AdvancedText2SpeechEditor

Sprint Report

<3334,3246, 3058>
Ανδρέας Κατσαντώνης Α.Μ:3246
Ευάγγελος Σταμάτης Α.Μ:3334
Χρήστος Παρασκευόπουλος Α.Μ:3058

VERSIONS HISTORY

Date	Version	Description	Author
26/5/2021	1.0	Final Report for AdvancedText2SpeechEditor	Ευάγγελος Σταμάτης Αντρέας Κατσαντώνης Χρήστος Παρασκευόπουλος

1 Introduction

This document provides information concerning the **<2>** sprint of the project.

1.1 Purpose

Purpose of this document is to give details about the Design. Such details are the Use Cases exhibited in a tabular form, the complete UML diagrams of each package imported from the Eclipse work space and the responsibilities of each class in our project.

1.2 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.

2 Scrum team and Sprint Backlog

2.1 Scrum team

Product Owner	Apostolos Zarras
Scrum Master	Andreas Katsantonis
Development Team	Evangelos Stamatis Andreas Katsantonis Christos Paraskevopoulos

2.2 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
1	**26/5/2021**	**27/5/2021**	**Zero**	**All**

3 Use Cases

3.1 <Use Case 1>

Use case ID	UC1
Actors	User
Pre conditions	Execution of the program
Main flow of events	1. The use case starts when the user clicks the <Open> button from the menu bar 2. The user clicks the Browse… button to select the file he wants to open 2.1. A pop up window appears and asks the user to input what encoding the document has
Alternative flow 1	If the user chose a wrong file to open he can press the <X> button and re do the process
Alternative flow 2	If the user types the wrong decoding method he can erase the text from the editor and he can re do the process
Post conditions	The user sees the decoded text of the file he opened (docx, xlsx) in the editor

3.2 <Use Case 2>

Use case ID	UC2
Actors	User
Pre conditions	A file has been opened in the Editor and it's contents are on the screen
Main flow of events	1. The use case starts when the user selects the <Edit> Button from the menu bar 2. The user comes across three different choices he can either copy, paste or cut a selected part of the contents
Alternative flow 1	To copy and paste a part of the file he can select that part with the mouse and after that, click copy button. Then to paste move the cursor in the appropriate place and hit the paste button

Alternative flow 2	To cut and paste a part of the file he can select that part with the mouse and after that, click cut button. Then to paste move the cursor in the appropriate place and hit the paste button
Post conditions	There might be some changes in the contents of the file.

3.3 <Use Case 3>

Use case ID	UC3
Actors	User
Pre conditions	Either an opened file with some changes in the contents or some new written text on the editor.
Main flow of events	1. The use case starts when the user presses the <Save> button on the menu bar 2. A pop up file explorer appears and the user must input the name of the file he wants to save after choosing the right directory 2.1. A pop up window appears and the user inputs the encoding method he wants to encode the file he is about to save
Alternative flow 1	If the user chooses a wrong path of directories to save the file he can press either <cancel> or the <X> button and re do the process
Alternative flow 2	If the user chooses the wrong encoding method he can press the <X> button and re do the process
Post conditions	The file is save in the users dick

3.4 <Use Case 4>

Use case ID	UC4
Actors	User
Pre conditions	Either an opened file with some changes in the contents or some new written text on the editor.
Main flow of events	1. The use case starts when the user presses the <Contents To Audio> button from the menu bar. 2. The user presses the <Contents> button and he listens to the contents of the file.
Post conditions	The contents of the file are being played from the speakers

3.5 <Use Case 5>

Use case ID	UC5
Actors	User
Pre conditions	Either an opened file with some changes in the contents or some new written text on the editor. Part of the contents is selected to be played.
Main flow of events	1. The use case starts when the user presses the <Contents To Audio> button from the menu bar. 2. The user presses the <Selected Area> button and he listens to the contents of the file.
Post conditions	The selected area of the file is being played from the speakers

3.6 <Use Case 6>

Use case ID	UC6
Actors	User
Pre conditions	None
Main flow of events	1. The use case starts when the user clicks the <Tune Audio> button from the menu bar 2. The user can either choose to tune the Volume, the Pitch or the Rate 2.1 The user picks a parameter to tune and the system depicts a pop up window with a field where the user can write the value of the parameter he chose.
Post conditions	The value of the parameter the user chose to tune has changed.

3.7 <Use Case 7>

Use case ID	UC7
Actors	User
Pre conditions	Either an opened file with some changes in the contents or some new written text on the editor.
Main flow of events	1. The use case starts when the user clicks the <Recording> button from the menu bar 2. The user clicks the <Activate Rec> button from the drop down menu 2.1 The system records every command the user executes.
Alternative flow 1	The user can run any command and after hit the <Recording> and <Activate Rec> buttons se he can record from now on
Post conditions	The commands the user executes are being saves so they can be played back

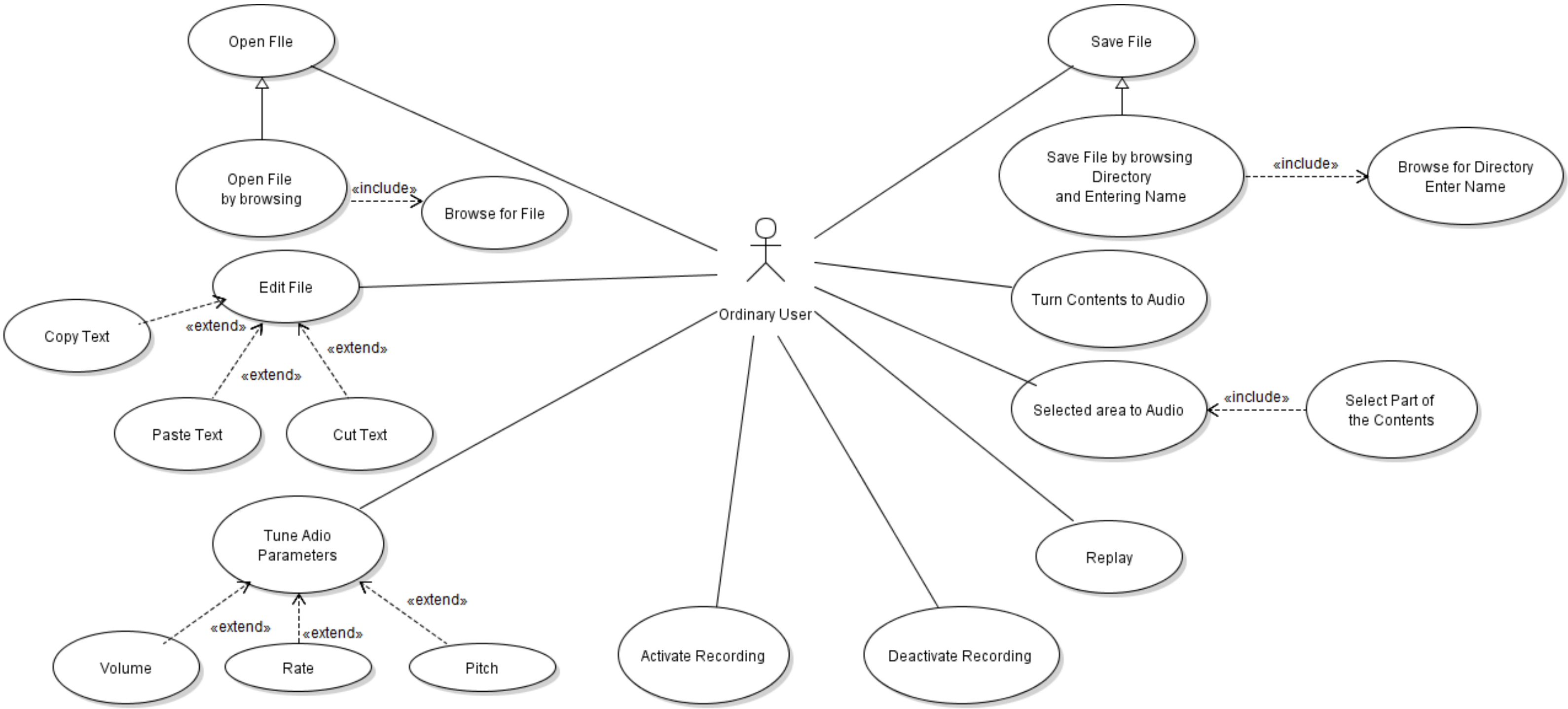
3.8 <Use Case 8>

Use case ID	UC8
Actors	User
Pre conditions	The user clicked <Activate Rec > and has executed a series of commands
Main flow of events	1. The use case starts when the user presses the <Replay> button from the drop down menu. 2. The system re does every command the user executed after it started recording
Post conditions	The series of recorded commands are being replayed

3.9 <Use Case 9>

Use case ID	UC9
Actors	User
Pre conditions	The user has executed the <Start Recording> command
Main flow of events	1. The use case starts when the user clicks the <Deactivate Rec> button from the drop down menu. 2 The systems deactivates the recording and from now on the commands are not being recorded
Post conditions	None

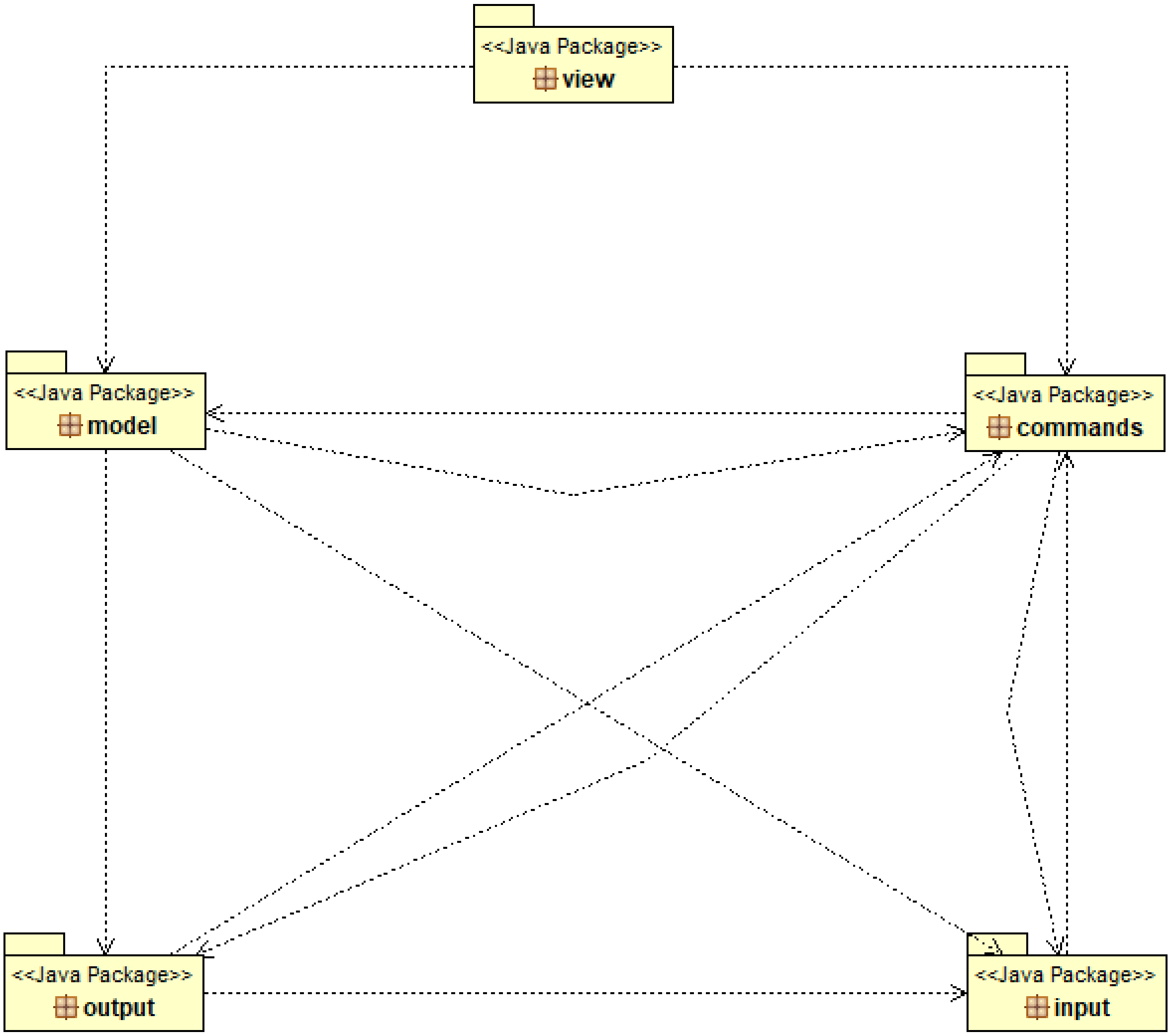
3.10 Use Case Diagram



4 Design

4.1 Architecture

UML package diagram.

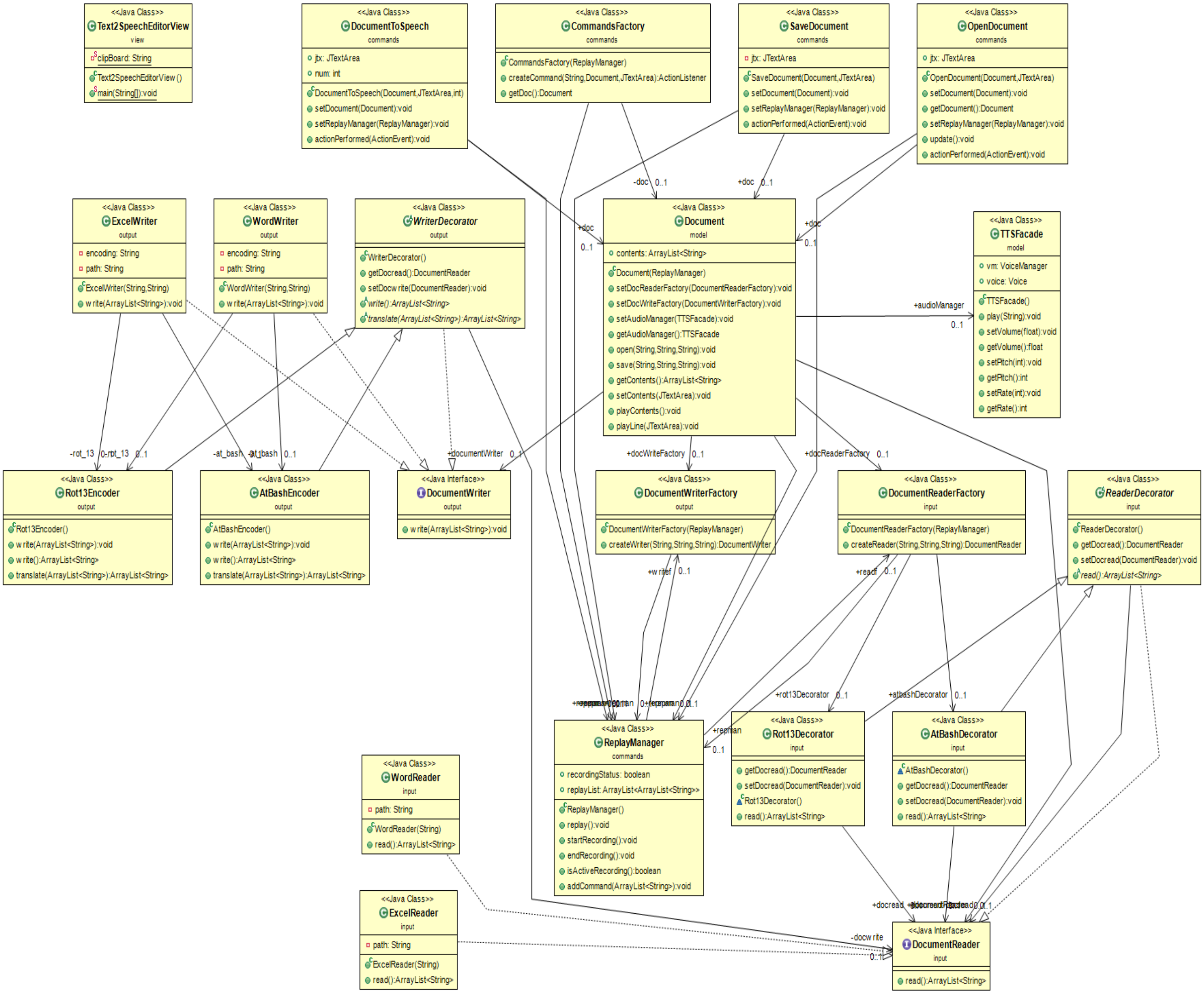


4.2 Design

UML class diagrams

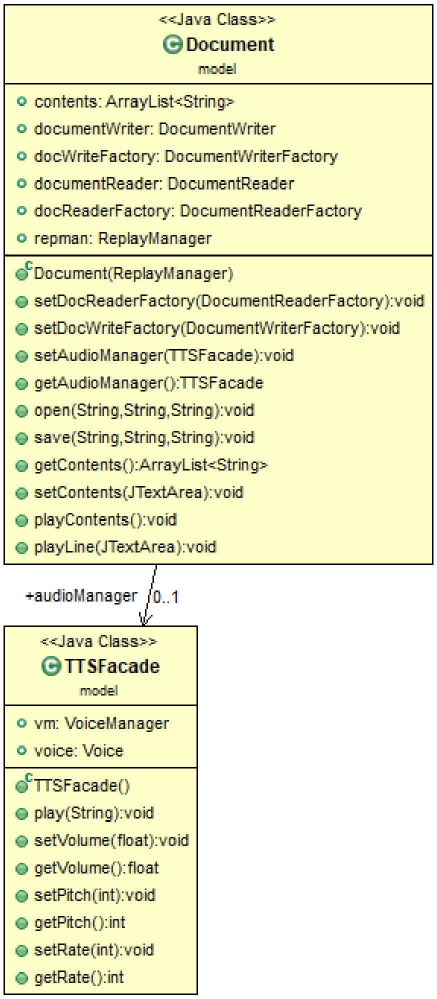
4.2.1 Design

UML class for View package



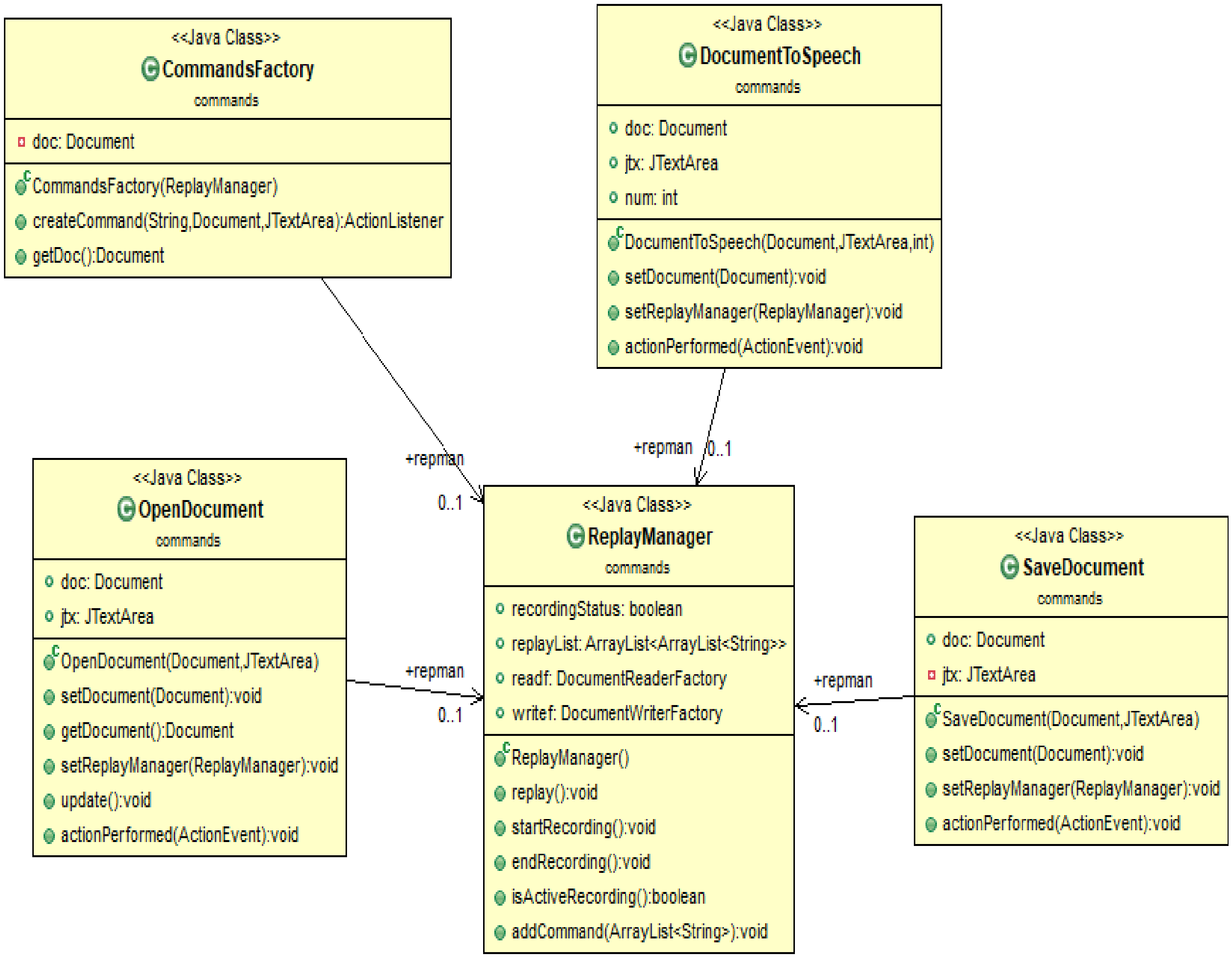
4.2.2 Design

UML class for Model package


<<Java Class>>
Document
model
contents: ArrayList<String>
documentWriter: DocumentWriter
docWriteFactory: DocumentWriterFactory
documentReader: DocumentReader
docReaderFactory: DocumentReaderFactory
repman: ReplayManager
Document(ReplayManager)
setDocReaderFactory(DocumentReaderFactory):void
setDocWriteFactory(DocumentWriterFactory):void
setAudioManager(TTSFacade):void
getAudioManager():TTSFacade
open(String,String,String):void
save(String,String,String):void
getContents():ArrayList<String>
setContents(JTextArea):void
playContents():void
playLine(JTextArea):void
+audioManager
0..1
<<Java Class>>
TTSFacade
model
vm: VoiceManager
voice: Voice
TTSFacade()
play(String):void
setVolume(float):void
getVolume():float
setPitch(int):void
getPitch():int
setRate(int):void
getRate():int

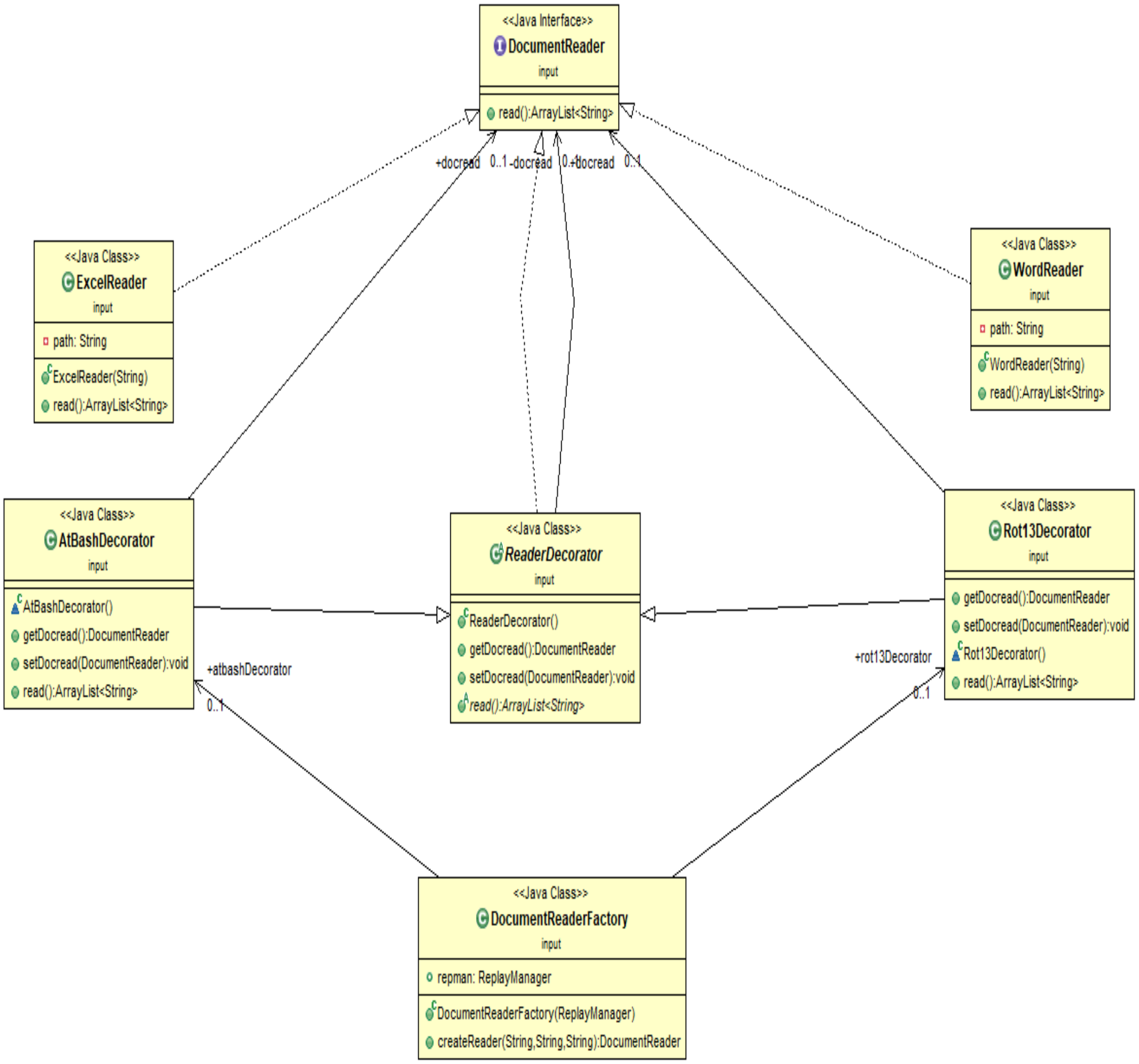
4.2.3 Design

UML class for Commands package


<<Java Class>>
CommandsFactory
commands
doc: Document
CommandsFactory(ReplayManager)
createCommand(String,Document,JTextArea):ActionListener
getDoc():Document
<<Java Class>>
DocumentToSpeech
commands
doc: Document
jtx: JTextArea
num: int
DocumentToSpeech(Document,JTextArea,int)
setDocument(Document):void
setReplayManager(ReplayManager):void
actionPerformed(ActionEvent):void
+repman
0..1
+repman
0..1
<<Java Class>>
OpenDocument
commands
doc: Document
jtx: JTextArea
OpenDocument(Document,JTextArea)
setDocument(Document):void
getDocument():Document
setReplayManager(ReplayManager):void
update():void
actionPerformed(ActionEvent):void
+repman
0..1
<<Java Class>>
ReplayManager
commands
recordingStatus: boolean
replayList: ArrayList<ArrayList<String>>
readf: DocumentReaderFactory
writef: DocumentWriterFactory
ReplayManager()
replay():void
startRecording():void
endRecording():void
isActiveRecording():boolean
addCommand(ArrayList<String>):void
+repman
0..1
<<Java Class>>
SaveDocument
commands
doc: Document
jtx: JTextArea
SaveDocument(Document,JTextArea)
setDocument(Document):void
setReplayManager(ReplayManager):void
actionPerformed(ActionEvent):void

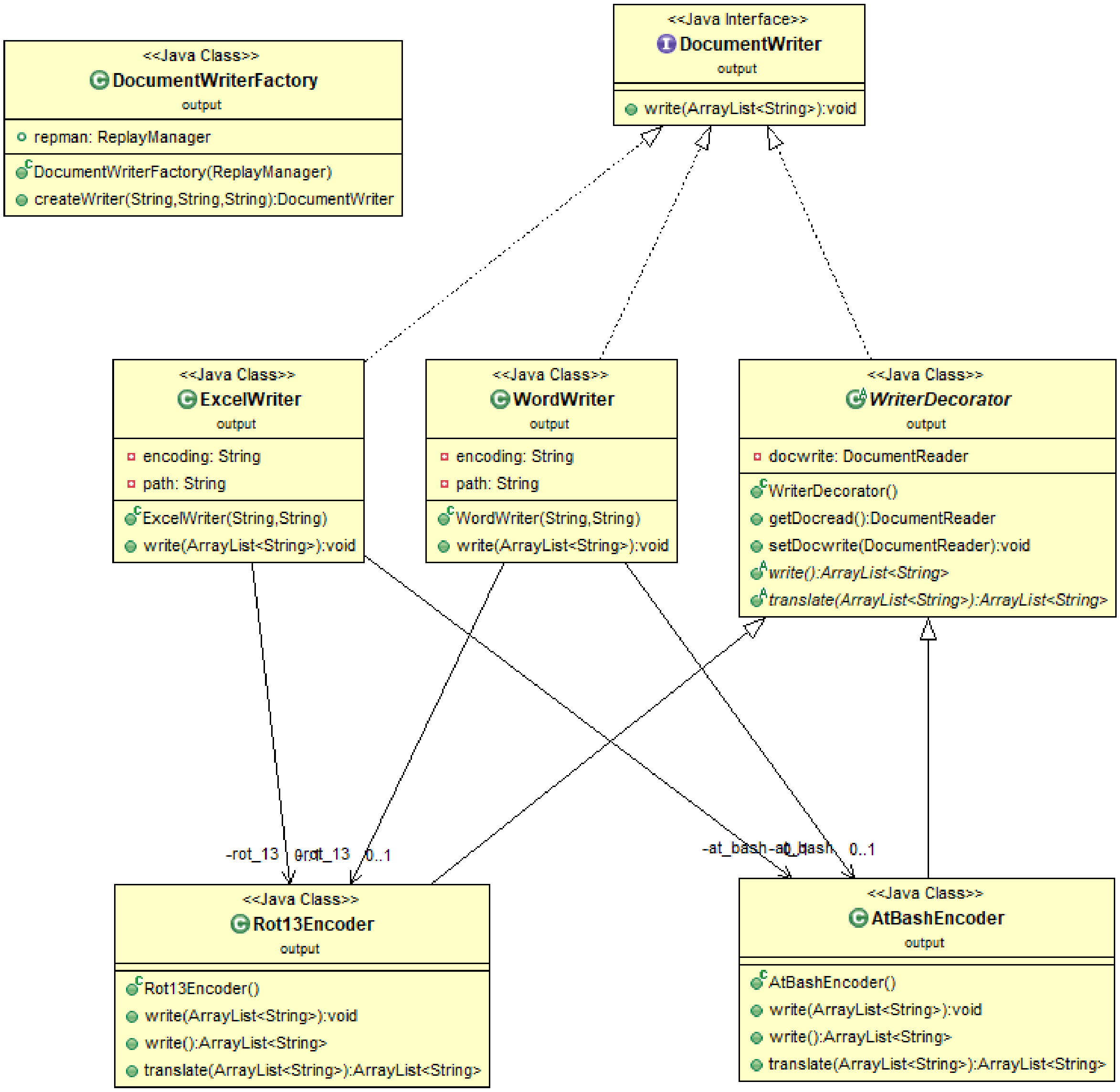
4.2.4 Design

UML class for Input package



4.2.5 Design

UML class for Output package



<<Java Class>>
DocumentWriterFactory
output
repman: ReplayManager
DocumentWriterFactory(ReplayManager)
createWriter(String,String,String):DocumentWriter
<<Java Interface>>
DocumentWriter
output
write(ArrayList<String>):void
<<Java Class>>
ExcelWriter
output
encoding: String
path: String
ExcelWriter(String,String)
write(ArrayList<String>):void
<<Java Class>>
WordWriter
output
encoding: String
path: String
WordWriter(String,String)
write(ArrayList<String>):void
<<Java Class>>
WriterDecorator
output
docwrite: DocumentReader
WriterDecorator()
getDocread():DocumentReader
setDocwrite(DocumentReader):void
write():ArrayList<String>
translate(ArrayList<String>):ArrayList<String>
-rot_13
0..1
-rot_13
0..1
-at_bash
0..1
-at_bash
0..1
<<Java Class>>
Rot13Encoder
output
Rot13Encoder()
write(ArrayList<String>):void
write():ArrayList<String>
translate(ArrayList<String>):ArrayList<String>
<<Java Class>>
AtBashEncoder
output
AtBashEncoder()
write(ArrayList<String>):void
write():ArrayList<String>
translate(ArrayList<String>):ArrayList<String>

4.3 Design

Class Name: Text2SpeechEditorView	
Responsibilities: ▪ G.U.I creation ▪ Bind buttons with Actions ▪ Main method	**Collaborations:** ▪ CommandFactory ▪ DocumentReaderFactory ▪ DocumentWriterFactory

Class Name: CommandsFactory	
Responsibilities: ▪ Commands Handling ▪ Connects Front-End and Back-End	**Collaborations:** ▪ ReplayManager ▪ Document ▪ OpenDocument ▪ SaveDocument ▪ DocumentToSpeech

Class Name: DocumentToSpeech	
Responsibilities: ▪ Turning contents of the document to speech	**Collaborations:** ▪ Document ▪ ReplayManager

Class Name: OpenDocument	
Responsibilities: ▪ Opens a file chooser so the user can input the file he/she wants to open	**Collaborations:** ▪ Document ▪ ReplayManager

Class Name: ReplayManager	
Responsibilities:	**Collaborations:**
▪ Records the actions of the user and stores the for later use	▪ DocumentWriterFactory ▪ DocumentReaderFactory

Class Name: SaveDocument	
Responsibilities:	**Collaborations:**
▪ Opens a file chooser so the user can find the directory and input the name of the file he/she wants to save	▪ Document

Class Name: AtBashDecorator	
Responsibilities:	**Collaborations:**
▪ Reads the file ▪ Decodes the file	▪ DocumentReader

Class Name: DocumentReaderFactory	
Responsibilities:	**Collaborations:**
▪ Chooses the way you have to open a file based on the file type and encoding	▪ AtBashDecorator ▪ Rot13Decorator ▪ ReplayManager ▪ WordReader ▪ ExcelReader

Class Name: ExcelReader	
Responsibilities:	**Collaborations:**
▪ Reads an Excel file into the text area of the editor	

Class Name: Rot13Decorator	
Responsibilities: ■ Reads The file ■ Decodes the file	**Collaborations:** ■ DocumentReader

Class Name: WordReader	
Responsibilities: ■ Reads a Word File into the text area of the Editor	**Collaborations:**

Class Name: Document	
Responsibilities: ■ Initializing the document ■ Handling the document’s functions	**Collaborations:** ■ DocumentWriterFactory ■ DocumentReaderFactory ■ TTSFacade ■ ReplayManager ■ DocumentWriter ■ DocumentReader

Class Name: TTSFacade	
Responsibilities: ■ Makes the tuning of the voice in TTS simple	**Collaborations:**

Class Name: AtBashEncoder	
Responsibilities: ■ Encodes the file	**Collaborations:**

Class Name: DocumentWriterFactory	
Responsibilities:	**Collaborations:**
▪ Decides the way that the system will write the file in the path that the user inputs	▪ ReplayManager ▪ WordWriter ▪ ExcelWriter

Class Name: ExcelWriter	
Responsibilities:	**Collaborations:**
▪ Writes an .xlsx file in the path that the user inputs	▪ Rot13Encoder ▪ AtBashEncoder

Class Name: Rot13Encoder	
Responsibilities:	**Collaborations:**
▪ Encodes the file	

Class Name: WordWriter	
Responsibilities:	**Collaborations:**
▪ Writes an .docx in the path that the user inputs	▪ Rot13Encoder ▪ AtBashEncoder