AdvancedText2SpeechEditor

Sprint Report

Tucaliuc Agnes Monalisa, 3346

Joanis Prifti, 3321

Ioanna Charpantidou, 4199

VERSIONS HISTORY

Date	Version	Description	Author
12/04/2021	v.0	Create packets and a diagram of the project	**Two girls and a half man**
22/04/2021	v.1	Execution of DocumentReader, ExcelReader, WordReader, OtherFileReader in packet input	**Two girls and a half man**
27/04/2021	v.2	Execution of DocumentDecoder, ReaderAtBashDecorator, ReaderRot13Decorator in packet input	**Two girls and a half man**
7/05/2021	v.3	Execution of DocumentReaderFactory in packet input	**Two girls and a half man**
8/05/2021	v.4	In packet model we implemented Document and TTSFacade classes	**Two girls and a half man**
11/05/2021	v.5	In packet view we started creating the gui	**Two girls and a half man**
19/05/2021	v.6	In packet commands we started implementing action listeners for the buttons	**Two girls and a half man**
24/05/2021	v.7	In packet commands we started implementing voice and voice manager	**Two girls and a half man**
25/05/2021	v.8	Last modifications to achieve scalability	**Two girls and a half man**
27/05/2021		REPORT	**Two girls and a half man**

1 Introduction

This document provides information concerning the sprints of the project.

1.1 Document Structure

The rest of this document is structured as follows. Section 2 describes out Scrum team and specifies the 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	**Two girls and a half man(TM)**
Scrum Master	**Two girls and a half man(TM)**
Development Team	**Two girls and a half man(TM)**

2.2 Sprints

Sprint No	Begin Date	End Date	Number of weeks	User stories
1	**22/04**	**7/05**	**2**	**US1: Open Document**
2	**7/05**	**24/05**	**2**	**US2: Save Document**
3	**19/05**	**25/05**	**1**	**US3: Play Contents**
4	**19/05**	**25/05**	**1**	**US4: Play Line**
5	**19/05**	**25/05**	**1**	**US5: Setting Volume**
6	**19/05**	**25/05**	**1**	**US6: Replay Contents**

3 Use Cases

3.1 Open Document

Use case ID	1
Actors	User
Pre conditions	To have the document saved to the PC, in a valid format..
Main flow of events	1. The use case starts when the user presses the “Open” button at the upper panel of the GUI. 2. A window opens. 3. The user finds the file at the PC’s path. 4. The user presses the “Open” button (at the new window) .
Alternative flow 1	At step 4 the user can double click the file to open it.
Alternative flow 2	At step 4 the user can select the file and press enter to open it.
Post conditions	The text area is overwritten with the file content the user chose.

3.2 Save Document

Use case ID	2
Actors	User
Pre conditions	-
Main flow of events	1. The use case starts when the user presses the “Save” button at the upper panel of the GUI. 2. A window opens.

	3. The user chooses a name for the file (the name must contain the .format type extension. 4. The user chooses the path the file will be saved to. 5. The user presses the “Save” button (at the new window) .
Alternative flow 1	Step 4 can happen at step 3 and step 4 can happen at step 3.
Alternative flow 2	-
Post conditions	The file is saved at the chosen path.

3.3 Clear

Use case ID	3
Actors	User
Pre conditions	The text area contains at least one character either by directly writing to the text area or by opening an existing file from the user’s system.
Main flow of events	1. The use case starts when the user presses the “Clear” button at the upper panel of the GUI.
Post Conditions	The text area clears up.

3.4 Encryption/Decryption

Use case ID	4
Actors	User
Pre conditions	The text area contains at least one character either by directly writing to the text area or by opening an existing file from the user’s system.

Main flow of events	1. The use case starts when the user presses one of the “At_Bash”, “Rot_13” or “None” button at the right panel of the GUI.
Alternative flow 1	1. The user can alternate between the encryption types. 2. If the same button is pressed again decryption occurs.
Post conditions	The file is encoded of decoded with the chosen encryption/decryption (or stays the same if “None” button is pressed).

3.5 Lock

Use case ID	5
Actors	User
Pre conditions	The text area contains at least one character either by directly writing to the text area or by opening an existing file from the user’s system.
Main flow of events	1. The use case starts when the user presses the “Lock” button at the right panel of the GUI.
Post Conditions	The text area locks up and user can not erase any character from the text area, clear the text area or write to the text area.

3.6 Play

Use case ID	6
Actors	User
Pre conditions	1. The text area contains at least one character either by directly writing to the text area or by opening an existing file from the user’s system. 2. The user must have his system volume at 100% for better hearing of the speech output.
Main flow of events	1. The use case starts when the user adjusts the volume by moving the bar from the slider, which is positioned at the right bottom of the GUI panel. 2. The user presses the “Play” button at the bottom panel of the GUI.

Post conditions	The contents of the text area are being spoken out loud by the app.

3.7 Play Line

Use case ID	7
Actors	User
Pre conditions	1. The text area contains at least one character either by directly writing to the text area or by opening an existing file from the user's system. 2. The user must have his system volume at 100% for better hearing of the speech output.
Main flow of events	1. The use case starts when the user adjusts the volume by moving the bar from the slider, which is positioned at the right bottom of the GUI panel. 2. The user enters a valid number in the text field at the bottom panel of the GUI. 3. The user presses the "Play Line" button.
Post conditions	The contents of the text area at the line the user chose are being spoken out loud.

3.8 Replay

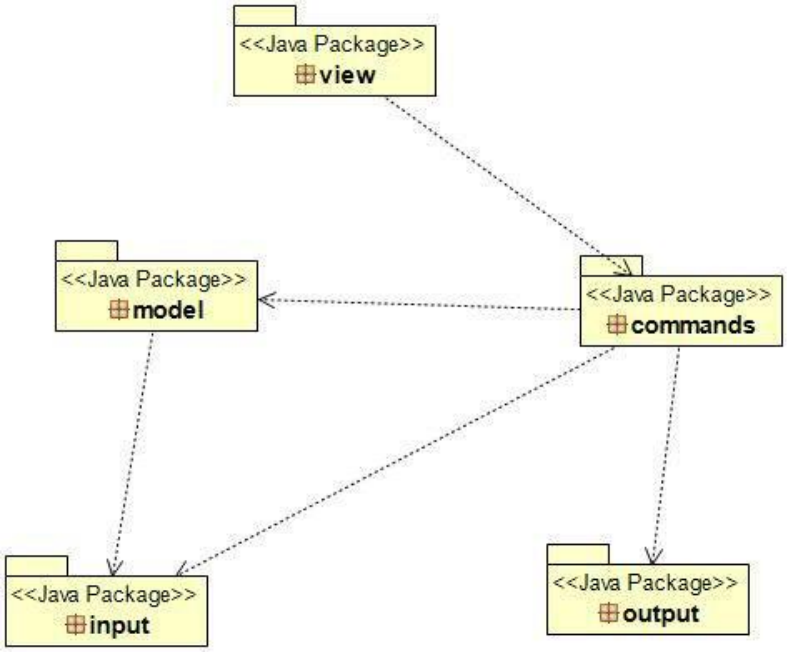
Use case ID	8
Actors	User
Pre conditions	1. The user must have played (or played a line) at least once. 2. The user must have his system volume at 100% for better hearing of the speech output.
Main flow of events	1. The use case starts when the user presses the "Replay" button.

Post conditions	All the played contents from the beginning are being spoken out loud by the system.

4 Design

4.1 Architecture

For this project we created 5 packets.



Package view: In this package we created the GUI class that contains the graphical user interface and AdvancedText2SpeechApp class that has the main method.

Package output: In this package we created classes that read and create the files for the project.

Package model: In this package we created Document and TTSFacade classes that help combining files with freetts library.

Package input: In this package we created the classes responsible for reading any kind of file type and decoding it in ROT13 or AtBash.

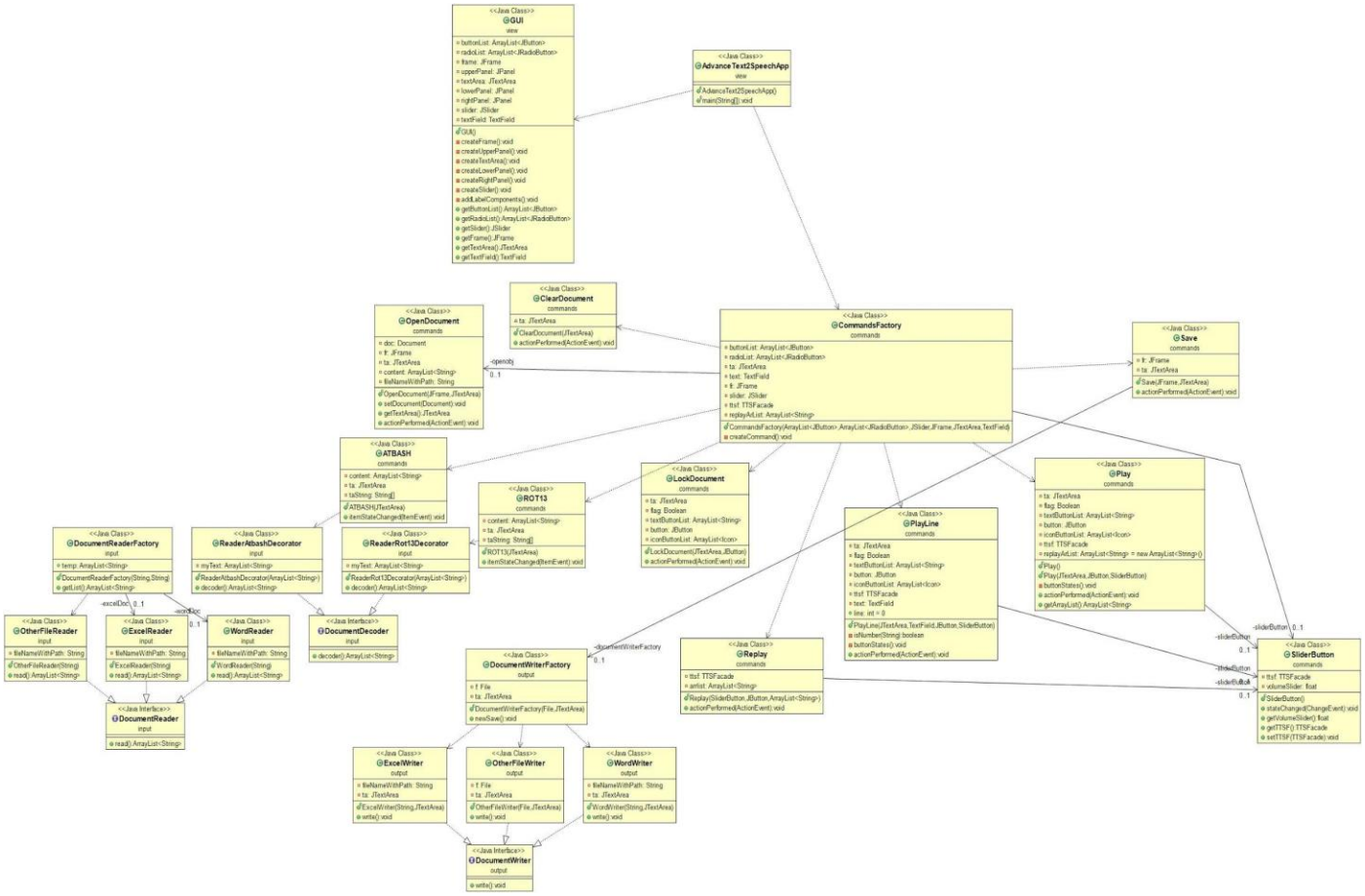
Package commands: In this package we have the CommandsFactory responsible for the buttons' operation.

3.1. 4.2 Patterns

Pattern No	Name	Packet or classes	Description
1	**GoF Command Pattern**	**commands packet**	**Allows us to pass actions as parameters to methods or objects, which subsequently execute those actions.**
2	**GoF Facade pattern**	**TTSFacade class**	**Facade defines a higher-level class that makes a subsystem or library easier to use.**
3	**GoF Strategy pattern**	**Document class, input class, output class**	**Reads any kind of Document type and decoding it in ROT13 or AtBash or reads and creates the Documents for the project.**
4	**GoF Decorator pattern**	**ReaderAtbashDecorator, ReaderRot13Decorator classes**	**Will allow to easily combine a decoding strategy with a particular file opening strategy.**

3.2. 4.3 Design

Below you can see the Class Uml of the project.



Class Name: AdvanceText2SpeechApp	
Responsibilities: ▪ has the main method ▪ calls the GUI class ▪ calls the CommandsFactory class	**Collaborations:** ▪ with the commands packet that contains CommandsFactory

Class Name: GUI	
Responsibilities: ▪ Creates the Frame	**Collaborations:** ▪ it is called from the other classes in the packet

▪ Creates the Panels ▪ Creates the TextArea ▪ Creates the Slider	

Class Name: ATBASH	
Responsibilities: ▪ AtBash itemListener for the radioButton, encrypts and decrypts	**Collaborations:** ▪ with the input package to get `ReaderAtbashDecorator` ▪ `and the GUI class to get TextArea`

Class Name: ClearDocument	
Responsibilities: ▪ ActionListener to clear the TextArea	**Collaborations:** ▪ With the GUI to get the TextArea

Class Name: CommandsFactory	
Responsibilities: ▪ calls every command on the GUI ▪ puts actions on the frame	**Collaborations:** ▪ with every class in the commands packet ▪ with packet model to get TTSFacade class

Class Name: LockDocument

Responsibilities:	Collaborations:
▪ actionListener to lock text area	▪ with the GUI to get text area and the JButton

Class Name: OpenDocument	
Responsibilities:	**Collaborations:**
▪ is responsible to open .docx , .xlsx or other type of file and adding their content in text area ▪ actionListener for the Open JButton	▪ package model to get Document class ▪ with the GUI to get Jframe and JTextArea

Class Name: Play	
Responsibilities:	**Collaborations:**
▪ plays content of the text area ▪ plays selected content in text area ▪ actionListener for the Play JButton	▪ packet model to access TTSFacade class ▪ with the GUI to get text area, JButton ▪ with SliderButton class to ger value of the volume

Class Name: PlayLine	
Responsibilities:	**Collaborations:**
▪ plays selected line that the user inputs in the text field ▪ actionListener for the PlayLine Jbutton ▪ restricts the text field just to get digits.	▪ with package model to get TTSFacade functions ▪ with SliderButton to get volume value ▪ with GUI to get text area, textfield and button

Class Name: Replay	
Responsibilities: ▪ actionListener for the Replay JButton ▪ replays content that has been played so far	**Collaborations:** ▪ with package model to get TTSFacade functions ▪ with GUI to get button

Class Name: ROT13	
Responsibilities: ▪ itemListener for the ROT13 RadioButton ▪ when selected decodes content in text area to Rot13	**Collaborations:** ▪ with GUI class to get radioList and text area ▪ with input package to get ReaderRot13Decoder

Class Name: Save	
Responsibilities: ▪ actionListener for the Save JButton ▪ saves content of the text area into a file that the user creates or in already existing file	**Collaborations:** ▪ with the GUI to get Jframe and JTextArea ▪ with output package to get access in DocumentWriterFactory class

Class Name: SliderButton	
Responsibilities: ▪ addChangeListener for the slider ▪ gets volume from TTSFacade	**Collaborations:** ▪ with the package model to get access in TTSFacade class

	▪ with any class that plays content from the text area to give volume value

Class Name: DocumentDecoder	
Responsibilities: ▪ interface for the ArrayList<String> decoder()	**Collaborations:** ▪ with ReaderAtBashDecorator class ▪ with ReaderRot13Decorator

Class Name: DocumentReader	
Responsibilities: ▪ interface for the ArrayList<String> read()	**Collaborations:** ▪ with WordReader class ▪ with ExcelReader class ▪ with OtherFileReader class

Class Name: DocumentReaderFactory	
Responsibilities: ▪ checks file type to call the proper class to open the file ▪ gets the the list that has file content	**Collaborations:** ▪ with WordReader class ▪ with ExcelReader class ▪ with OtherFileReader class

Class Name: ExcelReader	
Responsibilities: ▪ opens .xlsx file ▪ puts file's content in a list	**Collaborations:** ▪ with DocumentReader class ▪ with DocumentReaderFactory class

Class Name: WordReader	
Responsibilities: ▪ opens .docx file ▪ puts file's content in a list	**Collaborations:** ▪ with DocumentReader class ▪ with DocumentReaderFactory class

Class Name: OtherFileReader	
Responsibilities: ▪ opens any type of file ▪ puts file's content in a list	**Collaborations:** ▪ with DocumentReaderFactory class ▪ with DocumentReader class

Class Name: ReaderAtBashDecorator	
Responsibilities: ▪ decodes the list with the contents of the file that the user opens with AtBash	**Collaborations:** ▪ with DocumentDecoder class ▪ with packet commands in ATBASH class

Class Name: ReaderRot13	
Responsibilities: ▪ decodes the list with the contents of the file that the user opens with Rot13	**Collaborations:** ▪ with DocumentDecoder class ▪ with packet commands in Rot13 class

Class Name: DocumentWriter

Responsibilities:	**Collaborations:**
▪ interface with write method	▪ with ExcelWriter class ▪ with OtherFileWriter class ▪ with WordWriter class

Class Name: DocumentWriterFactory	
Responsibilities:	**Collaborations:**
▪ checks file type and opens proper file type to save the text area content when the user presses Save button	▪ with WordWriter class ▪ with ExcelWriter class ▪ with OtherFileWriter class

Class Name: ExcelWriter	
Responsibilities:	**Collaborations:**
▪ Saves content in .xlsx file	▪ with DocumentWriter class ▪ with DocumentWriterFactory class

Class Name: OtherFileWriter	
Responsibilities:	**Collaborations:**
▪ Saves content in .docx file	▪ with DocumentWriterFactory class ▪ with DocumentWriter class

Class Name: WordWriter	
Responsibilities:	**Collaborations:**
▪ Saves content in any file type the user chooses	▪ with DocumentWriterFactory class ▪ with DocumentWriter class

Class Name: Document	
Responsibilities: ▪ gets the path of the file that we open or we save ▪ gets content list from the files that the user selects to open ▪ opens files with the DocumentReaderFactory	**Collaborations:** ▪ with input package to access DocumentReaderFactory ▪ with TTSFacade class

Class Name: TTSFacade	
Responsibilities: ▪ gets voice functions for the audio from freetts package	**Collaborations:** ▪ with commands package with every kind of play button and the slider

3.3. 5 Acceptance Tests

US1: Open Document

Description: opens/reads a document.

Method to test: ArrayList<String> read(), which is in the classes: ExcelReader, WordReader, OtherFileReader.

Assertion: Check if the document is properly read (the returned "ArrayList<String> list" is not empty and it has the document's contents) (in ExcelReader/WordReader/OtherFileReader classes).

US2: Save Document

Description: saves a document.

Method to test: void write(), which is in the classes: ExcelWriter, WordWriter, OtherFileWriter

Assertion: Check if the document is properly saved. To do this we check if the following variables are not null and if the file they write to has the text area's contents.

1. XWPFDocument document (for the WordWriter).
2. XSSFWorkbook workbook (for the ExcelWriter).
3. FileWriter fw (for the OtherFileWriter).

US3: Play contents

Description: Plays all the contents of the text area.

Methods to test:

- TTSFacade getTTSF() (in SliderButton class).
- void actionPerformed(ActionEvent eve) (in Play class).

Assertion: in Play class check if:

1. the "TTSFacade ttsf" field is not null.
2. the local variable "String allText" contains all the text from the text area.
3. the local variable "String text" is null or not.

US4: Play Line

Description: Plays the contents of a line of the text area.

Method to test:

- void actionPerformed(ActionEvent eve) (in PlayLine class).
- TTSFacade getTTSF() (in SliderButton class).

Assertion: In PlayLine class check if:

1. the "int line" field contains a valid line NUMBER (number of a line in which there is at least a character).
2. the "textArea[line]" is not null and it contains the text of the line the user entered.
3. the "TTSFacade ttsf" field is not null.

US5: Setting Volume

Description: Sets the volume from the slider.

Method to test: void stateChanged(ChangeEvent e) (in SliderButton class).

Assertion: Check if the “float volumeSlider” field is correctly containing the float from the slider (in SliderButton class).

US6: Replay Contents

Description: Replays every “Play” and “PlayLine” the user has pressed from the time he/she opened the app.

Methods to test:

- void actionPerformed(ActionEvent eve) (in Play AND PlayLine classes).
- void createCommand() (in CommandsFactory class).

Assertion: Check if:

1. the “ArrayList<String> replayArList” field is correctly adding in it the text it is played out loud every time “Play” or “PlayLine” buttons are pressed (in Play/PlayLine classes).
2. the “ArrayList<String> replayArList” field is not null as well as if it contains the correct phrases (in CommandsFactory class).