Translated from Greek to English - www.onlinedoctranslator.com

T e xt 2S pee ch E ditor

Team members: Vassilis Mylonas 2777, Andreas Theofilopoulos 2701, Panos Vrachnos 2655

VERSION HISTORY

Date	Version	Description	Author
6/4/2020-10/4/2020		Implementation has taken place [US-1]	Mylonas Vassilis 2777 Theofilopoulos Andreas 2701 Vrachnos Panos 2655
11/4/2020-13/4/2020	1.0	Implementation has taken place [US-2]	Mylonas Vassilis 2777 Theofilopoulos Andreas 2701 Vrachnos Panos 2655
15/4/2020-18/4/2020		Implementation has taken place [US-3]	Mylonas Vassilis 2777 Theofilopoulos Andreas 2701 Vrachnos Panos 2655
19/4/2020-21/4/2020		Implementation has taken place [US-4]	Mylonas Vassilis 2777 Theofilopoulos Andreas 2701 Vrachnos Panos 2655
22/4/2020-24/4/2020		Implementation has taken place [US-5]	Mylonas Vassilis 2777 Theofilopoulos Andreas 2701 Vrachnos Panos 2655
24/4/2020-27/4/2020		Implementation has taken place [US-6]	Mylonas Vassilis 2777 Theofilopoulos Andreas 2701 Vrachnos Panos 2655
27/4/2020-30/4/2020		Implementation has taken place [US-7]	Mylonas Vassilis 2777 Theofilopoulos Andreas 2701 Vrachnos Panos 2655
30/4/2020-3/5/2020		Implementation has taken place [US-8]	Mylonas Vassilis 2777 Theofilopoulos Andreas 2701 Vrachnos Panos 2655
3/5/2020-6/5/2020		Implementation has taken place [US-	Mylonas Vassilis 2777

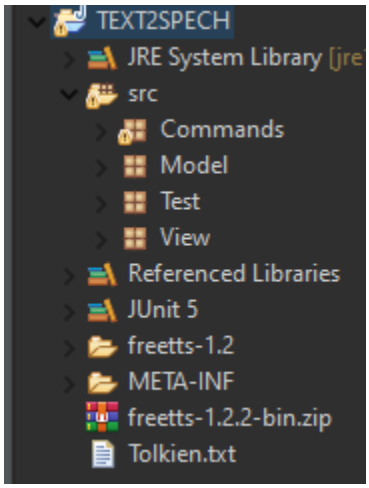
		9]	**Theofilopoulos Andreas 2701** **Vrachnos Panos 2655**
6/5/2020-8/5/2020		**Implementation has taken place [US-10]**	**Mylonas Vassilis 2777** **Theofilopoulos Andreas 2701** **Vrachnos Panos 2655**
8/5/2020-10/5/2020		**Implementation has taken place [US-11]**	**Mylonas Vassilis 2777** **Theofilopoulos Andreas 2701** **Vrachnos Panos 2655**
11/5/2020-12/5/2020		**Implementation has taken place [US-12]**	**Mylonas Vassilis 2777** **Theofilopoulos Andreas 2701** **Vrachnos Panos 2655**
13/5/2020-18/5/2020		**Implementation has taken place [US-13]**	**Mylonas Vassilis 2777** **Theofilopoulos Andreas 2701** **Vrachnos Panos 2655**

LINK for the video in case you didn't see the file named text2speech.mkv https://github.com/mulonas/TEXT2SPEECH

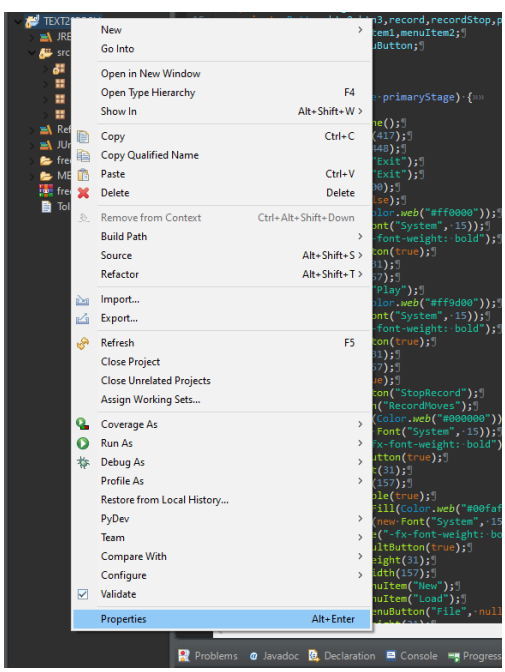
First we will mention that we all worked on almost every piece. We would finish one US and move on to the next. Also let's say we used all the patterns you gave in the hints and the model. In the tests because the design was done somewhat differently from the one you showed in the uml it is not exactly the same some tests could not be made according to what you gave as an idea whenever we made ours.

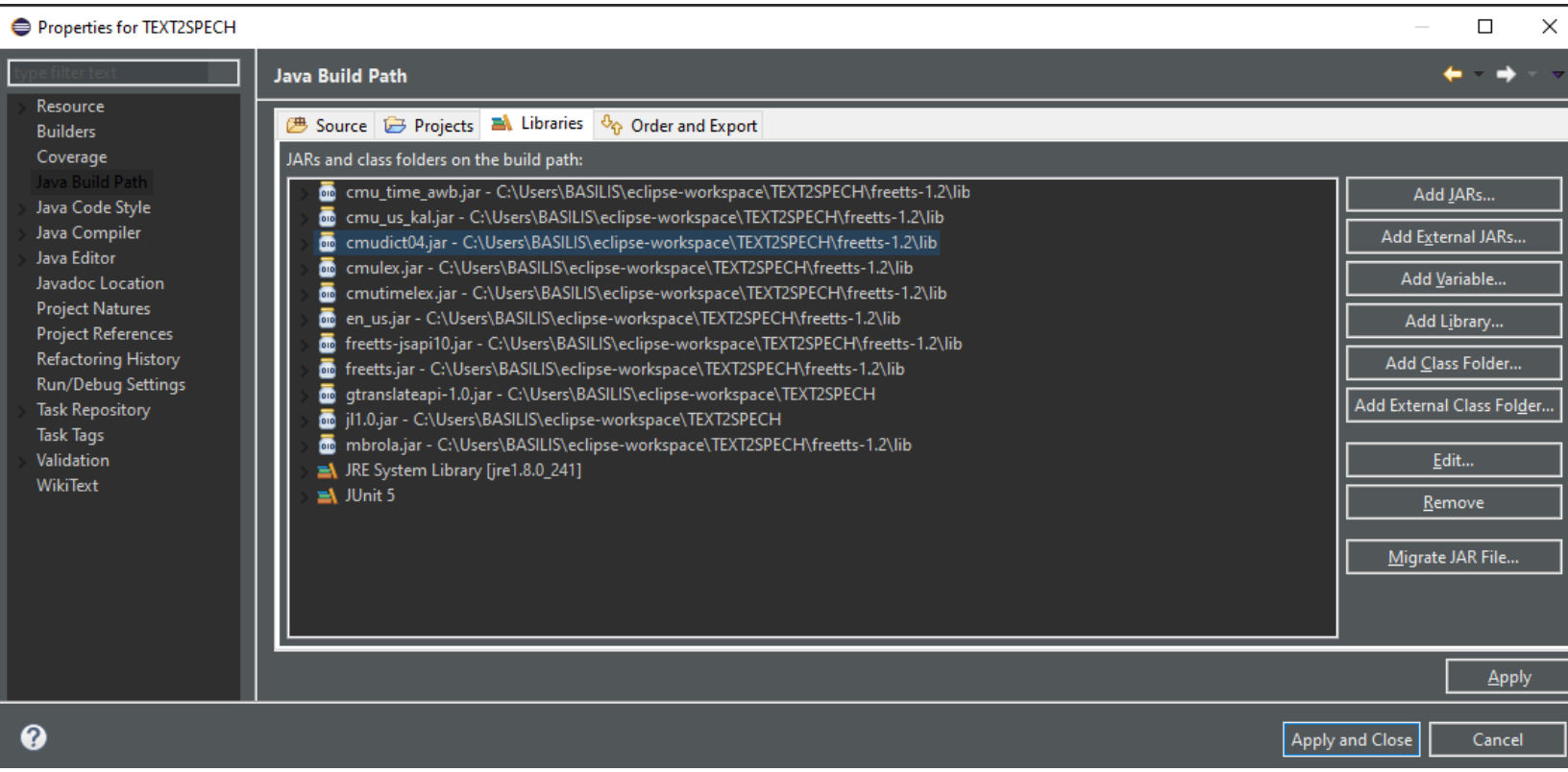
Job Breakdown:

First the image below shows us the MVC (Model View Command/Controller) model



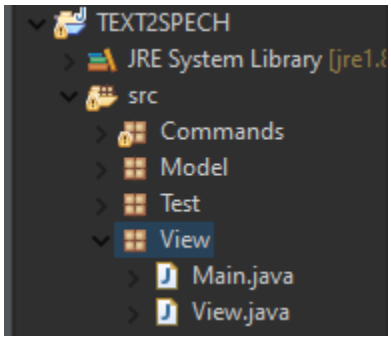
To run the program on your computer you will need to add these libraries:





Here we see that you need to enter the folder freetts-1.2\lib And add all the Jars you will find these our jars are necessary for voice . and the junit library you use for junit test.

First we will analyze the View package.



The View contains the Main where it is the class where we call to start the program and it also contains the Main window we will explain below what we mean. And the View file, which is the most basic one, which is, let's say, the second Main window.

Main.java class analysis

The code is heard:

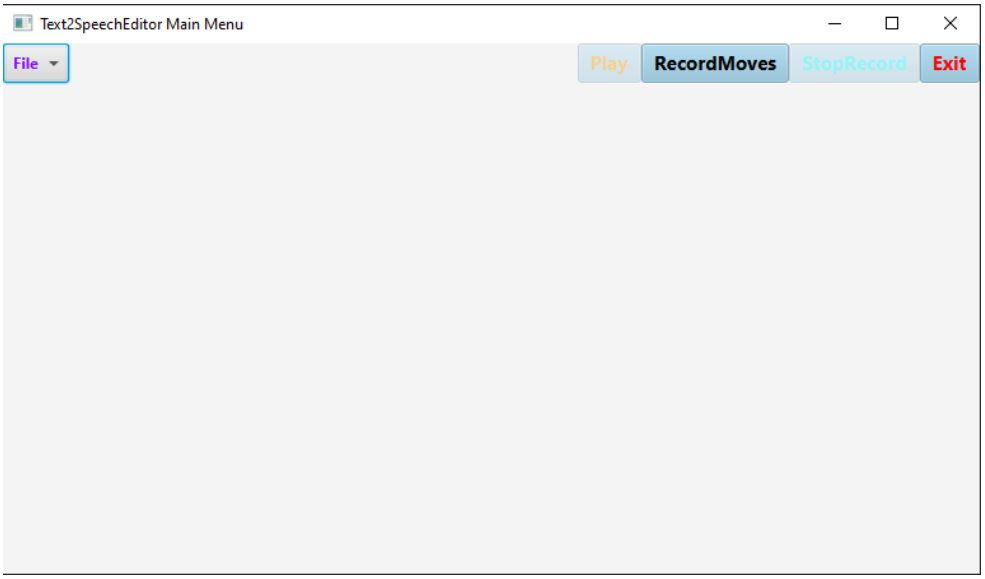
Main.java

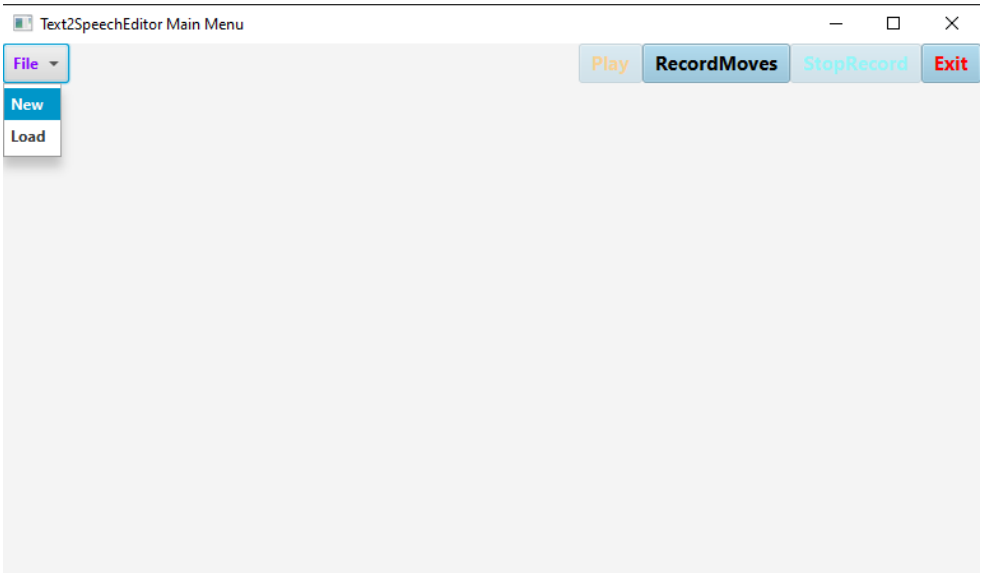
```java
package View;
import Commands.Factory;
public class Main extends Application {
    public static Factory PRESS=new Factory();;
    private GridPane grid1;
    private Button btn2,btn3,record,recordStop,play;
    private MenuItem menuItem1,menuItem2;
    private MenuButton menuButton;
    private  Stage stage;
    private Scene scene;
    @Override
    public void start(Stage primaryStage) {
        
        grid1 = new GridPane();
        grid1.setMinHeight(417);
        grid1.setMinWidth(448);
        btn2 = new Button("Exit");
        btn3 = new Button("Exit");
        btn3.setMinWidth(400);
        btn3.setVisible(false);
        btn2.setTextFill(Color.web("#ff0000"));
        btn2.setFont(new Font("System", 15));
        btn2.setStyle("-fx-font-weight: bold");
        btn2.setDefaultButton(true);
        btn2.setMaxHeight(31);
        btn2.setMaxWidth(157);
        play = new Button("Play");
        play.setTextFill(Color.web("#ff9d00"));
        play.setFont(new Font("System", 15));
        play.setStyle("-fx-font-weight: bold");
        play.setDefaultButton(true);
        play.setMaxHeight(31);
        play.setMaxWidth(157);
        play.setDisable(true);
        recordStop=new Button("StopRecord");
        record = new Button("RecordMoves");
        record.setTextFill(Color.web("#000000"));
        record.setFont(new Font("System", 15));
        record.setStyle("-fx-font-weight: bold");
        record.setDefaultButton(true);
        record.setMaxHeight(31);
        record.setMaxWidth(157);
        recordStop.setDisable(true);
        recordStop.setTextFill(Color.web("#00faff"));
        recordStop.setFont(new Font("System", 15));
        recordStop.setStyle("-fx-font-weight: bold");
        recordStop.setDefaultButton(true);
        recordStop.setMaxHeight(31);
        recordStop.setMaxWidth(157);
        menuItem1 = new MenuItem("New");
        menuItem2 = new MenuItem("Load");
        menuButton = new MenuButton("File", null, menuItem1, menuItem2);
```

```
    menuButton.setMaxHeight(31);
    menuButton.setMaxWidth(157);
    menuButton.setStyle("-fx-font-weight: bold");
    menuButton.setTextFill(Color.web("#8400ff"));
    menuButton.setFont(new Font("System", 15));
    grid1.add(btn2,6,0);
    grid1.add(recordStop,5,0);
    grid1.add(btn3,2, 0);
    grid1.add(record,4,0);
    grid1.add(menuButton, 0,0);
    grid1.add(play, 3, 0);
    stage = new Stage();
      stage.setTitle("Text2SpeechEditor Main Menu");
      stage.show();
      scene=new Scene(grid1);
      stage.setScene(scene);
      stage.centerOnScreen();
      menuItem1.setOnAction(e1 -> {
        PRESS.Factorybuttonsaction("MenuNew",null,null,null,null,null,null,null,-1,e1, null, null, null, null);

        PRESS.callReplayManager("MenuNew", null, null, null, null, null, null, null, -1, (float)-1, (float)-1, (float)-1,e1,null,null,null,null);
    });

    btn2.setOnAction(e2 -> {
        PRESS.Factorybuttonsaction("closeStage",stage,null,null,null,null,null,null,-1,e2, null, null, null, null);

    });
    menuItem2.setOnAction(e3 -> {
        PRESS.Factorybuttonsaction("MenuLoad",null,null,null,null,null,null,null,-1,e3, null, null, null, null);

    });
    record.setOnAction(e4 -> {
        PRESS.Factorybuttonsaction("True",null,null,null,null,null,null,null,-1,e4, null, null, null, null);
        recordStop.setDisable(false);
        play.setDisable(true);
    });
    recordStop.setOnAction(e5 -> {
        PRESS.Factorybuttonsaction("False",null,null,null,null,null,null,null,-1,e5, null, null, null, null);
        play.setDisable(false);
        recordStop.setDisable(true);
    });
    play.setOnAction(e6 -> {
        PRESS.PlayTheRecord(e6);
    });


  }
  public static Factory factRet() {
    return PRESS;
  }


  public static void main(String[] args) {

    launch(args);
  }

}
```

We will not refer to the design part which consists of Javafx commands for creating buttons label textfield textarea stage. Here let's say that there is a function factRet which returns the factory we have created in the next file which is the view because a new factory should not be created. The factorybuttonsaction function accepts some arguments and is responsible for executing the program's buttons. Here we have the call for the new Load Exit buttons as well as to start or stop recording for the replay function.

In order to have a more correct image when the program starts, the first image you create from this file is the following:





In the View class we have the following code which simply implements the following window and sends the action of each button to the Command, as well as printing some error windows. Below we give the code in miniature if you just want to have an image for the view.

View.java

```java
package View;

import Commands.Factory;

public class View {
    private int choicefrombox,arraylenght,flag=1;
    private String[] array1;
    private TextArea textA;
    private TextArea holdText;
    private TextField textf1,textf2,textf3,textf4,textf5,textPitch,textRate;
    private Factory PRESS;
    private Button btn1,btn2,btn3,btn4,btn5,btn6,btnSET;
    private Stop[] stops;
    private LinearGradient lg1;
    private MenuItem encode1,encode2,encode3;
    private MenuButton encode;
    private GridPane grid1,grid2,grid3,grid4;
    private Label warning,linelabel,l6,l5,l4,l3,l2,l1,SliderLabel,PitschLabel,RateLabel;
    private Button ok ,space,btnselect;
    private Stage WarnStage,SelectStage,ViewWindowStage,SetVoiceStAGE;
    private Scene WARNScene,Selectscene,ViewWindowScene,SetVoiceScene;
    private ComboBox<Integer> cb;
    private Slider slider;
    private Boolean bPitch,bRate;
    public void ViewWindow(String[] array) {
        holdText=new TextArea();
        flag=1;
        PRESS=new,factRet();
        grid1 = new GridPane();
        ViewWindowStage = new Stage();
        grid1.setMinHeight(417);
        grid1.setMinWidth(440);
        btn1 = new Button("Save");
        btn2 = new Button("Exit");
        btn3 = new Button("Speech");
        btn4 = new Button("Edit");
        btn5 = new Button("Reverse Speech");
        btn6 = new Button("VoiceSet");
        stops = new Stop[] { new Stop(0f,Color.web("#00eeff")), new Stop(1.0f,Color.web("#1aff03")),};
        lg1 = new LinearGradient(0.5619047619047619, 0.5190476190476190,0.5476190476190476, 0.5805210095238 1, true, CycleMethod.NO_CYCLE,stops);
        btn1.setTextFill(Color.web("#12ba4a"));
        btn2.setTextFill(Color.web("#ff0000"));
        btn3.setTextFill(Color.web("#4000ff"));
        btn4.setTextFill(lg1);
        btn5.setTextFill(Color.web("#ff00e9"));
        btn6.setTextFill(Color.web("#000000"));
        btn1.setFont(new Font("System", 15));
        btn2.setFont(new Font("System", 15));
        btn3.setFont(new Font("System", 15));
        btn4.setFont(new Font("System", 15));
        btn5.setFont(new Font("System", 15));
```

View.java

```java
        btn1.setStyle("-fx-font-weight: bold");
        btn2.setStyle("-fx-font-weight: bold");
        btn3.setStyle("-fx-font-weight: bold");
        btn4.setStyle("-fx-font-weight: bold");
        btn5.setStyle("-fx-font-weight: bold");
        btn6.setStyle("-fx-font-weight: bold");
        btn1.setDefaultButton(true);
        btn2.setDefaultButton(true);
        btn3.setDefaultButton(true);
        btn4.setDefaultButton(true);
        btn5.setDefaultButton(true);
        btn6.setDefaultButton(true);
        btn1.setMaxHeight(31);
        btn1.setMaxWidth(157);
        btn2.setMaxHeight(31);
        btn2.setMaxWidth(157);
        btn3.setMaxHeight(31);
        btn3.setMaxWidth(157);
        btn4.setMaxHeight(31);
        btn4.setMaxWidth(157);
        btn5.setMaxHeight(31);
        btn5.setMaxWidth(157);
        btn6.setMaxHeight(31);
        btn6.setMaxWidth(157);
        encode1 = new MenuItem("Atbash");
        encode2 = new MenuItem("Rot-13");
        encode3 = new MenuItem("utf-8");
        encode = new MenuButton("Encode",null, encode1, encode2,encode3);
        encode.setMaxHeight(31);
        encode.setMaxWidth(157);
        encode.setStyle("-fx-font-weight: bold");
        encode.setFont(new Font("System", 15));
        encode.setTextFill(Color.web("#ff00a1"));
        encode.setFont(new Font("System", 12));
        encode1.setStyle("-fx-font-weight: bold");
        encode2.setStyle("-fx-font-weight: bold");
        encode3.setStyle("-fx-font-weight: bold");
        l1=new Label("Name Of File ");
        l2=new Label("The Author");
        l3=new Label("The Title ");
        l4=new Label("The date that it was created. ");
        l5=new Label("The date that it was last saved. ");
        l6=new Label("Text2Speech");
        l1.setTextFill(Color.web("#000cff"));
        l2.setTextFill(Color.web("ff0000"));
        l3.setTextFill(Color.web("#8d0762"));
        l4.setTextFill(Color.web("#ae1d1d"));
        l5.setTextFill(Color.web("#008ae59"));
        l6.setTextFill(Color.web("#ff9d00"));
        l1.setFont(new Font("System", 15));
        l2.setFont(new Font("System", 15));
        l3.setFont(new Font("System", 15));
```

View.java

```java
        l4.setFont(new Font("System", 15));
        l5.setFont(new Font("System", 15));
        l6.setFont(new Font("System", 15));
        l1.setStyle("-fx-font-weight: bold");
        l2.setStyle("-fx-font-weight: bold");
        l3.setStyle("-fx-font-weight: bold");
        l4.setStyle("-fx-font-weight: bold");
        l5.setStyle("-fx-font-weight: bold");
        l6.setStyle("-fx-font-weight: bold");
        textf1=new TextField();
        textf2=new TextField();
        textf3=new TextField();
        textf4=new TextField();
        textf5=new TextField();
        textA=new TextArea();
        textf1.setPromptText("Tolkien "Need completed for Save");
        textf2.setPromptText("John Ronald Reuel Tolkien "Need completed for Save");
        textf3.setPromptText("The Lord of the Rings "Need completed for Save");
        textf4.setPromptText("No supplement required");
        textf5.setPromptText("No supplement required");
        textA.setPromptText("He was the author of the classic high fantasy works The Hobbit and The Lord of the Rings. "+"\n"+
            "He served as the Rawlinson and Bosworth Professor of Anglo-Saxon and "+"\n"+
            "Fellow of Pembroke College, Oxford, from 1925 to 1945 and "+"\n"+
            "Merton Professor of English Language and Literature and "+"\n"+
            "Fellow of Merton College, Oxford, from 1945 to 1959. "+"\n"+
            "He was at one time a close friend of C. S. Lewisthey were both members of the informal literary discussion group known as the Inklings. "+"\n"+
            " "Need completed for Save");
        if(!(array==null)) {
            textf1.setText(array[0]);
            textf2.setText(array[1]);
            textf3.setText(array[2]);
            textf4.setText(array[3]);
            textf5.setText(array[4]);
            textA.setText(array[5]);
            textf1.setEditable(false);
            textf2.setEditable(false);
            textf3.setEditable(false);
            textf4.setEditable(false);
            textf5.setEditable(false);
            textA.setEditable(false);
            grid1.add(btn4,0,0);
            ViewWindowStage.setTitle("Text2SpeechEditor Load File");

            PRESS.callReplayManager("MenuLoad",ViewWindowStage, textf1.getText(), textf2.getText(), textf3.getText(), textf4.getText(), textf5.getText(), textA.getText()
        }
        else {
            textf4.setEditable(false);
            textf5.setEditable(false);
            ViewWindowStage.setTitle("Text2SpeechEditor New File");
        }
        grid1.add(encode,1,0);
        grid1.add(l1,0,1);
```

View.java

```java
        grid1.add(l3,0,3);
        grid1.add(l4,0,4);
        grid1.add(l5,0,5);
        grid1.add(l6,0,6);
        grid1.add(textf1,1,1);
        grid1.add(textf2,1,2);
        grid1.add(textf3,1,3);
        grid1.add(textf4,1,4);
        grid1.add(textf5,1,5);
        grid1.add(textA,1,6);
        grid1.add(btn1,1,0);
        grid1.add(btn2,1,8);
        grid1.add(btn3,2,8);
        grid1.add(btn5,1,1);
        grid1.add(btn6,2,1);
        ViewWindowStage.show();
        ViewWindowScene=new Scene(grid1);
        ViewWindowStage.setScene(ViewWindowScene);
        ViewWindowStage.centerOnScreen();

        btn1.setOnAction(e1 -> {
            PRESS.Factorybuttonsaction("MenuSave",ViewWindowStage,textf1.getText(), textf2.getText(), textf3.getText(), textf4.getText(), textf5.getText(), textA.get

            PRESS.callReplayManager("MenuSave", ViewWindowStage, textf1.getText(), textf2.getText(), textf3.getText(), textf4.getText(), textf5.getText(), textA.getT
        });

        btn2.setOnAction(e2 -> {
            PRESS.Factorybuttonsaction("closeStage",ViewWindowStage,null,null,null,null,null,null,-1,e2,null,null,null,null);

            PRESS.callReplayManager("closeStage", ViewWindowStage, null, null, null, null, null, null, -1, (float)-1,(float)-1, (float)-1, e2,null,null,null,null);

        });
        btn3.setOnAction(e3 -> {
            if (textA.getText().isEmpty()) {
                WarningWindow("SpeechButton");
            }
            else {
                Choicewindow(textA,"speechButton","read");
            }
        });
        btn4.setOnAction(e4 -> {
            PRESS.Factorybuttonsaction("Edit",null,null, null, null,null,null, null, -1, -1, e4,textf1,textf2,textf3,textA);

            PRESS.callReplayManager("Edit", null, null, null, null, null, null, null, -1, -1, (float)-1,(float)-1, (float)-1, e4,textf1,textf2,textf3,textA);
        });
        btn5.setOnAction(e3 -> {
            if (textA.getText().isEmpty()) {
                WarningWindow("ReverseSpeechButton");
            }
            else {
                Choicewindow(textA,"ReverseSpeechButton","read");
```

View.java

```java
        btn6.setOnAction(e3 -> {
            grid4 = new GridPane();
            slider = new Slider(0, 1, 0.5);
            slider.setShowTickMarks(true);
            slider.setShowTickLabels(true);
            slider.setMajorTickUnit(0.1f);
            slider.setBlockIncrement(0.1f);
            slider.setMinWidth(200);
            SliderLabel=new Label("Volume");
            PitschLabel=new Label("Sets the baseline pitch");
            RateLabel=new Label("Sets the rate of speech.");
            SliderLabel.setTextFill(Color.web("#00b4ff"));
            PitschLabel.setTextFill(Color.web("#007fff"));
            RateLabel.setTextFill(Color.web("#0b00e7"));
            SliderLabel.setFont(new Font("System", 15));
            PitschLabel.setFont(new Font("System", 15));
            RateLabel.setFont(new Font("System", 15));
            SliderLabel.setStyle("-fx-font-weight: bold");
            PitschLabel.setStyle("-fx-font-weight: bold");
            RateLabel.setStyle("-fx-font-weight: bold");
            textPitch=new TextField();
            textRate=new TextField();
            textPitch.setPromptText("Only numbers integer or Decimal");
            textRate.setPromptText("Only numbers integer or Decimal");
            btnSET=new Button("Set the voice");
            btnSET.setTextFill(Color.web("#ff4c00"));
            btnSET.setFont(new Font("System", 15));
            btnSET.setStyle("-fx-font-weight: bold");
            btnSET.setDefaultButton(true);
            grid4.add(SliderLabel,0,0);
            grid4.add(slider,1,0);
            grid4.add(PitschLabel,0,1);
            grid4.add(textPitch,1,1);
            grid4.add(RateLabel,0,2);
            grid4.add(textRate,1,2);
            grid4.add(btnSET,2,3);
            SetVoiceStAGE = new Stage();
            SetVoiceStAGE.setTitle("Set Voice");
            SetVoiceScene=new Scene(grid4);
            SetVoiceStAGE.setScene(SetVoiceScene);
            SetVoiceStAGE.centerOnScreen();
            SetVoiceStAGE.initModality(Modality.APPLICATION_MODAL);
            SetVoiceStAGE.show();
```

View.java

```java
            btnSET.setOnAction(enc1->{
                if(textPitch.getText().isEmpty()) {
                    bPitch=true;
                }
                else {
                    try
                    {
                        // checking valid float using parseInt() method
                        Float.parseFloat(textPitch.getText());
                        bPitch=true;

                    }
                    catch (NumberFormatException e)
                    {
                        try
                        {
                            // checking valid float using parseInt() method
                            Integer.parseInt(textPitch.getText());
                            bPitch=true;
                        }
                        catch (NumberFormatException ee)
                        {
                            bPitch=false;
                        }
                    }
                }
                if(textRate.getText().isEmpty()) {
                    bRate=true;
                }
                else {
                    try
                    {
                        // checking valid float using parseInt() method
                        Float.parseFloat(textRate.getText());
                        bRate=true;
                    }
                    catch (NumberFormatException e)
                    {
                        try
                        {
                            // checking valid float using parseInt() method
                            Integer.parseInt(textRate.getText());
                            bRate=true;
                        }
                        catch (NumberFormatException ee)
                        {
                            bRate=false;
                        }
                    }
                }
```

View.java

```java
                if(bRate&&bPitch) {
                    if(textRate.getText().isEmpty()&&textPitch.getText().isEmpty()) {
                        PRESS.Factorybuttonsaction((float)slider.getValue(), (float)-1,(float)-1);

                        PRESS.callReplayManager("setVoice", null, null, null, null, null, null, null, -1,(float)slider.getValue(), (float)-1, (float)-1, enc1, null, null, null, null);
                    }
                    else if(!(textRate.getText().isEmpty())&&textPitch.getTextt().isEmpty()) {

                        PRESS.Factorybuttonsaction((float)slider.getValue(), Float.parseFloat(textRate.getTextt()), (float)-1);

                        PRESS.callReplayManager("setVoice", null, null, null, null, null, null, null, -1,(float)slider.getValue(),Float.parseFloat(textRate.getTextt()), (float)-1, enc1, null, null, null, null);
                    }
                    else if(textRate.getTextt().isEmpty()&&!(textPitch.getTextt().isEmpty())) {

                        PRESS.Factorybuttonsaction((float)slider.getValue(),  (float)-1,  Float.parseFloat(textPitch.getTextt()));

                        PRESS.callReplayManager("setVoice", null, null, null, null, null, null, null, -1,(float)slider.getValue(),(float)-1,Float.parseFloat(textPitch.getTextt()), enc1, null, null, null, null);
                    }
                    else {
                        PRESS.Factorybuttonsaction((float)slider.getValue(), Float.parseFloat(textRate.getTextt()),  Float.parseFloat(textPitch.getTextt()));

                        PRESS.callReplayManager("setVoice", null, null, null, null, null, null, null, -1,(float)slider.getValue(),Float.parseFloat(textRate.getTextt()),Float.parseFloat(textPitch.getTextt()), enc1, null, null, null, n
                    }
                    SetVoiceStAGE.close();
                }
                else {
                    WarningWindow("Set the voice Button");
                }

                });

            });
```

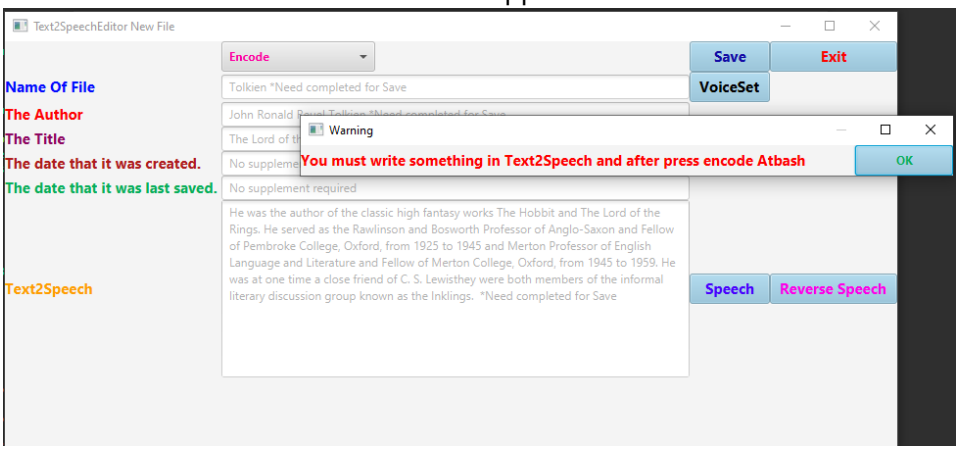
View.java

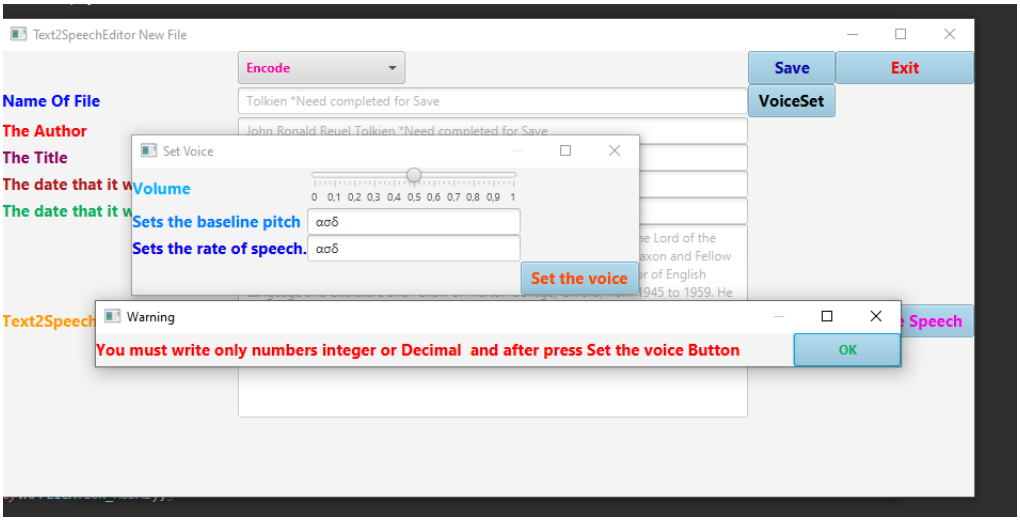
```java
        encode1.setOnAction(enc1->{
            if (flag==1) {
                holdText.setText(textA.getText());
                flag=0;
            }
            if (textA.getText().isEmpty()) {
                WarningWindow("encode Atbash");
                }
            else {
                Choicewindow(textA,"Atbash","Encode");
            }
        });
        encode2.setOnAction(enc1->{
            if (flag==1) {
                holdText.setText(textA.getText());
                flag=0;
            }
            if (textA.getText().isEmpty()) {
                WarningWindow("encode Rot13");
            }
            else {
                Choicewindow(textA,"Rot13","Encode");
                }
        });
        encode3.setOnAction(enc3->{
            if (holdText.getText().isEmpty()) {
                WarningWindow("UTF-8");
            }
            else {
                Choicewindow(textA,"UTF-8","Encode");
                flag=1;
            }

        });

    }

    }
```

In the code you will notice that WarningWindow is called which does nothing but print a window that you did something wrong. The code and window are heard

```java
public void WarningWindow(String ButtoName) {
    grid3 = new GridPane();
    if(ButtoName.equals("Set the voice Button")) {

        warning=new Label("You must write only numbers integer or Decimal  and after press "+ButtoName);
    }
    else {

        warning=new Label("You must write something in Text2Speech and after press "+ButtoName);
    }
    warning.setTextFill(Color.web("#ff0000"));
    warning.setStyle("-fx-font-weight: bold");
    warning.setFont(new Font("System", 15));
    ok=new Button("OK");
    space=new Button();
    space.setVisible(false);
    space.setMinWidth(50);
    ok.setMinHeight(31);
    ok.setMinWidth(100);
    ok.setTextFill(Color.web("#08ae59"));
    ok.setFont(new Font("System", 15));
    ok.setStyle("-fx-font-weight: bold");
    ok.setDefaultButton(true);
    grid3.add(warning, 0, 0);
    grid3.add(space,1,0);
    grid3.add(ok, 2, 0);
    WarnStage = new Stage();
    WarnStage.setTitle("Warning");
    WARNScene=new Scene(grid3);
    WarnStage.setScene(WARNScene);
    WarnStage.centerOnScreen();
    WarnStage.initModality(Modality.APPLICATION_MODAL);
    WarnStage.show();
    ok.setOnAction(ok1->{
        WarnStage.close();
    });
}
```

For example, if you go to encode an empty text, it is wrong whenever you get the corresponding window, or if you want to put letters where you should put numbers, the second window appears.



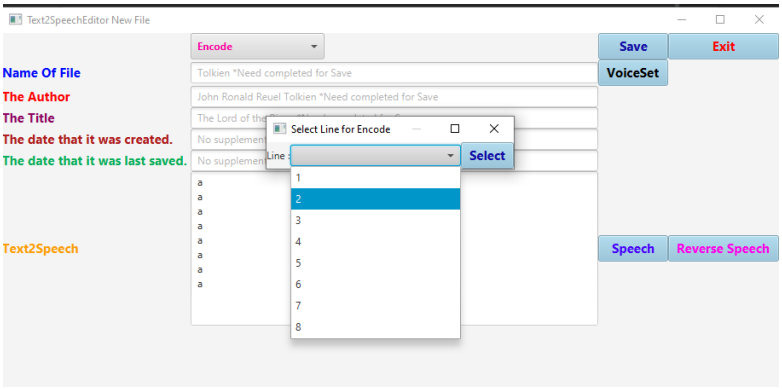


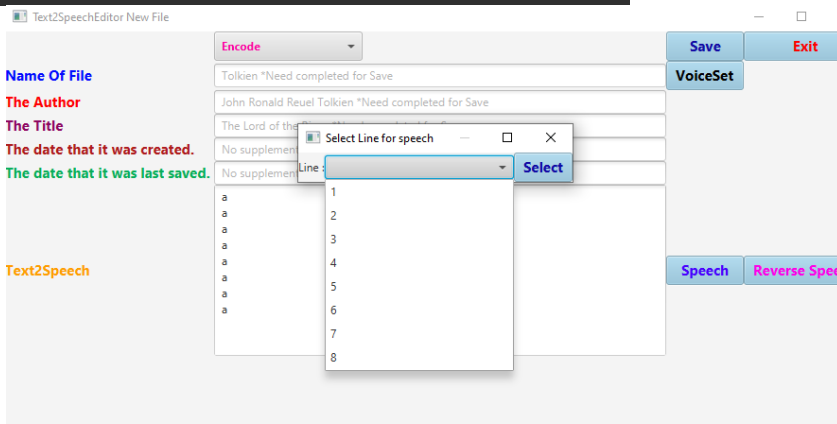
You also use the Choicewindow function, which in turn calls the corresponding function that should be called depending on which button the user pressed.

View.java

```java
public void Choicewindow(TextArea text,String type,String TypeOfButton) {
    choicefrombox=0;
    linelabel=new Label("Line :");
    btnselect = new Button("Select");
    btnselect.setTextFill(Color.web("#120aa4"));
    btnselect.setFont(new Font("System", 15));
    btnselect.setStyle("-fx-font-weight: bold");
    btnselect.setDefaultButton(true);
    btnselect.setMaxHeight(31);
    btnselect.setMaxWidth(157);
    final Tooltip tooltip = new Tooltip();
    tooltip.setText("\n If you do not select something,the whole text will be "+TypeOfButton+" \n");
    cb = new ComboBox<>();
    cb.setTooltip(tooltip);
    cb.setMinWidth(200);
    cb.setMinHeight(20);
    array1=textA.getText().split("\n");
    arraylenght=array1.length;
    for (int i = 1; i <=arraylenght; i++) {
        cb.getItems().add(i);
    }
    grid2 = new GridPane();
    grid2.add(cb, 1, 0);
    grid2.add(linelabel,0,0);
    grid2.add(btnselect,2,0);
    SelectStage = new Stage();
    if( type.equals("UTF-8")||type.equals("Rot13")||type.equals("Atbash")){

        SelectStage.setTitle("Select Line for Encode");
    }else {

        SelectStage.setTitle("Select Line for speech");
    }
    Selectscene=new Scene(grid2);
    SelectStage.setScene(Selectscene);
    SelectStage.centerOnScreen();
    SelectStage.initModality(Modality.APPLICATION_MODAL);
    SelectStage.show();
```

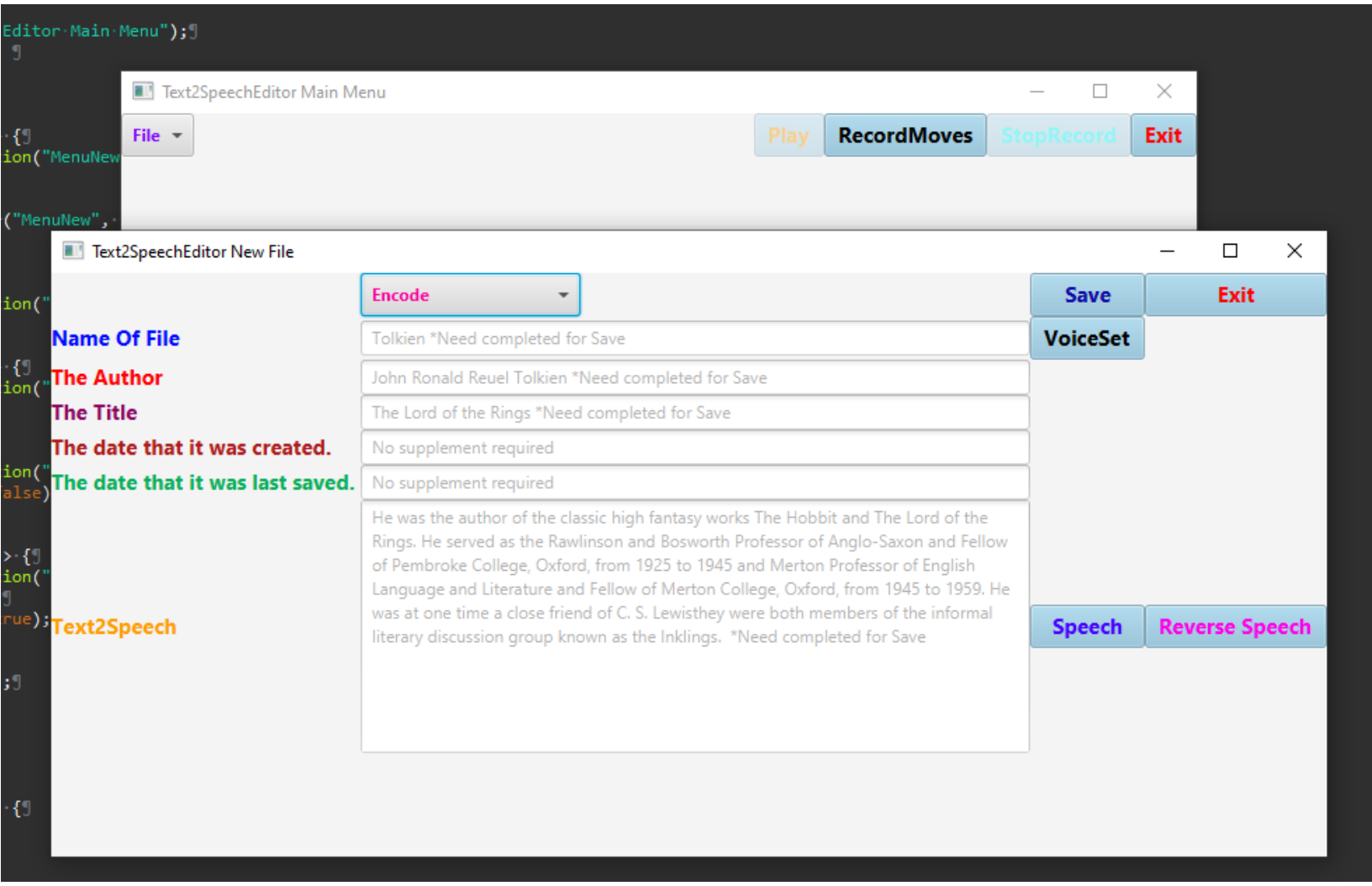
```java
    btnselect.setOnAction(e4 -> {

        if (cb.getValue()!=null) {

            choicefrombox= cb.getValue();
        }
        SelectStage.close();
        if( type.equals("UTF-8")){
            textA.setText(PRESS.Factorybuttonsaction("UTF-8",holdText.getText(),choicefrombox,e4));
            PRESS.callReplayManager("UTF", null, null, null, null, null, null, holdText.getText(), choicefrombox, (float)-1,(float)-1, (float)-1, e4, null, null, null, null);
        }else if( type.equals("Rot13")){
            textA.setText(PRESS.Factorybuttonsaction("Rot13",textA.getText(),choicefrombox,e4));
            PRESS.callReplayManager("Rot13", null, null, null, null, null, null, textA.getText(), choicefrombox, (float)-1,(float)-1, (float)-1, e4, null, null, null, null);
        }
        else if( type.equals("Atbash")){
            textA.setText(PRESS.Factorybuttonsaction("Atbash",textA.getText(),choicefrombox,e4));

            PRESS.callReplayManager("Atbash", null, null, null, null, null, null, textA.getText(), choicefrombox, (float)-1,(float)-1, (float)-1, e4, null, null, null, null);
        }else if( type.equals("ReverseSpeechButton")){
            PRESS.Factorybuttonsaction("ReverseSpeechButton",null,null,null,null,null,null,textA.getText(),choicefrombox, e4,null,null,null,null);

            PRESS.callReplayManager("ReverseSpeechButton", null, null, null, null, null, null, textA.getText(), choicefrombox, (float)-1,(float)-1, (float)-1, e4, null, null, null, null);
        }
        else if( type.equals("speechButton")){
            PRESS.Factorybuttonsaction("speechButton",null,null,null,null,null,null,textA.getText(),choicefrombox, e4,null,null,null,null);
            PRESS.callReplayManager("speechButton", null, null, null, null, null, null, textA.getText(), choicefrombox, (float)-1,(float)-1, (float)-1, e4, null, null, null, null);
        }
    });
}
```

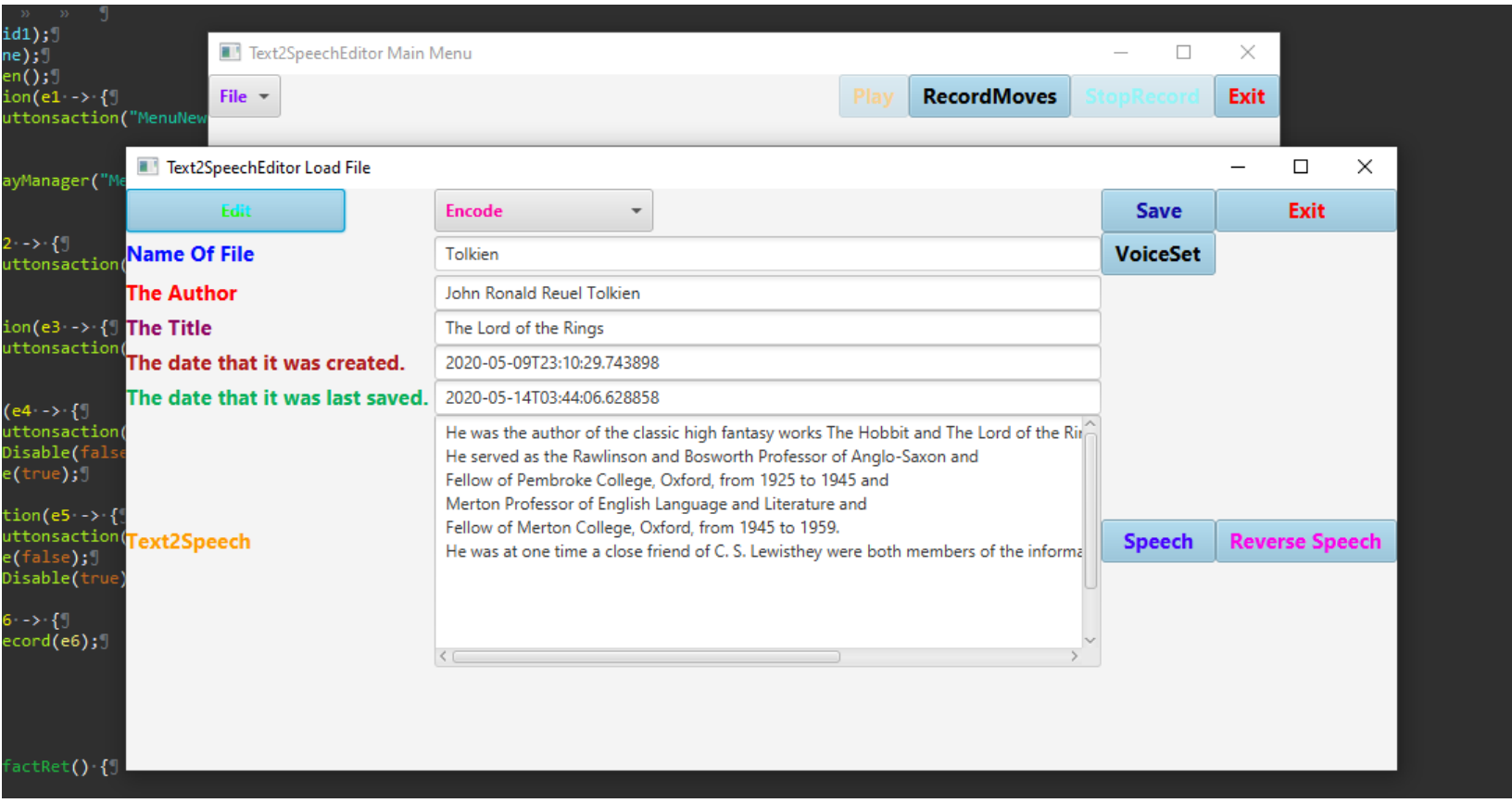




It displays the result from pressing new and Load and is responsible for calling the factory in turn to execute the different functions.

Following are 2 screenshots, the first refers to what the view prints when new is pressed and the second image what will appear when Load is pressed.





Note that we selected the Tolkien.txt file to achieve this effect.

We don't go into detail about how we made what is shown because we think it's more important to analyze how we made the Patterns for the function of the buttons than design wise. In general, the structure where the windows were implemented is that we took a grid and put buttons on it in specific positions and manipulated their effects, nothing remarkable.

Now we will start to analyze how each pattern was made with what logic and how the whole application works. Everything will be analyzed in the Model package.

We used the command Pattern to implement the New Load Edit Save buttons.

But since you wanted everything to have a factory, we took and made a general factory which takes us to the appropriate functions, first we will show how the command pattern was implemented, then how we added it to its factory and then how this factor was connected to the general factory that is for everything actions And is the function that joins view with Model.

First we have an Interface execute:

commandsExecute.java

```java
package Model;

import javafx.scene.control.TextArea;

public interface commandsExecute {

    public void execute(String[] array, Stage stage, TextField textf1, TextField textf2, TextField textf3, TextArea textA);

    

}

```

This Interface is implemented in the files Exit Load CreateNewFile and Save As you can see below

Exit.java | EditFile.java | LoadFile.java | CreateNewFile.java

```java
package Model;

import javafx.scene.control.TextArea;

public class Exit implements commandsExecute{

    private Filebuttons file;
    public Exit(Filebuttons newfile) {
        
        file=newfile;
    }
    @Override
    public void execute(String[] array,Stage stage,TextField textf1,TextField textf2,TextField textf3,TextArea textA) {
        file.Exit(stage);
    }
}
```

Exit.java | EditFile.java | LoadFile.java | CreateNewFile.java

```java
package Model;

import javafx.scene.control.TextArea;

public class EditFile implements commandsExecute{

    private Filebuttons file;
    public EditFile(Filebuttons newfile) {
        file=newfile;
    }
    @Override
    public void execute(String[] array,Stage stage,TextField textf1,TextField textf2,TextField textf3,TextArea textA) {
        file.Edit(textf1,textf2,textf3,textA);
    }

}

```

Exit.java EditFile.java LoadFile.java CreateNewFile.java

```java
package Model;

import javafx.scene.control.TextArea;

public class LoadFile implements commandsExecute{

    private Filebuttons file;
    public LoadFile(Filebuttons newfile) {

        file=newfile;
    }
    @Override
    public void execute(String[] array,Stage stage,TextField textf1,TextField textf2,TextField textf3,TextArea textA) {
        file.MLoad();
    }
}
```

Exit.java EditFile.java LoadFile.java CreateNewFile.java

```java
package Model;

import javafx.scene.control.TextArea;

public class CreateNewFile implements commandsExecute{

    private Filebuttons file;
    public CreateNewFile(Filebuttons newfile) {
        file=newfile;
    }
    @Override
    public void execute(String[] array,Stage stage,TextField textf1,TextField textf2,TextField textf3,TextArea textA) {
        file.Mnew();
    }
}
```

As we can see, each one calls its own function with the arguments it needs, if necessary we will explain the Filebuttons function below

We have made a controller for these executes

ControllerOfExecute.java

```java
package Model;

import javafx.scene.control.TextArea;

public class ControllerOfExecute {
    private commandsExecute command;
    public ControllerOfExecute(commandsExecute newcommand) {
        command=newcommand;
    }
    public void executecommand(String[] array,Stage stage,TextField textf1,TextField textf2,TextField textf3,TextArea textA) {
        command.execute(array,stage,textf1,textf2,textf3,textA);
    }
}
```

We have made an interface for all the functions called by edit, exit, Load, CreateNewFile

AllFileButtons.java

```java

package Model;

import javafx.scene.control.TextArea;

public interface AllFileButtons {
    public void Mnew();
    public void MLoad();
    public void MSave(String [] array);
    public void Exit(Stage stage);
    public void Edit(TextField textf1,TextField textf2,TextField textf3,TextArea textA) ;
}
```

We implement the interface in the FileButtons file

Filebuttons.java

```java

package Model;


import View.View;
public class Filebuttons implements AllFileButtons{

    private int savetag;
    private String[] loadarray=new String[5];
    private SaveModel savefile = new SaveModel();
    private LoadModel loadfile=new LoadModel();
    public View view= new View();
    final FileChooser chosefile= new FileChooser();
    public void Mnew() {

        view.ViewWindow(null);

    }
    public void MLoad() {

        loadarray=loadfile.LoadFile(chosefile.showOpenDialog(null).getPath());
        if(loadarray==null) {
            System.out.print("Load unsuccessful");
        }
        else {
            System.out.print("Load successful");
            view.ViewWindow(loadarray);

        }

    }
    public void MSave(String[] array) {
        savetag=savefile.savefile(array);

        if (savetag==1) {
            System.out.print("Save successful");
        }
        else {
            System.out.print("Save unsuccessful");
        }
```

```java
    }
    public void Edit(TextField textf1,TextField textf2,TextField textf3,TextArea textA) {
        if (textf1!=null) {
            textf1.setEditable(true);
        }
        if (textf2!=null) {
            textf2.setEditable(true);
        }
        if (textf3!=null) {
            textf3.setEditable(true);
        }
        if (textA!=null) {
            textA.setEditable(true);
        }
    }
    public void Exit(Stage stage) {
        stage.close();
    }

}
```

Now let's see what these are called. Initially, when the user presses one of these 4 buttons, it will take us to the Factory. Where we said you can find it in the command package

Factory.java

```java
package Commands;
import Model.Atbash;

public class Factory {
    private Filebuttons file=new Filebuttons();
    private CreateNewFile CreateNewFile=new CreateNewFile(file);
    private LoadFile load=new LoadFile(file);
    private SaveFile Save=new SaveFile(file);
    private Exit exit=new Exit(file);
    private EditFile edit=new EditFile(file);
    private ControllerOfExecute controller;
    private FactoryCommand FactoryCommands= new FactoryCommand();
    private SpeechAdapter voicetext=new SpeechAdapter();
    private Encode typeencode;
    private ReplayManager replayer;
    private Boolean Active=false;
    public void Factorybuttonsaction(String typeofbutton,Stage stage,String textf1,String textf2,String textf3,String textf4,String textf5,String textA,int line,ActionEvent event,TextField txt1,TextField txt2,TextField tx
        if (typeofbutton.equals("MenuNew")) {
            controller=new ControllerOfExecute(CreateNewFile);
            FactoryCommands.MenuNew(controller);
        }
        else if (typeofbutton.equals("MenuLoad")) {
            controller=new ControllerOfExecute(load);
            FactoryCommands.MenuLoad(controller);
        }
        else if (typeofbutton.equals("closeStage")) {
            controller=new ControllerOfExecute(exit);
            FactoryCommands.closeStage(controller,stage);
        }
        else if (typeofbutton.equals("MenuSave")) {
            controller=new ControllerOfExecute(Save);
            FactoryCommands.MenuSave(controller,stage,textf1,textf2,textf3,textf4,textf5,textA);
        }
        else if(typeofbutton.equals("Edit")){
            controller=new ControllerOfExecute(edit);
            FactoryCommands.Edit(controller,txt1, txt2, txt3, txtA);

        }
        else if(typeofbutton.equals("ReverseSpeechButton")) {
            voicetext.ReverseSpeech(textA,line);
        }
        else if(typeofbutton.equals("speechButton")) {
            voicetext.speech(textA,line);
        }
        else if(typeofbutton.equals("True")) {
            Active=true;
            replayer=new ReplayManager();
        }
        else if(typeofbutton.equals("False")) {
            Active=false;
        }
    }
```

When a button is pressed Along with the necessary data we obviously also send a string to check which button was pressed. In this class we define everything that needs to be defined so that later when a button is pressed I can go to the corresponding if and call execute which you will find on the server.

Here we see one more class that you call is factoryCommands. This is the factory only for the 4 buttons, typically we made an even bigger factory that is for all the buttons because you mentioned it gradually in the hints (To avoid wasting factories and we have a factory for each pattern). Whenever the factory in the command pattern works as a factory of all the buttons in the program but also as a view mediator with model .

So below we have FactoryCommand

FactoryCommand.java

```java
package Model;
import javafx.fxml.FXML;
public class FactoryCommand {
    @FXML private javafx.scene.control.Button  Exit;
    private int number=1;
    private String []ArrayForSave=new String[6];
    public void MenuNew(ControllerOfExecute controller)  {
        controller.executecommand(null,null,null,null,null,null);
    }
    public void MenuLoad(ControllerOfExecute controller) {
        controller.executecommand(null,null,null,null,null,null);
    }
    public void MenuSave(ControllerOfExecute controller,Stage stage,String textf1,String textf2,String textf3,String textf4,String textf5,String textA) {
        stage.close();
        ArrayForSave[0]=textf1;
        ArrayForSave[1]=textf2;
        ArrayForSave[2]=textf3;
        ArrayForSave[3]=textf4;
        ArrayForSave[4]=textf5;
        ArrayForSave[5]=textA;
        controller.executecommand(ArrayForSave,null,null,null,null,null);
        number=number+1;
    }
    public void Edit(ControllerOfExecute controller,TextField textf1,TextField textf2,TextField textf3,TextArea textA) {
        controller.executecommand(null,null,textf1,textf2,textf3,textA);
    }
    public void closeStage(ControllerOfExecute controller,Stage stage) {
        controller.executecommand(null,stage,null,null,null,null);
    }
}
```

Here, in essence, it accepts a controller and calls execute with the arguments if they are needed or does some processing such as keeping all the text elements in an array.

Now we will analyze the Speech and Reversespeech and Setvoice buttons (you added this, we were not in your hints, we thought it was more appropriate to include it together)

Again, when one of these is pressed, we go to the Factory.

Factory.java

```java
package Commands;
import Model.Atbash;

public class Factory {
    private Filebuttons file=new Filebuttons();
    private CreateNewFile CreateNewFile=new CreateNewFile(file);
    private LoadFile load=new LoadFile(file);
    private SaveFile Save=new SaveFile(file);
    private Exit exit=new Exit(file);
    private EditFile edit=new EditFile(file);
    private ControllerOfExecute controller;
    private FactoryCommand FactoryCommands= new FactoryCommand();
    private SpeechAdapter voicetext=new SpeechAdapter();
    private Encode typeencode;
    private ReplayManager replayer;
    private Boolean Active=false;
    public void Factorybuttonsaction(String typeofbutton,Stage stage,String textf1,String textf2,String textf3,String textf4,String textf5,String textA,int line,ActionEvent event,TextField txt1,TextField txt2,TextField tx
        if (typeofbutton.equals("MenuNew")) {
            controller=new ControllerOfExecute(CreateNewFile);
            FactoryCommands.MenuNew(controller);
        }
        else if (typeofbutton.equals("MenuLoad")) {
            controller=new ControllerOfExecute(load);
            FactoryCommands.MenuLoad(controller);
        }
        else if (typeofbutton.equals("closeStage")) {
            controller=new ControllerOfExecute(exit);
            FactoryCommands.closeStage(controller,stage);
        }
        else if (typeofbutton.equals("MenuSave")) {
            controller=new ControllerOfExecute(Save);
            FactoryCommands.MenuSave(controller,stage,textf1,textf2,textf3,textf4,textf5,textA);
        }
        else if(typeofbutton.equals("Edit")){
            controller=new ControllerOfExecute(edit);
            FactoryCommands.Edit(controller,txt1, txt2, txt3, txtA);

        }
        else if(typeofbutton.equals("ReverseSpeechButton")) {
            voicetext.ReverseSpeech(textA,line);

        }
        else if(typeofbutton.equals("speechButton")) {
            voicetext.speech(textA,line);
        }
        else if(typeofbutton.equals("True")) {
            Active=true;
            replayer=new ReplayManager();
        }
        else if(typeofbutton.equals("False")) {
            Active=false;
        }
    }

```

Originally the Pattern Adapter was used here.

First we have an Interface textToSpeechInterface

textToSpeechInterface.java

```java
package Model;

import com.sun.speech.freetts.Voice;


public interface textToSpeechInterface {
    public Boolean ReverseSpeechButton(Voice voice,String text,int i) ;
    public Boolean speechButton(Voice voice,String text,int i) ;
    public Voice Volume(Voice voice,float volume) ;
    public Voice  speechRate (Voice voice,float wordpersech);
    public Voice  pitch (Voice voice,float hrz);
}

```

Whichever we do implements in the Speech and ReverseSpeech files

Speech.java

```java
package Model;

import com.sun.speech.freetts.Voice;
public class Speech implements textToSpeechInterface
{
    private String[] array1;
    @Override
    public Boolean ReverseSpeechButton(Voice voice,String text,int i) {
        //do nothing
        return false;
    }
    @Override
    public Boolean speechButton(Voice voice,String text,int i) {
        if (i<0) {
            return false;
        }
    ....if (i==0) {
    ....    voice.speak(text);
    ....}
    ....else {
    ....    array1=text.split("\n");
    ....    voice.speak(array1[i-1]);
    ....}
    ....return true;
    }
    @Override
    public Voice Volume(Voice voice,float volume) {
        // TODO Auto-generated method stub
        //do nothing
        return voice;
    }
    @Override
    public Voice speechRate(Voice voice,float wordpersech) {
        // TODO Auto-generated method stub
        //do nothing
        return voice;
    }
    @Override
    public Voice pitch(Voice voice,float hrz) {
        // TODO Auto-generated method stub
        //do nothing
        return voice;
    }
```

ReverseSpeech.java

```java
package Model;
import com.sun.speech.freetts.Voice;
public class ReverseSpeech implements textToSpeechInterface
{
    private String[] array1,input,input1;
    private String voiceString;
    @Override
    public Boolean ReverseSpeechButton(Voice voice ,String text,int i) {
        if (i<0) {
            return false;
        }
        voiceString="";
        if (i==0) {
            input = text.split(" ");
        }
        else {
            array1=text.split("\n");
            input =array1[i-1].split(" ");
        }
        input1=new String[input.length];
        int k1=0;
        for (int k=input.length-1;k>=0;k--) {
            input1[k1]=input[k];
            k1=k1+1;
        }
        for(int n=0;n<input1.length;n++) {
            voiceString=voiceString+input1[n]+"\n";
        }
        voice.speak(voiceString);
        return true;
    }
    @Override
    public Boolean speechButton(Voice voice ,String text,int i) {
        //do nothing
        return false;
    }
    @Override
    public Voice Volume(Voice voice,float volume) {
        // TODO Auto-generated method stub
        //do nothing
        return voice;
    }
```

Which have some functions which are for the voice set and do not use. Basically they accept an Object voice, a string and an Int which represents the series. If something is not selected by the user it means that he wants it all to be read whenever we simply read the whole text or split the string And get the line we want.

And obviously we also have the adapter Which, as we saw in the package command factory, calls depending on what the user wants speech Reverse Speech

SpeechAdapter.java

```java
package Model;

import com.sun.speech.freetts.Voice;
public class SpeechAdapter {
    private VoiceManager vm = VoiceManager.getInstance();
    private Voice voice = vm.getVoice("kevin16");
    private textToSpeechInterface voicetext;
    private Boolean condition;
    public void speech(String text,int i) {
        voice.allocate();
        voicetext=new Speech();
        condition=voicetext.speechButton(voice ,text, i) ;
        if(condition=false) {
            System.out.print("Text conversion Fail in speeches");
        }
        else {
            System.out.print("Text conversion successes in speeches");
        }
    }
    public void ReverseSpeech(String text,int i) {
        voice.allocate();
        voicetext=new ReverseSpeech();
        condition=voicetext.ReverseSpeechButton(voice ,text, i) ;
        if(condition=false) {
            System.out.print("Text conversion Fail in Reverse speeches");
        }
        else {
            System.out.print("Text conversion successes in Reverse speeches");
        }
    }
    public void SetParametersofVoice(float volume,float wordpersech,float hrz) {
        voicetext=new VoiceParametersSet();
        voice=voicetext.Volume(voice, volume);
        voice=voicetext.speechRate(voice, wordpersech);
        voice=voicetext.pitch(voice, hrz);

    }
}
```

And depending on how we define the adapter in the factory, the corresponding function is called.

You call SetParametersofVoice separately from the other buttons and it is shown below (we are again talking about the same factory of the package command)

```java
Factory.java   textToSpeechInterface.java   SpeechAdapter.java   Speech.java   ReverseSpeech.java
public void Factorybuttonsaction(String typeofbutton,Stage stage,String textf1,String textf2,String textf3,String textf4,String textf5,String
    if (typeofbutton.equals("MenuNew")) {
        controller=new ControllerOfExecute(CreateNewFile);
        FactoryCommands.MenuNew(controller);
    }
    else if (typeofbutton.equals("MenuLoad")) {
        controller=new ControllerOfExecute(load);
        FactoryCommands.MenuLoad(controller);
    }
    else if (typeofbutton.equals("closeStage")) {
        controller=new ControllerOfExecute(exit);
        FactoryCommands.closeStage(controller,stage);
    }
    else if (typeofbutton.equals("MenuSave")) {
        controller=new ControllerOfExecute(Save);
        FactoryCommands.MenuSave(controller,stage,textf1,textf2,textf3,textf4,textf5,textA);
    }
    else if(typeofbutton.equals("Edit")){
        controller=new ControllerOfExecute(edit);
        FactoryCommands.Edit(controller,txt1, txt2, txt3, txtA);

    }
    else if(typeofbutton.equals("ReverseSpeechButton")) {
        voicetext.ReverseSpeech(textA,line);
    }
    else if(typeofbutton.equals("speechButton")) {
        voicetext.speech(textA,line);
    }
    else if(typeofbutton.equals("True")) {
        Active=true;
        replayer=new ReplayManager();
    }
    else if(typeofbutton.equals("False")) {
        Active=false;
    }
}

public String Factorybuttonsaction(String typeofbutton,String textA,int line,ActionEvent event) {
    if (typeofbutton.equals("Atbash")) {
        typeencode=new Atbash();
        return typeencode.MakeMainEncode(textA,line);
    }
    else if (typeofbutton.equals("Rot13")) {
        typeencode=new Rot_13();
        return typeencode.MakeMainEncode(textA,line);
    }else {
        typeencode=new Utf8();
        return typeencode.MakeMainEncode(textA,line);
    }

}
```

Which obviously sets the sound and there is another file in which it implements to textToSpeechInterface which we saw, in the VoiceParametersSet file

VoiceParametersSet.java

```java
package Model;

import com.sun.speech.freetts.Voice;
public class VoiceParametersSet implements textToSpeechInterface{
    @Override
    public Boolean ReverseSpeechButton(Voice voice, String text, int i) {
        //do nothing
        return false;
    }
    @Override
    public Boolean speechButton(Voice voice, String text, int i) {
        //do nothing
        return false;
    }
    @Override
    public Voice Volume(Voice voice,float volume) {
        if(volume>=0.0&&volume<=1.0) {
            voice.setVolume(volume);
        }
        else {
            System.out.print("unsuccessful change volume ");
        }
        return voice;
    }
    @Override
    public Voice speechRate(Voice voice,float wordpersech) {
        if(wordpersech!=-1) {
            voice.setRate(wordpersech);
        }
        else if(wordpersech==-1){
            //dont do something
        }
        else {
            System.out.print("unsuccessful change  speech rate ");
        }
        return voice;
    }
    @Override
    public Voice pitch(Voice voice,float hrz) {
        if(!(hrz==-1)) {
            voice.setPitch(hrz);
        }
        else if(hrz==-1){
            System.out.print("unsuccessful change  speech pitch ");
            //dont do something
        }
        return voice;
    }
}
```

Which obviously has the other functions speechButton and ReverseSpeechButton but does nothing for them. This is how we completed the adapter Pattern.

Now we will analyze the coding in which we used statergy and template Pattern

First we have the Encode class:

Encode.java

```java
package Model;
public abstract class Encode {
    private char[] SmallAlphabet = new char[26],BigAlphabet = new char[26];
    private String[] array,arrayline;
    private String arraytostring="",arraytostringline="";
    public final String MakeMainEncode(String text,int line){
        for(char ch = 'a'; ch <= 'z'; ++ch)// fills alphabet array with the alphabet
        {
            SmallAlphabet[ch-'a']=ch;
        } 
        for(char ch = 'A'; ch <= 'Z'; ++ch)// fills alphabet array with the alphabet
        {
            BigAlphabet[ch-'A']=ch;
        }
        if (line==0) {
            array=text.split(" ");
            for (int i=0; i<array.length;i++) {
                if (!array[i].equals("\n")) {
                    if (typeEncode().equals("Atbash")) {
                        array[i]=EncodeAtbash(array[i],SmallAlphabet,BigAlphabet);
                    }                    
                    if(typeEncode().equals("Rot13")) {
                        array[i]=EncodeRot_13(array[i],SmallAlphabet,BigAlphabet);
                    }  
                    if(typeEncode().equals("UTF8")) {
                        array[i]= EncodeUTF_8(array[i],SmallAlphabet,BigAlphabet);
                        
                    }
                }
            }
            for (int i=0;i<array.length;i++) {
                if (i==array.length-1) {
                    arraytostring=arraytostring+array[i];
                }else {
                    arraytostring=arraytostring+array[i]+" ";
                }
            }
        }
        else {
            array=text.split("\n");
            arrayline=array[line-1].split(" ");
            for (int i=0; i<arrayline.length;i++) {
                if (typeEncode().equals("Atbash")) {
                    arrayline[i]=EncodeAtbash(arrayline[i],SmallAlphabet,BigAlphabet);
                }                    
                if(typeEncode().equals("Rot13")) {
                    arrayline[i]=EncodeRot_13(arrayline[i],SmallAlphabet,BigAlphabet);
                }
                if(typeEncode().equals("UTF8")) {
                    arrayline[i]= EncodeUTF_8(arrayline[i],SmallAlphabet,BigAlphabet);
                }
            }
            for (int i=0;i<arrayline.length;i++) {
                if (i==arrayline.length-1) {
                    arraytostringline=arraytostringline+arrayline[i];
                }else {
                    arraytostringline=arraytostringline+arrayline[i]+" ";
                }
            }
            for (int i=0;i<array.length;i++) {
                if(i!=line-1) {
                    if(i==array.length-1) {
                        arraytostring=arraytostring+array[i];
                    }
                    else {
                        arraytostring=arraytostring+array[i]+"\n";
                    }
                }
                else {
                    if(i==array.length-1) {
                        arraytostring=arraytostring+arraytostringline;
                    }
                    else {
                        arraytostring=arraytostring+arraytostringline+"\n";
                    }
                }
            }
        }
        return arraytostring;
    }
    abstract String EncodeAtbash(String array, char[] SmallAlphabet,char[] BigAlphabet);
    abstract String EncodeRot_13(String array, char[] SmallAlphabet,char[] BigAlphabet);
    abstract String  EncodeUTF_8(String array, char[] SmallAlphabet,char[] BigAlphabet);
    String typeEncode() {
        return "kati" ;
    }

}
```

In this abstract class in the function makeMainEncode we have the common code of all 3 encodings. And depending on the encoding the user has chosen we call a sixth atbash rot and utf8 to each do its own part. We also check for the user's line selections if any of them are selected. And depending on what she has selected I do the corresponding process to return a thong back that will have the appropriate changes.

I have also defined the 3 encodings as abstract Methods that I call later in different files. And a typeEncode that I do in each overwrite method, this helped me to choose the right encoding. To Elena with If.

We have an InterfaceEncode which implements these 3 classes as shown below to return the appropriate type to If.

InterfaceEncode.java

```java
package Model;
public interface InterfaceEncode {
    String encode();
}
class AtbaashReturn implements InterfaceEncode{
    public String encode() {
        return"Atbash";
    }
}
class Rot13Return implements InterfaceEncode{
    public String encode() {
        return"Rot13";
    }
}
class UTF implements InterfaceEncode{
    public String encode() {
        return"UTF8";
    }
}

```

Below we have the 3 encodings:

Utf8.java | Rot_13.java | Atbash.java

```java
package Model;
public class Utf8 extends Encode{
    private InterfaceEncode encode;
    public  Utf8() {
        super();
        encode=new UTF();
    }
    String EncodeAtbash(String array, char[] SmallAlphabet,char[] BigAlphabet) {
        return "" ;
    }
    String  EncodeRot_13(String array, char[] SmallAlphabet,char[] BigAlphabet) {
        //do nothing
        return "" ;
    }
    String  EncodeUTF_8(String array, char[] SmallAlphabet,char[] BigAlphabet) {
        return array;
    }
    String typeEncode() {
        return encode.encode() ;
    }
}
```

Here Utf8 is responsible for returning the text it received at the beginning before encoding whenever the way I call Utf8 is such that I have previously kept the original text and send it but not the encoded one. so when I call it, it returns the originally stored text unencoded. In the rest of the methods we do not do anything as we see in endoce rot13 and encodeatbash.

As we said before, typeEncode is done in every Overwrite encoding.

Below we see rot13

Utf8.java | Rot_13.java | Atbash.java

```java
package Model;
public class Rot_13 extends Encode{
    InterfaceEncode encode;
    private char[] SmallAlphabetPlus13 = new char[39],BigAlphabetPlus13 = new char[39],change;
    private int counter;
    private String StringForReturn;
    public  Rot_13() {
        super();
        encode=new Rot13Return();
    }
    String typeEncode() {
        return encode.encode() ;
    }
    String EncodeAtbash(String array, char[] SmallAlphabet,char[] BigAlphabet) {
        //do nothing
        return "" ;
    }
    String EncodeRot_13(String array, char[] SmallAlphabet,char[] BigAlphabet) {
        StringForReturn="";
        counter=0;
        for (int i=0; i<SmallAlphabet.length;i++) {
            SmallAlphabetPlus13[counter]=SmallAlphabet[i];
            counter=counter+1;
        }
        for (int i=0; i<SmallAlphabet.length-13;i++) {
            SmallAlphabetPlus13[counter]=SmallAlphabet[i];
            counter=counter+1;
        }
        counter=0;
        for (int i=0; i<BigAlphabet.length;i++) {
            BigAlphabetPlus13[counter]=BigAlphabet[i];
            counter=counter+1;
        }
        for (int i=0; i<BigAlphabet.length-13;i++) {
            BigAlphabetPlus13[counter]=BigAlphabet[i];
            counter=counter+1;
        }
        change=array.toCharArray();
        for (int i=0;i<change.length;i++) {
            for (int j=0;j<SmallAlphabet.length;j++) {
                if (change[i]==SmallAlphabet[j]) {
                    change[i]=SmallAlphabetPlus13[j+13];
                    break;
                }
                else if (change[i]==BigAlphabet[j]) {
                    change[i]=BigAlphabetPlus13[j+13];
                    break;
                }
            }

        }
        for (int i=0;i<change.length;i++) {
            StringForReturn=StringForReturn+change[i];
        }
        return StringForReturn;
    }
```

```java
    String  EncodeUTF_8(String array, char[] SmallAlphabet,char[] BigAlphabet) {
        //donothing
        return"";
    }
}
```

Which in turn does the coding as mentioned in the pronunciation and returns the text to replace the textarea we have in the view.

And apparently it also overwrites the typeEncode we use in the If As we said

We follow a similar logic in atbash, the only difference it has is in its coding, which does it in a different way than rot

Utf8.java | Rot_13.java | Atbash.java

```java
package Model;
public class Atbash extends Encode{
    InterfaceEncode encode;
    private char[] ReverseSmallAlphabet = new char[26],ReverseBigAlphabet = new char[26],change;
    private int counter;
    private String StringForReturn;
    public  Atbash() {
        super();
        encode=new AtbaashReturn();
    }
    @Override
    String typeEncode() {
        return encode.encode() ;
    }
    @Override
    String EncodeAtbash(String array, char[] SmallAlphabet,char[] BigAlphabet) {
        StringForReturn="";
        counter=0;
        for (int i=SmallAlphabet.length-1;i>=0;i--) {
            ReverseSmallAlphabet[counter]=SmallAlphabet[i];
            counter=counter+1;
        }
        counter=0;
        for (int i=BigAlphabet.length-1;i>=0;i--) {
            ReverseBigAlphabet[counter]=BigAlphabet[i];
            counter=counter+1;
        }
        change=array.toCharArray();
        for (int i=0;i<change.length;i++) {
            for (int j=0;j<SmallAlphabet.length;j++) {
                if (change[i]==SmallAlphabet[j]) {
                    change[i]=ReverseSmallAlphabet[j];
                    break;
                }
                else if (change[i]==BigAlphabet[j]) {
                    change[i]=ReverseBigAlphabet[j];
                    break;
                }
            }
        }
        for (int i=0;i<change.length;i++) {
            StringForReturn=StringForReturn+change[i];
        }
        return StringForReturn;
    }
    @Override
    String EncodeRot_13(String array, char[] SmallAlphabet,char[] BigAlphabet) {
        //do nothing
        return "";
    }
    String  EncodeUTF_8(String array, char[] SmallAlphabet,char[] BigAlphabet) {
        //do nothing
        return "";
    }
}
```

And obviously we overwrite typeEncode again. And we don't bother with the other methods.

Finally, we are left with the last US which has to do with the replay.

This is what ReplayManager does. The one that has 2 methods, one is the save, the one that every time we do an action, given that we have always pressed to record, it goes ahead and records our movements, in essence it records every field in parallel tables.

For example if it saves something it will keep all the fields needed to call it later when Play is pressed the relevant function. And below we have the code.

ReplayManager.java

```java
package Model;
import java.util.ArrayList;
public class ReplayManager {
    private ArrayList<String> typeofbutton = new ArrayList<String>();
    private ArrayList<String[]> TextList = new ArrayList<String[]>();
    private ArrayList<TextArea> TxtAREAList = new ArrayList<TextArea>();
    private ArrayList<TextField[]> TxtList = new ArrayList<TextField[]>();
    private ArrayList<String> TextAList = new ArrayList<String>();
    private ArrayList<Integer> lineList = new ArrayList<Integer>();
    private ArrayList<float[]> setVOICE = new ArrayList<float[]>();
    private ArrayList<ActionEvent> events = new ArrayList<ActionEvent>();
    private float[] Sets=new float[3];
    private ArrayList<Stage> stageList = new ArrayList<Stage>();
    private String[] Text=new String[5];
    private TextField[] Txt=new TextField[3];
    private Factory record;
    private String[] array=new String[6];
    private View view=new View();
    private int k;
    public void SaveActions(String typeofB,Stage stage,String textf1,String textf2,String textf3,String textf4,String textf5,String textA,int line,float volume,float wordpersech,float hrz,ActionEvent event,TextField tx
        typeofbutton.add(typeofB);
        stageList.add(stage);
        Text[0]=textf1;
        Text[1]=textf2;
        Text[2]=textf3;
        Text[3]=textf4;
        Text[4]=textf5;
        Txt[0]=txt1;
        Txt[1]=txt2;
        Txt[2]=txt3;
        TxtAREAList.add(txtA);
        TxtList.add(Txt);
        TextList.add(Text);
        TextAList.add(textA);
        lineList.add(line);
        if (!(volume==-1)) {
            Sets[0]=volume;
            Sets[1]=wordpersech;
            Sets[2]=hrz;
            setVOICE.add(Sets);
        }
        events.add(event);
        for (int k=0;k<Text.length;k++) {
            if(!(Text[k]==null)) {
                array[k]=Text[k];
            }
        }
        if(!(textA==null)) {
            array[5]=textA;
        }
    }

    public void replay() {
        record=new Factory();
        int listSize = events.size();
        k=0;
        for (int i = 0; i<listSize; i++){
            if (typeofbutton.get(i).equals("ReverseSpeechButton")||typeofbutton.get(i).equals("speechButton")) {
                record.Factorybuttonsaction(typeofbutton.get(i), stageList.get(i), TextList.get(i)[0], TextList.get(i)[1], TextList.get(i)[2], TextList.get(i)[3], TextList.get(i)[4], TextAList.get(i), lineList.get(i),
            }
            else if (typeofbutton.get(i).equals("Atbsash")||typeofbutton.get(i).equals("Rot13")||typeofbutton.get(i).equals("UTF-8")){
                view.TEXTAset(record.Factorybuttonsaction(typeofbutton.get(i), TextAList.get(i), lineList.get(i), events.get(i)));
            }
            else if (typeofbutton.get(i).equals("setVoice")){

                record.Factorybuttonsaction(setVOICE.get(k)[0], setVOICE.get(k)[1], setVOICE.get(k)[2]);
            }else if(typeofbutton.get(i).equals("MenuNew")){
                view.ViewWindow(array);
            }
            else if (typeofbutton.get(i).equals("MenuLoad")){
                view.ViewWindow(array);

            }else if(typeofbutton.get(i).equals("Edit")) {
                view.TEXTeditable();
            }
            else if(typeofbutton.get(i).equals("closeStage")) {
                view.Closestage();
            }
            else if (typeofbutton.get(i).equals("MenuSave")) {
                record.Factorybuttonsaction(typeofbutton.get(i), stageList.get(i), TextList.get(i)[0], TextList.get(i)[1], TextList.get(i)[2], TextList.get(i)[3], TextList.get(i)[4], TextAList.get(i), lineList.get(i),
                view.Closestage();
            }
            else {
                System.out.print(typeofbutton.get(i));
                System.out.print("Something going wrong Fail inputs.Try again");
            }
        }
    }

}
```

Whenever, as we see here, at the beginning we save all the fields in appropriate tables and later when the time comes to reproduce them, we call replay which in turn will call whatever function is needed to do the appropriate action.

And these functions are called again from the factory by the package command as we see below

```java
public void callReplayManager(String typeofB,Stage stage,String textf1,String textf2,String textf3,String textf4,String textf5,String textA,int line,float volume,float w
    if (Active) {
        replayer.SaveActions(typeofB, stage, textf1, textf2, textf3, textf4, textf5, textA, line, volume, wordpersech, hrz, event,txt1,txt2,txt3,txtA);
    }
```

Regarding active, it is a Boolean variable that when the record is pressed we set it to true otherwise false

```
    }
    else if(typeofbutton.equals("True")) {
        Active=true;
        replayer=new ReplayManager();
    }
    else if(typeofbutton.equals("False")) {
        Active=false;
    }
```

So as we understand the saving is done every time we call a function from the view and this is also visible in the code as for example here for Load H for new .

```
    PRESS.callReplayManager("MenuNew", null, null, null, null, null, null, null, -1, (float)-1, (float)-1, (float)-1,e1,null,null,null,null);
});
```

```
PRESS.callReplayManager("MenuLoad",ViewWindowStage, textf1.getText(), textf2.getText(), textf3.getText(), textf4.getText(), textf5.getText(), textA.getText(), -1, (float)-1,(float)-1, (float
```

So every time a button is pressed we call save here to say that while When we go New we call save right below in the main file when we do .

And when we now press Play in the view, the competent PlayTheRecord is called and the playback of the moves we made is finished

```
public void PlayTheRecord(ActionEvent event) {
    replayer.replay();
}
```

Here in this package we have 2 files save And Load To save And Load you call the Filebuttons file. The save code in filebuttons is mentioned above and the saveModel code is shown below.

Factory.java | Filebuttons.java | SaveModel.java

```java
package Model;

import java.io.File;
public class SaveModel {

    private String namefile;
    private FileWriter myWriter;
    private BasicFileAttributes attr;
    private FileWriter fstream;
    public int savefile(String[] array) {

        namefile=array[0];
        if(array.length<6) {
            return 0;
        }
        for (int i = 0; i <=5; i++) {
            if(array[i].isEmpty()&& i!=3&&i!=4) {
                return 0;
            }
        }
        try {
            Integer.parseInt(array[1]);
            
            
            return 0;
        } catch (final NumberFormatException e) {
            
        }
        try {
            
            
            Integer.parseInt(array[2]);
            return 0;
        } catch (final NumberFormatException e) {
            
        }
        try {
            myWriter = new FileWriter(namefile+".txt");
            myWriter.close();
            File file = new File(namefile+".txt");
            attr = Files.readAttributes(file.toPath(), BasicFileAttributes.class);
            fstream = new FileWriter(namefile+".txt", true);
            for (int i = 0; i <=5; i++) {
                if(i==3) {
                    fstream.write(attr.creationTime().toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()+"\n");
                }
                else if(i==4){
                    
                    fstream.write(attr.lastModifiedTime().toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()+"\n");
                }
                else {
                    fstream.write(array[i]+"\n");
                }
            }
            fstream.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            return 0;
        }
        return 1;

    }
}
```

Here we have a typical process where we check some data to not be numbers like for example author can't be a number and later we switch to opening file and writing with a structure so we can read it later.

We have also adjusted the writing so that the date of creation or the last conversion does not depend on the user dll, even if there is a bug and it manages to send information it will not affect the result, this has more to do with unitTEST. And you return the value 1 if everything went well

Finally, we have Load, which you also call from the appropriate point in Filebuttons. And we have the code below:

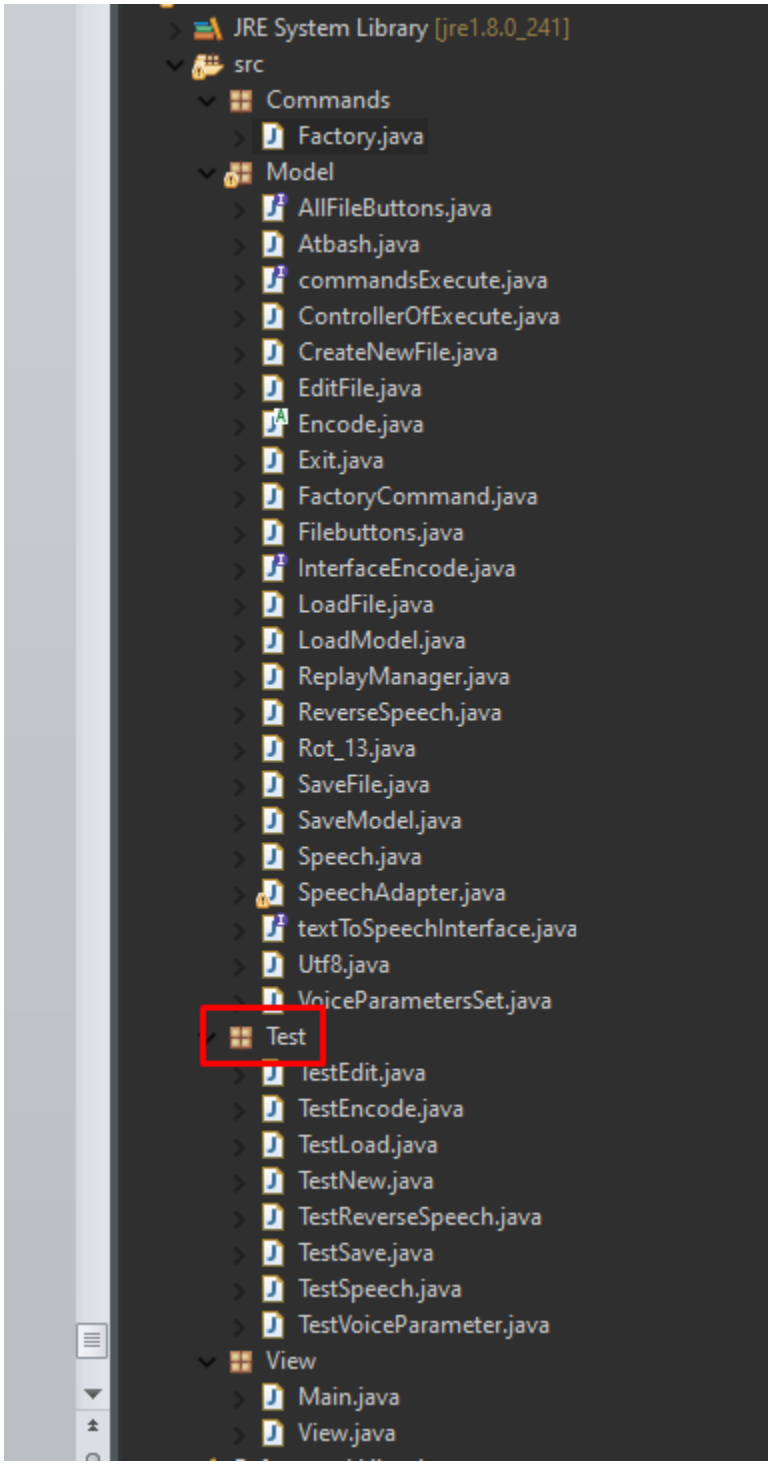
Factory.java | Filebuttons.java | LoadModel.java

```java
package Model;


import java.io.FileNotFoundException;
public class LoadModel {

    private String[] array;
    private String  dataTextA;
    private Scanner myReader;
    private int loadtag=1;
    public String[] LoadFile(String path) {
        File file =new File(path);
        dataTextA="";

        try {
            myReader = new Scanner(file);
            int i =0;
            if(!myReader.hasNextLine()) {
                myReader.close();
                return null;

            }
            else {
                array=new String[6];
            }
            while (myReader.hasNextLine()) {

                if(i!=5) {
                    String data = myReader.nextLine();
                    if (data=="") {
                        array[i]="";
                    }
                    array[i]=data;
                    i=i+1;
                }
                else {
                    dataTextA=dataTextA+myReader.nextLine()+"\n";
                }

            }
            array[i]=dataTextA;
            myReader.close();

        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

```java
        if(array==null) {
            System.out.print("Load unsuccessful");
            return null;
        }
        for (int i = 0; i <=5; i++) {
            if(loadtag==1&&array[i].isEmpty()) {
                loadtag=0;
            }
        }
        try {
            Integer.parseInt(array[1]);
            loadtag=0;
        } catch (final NumberFormatException e) {

        }
        try {
            Integer.parseInt(array[2]);
            loadtag=0;
        } catch (final NumberFormatException e) {
        }

        if (loadtag==1) {
            System.out.print("Load successful");

            return array;
        }else {
            System.out.print("Load unsuccessful");
            loadtag=1;
            return null;
        }

    }
}
```

And here we have the file open for reading save to an array some checks so we can't read any structure and read specific results and finally we have the return of the array if everything went well Null if something didn't go as it would we wanted .

We have finished the Unit TEST!



We made a Package that we named so you don't get involved in the model, we just want to make it clear for better visuals to have a package with the tests rather than just tests in src.

We will break down each test to 1 because we have done enough.

First of all let's say that for the files that should fail because we have set assertfalse it outputs correctly whenever don't be surprised if you see everything correctly that has the name test fail it means that it should have gone wrong and that we sent data wrong and it did come out wrong

Test edit:

TestEdit.java

```java
package Test;

import java.util.concurrent.CountDownLatch;
public class TestEdit {

    private Filebuttons file=new Filebuttons();
    private TextField text1=new TextField();
    private TextField text2=new TextField();
    private TextField text3=new TextField();
    private TextArea textA=new TextArea();
    @BeforeClass
    public static void initializefortextfield() throws ExceptionInInitializerError, InterruptedException {

        final CountDownLatch latch = new CountDownLatch(1);
        SwingUtilities.invokeLater(() -> {
            new JFXPanel(); // initializes JavaFX items epeidh einai to edit kai prepei na kanei edit anagkastika textfield items
            latch.countDown(); });
        if (!latch.await(5L, TimeUnit.SECONDS)) {
            throw new ExceptionInitializerError();
        }

        }
    }
    @Test
    public void TestEditPass()  {

        text1.setText("Text1");
        text2.setText("Text2");
        text3.setText("Text3");
        textA.setText("TextA");
        textA.setEditable(false);
        text1.setEditable(false);
        text2.setEditable(false);
        text3.setEditable(false);
        file.Edit(text1,text2,text3,textA) ;
        text1.setText("EditableText1");
        text2.setText("EditableText2");
        text3.setText("EditableText3");
        textA.setText("EditableTextA");
        Assert.assertTrue(text1.getText().equals("EditableText1")&&text2.getText().equals("EditableText2")&&text3.getText().equals("EditableText3")&&textA.getText().equals("EditableTextA"));
    }
```

Here, because we are messing with textfield information, we must create textfield fields by necessity and make them false in the editable, send them to the appropriate function and see if we then sort their text, we will get correct results.

The initialize is because obviously in test we can't have javafx access an error that gives .

Test Save

TestSave.java TestLoad.java

```java
package Test;
import org.junit.Assert;
public class TestSave {
    private SaveModel save;

    private String array[];

    @Test
    public void TestSaveFail1() {
        save=new SaveModel();
        array=new String[1];
        array[0]="TESTNAMEFail1";

        Assert.assertFalse(1==save.savefile(array));
    }
    @Test
    public void TestSaveFail2() {
        save=new SaveModel();
        array=new String[2];
        array[0]="TESTNAMEFail2";
        array[1]="TestAuthor";

        Assert.assertFalse(1==save.savefile(array));
    }
    @Test
    public void TestSaveFail3() {
        save=new SaveModel();
        array=new String[3];
        array[0]="TESTNAMEFail3";
        array[1]="TestAuthor";
        array[2]="TestTitle";

        Assert.assertFalse(1==save.savefile(array));
    }
    @Test
    public void TestSaveFail4() {
        save=new SaveModel();
        array=new String[4];
        array[0]="TESTNAMEFail4";
        array[1]="TestAuthor";
        array[2]="TestTitle";
        array[3]="Testsomething here";


        Assert.assertFalse(1==save.savefile(array));
    }
```

Here we see that we failed to save something wrong, obviously the result is negative because we have to write in the first 2 (textfiled) and the last field of the table (textarea).

While below we have correct saves

```java
	@Test
	public void TestSavePass1() {
		save=new SaveModel();
		array=new String[6];
		array[0]="TESTNAMEPass1";
		array[1]="TestAuthor";
		array[2]="TestTitle";
		array[3]="";
		array[4]="";
		array[5]="Test Text2speech something here";
		Assert.assertTrue(1==save.savefile(array));
	}
	@Test
	public void TestSavePass2() {
		save=new SaveModel();
		array=new String[9];
		array[0]="TESTNAMEPass2";
		array[1]="TestAuthor";
		array[2]="TestTitle";
		array[3]="";
		array[4]="";
		array[5]="Test Text2speech something here";//unotithete otidipote meta to 5 einai seires tou text2speech
		array[6]="Test Text2speech something here";
		array[7]="Test Text2speech something here";
		array[8]="Test Text2speech something here";
		Assert.assertTrue(1==save.savefile(array));
	}
```

Test Load

Here we try to load any saves from the previous test that passed to make sure they load as we made some test files manually for them to check.

TestSave.java | TestLoad.java

```java
package Test;


import org.junit.Assert;

public class TestLoad {
	public final ExpectedException exception = ExpectedException.none();
	private LoadModel file=new LoadModel();

	@Test
	public void TestLoadPass() {

		Assert.assertTrue(file.LoadFile("C:\\Users\\BASILIS\\eclipse-workspace\\TEXT2SPECH\\Tolkien.txt")!=null);
	}
	@Test
	public void TestLoadPass1() {

		Assert.assertTrue(file.LoadFile("C:\\Users\\BASILIS\\eclipse-workspace\\TEXT2SPECH\\TESTNAMEPass1.txt")!=null);
	}
	@Test
	public void TestLoadPass2() {

		Assert.assertTrue(file.LoadFile("C:\\Users\\BASILIS\\eclipse-workspace\\TEXT2SPECH\\TESTNAMEPass2.txt")!=null);
	}
	@Test
	public void TestLoadPass3() {

		Assert.assertTrue(file.LoadFile("C:\\Users\\BASILIS\\eclipse-workspace\\TEXT2SPECH\\TESTNAMEPass3.txt")!=null);
	}
	@Test
```

```java
	@Test
	public void TestLoadFail1() {
		Assert.assertFalse(file.LoadFile("C:\\Users\\BASILIS\\eclipse-workspace\\TEXT2SPECH\\emty.txt")!= null);

	}
	@Test
	public void TestLoadFail2() {
		Assert.assertFalse(file.LoadFile("C:\\Users\\BASILIS\\eclipse-workspace\\TEXT2SPECH\\failTestForLoad1.txt")!= null);

	}
	@Test
	public void TestLoadFail3() {
		Assert.assertFalse(file.LoadFile("C:\\Users\\BASILIS\\eclipse-workspace\\TEXT2SPECH\\failTestForLoad2.txt")!= null);

	}
	@Test
	public void TestLoadFail4() {
		Assert.assertFalse(file.LoadFile("C:\\Users\\BASILIS\\eclipse-workspace\\TEXT2SPECH\\failTestForLoad3.txt")!= null);

	}
```

Test New

Here there is obviously no margin for error as we have built the program the user simply presses a button whenever there is a test only

TestNew.java

```java
package Test;


import java.util.concurrent.CountDownLatch;
public class TestNew {

    private Filebuttons file=new Filebuttons();
    @BeforeClass
    public static void initializefortextfield() throws ExceptionInInitializerError, InterruptedException {

        final CountDownLatch latch = new CountDownLatch(1);
        SwingUtilities.invokeLater(() -> {
            new JFXPanel(); // initializes JavaFX
            latch.countDown(); });
            if (!latch.await(5L, TimeUnit.SECONDS)) {
                throw new ExceptionInInitializerError();
            }

            
    }
    @Test
    public void NewTest() {
        try
        {
            file.Mnew();//epeidi o tropos apou exw ftiaksei tin new den dinei perithorio lathous profanws den gine
        Assert.assertTrue(true);
        }
        catch(Exception ex) {

        }
    }
```

Test Encode

Here we simply expect the text to be encoded correctly and covering the possibility that the user may provide some or all of the sequence. And for all Encode types.

TestEncode.java

```java
package Test;
import org.junit.Assert;
public class TestEncode {

    private Encode Encode;
    private String startString;
    private String testString,converString;
    public char[] SmallAlphabet = new char[26],BigAlphabet = new char[26];

    

    @BeforeAll
    public void initializefortextfield() {    
        for(char ch = 'a'; ch <= 'z'; ++ch)// fills alphabet array with the alphabet
        {
            SmallAlphabet[ch-'a']=ch;
        }
        for(char ch = 'A'; ch <= 'Z'; ++ch)// fills alphabet array with the alphabet
        {
            BigAlphabet[ch-'A']=ch;
        }

    }

    @Test
    public void TestAtbashPass1() {
        Encode=new Atbash();
        startString="AAA";
        converString="ZZZ";
        testString=Encode.MakeMainEncode(startString,0);
        Assert.assertTrue(testString.equals(converString));
    }
    @Test
    public void TestAtbashPass2() {
        Encode=new Atbash();
        startString="aaa";
        converString="zzz";
        testString=Encode.MakeMainEncode(startString,0);
        Assert.assertTrue(testString.equals(converString));
    }
```

Test Speech-ReverseSpeech

Here we have a test by sending a text to see if it is actually read since the user does not have the option to do anything else so what remains to be tested is if it works outside of the program execution.

TestSpeech.java | TestReverseSpeech.java

```java
package Test;

import org.junit.Assert;
public class TestSpeech{
    private Speech Speech=new Speech();
    private VoiceManager vm = VoiceManager.getInstance();
    private Voice voice = vm.getVoice("kevin16");

    @Test
    public void TestSpeechPass() {
        voice.allocate();
        Assert.assertTrue(Speech.speechButton(voice,"a b c 1 2 3 ! @ # $ % ^ & * _ -",0));
    }
    @Test
    public void TestSpeechPass1() {
        voice.allocate();
        Assert.assertTrue(Speech.speechButton(voice,"1 2 3 ",0));
    }
    @Test
    public void TestSpeechPass2() {
        voice.allocate();
        Assert.assertTrue(Speech.speechButton(voice,"1 \n 2 \n 3",1));
    }
    @Test
    public void TestSpeechPass3() {
        voice.allocate();
        Assert.assertTrue(Speech.speechButton(voice,"1 \n 2 \n 3",2));
    }

    @Test
    public void TestSpeechPass4() {
        voice.allocate();
        Assert.assertTrue(Speech.speechButton(voice,"1 \n 2 \n 3",3));
    }
    @Test
    public void TestSpeechFail() {
        voice.allocate();
        Assert.assertFalse(Speech.speechButton(voice,"1 \n 2 \n 3",-1));
    }

    

}
```

TestSpeech.java | TestReverseSpeech.java

```java
package Test;

import org.junit.Assert;

public class TestReverseSpeech {
    private ReverseSpeech Speech=new ReverseSpeech();
    private VoiceManager vm = VoiceManager.getInstance();
    private Voice voice = vm.getVoice("kevin16");
    @Test
    public void TestSpeechPass() {
        voice.allocate();
        Assert.assertTrue(Speech.ReverseSpeechButton(voice,"a b c 1 2 3 ! @ # $ % ^ & * _ - ",0));
    }
    @Test
    public void TestSpeechPass1() {
        voice.allocate();
        Assert.assertTrue(Speech.ReverseSpeechButton(voice,"1 2 3 ",0));
    }
    @Test
    public void TestSpeechPass2() {
        voice.allocate();
        Assert.assertTrue(Speech.ReverseSpeechButton(voice,"1 2 3 \n 2 3 1  \n 3 2 1",1));
    }
    @Test
    public void TestSpeechPass3() {
        voice.allocate();
        Assert.assertTrue(Speech.ReverseSpeechButton(voice,"1 2 3 \n 2 3 1 \n 3 2 1",2));
    }

    @Test
    public void TestSpeechPass4() {
        voice.allocate();
        Assert.assertTrue(Speech.ReverseSpeechButton(voice,"1 2 3 \n 2 3 1 \n 3 2 1",3));
    }
    @Test
    public void TestSpeechFail() {
        voice.allocate();
        Assert.assertFalse(Speech.ReverseSpeechButton(voice,"1 2 3 \n 2 3 1 \n 3 2 1",-1));
    }
}
```

Test SetVoiceParameter

Here we play the parameters we send all the combinations and wait to see that it returns a correct result when we send allowed values because some of the values we send must be positive whenever we check all the cases and we wait to see a correct result in the error and in the correct way to send data to the class.

TestSpeech.java | TestReverseSpeech.java | TestVoiceParameter.java

```java
package Test;

import org.junit.Assert;
import org.junit.Test;
import com.sun.speech.freetts.Voice;
import com.sun.speech.freetts.VoiceManager;

import Model.VoiceParametersSet;

public class TestVoiceParameter {
    private VoiceManager vm = VoiceManager.getInstance();
    private Voice voice = vm.getVoice("kevin16");
    private VoiceParametersSet voicechange=new VoiceParametersSet();
    private float volume;
    private float speechRate;
    private float pitch;

    @Test
    public void TestVoiceParameterPass1() {
        voice.allocate();
        volume=voice.getVolume();
        speechRate=voice.getRate();
        pitch=voice.getPitch();
        voice=voicechange.Volume(voice, (float)0.5);
        Assert.assertTrue(volume!=voice.getVolume()&&speechRate==voice.getRate()&&pitch==voice.getPitch());
    }
    @Test
    public void TestVoiceParameterPass2s() {
        voice.allocate();
        volume=voice.getVolume();
        speechRate=voice.getRate();
        pitch=voice.getPitch();
        voice=voicechange.Volume(voice, (float)0.5);
        voice=voicechange.speechRate(voice, (float)100);
        Assert.assertTrue(volume!=voice.getVolume()&&speechRate!=voice.getRate()&&pitch==voice.getPitch());
    }
```