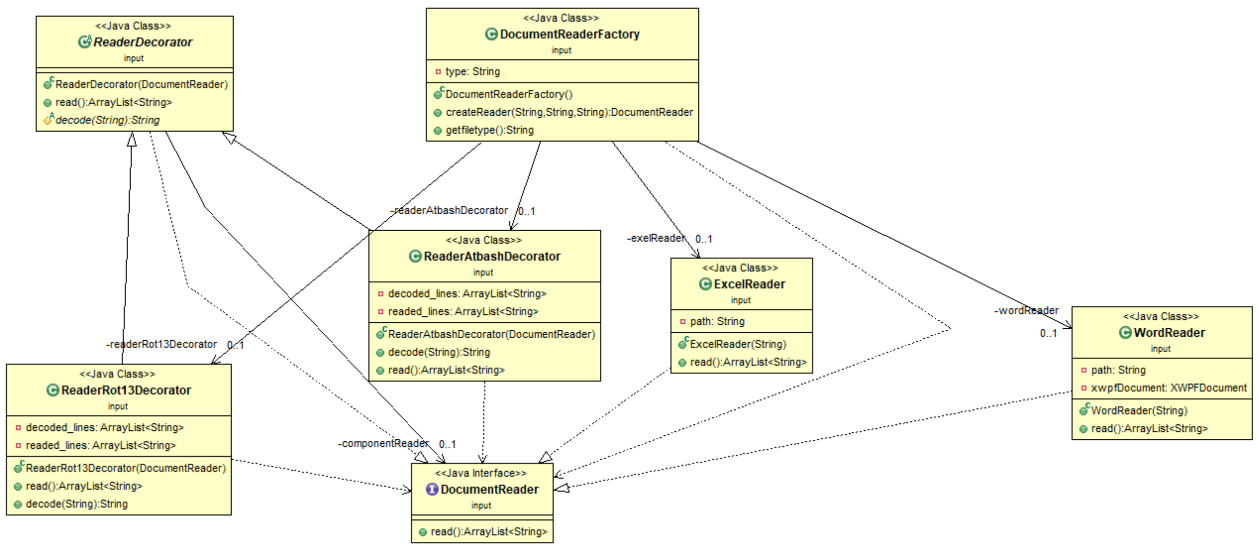
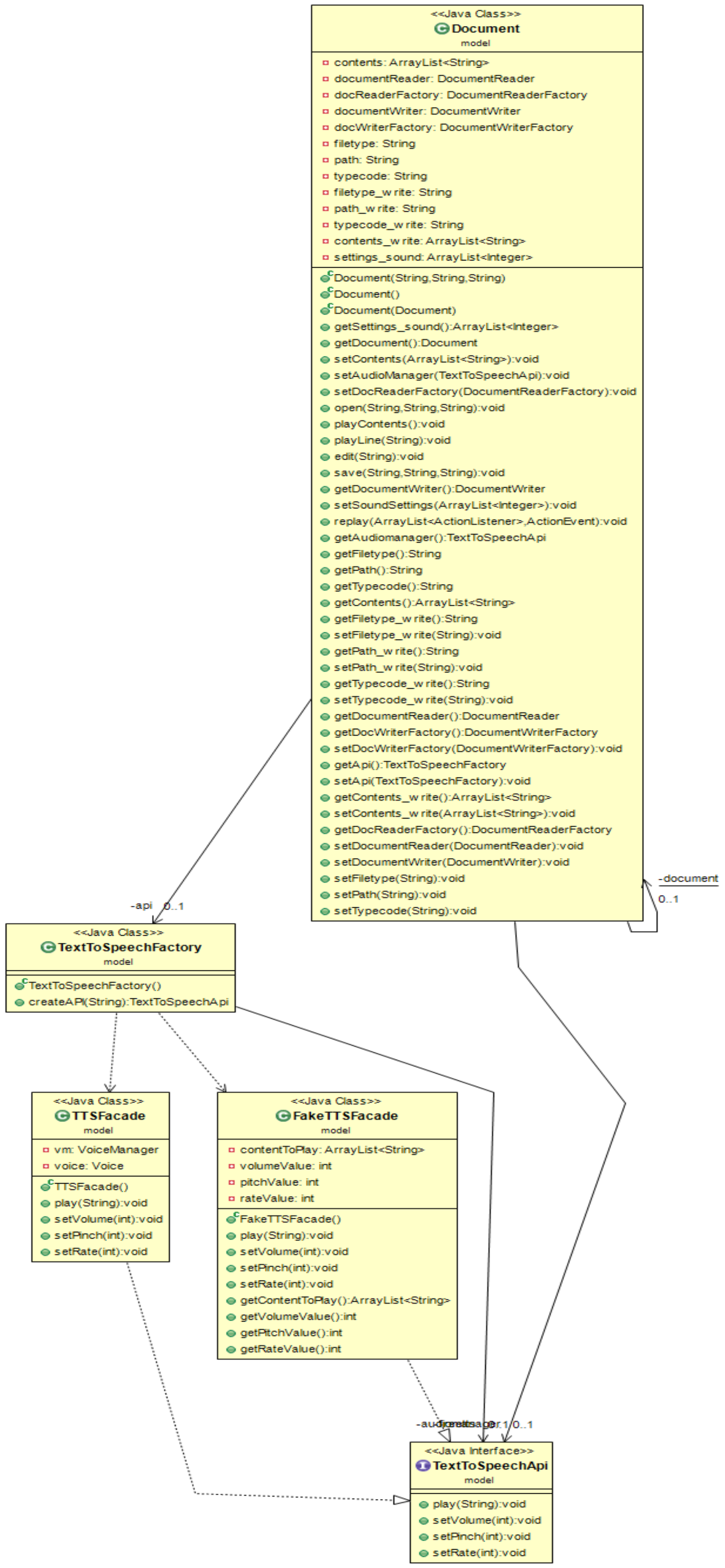


FIGURE 4: UML INPUT PACKAGE CLASS DIAGRAM



<<Java Class>>
Document
model
contents: ArrayList<String>
documentReader: DocumentReader
docReaderFactory: DocumentReaderFactory
documentWriter: DocumentWriter
docWriterFactory: DocumentWriterFactory
filetype: String
path: String
typecode: String
filetype_write: String
path_write: String
typecode_write: String
contents_write: ArrayList<String>
settings_sound: ArrayList<Integer>
Document(String,String,String)
Document()
Document(Document)
getSettings_sound():ArrayList<Integer>
getDocument():Document
setContents(ArrayList<String>):void
setAudioManager(TextToSpeechApi):void
setDocReaderFactory(DocumentReaderFactory):void
open(String,String,String):void
playContents():void
playLine(String):void
edit(String):void
save(String,String,String):void
getDocumentWriter():DocumentWriter
setSoundSettings(ArrayList<Integer>):void
replay(ArrayList<ActionListener>,ActionEvent):void
getAudiomanager():TextToSpeechApi
getFiletype():String
getPath():String
getTypecode():String
getContents():ArrayList<String>
getFiletype_write():String
setFiletype_write(String):void
getPath_write():String
setPath_write(String):void
getTypecode_write():String
setTypecode_write(String):void
getDocumentReader():DocumentReader
getDocWriterFactory():DocumentWriterFactory
setDocWriterFactory(DocumentWriterFactory):void
getApi():TextToSpeechFactory
setApi(TextToSpeechFactory):void
getContents_write():ArrayList<String>
setContents_write(ArrayList<String>):void
getDocReaderFactory():DocumentReaderFactory
setDocumentReader(DocumentReader):void
setDocumentWriter(DocumentWriter):void
setFiletype(String):void
setPath(String):void
setTypecode(String):void
-document
0..1
-api
0..1
<<Java Class>>
TextToSpeechFactory
model
TextToSpeechFactory()
createAPI(String):TextToSpeechApi
<<Java Class>>
TTSFacade
model
vm: VoiceManager
voice: Voice
TTSFacade()
play(String):void
setVolume(int):void
setPinch(int):void
setRate(int):void
<<Java Class>>
FakeTTSFacade
model
contentToPlay: ArrayList<String>
volumeValue: int
pitchValue: int
rateValue: int
FakeTTSFacade()
play(String):void
setVolume(int):void
setPinch(int):void
setRate(int):void
getContentToPlay():ArrayList<String>
getVolumeValue():int
getPitchValue():int
getRateValue():int
-audiomanager
0..1
0..1
<<Java Interface>>
TextToSpeechApi
model
play(String):void
setVolume(int):void
setPinch(int):void
setRate(int):void

FIGURE 5: UML MODEL PACKAGE CLASS DIAGRAM

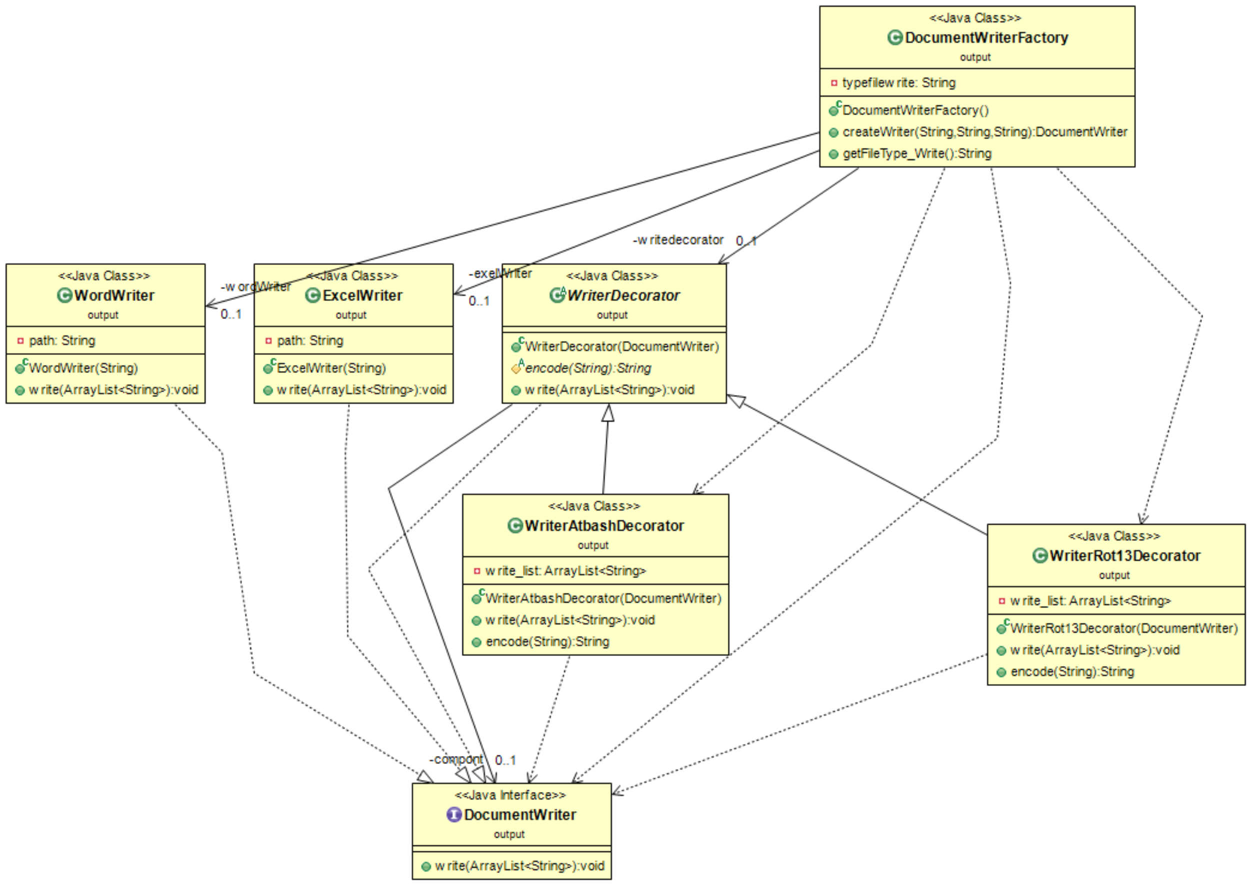
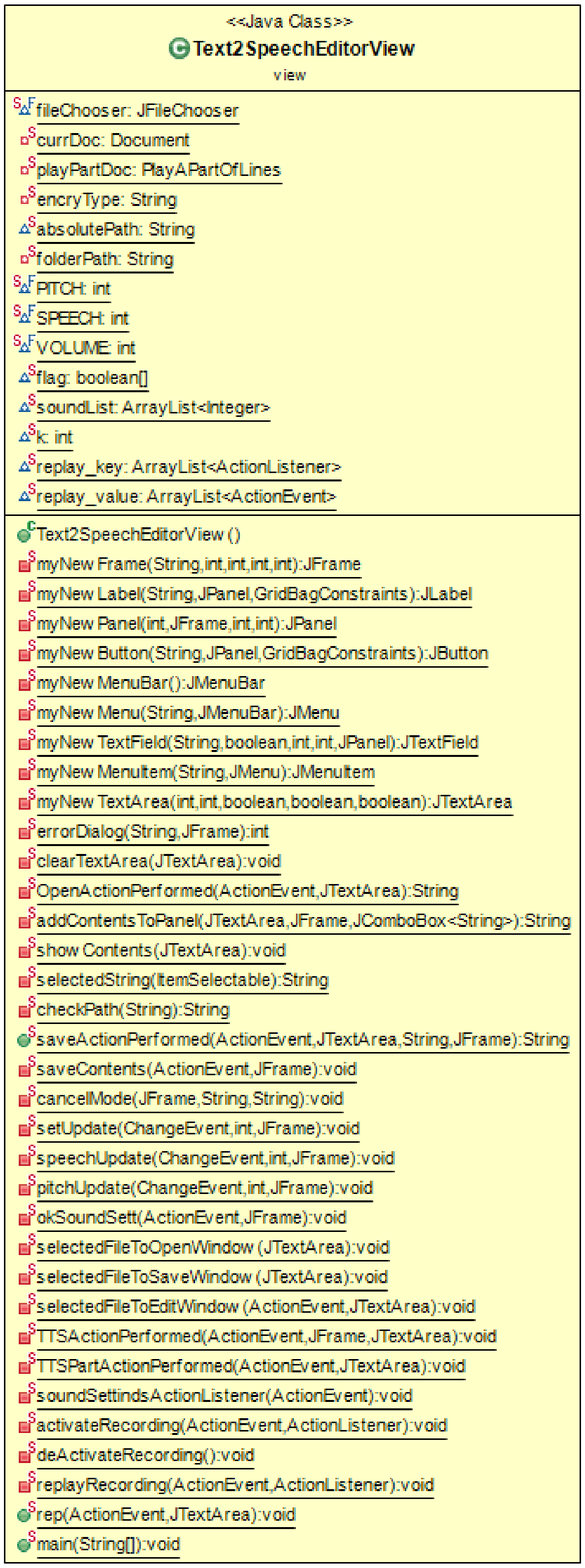


FIGURE 6: UML OUTPUT PACKAGE CLASS DIAGRAM



```
<<Java Class>>
Text2SpeechEditorView
view

fileChooser: JFileChooser
currDoc: Document
playPartDoc: PlayAPartOfLines
encryType: String
absolutePath: String
folderPath: String
PITCH: int
SPEECH: int
VOLUME: int
flag: boolean[]
soundList: ArrayList<Integer>
k: int
replay_key: ArrayList<ActionListener>
replay_value: ArrayList<ActionEvent>

Text2SpeechEditorView ()
myNewFrame(String,int,int,int,int):JFrame
myNewLabel(String,JPanel,GridBagConstraints):JLabel
myNewPanel(int,JFrame,int,int):JPanel
myNewButton(String,JPanel,GridBagConstraints):JButton
myNewMenuBar():JMenuBar
myNewMenu(String,JMenuBar):JMenu
myNewTextField(String,boolean,int,int,JPanel):JTextField
myNewMenuItem(String,JMenu):JMenuItem
myNewTextArea(int,int,boolean,boolean,boolean):JTextArea
errorDialog(String,JFrame):int
clearTextArea(JTextArea):void
OpenActionPerformed(ActionEvent,JTextArea):String
addContentsToPanel(JTextArea,JFrame,JComboBox<String>):String
showContents(JTextArea):void
selectedString(ItemSelectable):String
checkPath(String):String
saveActionPerformed(ActionEvent,JTextArea,String,JFrame):String
saveContents(ActionEvent,JFrame):void
cancelMode(JFrame,String,String):void
setUpdate(ChangeEvent,int,JFrame):void
speechUpdate(ChangeEvent,int,JFrame):void
pitchUpdate(ChangeEvent,int,JFrame):void
okSoundSett(ActionEvent,JFrame):void
selectedFileToOpenWindow (JTextArea):void
selectedFileToSaveWindow (JTextArea):void
selectedFileToEditWindow (ActionEvent,JTextArea):void
TTSActionPerformed(ActionEvent,JFrame,JTextArea):void
TTSPartActionPerformed(ActionEvent,JTextArea):void
soundSettindsActionListener(ActionEvent):void
activateRecording(ActionEvent,ActionListener):void
deActivateRecording():void
replayRecording(ActionEvent,ActionListener):void
rep(ActionEvent,JTextArea):void
main(String[]):void
```

FIGURE 7: UML VIEW PACKAGE DIAGRAM

Below are the classes of the system in a CRC form to observe their responsibilities regarding the functionality of system and their collaboration in order to be operational. So the application can be functional and run without problems.

Class Name: CommandsFactory	
Responsibilities:	**Collaborations:**
▪ Create Commands objects ▪ Connects command actions with back-end	▪ Document ▪ OpenDocument ▪ EditDocument ▪ SaveDocument ▪ DocumentToSpeech ▪ PlayAPartOfLines ▪ SoundSettings ▪ ReplayManager ▪ Text2SpeechEditorView

Class Name: DocumentToSpeech	
Responsibilities:	**Collaborations:**
▪ Transforms document content to speech	▪ Document

Class Name: EditDocument	
Responsibilities:	**Collaborations:**
▪ Edit the selected document	▪ Document

Class Name: OpenDocument	
Responsibilities:	**Collaborations:**
▪ Open the selected document from the disk	▪ Document

Class Name: PlayAPartOfLines	
Responsibilities: ▪ Plays a part of transformed document to speech	**Collaborations:** ▪ Document

Class Name: ReplayManager	
Responsibilities: ▪ Triggers the commands to a document	**Collaborations:** ▪ Document ▪ CommandsFactory ▪ DocumentToSpeech ▪ EditDocument ▪ OpenDocument ▪ PlayAPartOfLines ▪ SaveDocument ▪ SoundSettings

Class Name: SaveDocument	
Responsibilities: ▪ Saves the document to disk	**Collaborations:** ▪ Document

Class Name: SoundSettings	
Responsibilities: ▪ Creates the command settings	**Collaborations:** ▪ Document

Class Name: DocumentReader	
Responsibilities:	**Collaborations:**
▪ Interface for read functionality	▪ ExcelReader ▪ ReaderAtbashDecorator ▪ ReaderDecorator ▪ ReaderRot13Decorator ▪ WordReader

Class Name: DocumentReaderFactory	
Responsibilities:	**Collaborations:**
▪ Creates reader objects with typefile and encoding type ▪ Connects input with commands functionalities	▪ WordReader ▪ ExcelReader ▪ ReaderAtbashDecorator ▪ ReaderRot13Decorator

Class Name: ExcelReader	
Responsibilities:	**Collaborations:**
▪ Opens an excel document ▪ Reads its content	▪ DocumentReader ▪ DocumentReaderFactory

Class Name: ReaderAtbashDecorator	
Responsibilities:	**Collaborations:**
▪ Reads document and decodes with Atbash encoding type	▪ ReaderDecorator ▪ DocumentReaderFactory

Class Name: ReaderDecorator	
Responsibilities:	**Collaborations:**
▪ Opens and reads encoded documents ▪ Decodes content with an encoding type	▪ DocumentReader

Class Name: ReaderRot13Decorator	
Responsibilities: ▪ Reads document and decodes with Rot13 encoding type	**Collaborations:** ▪ ReaderDecorator ▪ DocumentReaderFactory

Class Name: WordReader	
Responsibilities: ▪ Opens a word document ▪ Reads its content	**Collaborations:** ▪ DocumentReader ▪ DocumentReaderFactory

Class Name: Document	
Responsibilities: ▪ Creates Document objects ▪ Implements all the software functionalities ▪ Connects Gui with back-end ▪ Applies all commands to documents	**Collaborations:** ▪ DocumentReader ▪ DocumentReaderFactory ▪ DocumentWriter ▪ DocumentWriterFactory ▪ Text2SpeechFactory ▪ TTSFacade ▪ CommandsFactory ▪ DocumentToSpeech ▪ EditDocument ▪ NewDocument ▪ OpenDocument ▪ PlayAPartOfLines ▪ ReplayManager ▪ SaveDocument ▪ SoundSettings ▪ Text2SpeechEditorView

Class Name: TextToSpeechApi	
Responsibilities: ▪ Interface for speech parameters functionality	**Collaborations:** ▪ TTSFacade ▪ TextToSpeechFactory

Class Name: TTSFacade	
Responsibilities: ▪ Implements tune parameters ▪ Sets speech content as a VoiceManager object from the FreeTTS library	**Collaborations:** ▪ TextToSpeechApi ▪ TextToSpeechFactory ▪ Document

Class Name: FakeTTSFacade	
Responsibilities: ▪ Implements tune parameters ▪ Sets speech content as a VoiceManager object from the FreeTTS library ▪ Gets the tune parameters and apply to content ▪ Used for junit tests	**Collaborations:** ▪ TextToSpeechApi ▪ TextToSpeechFactory

Class Name: TextToSpeechFactory	
Responsibilities: ▪ Creates freetts objects ▪ Implements all speech functionalities	**Collaborations:** ▪ TextToSpeechApi ▪ TTSFacade ▪ Document

Class Name: DocumentWriter	
Responsibilities:	**Collaborations:**
▪ Interface for write functionality	▪ ExcelWriter ▪ WordWriter ▪ WriterAtbashDecorator ▪ WriterDecorator ▪ WriterRot13Decorator

Class Name: DocumentWriterFactory	
Responsibilities:	**Collaborations:**
▪ Creates writer objects with typefile and encoding type ▪ Connects output with commands functionalities	▪ ExcelWriter ▪ WordWriter ▪ WriterAtbashDecorator ▪ WriterRot13Decortor

Class Name: ExcelWriter	
Responsibilities:	**Collaborations:**
▪ Saves an excel document to the disk ▪ Writes either new content to the document or the same exists	▪ DocumentWriter ▪ DocumentWriterFactory

Class Name: WordWriter	
Responsibilities:	**Collaborations:**
▪ Saves a word document to the disk ▪ Writes either new content to the document or the same exists	▪ DocumentWriter ▪ DocumentWriterFactory

Class Name: WriterDecorator	
Responsibilities: ▪ Writes and saves encoded documents	**Collaborations:** ▪ DocumentWriter

Class Name: WriterAtbashDecorator	
Responsibilities: ▪ Writes the document and encodes with Atbash encoding type	**Collaborations:** ▪ WriterDecorator ▪ DocumentWriterFactory

Class Name: WriterRot13Decorator	
Responsibilities: ▪ Writes the document and encodes with Rot13 encoding type	**Collaborations:** ▪ WriterDecorator ▪ DocumentWriterFactory

Class Name: Text2SpeechEditorView	
Responsibilities: ▪ Creates the graphical content of the application ▪ A gui format for the user ▪ Interaction with various commands and functionalities	**Collaborations:** ▪ Document ▪ CommandsFactory