AdvancedText2SpeechEditor

Sprint Report

Konstantinos Gerogiannis 3196

Marios Gkikopoulos 3205

Alexandros Papathanasiou 3308

VERSIONS HISTORY

Date	Version	Description	Author
29/05/2021	Final	All user-stories and tests working correctly.	Konstantinos Gerogiannis Marios Gkikopoulos Alexandros Papathanasiou

1 Introduction

This document provides information concerning the **final** sprint of the project.

1.1 Purpose

This application was created to transform text into speech, that is provided by the user.

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

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

Test Case ID:	Testcase1
Test Title:	OpenDocumentTest
Description:	Tests if the documents are opened correctly. This test checks for different types of documents and encoding.

Test Case ID:	Testcase2
Test Title:	EditDocumentTest
Description:	Tests if the user can make changes in his currently opened document.

Test Case ID:	Testcase3
Test Title:	SaveDocumentTest
Description:	Tests if the documents are saved correctly. This test checks for different types of documents and encoding.

Test Case ID:	Testcase4
Test Title:	DocumentToSpeechTest
Description:	Tests if the document is properly transformed into speech.

Test Case ID:	Testcase5
Test Title:	SelectedToSpeechTest
Description:	Tests if the selected text is properly transformed into speech.

Test Case ID:	Testcase6
Test Title:	SetAudiotTest
Description:	Tests if the Volume, Speech Rate and Pitch work correctly .

Test Case ID:	Testcase7
Test Title:	StartRecordingTest
Description:	Tests if the recording process has started correctly.

Test Case ID:	Testcase8
Test Title:	ReplayTest
Description:	Tests if all the recorded texts are transformed to speech properly.

Test Case ID:	Testcase9
Test Title:	StopRecordingTest
Description:	Tests if the recording process has stopped correctly.

2.1 Scrum team

Product Owner	Apostolos Zarras
Scrum Master	Marios Gkikopoulos
Development Team	Marios Gkikopoulos, Alexandros Papathanasiou, Konstantinos Gerogiannis

2.2 Sprints

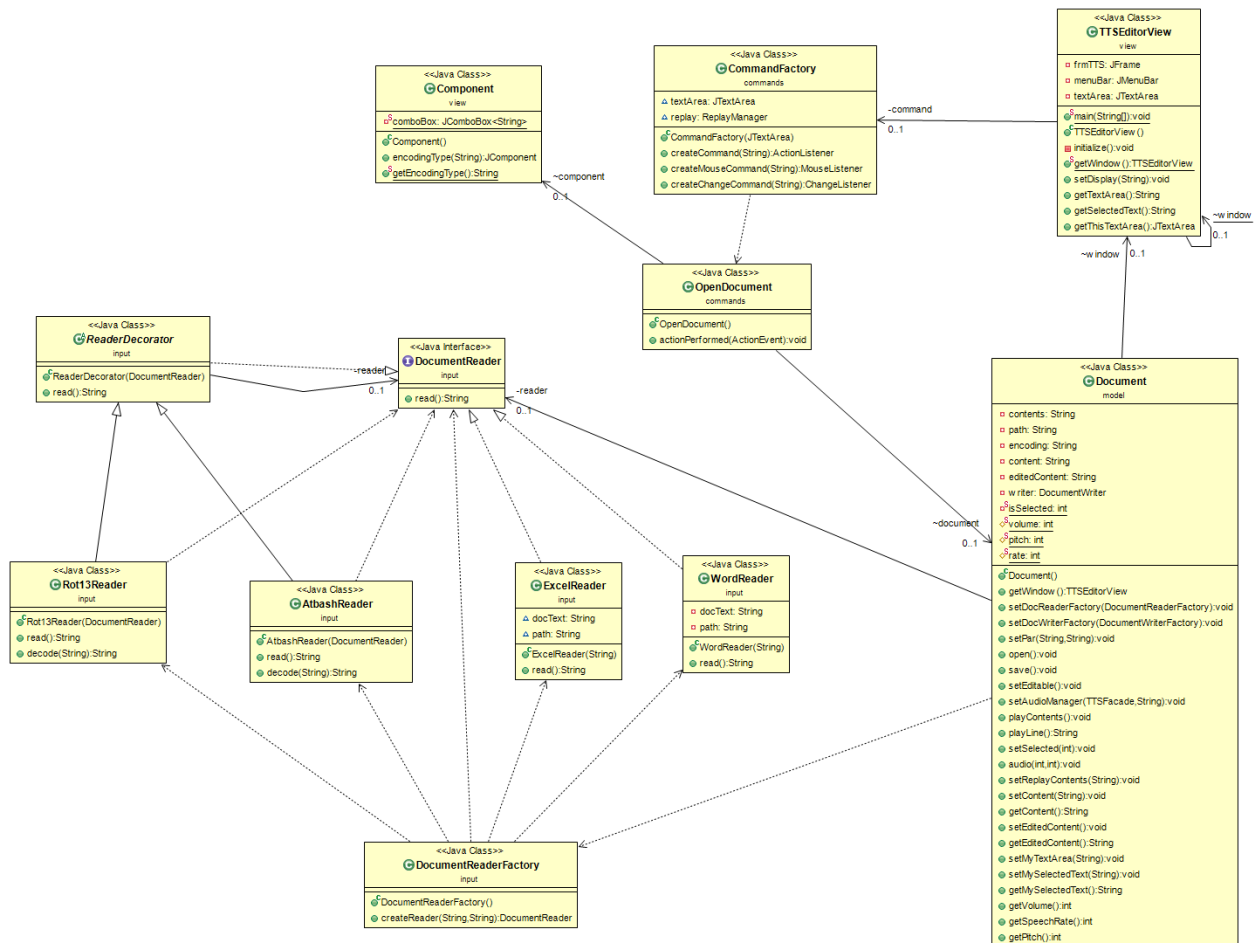| Sprint No	Begin Date	End Date	Number of weeks	User stories
1	01/04/2021	16/04/2021	2 weeks	US1, US2, US3
2	17/04/2021	01/05/2021	2 weeks	US4, US5
3	02/05/2021	23/05/2021	3 weeks	US6, US7, US8, US9

3   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**.>

3.1 <Use Case 1>

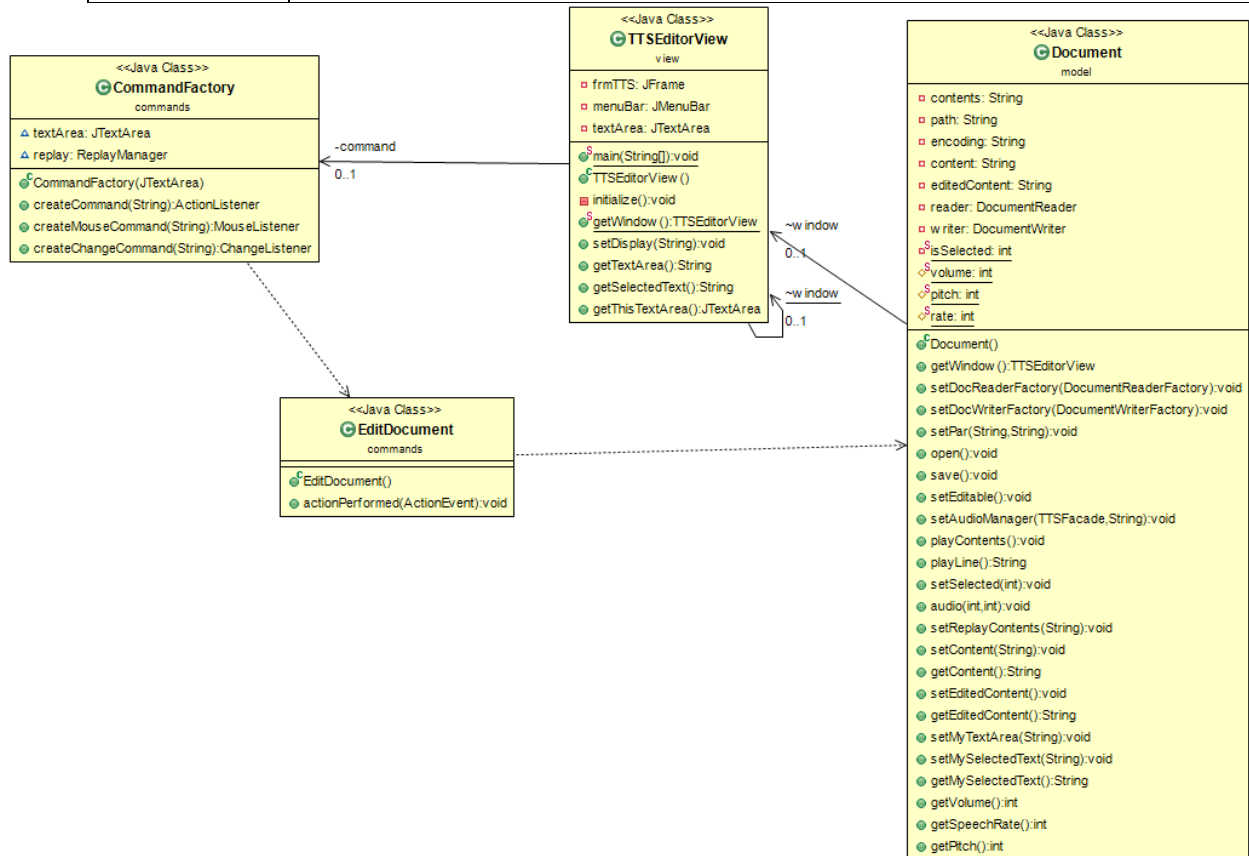|	
Use case ID	US1
Actors	The User
Pre conditions	The program is running and the user has also pressed the "File" menu button.
Main flow of events	1. The use case starts when the user presses the "Open File" button. 2. The system pops up a new file-chooser window and prompts the user to choose a document to open. 3. The user chooses a document and a decoding type if he wishes to.

	4. The system now will show the selected document's contents on its text area.
Alternative flow	The user presses the cancel button and the system pops a "Canceled" pane and goes back to the initial window.
Post conditions	The user can edit or listen to the opened document.

3.2 <Use Case 2>

Use case ID	US2

Actors	The User
Pre conditions	The program is running and the user has also pressed the "File" menu button.
Main flow of events	5. The use case starts when the user presses the "Edit File" button. 6. The systems sets the text area open for editing
Post conditions	The user can type in the text area and change a document if it is opened.

3.3 <Use Case 3>

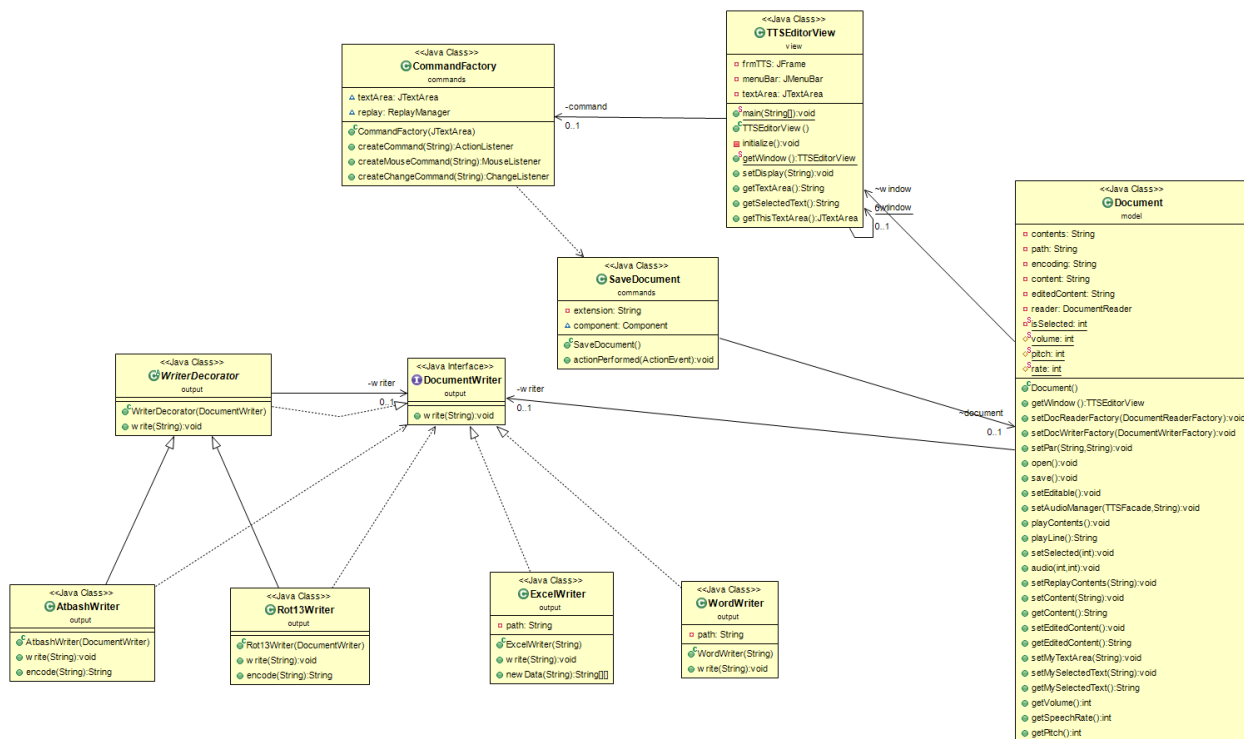| Use case ID	US3
Actors	The User
Pre conditions	The program is running and the user has also pressed the "File" menu button.

Main flow of events	
	7. The use case starts when the user presses the "Save File" button.
	8. The system pops up a new file-chooser window and prompts the user to choose a file to save his document. It also provides an document type, encoding and a name
	9. The user chooses a file and a name, a document and an encoding type if he wishes to.
	10. The system saves the document in the selected file and goes back to initial window.
Alternative flow	The user presses the cancel button and the system pops a "Canceled" pane and goes back to the initial window.
Post conditions	The user can open the saved document or access it outside the application.

3.4 <Use Case 4>

Use case ID	US4
Actors	The User
Pre conditions	The program is running and the user has opened a file or entered any text on the text area.
Main flow of events	11. The use case starts when the user presses the "Play" menu button. 12. The system transforms the text to speech
Alternative flow	The text area is empty and the system remains silent.
Post conditions	The user can listen the text that appears on the text area.

3.5 <Use case 5>

Use case ID	US5
Actors	The User
Pre conditions	The program is running and the user has opened a file or entered any text on the text area.
Main flow of events	13. The use case starts when the user presses the "Selected to Speech" button. 14. The user selects a part of text to listen to. 15. The user presses the "Play" menu button. 16. The system transforms to speech only the text that the user selected
Alternative flow 1	The user has pressed the "Selected to Speech" button and then the "Play" button, but hasn't selected any text. The system will show a "Please select some text" pane.
Post conditions	The user can listen to the selected text.
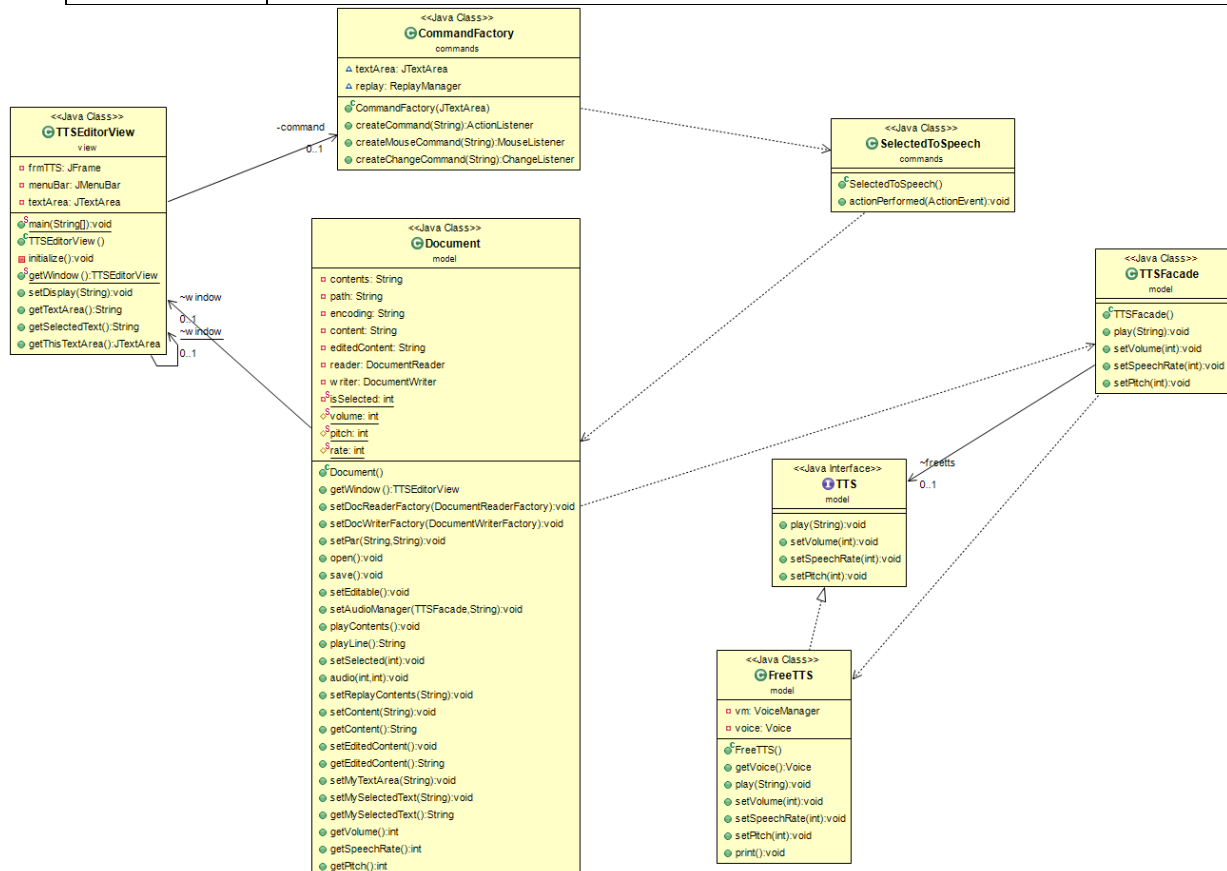
3.6 <Use Case 6>

Use case ID	US6
Actors	The User
Pre conditions	The program is running.
Main flow of events	17. The use case starts when the user presses the "Audio" menu button. 18. The user can set the Volume, Speech Rate and Pitch settings from 0 to 100.
Post conditions	When the user presses the "Play" button he will listen to the text with the modified settings.
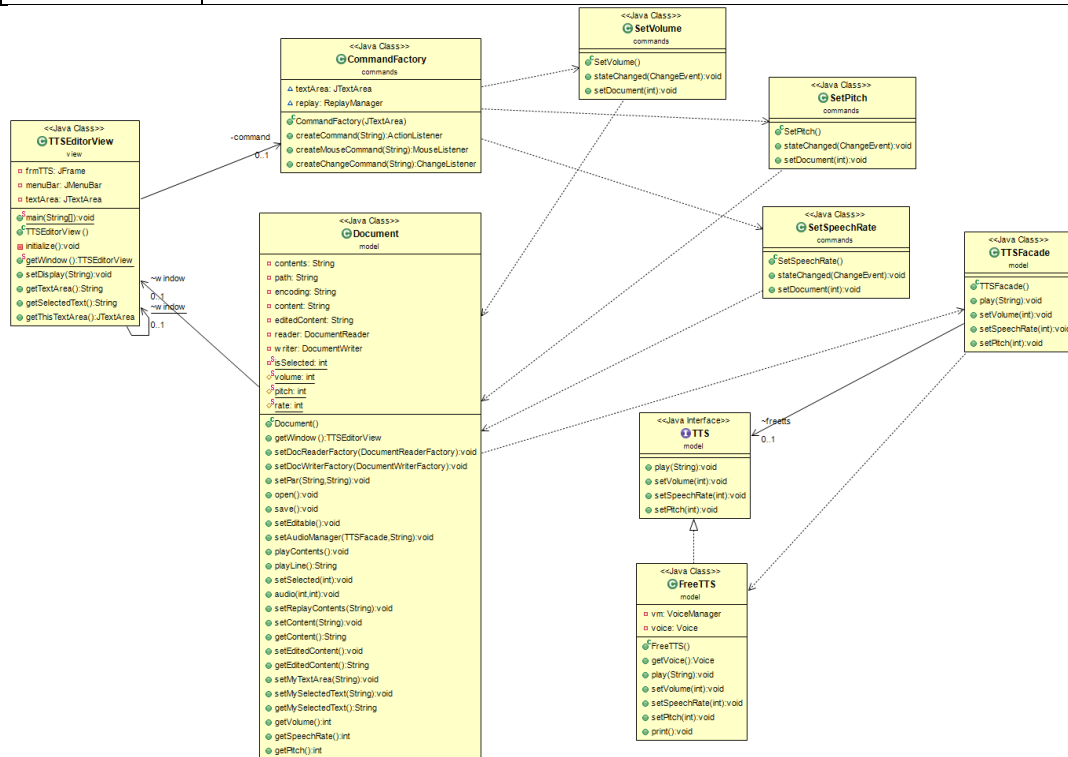
3.7 <Use Case 7>

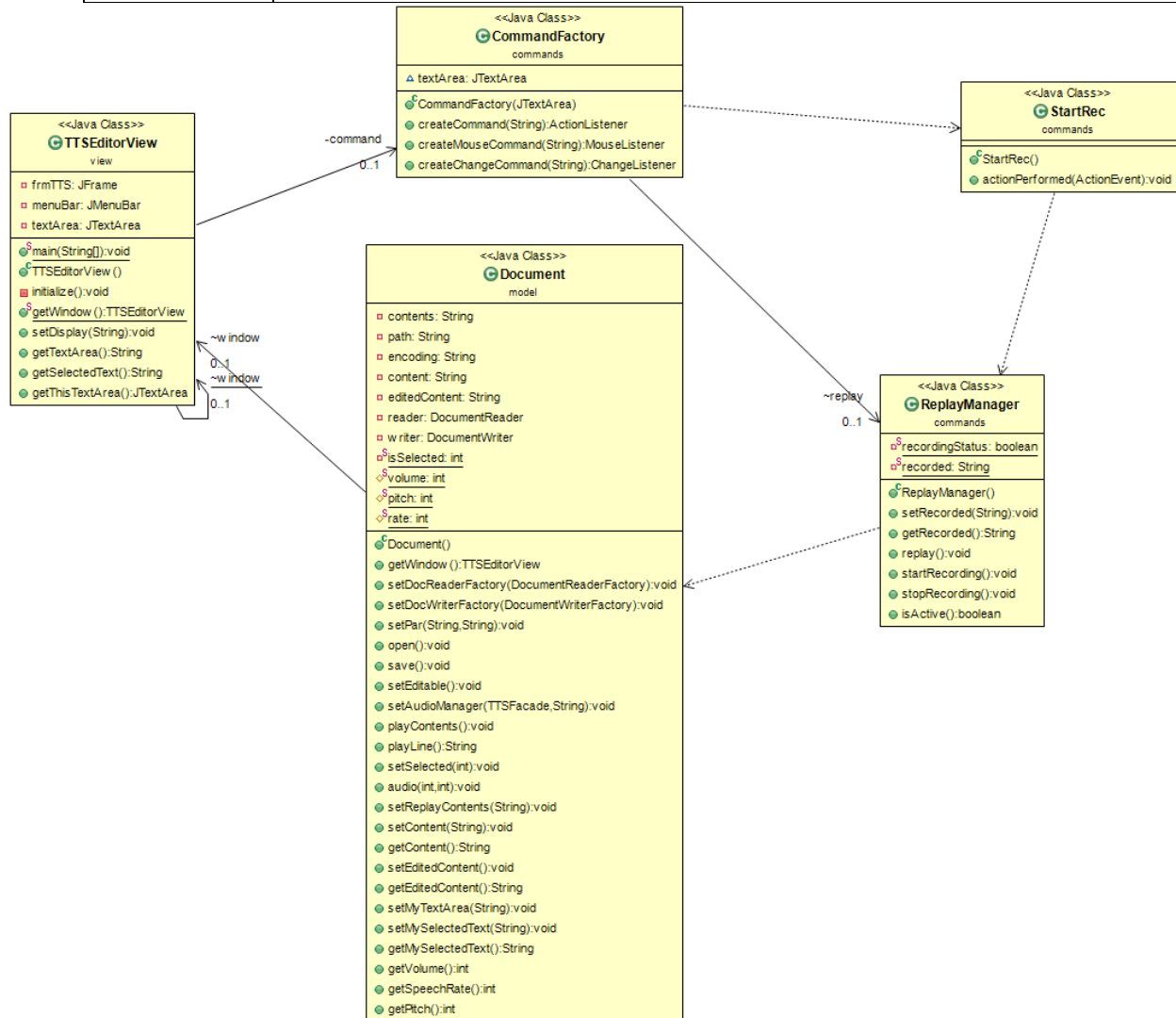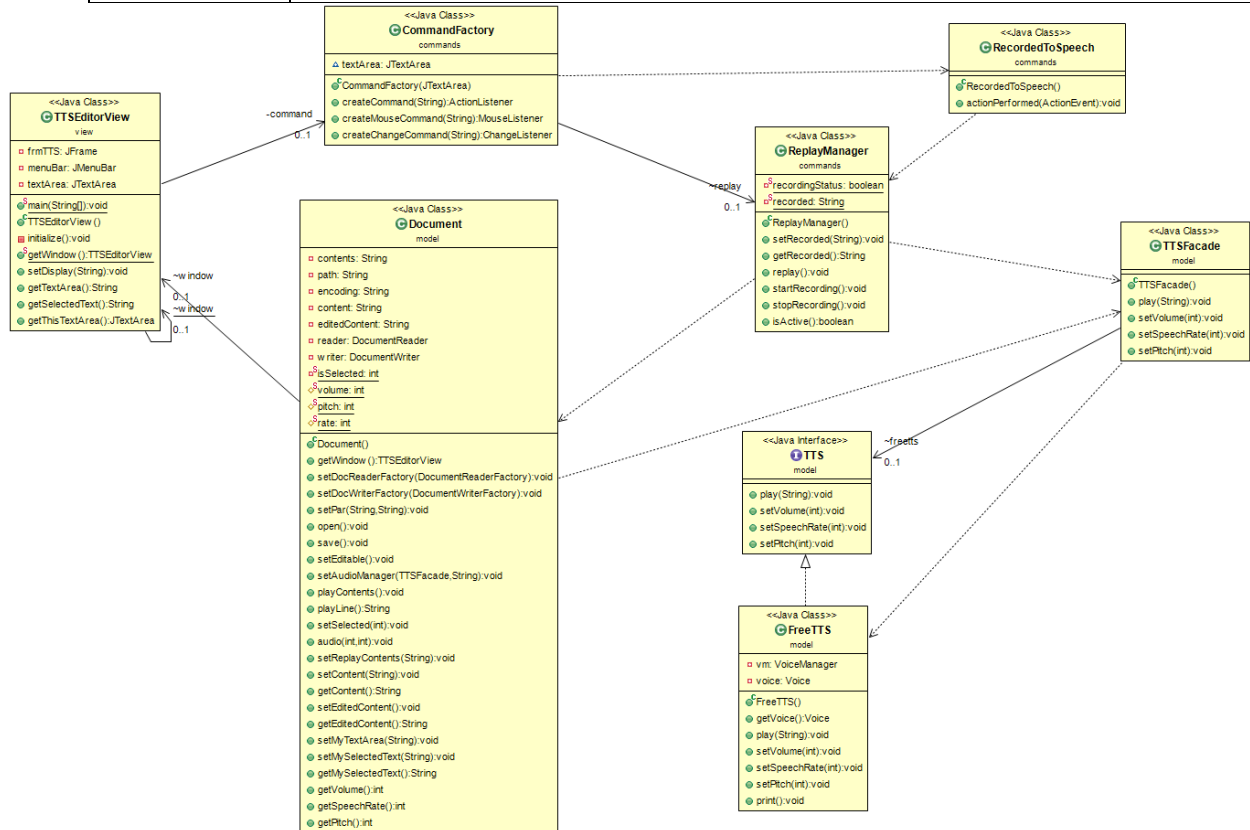| Use case ID	US7
Actors	The User
Pre conditions	The program is running.

Main flow of events	19. The use case starts when the user presses the "Activate Recording" button.
	20. The system now records every text the user chooses to transform into speech.
Post conditions	When the user presses the "Replay Recorded" button he can now listen to all the previously transformed to speech texts.

<<Java Class>>
CommandFactory
commands

△ textArea: JTextArea

- CommandFactory(JTextArea)
- createCommand(String):ActionListener
- createMouseCommand(String):MouseListener
- createChangeCommand(String):ChangeListener

<<Java Class>>
StartRec
commands

- StartRec()
- actionPerformed(ActionEvent):void

<<Java Class>>
TTSEditorView
view

- frmTTS: JFrame
- menuBar: JMenuBar
- textArea: JTextArea

- main(String[]):void
- TTSEditorView ()
- initialize():void
- getWindow ():TTSEditorView
- setDisplay(String):void
- getTextArea():String
- getSelectedText():String
- getThisTextArea():JTextArea

-command
0..1

~window
0..1
~window
0..1

<<Java Class>>
Document
model

- contents: String
- path: String
- encoding: String
- content: String
- editedContent: String
- reader: DocumentReader
- writer: DocumentWriter
- isSelected: int
- volume: int
- pitch: int
- rate: int

- Document()
- getWindow ():TTSEditorView
- setDocReaderFactory(DocumentReaderFactory):void
- setDocWriterFactory(DocumentWriterFactory):void
- setPar(String,String):void
- open():void
- save():void
- setEditable():void
- setAudioManager(TTSFacade,String):void
- playContents():void
- playLine():String
- setSelected(int):void
- audio(int,int):void
- setReplayContents(String):void
- setContent(String):void
- getContent():String
- setEditedContent():void
- getEditedContent():String
- setMyTextArea(String):void
- setMySelectedText(String):void
- getMySelectedText():String
- getVolume():int
- getSpeechRate():int
- getPitch():int

~replay
0..1

<<Java Class>>
ReplayManager
commands

- recordingStatus: boolean
- recorded: String

- ReplayManager()
- setRecorded(String):void
- getRecorded():String
- replay():void
- startRecording():void
- stopRecording():void
- isActive():boolean

3.8 <Use Case 8>

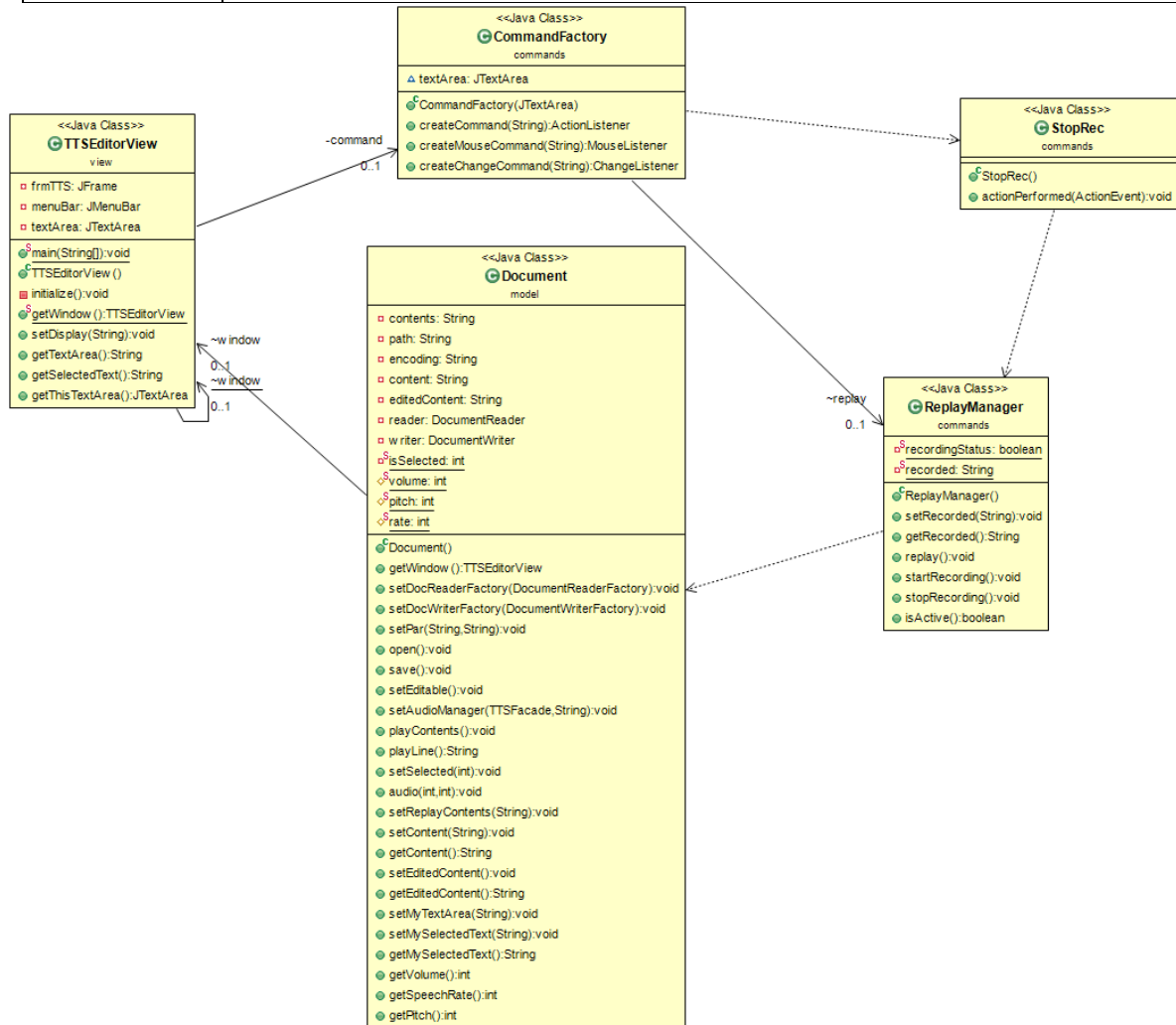| Use case ID	US8
Actors	The User

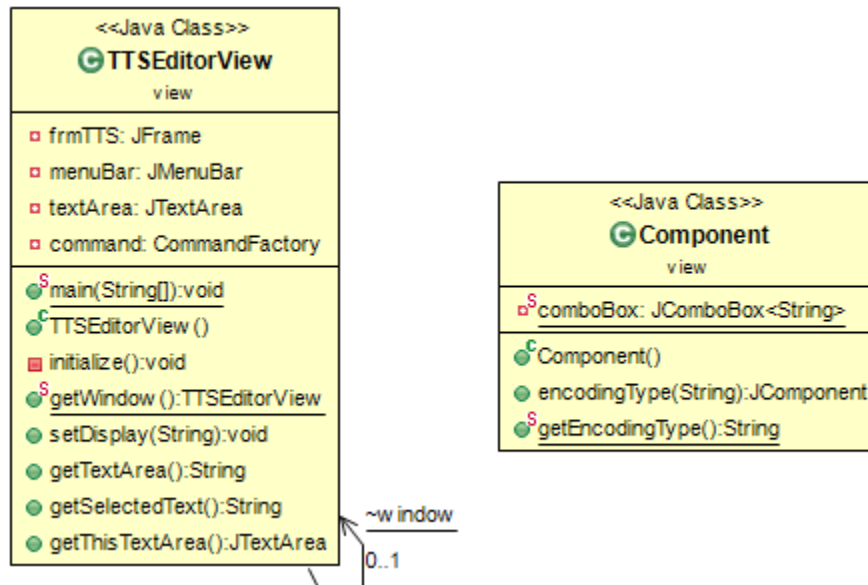| Pre conditions	The program is running.
Main flow of events	21. The use case starts when the user presses the "Replay Recorded" button. 22. The system replays the, so far, recorded text.
Alternative flow 1	If the user has not pressed the "Activate Recording" button or hasn't listened to any texts since pressing the "Activate Recording" button, the system will pop up a "Please activate the recording first" pane.

3.9 <Use Case 9>

Use case ID	US9
Actors	The User
Pre conditions	The program is running .

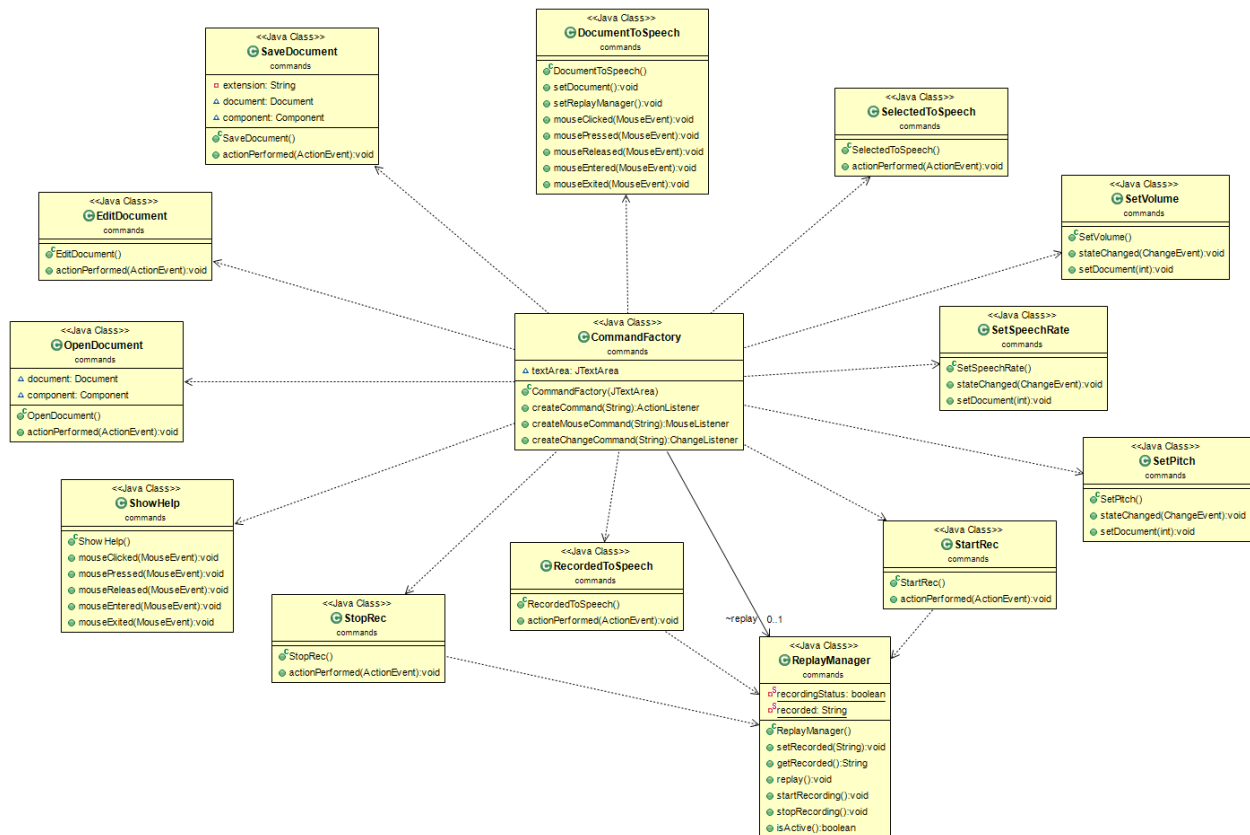| Main flow of events	23. The use case starts when the user presses the "Deactivate Recording" button. 24. The system now empties all the recorded entries.
Post conditions	If the user presses the "Replay Recorded" button the system won't transform anything to speech.

<<Java Class>>
CommandFactory
commands

△ textArea: JTextArea

- CommandFactory(JTextArea)
- createCommand(String):ActionListener
- createMouseCommand(String):MouseListener
- createChangeCommand(String):ChangeListener

<<Java Class>>
StopRec
commands

- StopRec()
- actionPerformed(ActionEvent):void

<<Java Class>>
TTSEditorView
view

- frmTTS: JFrame
- menuBar: JMenuBar
- textArea: JTextArea

- main(String[]):void
- TTSEditorView ()
- initialize():void
- getWindow ():TTSEditorView
- setDisplay(String):void
- getTextArea():String
- getSelectedText():String
- getThisTextArea():JTextArea

-command
0..1

~window
0..1
~window
0..1

<<Java Class>>
Document
model

- contents: String
- path: String
- encoding: String
- content: String
- editedContent: String
- reader: DocumentReader
- writer: DocumentWriter
- isSelected: int
- volume: int
- pitch: int
- rate: int

- Document()
- getWindow ():TTSEditorView
- setDocReaderFactory(DocumentReaderFactory):void
- setDocWriterFactory(DocumentWriterFactory):void
- setPar(String,String):void
- open():void
- save():void
- setEditable():void
- setAudioManager(TTSFacade,String):void
- playContents():void
- playLine():String
- setSelected(int):void
- audio(int,int):void
- setReplayContents(String):void
- setContent(String):void
- getContent():String
- setEditedContent():void
- getEditedContent():String
- setMyTextArea(String):void
- setMySelectedText(String):void
- getMySelectedText():String
- getVolume():int
- getSpeechRate():int
- getPitch():int

~replay
0..1

<<Java Class>>
ReplayManager
commands

- recordingStatus: boolean
- recorded: String

- ReplayManager()
- setRecorded(String):void
- getRecorded():String
- replay():void
- startRecording():void
- stopRecording():void
- isActive():boolean

4   Design

Class Name: TTSEditorView

Responsibilities:	Collaborations:
▪ Sets up the user interface ▪ Calls the CommandFactory to pass through all of the commands.	▪ CommandFactory ▪ Document

Class Name: Component

Responsibilities:	Collaborations:
▪ Creates an encoding/decoding component every time the file chooser opens	▪ OpenDocument ▪ SaveDocument

Class Diagram

<<Java Class>> SaveDocument (commands)
- extension: String
- document: Document
- component: Component
- SaveDocument()
- actionPerformed(ActionEvent):void

<<Java Class>> DocumentToSpeech (commands)
- DocumentToSpeech()
- setDocument():void
- setReplayManager():void
- mouseClicked(MouseEvent):void
- mousePressed(MouseEvent):void
- mouseReleased(MouseEvent):void
- mouseEntered(MouseEvent):void
- mouseExited(MouseEvent):void

<<Java Class>> SelectedToSpeech (commands)
- SelectedToSpeech()
- actionPerformed(ActionEvent):void

<<Java Class>> SetVolume (commands)
- SetVolume()
- stateChanged(ChangeEvent):void
- setDocument(int):void

<<Java Class>> EditDocument (commands)
- EditDocument()
- actionPerformed(ActionEvent):void

<<Java Class>> CommandFactory (commands)
- textArea: JTextArea
- CommandFactory(JTextArea)
- createCommand(String):ActionListener
- createMouseCommand(String):MouseListener
- createChangeCommand(String):ChangeListener

<<Java Class>> SetSpeechRate (commands)
- SetSpeechRate()
- stateChanged(ChangeEvent):void
- setDocument(int):void

<<Java Class>> OpenDocument (commands)
- document: Document
- component: Component
- OpenDocument()
- actionPerformed(ActionEvent):void

<<Java Class>> SetPitch (commands)
- SetPitch()
- stateChanged(ChangeEvent):void
- setDocument(int):void

<<Java Class>> ShowHelp (commands)
- Show Help()
- mouseClicked(MouseEvent):void
- mousePressed(MouseEvent):void
- mouseReleased(MouseEvent):void
- mouseEntered(MouseEvent):void
- mouseExited(MouseEvent):void

<<Java Class>> RecordedToSpeech (commands)
- RecordedToSpeech()
- actionPerformed(ActionEvent):void

<<Java Class>> StartRec (commands)
- StartRec()
- actionPerformed(ActionEvent):void

<<Java Class>> StopRec (commands)
- StopRec()
- actionPerformed(ActionEvent):void

~replay 0..1

<<Java Class>> ReplayManager (commands)
- recordingStatus: boolean
- recorded: String
- ReplayManager()
- setRecorded(String):void
- getRecorded():String
- replay():void
- startRecording():void
- stopRecording():void
- isActive():boolean

Class Name: CommandFactory	
Responsibilities:	**Collaborations:**
▪ Calls a new class for every command the user gives.	▪ All the package classes

Class Name: OpenDocument	
Responsibilities:	**Collaborations:**
▪ Opens a file chooser window to the user. ▪ Calls Document to read the document the user has chosen.	▪ CommandFactory ▪ Component ▪ Document

Class Name: EditDocument	
Responsibilities:	**Collaborations:**
▪ Calls Document to set the text area editable.	▪ Document ▪ CommandFactory

Class Name: SaveDocument	
Responsibilities:	**Collaborations:**
▪ Opens a file chooser window to the user. ▪ Calls Document to save the document the user has chosen.	▪ CommandFactory ▪ Component ▪ Document

Class Name: DocumentToSpeech	
Responsibilities:	**Collaborations:**
▪ Calls Document to play the contents of the text area.	▪ Document ▪ CommandFactory

Class Name: SelectedToSpeech	
Responsibilities:	**Collaborations:**
▪ Calls Document to play the selected text of the text area.	▪ Document ▪ CommandFactory

Class Names: SetVolume, SetSpeechRate, SetPitch	
Responsibilities:	**Collaborations:**
▪ Calls Document to set the volume/speech rate/pitch values the user has chosen.	▪ Document ▪ CommandFactory

Class Name: StartRec	
Responsibilities:	**Collaborations:**
▪ Calls ReplayManager to activate recording.	▪ ReplayManager ▪ CommandFactory

Class Name: RecordedToSpeech	
Responsibilities:	**Collaborations:**
▪ Calls ReplayManager to replay everything that is recorded.	▪ ReplayManager ▪ CommandFactory

Class Name: StopRec	
Responsibilities:	**Collaborations:**
▪ Calls ReplayManager to deactivate recording.	▪ ReplayManager ▪ CommandFactory

Class Name: ReplayManager	
Responsibilities:	**Collaborations:**
▪ Saves the recordings and calls Document to replay them.	▪ Document ▪ StartRec ▪ RecordedToSpeech ▪ StopRec ▪ CommandFactory

Class Name: ShowHelp	
Responsibilities:	**Collaborations:**
▪ Calls HelpReader to open the help.docx document.	▪ HelpReader ▪ CommandFactory
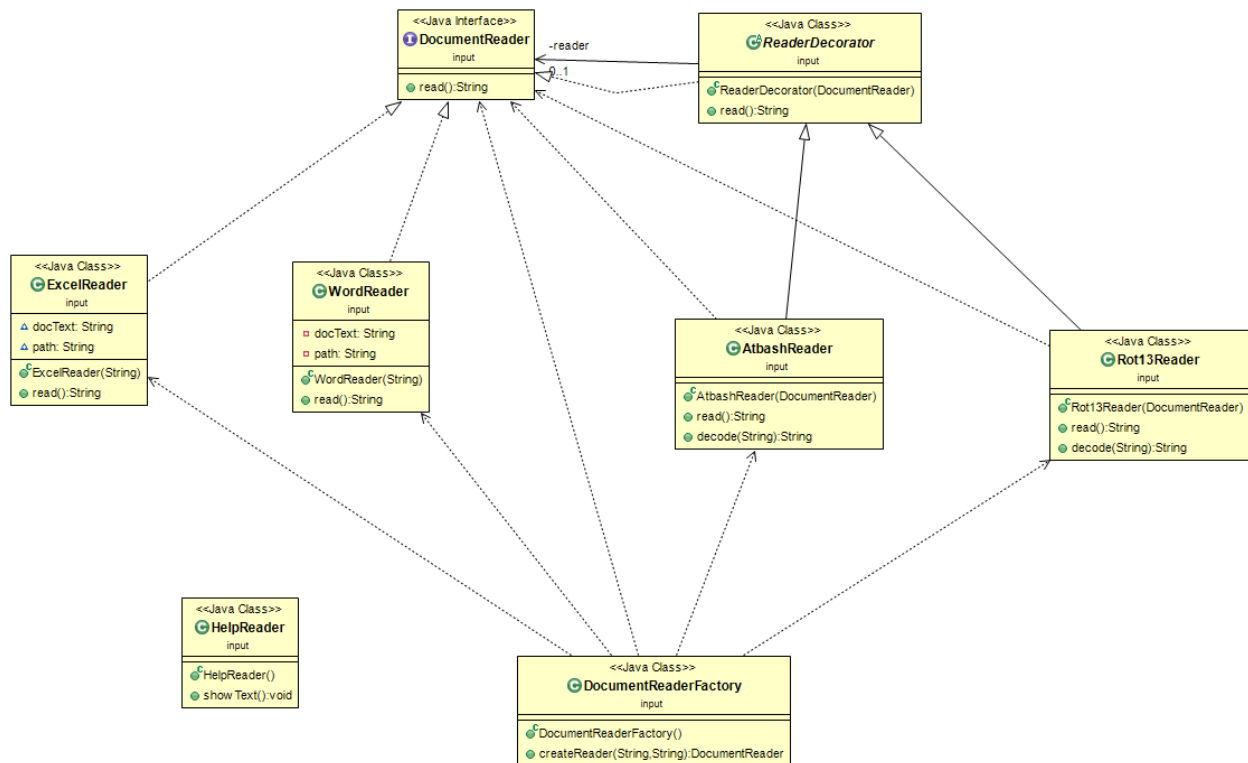
Class Name: DocumentReader	
Responsibilities:	**Collaborations:**
▪ Is the interface that passes the read() method in the "Reader" classes	▪ All the package classes except for HelpReader.

Class Name: DocumentReaderFactory	
Responsibilities:	**Collaborations:**
▪ Sets up the Word/Excel Reader depending on the path and decoding of the file the user has selected.	▪ DocumentReader ▪ WordReader ▪ ExcelReader ▪ AtbashReader ▪ Rot13Reader

Class Name: WordReader	
Responsibilities:	**Collaborations:**
Takes the path of a docx document and returns the contents of it as a string.	- DocumentReader - DocumentReaderFactory

Class Name: ExcelReader	
Responsibilities:	**Collaborations:**
Takes the path of a xlsx document and returns the contents of it as a string.	- DocumentReader - DocumentReaderFactory

Class Name: ReaderDecorator	
Responsibilities:	**Collaborations:**
- Abstract class that implements the DocumentReader interface. - Sets up the decorator pattern.	- DocumentReader - AtbashReader - Rot13Reader

Class Name: AtbashReader	
Responsibilities:	**Collaborations:**
Reads the opened file through Word/Excel Reader and decodes the returned string.	- DocumentReader - DocumentReaderFactory - ReaderDecorator

Class Name: Rot13Reader	
Responsibilities:	**Collaborations:**
Reads the opened file through Word/Excel Reader and decodes the returned string.	- DocumentReader - DocumentReaderFactory - ReaderDecorator - Rot13

Class Name: HelpReader	
Responsibilities: ▪ Sets the help.docx to be opened by Document.	**Collaborations:** ▪ Document
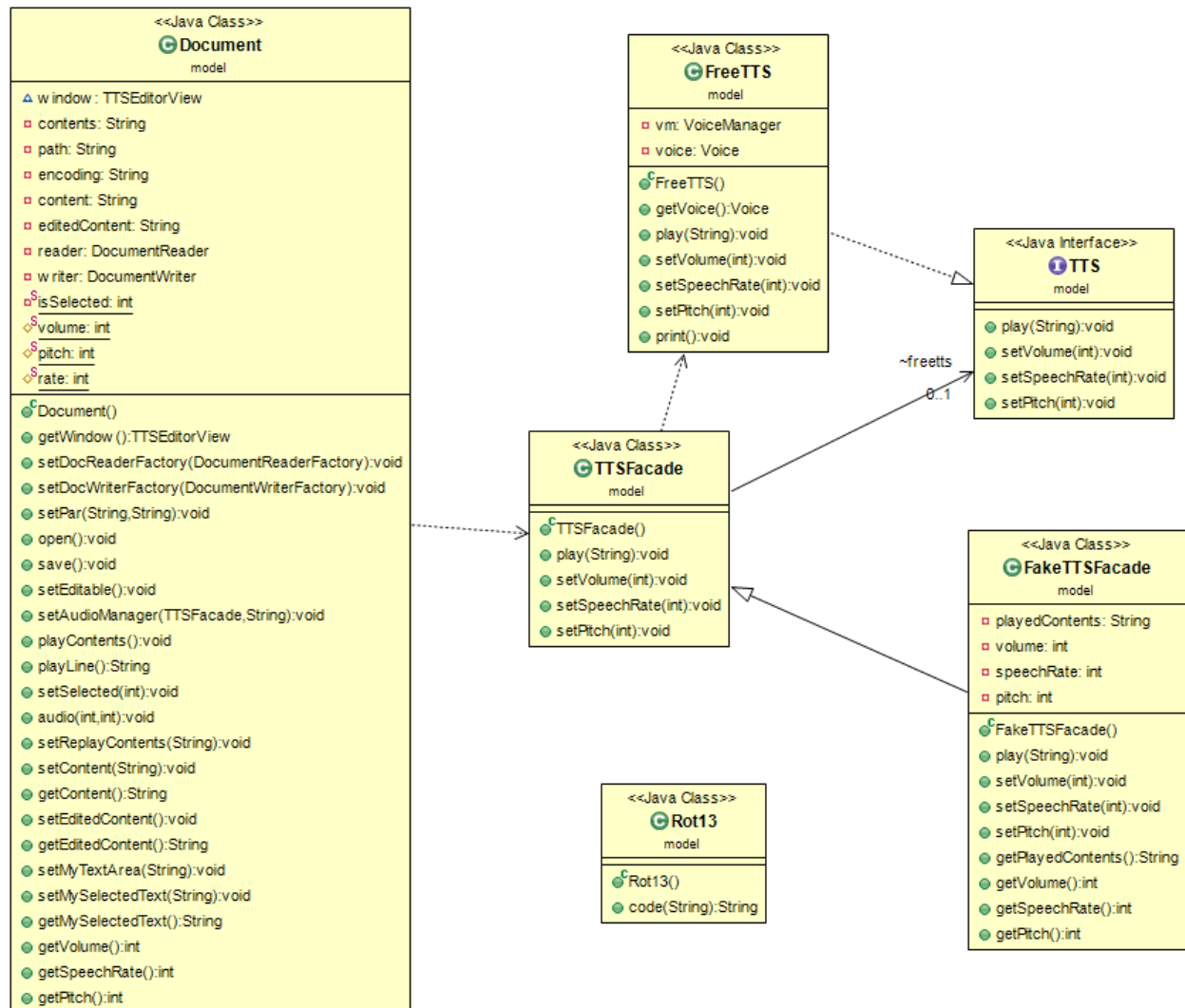
Class Name: Document	
Responsibilities: ▪ Document class sets almost everything up.	**Collaborations:** ▪ TTSFacade

▪ Creates a window instance so we can interact with the text area ▪ Sets up the reader/writer factories to open/save a document. ▪ Sets up the audio of the application. ▪ Associates all the tests with setter and getter methods.	▪ DocumentReader(Writer) ▪ DocumentReader(Writer)Factory ▪ TTSEditorView ▪ ReplayManager ▪ Most of the Commands ▪ All the tests

Class Name: TTS	
Responsibilities:	**Collaborations:**
▪ Is the interface that passes the play(), setVolume(), setSpeechRate(), setPitch() methods in the FreeTTS class.	▪ FreeTTS ▪ TTSFacade

Class Name: FreeTTS	
Responsibilities:	**Collaborations:**
▪ Implements the TTS interface ▪ Sets up the freetts library correctly. ▪ Sets up the play(), setVolume(), setSpeechRate(), setPitch() methods.	▪ TTS ▪ TTSFacade

Class Name: TTSFacade	
Responsibilities:	**Collaborations:**
▪ Implements the facade pattern. ▪ Makes the use of FreeTTS easier and isolates the rest of the application code from the actual FreeTTS Class.	▪ Document ▪ FreeTTS ▪ TTS ▪ FakeTTSFacade

Class Name: FakeTTSFacade	
Responsibilities:	**Collaborations:**
- Is a subclass of the TTSFacade. - Stores the text that is played and returns it. - Used in Tests.	- TTSFacade - DocumentToSpeechTest - SelectedToSpeechTest - ReplayTest

Class Name: Rot13	
Responsibilities:	**Collaborations:**
- Decodes and encodes text in Rot-13 coding. - Called by Rot13Writer, Rot13Reader.	- Rot13Writer - Rot13Reader
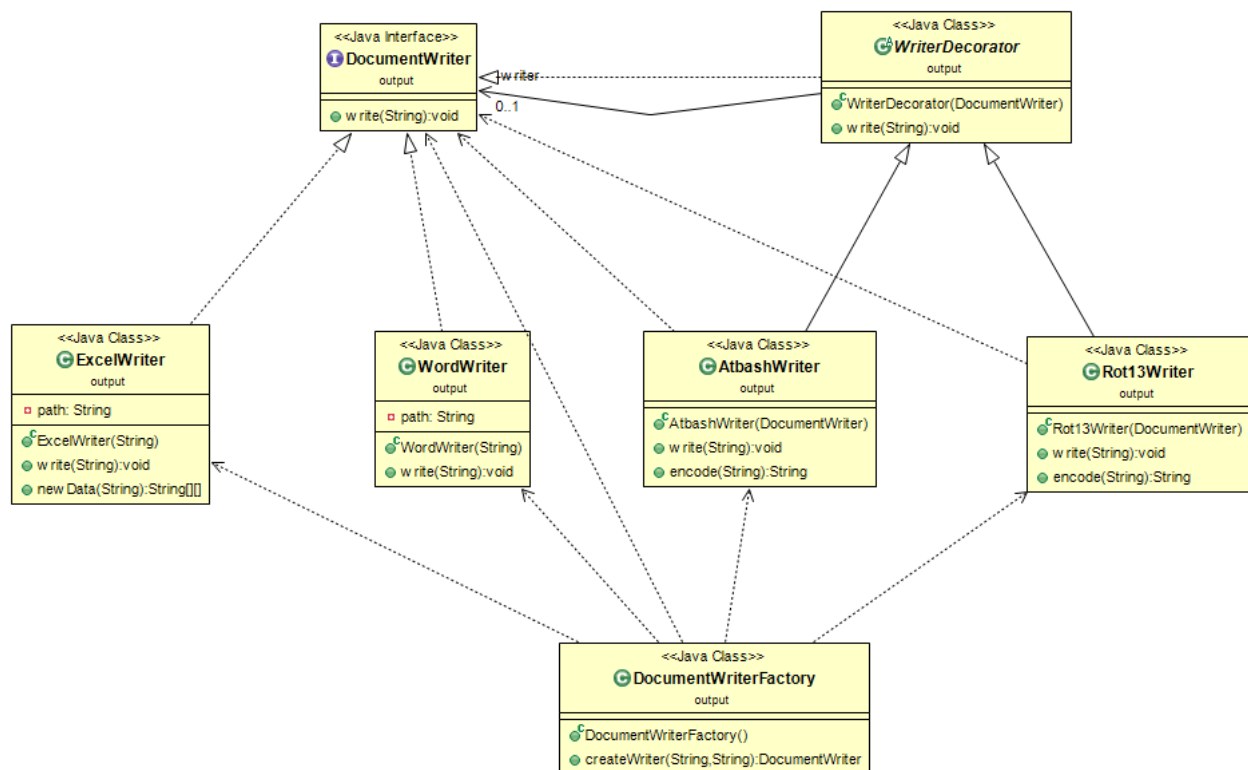
Class Name: DocumentWriter	
Responsibilities:	**Collaborations:**
▪ Is the interface that passes the write() method in the Writer classes	▪ All the package classes.

Class Name: DocumentWriterFactory	
Responsibilities:	**Collaborations:**
▪ Sets up the Word/Excel Writer depending on the path and decoding of the file the user has selected.	▪ DocumentWriter ▪ WordWriter ▪ ExcelWriter ▪ AtbashWriter ▪ Rot13Writer

Class Name: WordWriter	
Responsibilities:	**Collaborations:**
▪ Takes the path of a file, the contents of the text area as a string and saves it a docx document.	▪ DocumentWriter ▪ DocumentWriterFactory

Class Name: ExcelWriter	
Responsibilities:	**Collaborations:**
▪ Takes the path of a file, the contents of the text area as a string and saves it a xlsx document. ▪ Also transforms a string into a 2D array.	▪ DocumentWriter ▪ DocumentWriterFactory

Class Name: WriterDecorator	
Responsibilities:	**Collaborations:**
▪ Abstract class that implements the	▪ DocumentWriter

| DocumentWriter interface. | ■ AtbashWriter |
| ■ Sets up the decorator pattern. | ■ Rot13Writer |

Class Name: AtbashWriter	
Responsibilities:	**Collaborations:**
■ Encodes and writes the given string to a file through Word/Excel Writer.	■ DocumentWriter ■ DocumentWriterFactory ■ WriterDecorator

Class Name: Rot13Writer	
Responsibilities:	**Collaborations:**
■ Encodes and writes the given string to a file through Word/Excel Writer.	■ DocumentWriter ■ DocumentWriterFactory ■ WriterDecorator ■ Rot13