**ДОДАТОК Б**

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

\_\_\_\_\_\_\_\_ Ілля АХАЛАДЗЕ

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 р.

**ГОЛОСОВИЙ ПОМІЧНИК**

**Текст програми**

КПІ.ІП-1315.045490.03.12

“ПОГОДЖЕНО”

Керівник роботи:

\_\_\_\_\_\_\_\_\_\_\_\_ Ілля АХАЛАДЗЕ

|  |  |
| --- | --- |
| Консультант: | Виконавець: |
| \_\_\_\_\_\_\_\_\_\_\_ Максим ГОЛОВЧЕНКО | \_\_\_\_\_\_\_\_\_\_ Микита КИСЕЛЬОВ |

Київ – 2024

**Файл VoiceAssistant.java:**

/\*  
 \* Click nbfs:\\nbhost\\SystemFileSystem\\Templates\\Licenses\\license-default.txt to change this license  
 \* Click nbfs:\\nbhost\\SystemFileSystem\\Templates\\Classes\\Class.java to edit this template  
 \*/  
package com.nickmegistone.ai;  
  
import com.nickmegistone.ai.sphinxextextension.LiveSpeechRecognizerExtension;  
import edu.cmu.sphinx.api.Configuration;  
import javazoom.jl.decoder.JavaLayerException;  
import javazoom.jl.player.advanced.AdvancedPlayer;  
import org.jetbrains.annotations.NotNull;  
  
import java.io.FileInputStream;  
import java.io.IOException;  
  
*/\*\*  
 \* This class represents a voice assistant that can perform various tasks based on voice commands.  
 \*  
 \* @author Mykyta Kyselov - <a href="https://github.com/TheMegistone4Ever">Github</a>  
 \*/*public final class VoiceAssistant {  
  
 private final LiveSpeechRecognizerExtension lsr;  
 private boolean isRecognizing = false;  
  
 */\*\*  
 \* Constructs a VoiceAssistant object with the specified parameters.  
 \*  
 \* @param dictFilename A string representing the filename of the dictionary for speech recognition.  
 \* @param LMFilename A string representing the filename of the language model for speech recognition.  
 \*/* public VoiceAssistant(String dictFilename, String LMFilename) {  
 Configuration configuration = new Configuration();  
 configuration.setAcousticModelPath(  
 String.*format*("resource:%s", "/edu/cmu/sphinx/models/en-us/en-us"));  
 configuration.setDictionaryPath(  
 String.*format*("file:%s/src/main/java/com/nickmegistone/resources/%s", System.*getProperty*("user.dir"), dictFilename));  
 configuration.setLanguageModelPath(  
 String.*format*("file:%s/src/main/java/com/nickmegistone/resources/%s", System.*getProperty*("user.dir"), LMFilename));  
 try {  
 lsr = new LiveSpeechRecognizerExtension(configuration);  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
 }  
  
 */\*\*  
 \* Retrieves the command from the voice recognition result.  
 \*  
 \* @return The command extracted from the voice recognition result, converted to lowercase.  
 \*/* public @NotNull String getCommand() {  
 return lsr.getResult().getHypothesis().toLowerCase();  
 }  
  
 */\*\*  
 \* Retrieves the code corresponding to the given voice command.  
 \*  
 \* @param voiceCommand The voice command to be checked for code mapping.  
 \* @return The code associated with the voice command. Returns -1 if no matching code is found.  
 \*/* public int getCode(@NotNull String voiceCommand) {  
 if (voiceCommand.contains("play music")) {  
 return 0;  
 } else if (voiceCommand.contains("tell me a joke")) {  
 return 1;  
 } else if (voiceCommand.contains("weather forecast")) {  
 return 2;  
 } else if (voiceCommand.contains("search for")) {  
 return 3;  
 } else if (voiceCommand.contains("translate")) {  
 return 4;  
 } else if (voiceCommand.contains("hey vocalia")) {  
 return 5;  
 } else if (voiceCommand.contains("bye vocalia")) {  
 return 6;  
 }  
 return -1;  
 }  
  
 */\*\*  
 \* Starts the speech recognition process.  
 \*/* public void startRecognizing() {  
 lsr.startRecognition();  
 isRecognizing = true;  
 playMP3("greetings.mp3");  
 }  
  
 */\*\*  
 \* Stops the speech recognition process.  
 \*/* public void stopRecognizing() {  
 if (isRecognizing) {  
 lsr.stopRecognition();  
 isRecognizing = false;  
 }  
 playMP3("farewell.mp3");  
 }  
  
 */\*\*  
 \* Retrieves the substring after a specified search term in a given input string.  
 \*  
 \* @param input The input string to search within.  
 \* @param searchTerm The search term to find the substring after.  
 \* @return The substring after the search term.  
 \*/* public @NotNull String getSubstringAfter(@NotNull String input, @NotNull String searchTerm) {  
 if (input.length() <= searchTerm.length()) return input;  
 return input  
 .substring(input.indexOf(searchTerm) + searchTerm.length())  
 .trim()  
 .replaceAll("\\s", "%20");  
 }  
  
 */\*\*  
 \* Plays an MP3 file.  
 \*  
 \* @param filename A string representing the filename of the MP3 file to be played.  
 \*/* public void playMP3(String filename) {  
 try (FileInputStream in = new FileInputStream(String.*format*("%s/src/main/java/com/nickmegistone/resources/%s", System.*getProperty*("user.dir"), filename))) {  
 new AdvancedPlayer(in).play();  
 } catch (IOException | JavaLayerException e) {  
 throw new RuntimeException(e);  
 }  
 }  
}

**Файл GoogleTranslator.java:**

package com.nickmegistone.ai;  
  
import static com.nickmegistone.apputils.AppUtils.*getUrlContent*;  
import org.jetbrains.annotations.NotNull;  
  
import java.net.URLEncoder;  
import java.nio.charset.StandardCharsets;  
  
*/\*\*  
 \* The GoogleTranslator class represents a translator that can translate text from one language to another using  
 \* Google Apps Script. It utilizes the Google Apps Script web application deployed with a specific deploy ID.  
 \*  
 \* @author Mykyta Kyselov - <a href="https://github.com/TheMegistone4Ever">Github</a>  
 \*/*public class GoogleTranslator {  
  
 private final String deployId;  
  
 */\*\*  
 \* Constructs a GoogleTranslator object with the specified deploy ID.  
 \*  
 \* @param deployId The deployment ID for the Google Apps Script web application.  
 \*/* public GoogleTranslator(String deployId) {  
 this.deployId = deployId;  
 }  
  
 */\*\*  
 \* Translates the given text from the source language to the target language.  
 \*  
 \* @param langFrom The source language code.  
 \* @param langTo The target language code.  
 \* @param text The text to be translated.  
 \* @return The translated text.  
 \*/* public @NotNull String translate(String langFrom, String langTo, String text) {  
 return *getUrlContent*(  
 String.*format*(  
 "https://script.google.com/macros/s/%s/exec?q=%s&target=%s&source=%s",  
 deployId,  
 URLEncoder.*encode*(text, StandardCharsets.*UTF\_8*),  
 langTo,  
 langFrom  
 )  
 );  
 }  
}

**Файл MCNPLNN.java:**

package com.nickmegistone.ai;  
  
import org.apache.commons.lang3.StringUtils;  
import org.jetbrains.annotations.NotNull;  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
import java.util.\*;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
*/\*\*  
 \* This class represents a Markov Chain Natural Language Processing Neural Network.  
 \*  
 \* @author Mykyta Kyselov - <a href="https://github.com/TheMegistone4Ever">Github</a>  
 \*/*public class MCNPLNN {  
  
 private final Random random;  
 private final Map<String, Map<String, Double>> MCModel;  
 private final int nGram;  
  
 */\*\*  
 \* This constructor creates a Markov Chain model object from a file of raw text.  
 \*  
 \* @param filename A filename that will be a list of cleaned words to be used as the basis for the model.  
 \* @param nGram An integer representing the number of words in each state. Default value is 3.  
 \*/* public MCNPLNN(String filename, int nGram) {  
 random = new Random();  
 this.nGram = nGram;  
 List<String> clearText = *readAndCleanText*(String.*format*(System.*getProperty*("user.dir") +  
 "/src/main/java/com/nickmegistone/resources/%s", filename));  
 MCModel = new HashMap<>();  
 for (int i = 0; i <= clearText.size() - (nGram << 1); ++i) {  
 String currState = String.*join*(" ", clearText.subList(i, i + nGram));  
 String nextState = String.*join*(" ", clearText.subList(i + nGram, i + (nGram << 1)));  
 MCModel.putIfAbsent(currState, new HashMap<>());  
 MCModel.get(currState).put(nextState, MCModel.get(currState).getOrDefault(nextState, .0) + 1);  
 }  
 // Relative frequency  
 for (Map.Entry<String, Map<String, Double>> entry : MCModel.entrySet()) {  
 double totalNeighbours = entry.getValue().values().stream().mapToDouble(Double::doubleValue).sum();  
 Map<String, Double> transition = entry.getValue();  
 for (Map.Entry<String, Double> transitionEntry : transition.entrySet()) {  
 transitionEntry.setValue(transitionEntry.getValue() / totalNeighbours);  
 }  
 }  
 }  
  
 */\*\*  
 \* This function generates a string of text based on a Markov chain model.  
 \*  
 \* @param maxTokens An integer representing the maximum tokens in the generated text. Default value is 30.  
 \* @param start A string representing the initial state to begin text generation.  
 \* @return A processed string of generated text.  
 \*/* public String getSentence(int maxTokens, @NotNull String start) {  
 if (start.chars().filter(ch -> ch == ' ').count() + 1 != nGram) {  
 return "Count of words must be same as nGram!";  
 }  
 List<String> text = new ArrayList<>(Collections.*singletonList*(start));  
 for (int i = 0; i < maxTokens; ++i) {  
 Map<String, Double> possibleWords = MCModel.get(text.get(text.size() - 1));  
 String token = (String) possibleWords.keySet().toArray()[getRandomIndexByWeights(possibleWords.values().stream().toList())];  
 if (token.equals(start)) {  
 break;  
 }  
 text.add(token);  
 }  
 return StringUtils.*capitalize*(String.*join*(" ", text) + ".")  
 .replaceFirst(" Okay", " Okay,")  
 .replaceFirst(" heres", " here's")  
 .replaceFirst(" joke", " joke:")  
 .replaceFirst(" it", ", it")  
 .replaceFirst(" because", "? - because")  
 .replaceFirst(" youre", ", you're")  
 .replaceFirst(" haha", " - :) ha-ha ;),");  
 }  
  
 */\*\*  
 \* This function reads text from a file and returns a list of cleaned words.  
 \*  
 \* @param filePath A string representing the path to the file to be read.  
 \* @return A list of cleaned words extracted from the file.  
 \*/* private static @NotNull List<String> readAndCleanText(String filePath) {  
 List<String> cleanedWords = new ArrayList<>();  
 Pattern pattern = Pattern.*compile*("\\b\\w+\\b");  
 try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {  
 String line;  
 while ((line = br.readLine()) != null) {  
 Matcher matcher = pattern.matcher(line.replaceAll("[^\\w\\s\\a(){}-]", "").toLowerCase());  
 while (matcher.find()) {  
 cleanedWords.add(matcher.group());  
 }  
 }  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
 return cleanedWords;  
 }  
  
 */\*\*  
 \* This function returns a random index based on a list of probabilities.  
 \*  
 \* @param probabilities A list of doubles representing the probabilities.  
 \* @return An integer representing the random index.  
 \*/* private int getRandomIndexByWeights(@NotNull List<Double> probabilities) {  
 int i = 0;  
 for (double cumulativeProbability = 0, rnd = random.nextDouble(); i < probabilities.size(); ++i) {  
 cumulativeProbability += probabilities.get(i);  
 if (rnd < cumulativeProbability) {  
 break;  
 }  
 }  
 return i;  
 }  
}

**Файл OWMForecaster.java:**

package com.nickmegistone.ai;  
  
import io.ipinfo.api.IPinfo;  
import io.ipinfo.api.errors.RateLimitedException;  
import io.ipinfo.api.model.IPResponse;  
import org.json.JSONObject;  
  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.net.URL;  
import java.util.Locale;  
  
import static com.nickmegistone.apputils.AppUtils.*getUrlContent*;  
  
*/\*\*  
 \* This class represents an OWMForecaster that retrieves weather forecast information from the OpenWeatherMap API.  
 \*  
 \* @author Mykyta Kyselov - <a href="https://github.com/TheMegistone4Ever">Github</a>  
 \*/*public class OWMForecaster {  
  
 private final String OWMId;  
 private final IPResponse response;  
  
 */\*\*  
 \* Constructs an OWMForecaster object with the specified IPInfo ID and OpenWeatherMap ID.  
 \*  
 \* @param IPInfoId The IPInfo ID for retrieving location information.  
 \* @param OWMId The OpenWeatherMap ID for accessing the weather API.  
 \*/* public OWMForecaster(String IPInfoId, String OWMId) {  
 this.OWMId = OWMId;  
 // *TODO: Replace with a more reliable method of retrieving the IP address.* try (BufferedReader br = new BufferedReader(new InputStreamReader(new URL("https://checkip.amazonaws.com/").openStream()))) {  
 response = new IPinfo.Builder().setToken(IPInfoId).build().lookupIP(br.readLine());  
 } catch (IOException | RateLimitedException e) {  
 throw new RuntimeException(e);  
 }  
 }  
  
 */\*\*  
 \* Retrieves the weather forecast for the current location.  
 \*  
 \* @return The weather forecast as a formatted string.  
 \*/* public String forecast() {  
 String forecast = "Hello there! It seems we've encountered a little hiccup in our weather system, and " +  
 "unfortunately, we don't have your specific location information at the moment.";  
 JSONObject json = new JSONObject(  
 *getUrlContent*(  
 String.*format*(  
 "https://api.openweathermap.org/data/2.5/forecast?lat=%s&lon=%s&appid=%s&units=%s&lang=%s",  
 response.getLatitude(),  
 response.getLongitude(),  
 OWMId,  
 "metric",  
 "en"  
 )  
 )  
 );  
 if (json.getString("cod").equals("200")) {  
 JSONObject infoObj = json.getJSONArray("list").getJSONObject(1);  
 JSONObject mainObj = infoObj.getJSONObject("main");  
 JSONObject cityObj = json.getJSONObject("city");  
 forecast = String.*format*(  
 "Good evening, folks! It's time for your weather update. In the beautiful town of %s, located " +  
 "in %s at coordinates %,.1f latitude and %,.1f longitude, we're experiencing next " +  
 "conditions. Currently, the temperature is %,.1f degrees Celsius, with a feels-like " +  
 "temperature of %,.1f degrees Celsius. We're expecting a maximum temperature of %,.1f " +  
 "degrees Celsius and a minimum temperature of %,.1f degrees Celsius. The atmospheric " +  
 "pressure stands at %d millibars, and the humidity is %d%%. As for the skies, %s. The " +  
 "wind is blowing at a speed of %,.1f meters per second. That's all for now from your " +  
 "weather team. Stay tuned for more updates!",  
 cityObj.getString("name"),  
 new Locale.Builder().setRegion(cityObj.getString("country")).build().getDisplayCountry(Locale.*ENGLISH*),  
 Double.*parseDouble*(response.getLatitude()),  
 Double.*parseDouble*(response.getLongitude()),  
 mainObj.getDouble("temp"),  
 mainObj.getDouble("feels\_like"),  
 mainObj.getDouble("temp\_max"),  
 mainObj.getDouble("temp\_min"),  
 mainObj.getInt("pressure"),  
 mainObj.getInt("humidity"),  
 infoObj.getJSONArray("weather").getJSONObject(0).getString("description"),  
 infoObj.getJSONObject("wind").getDouble("speed")  
 );  
 }  
 return forecast;  
 }  
}

**Файл Synthesizer.java:**

package com.nickmegistone.ai;  
  
import com.sun.speech.freetts.Voice;  
import com.sun.speech.freetts.VoiceManager;  
  
*/\*\*  
 \* The Synthesizer class represents a text-to-speech synthesizer that can convert text into speech.  
 \* It uses the FreeTTS library for speech synthesis. <a href="https://cmusphinx.github.io/wiki/tutorial/">Tutorial</a>.  
 \*  
 \* @author Mykyta Kyselov - <a href="https://github.com/TheMegistone4Ever">Github</a>  
 \*/*public class Synthesizer implements AutoCloseable {  
  
 private final Voice voice;  
  
 */\*\*  
 \* Constructs a Synthesizer object with the specified voice name and pitch.  
 \*  
 \* @param voiceName The name of the voice to be used for speech synthesis.  
 \* @param pitch The pitch of the synthesized speech (range: 0.0 to 500.0).  
 \*/* private Synthesizer(String voiceName, float pitch) {  
 System.*setProperty*("freetts.voices", "com.sun.speech.freetts.en.us.cmu\_us\_kal.KevinVoiceDirectory");  
 voice = VoiceManager.*getInstance*().getVoice(voiceName);  
 voice.setPitch(pitch);  
 voice.allocate();  
 }  
  
 private static final class SynthesizerHolder {  
 private static final Synthesizer *instance* = new Synthesizer("kevin16", 120);  
 }  
  
 */\*\*  
 \* Returns the singleton instance of the Synthesizer class.  
 \*  
 \* @return The Synthesizer instance.  
 \*/* public static Synthesizer getInstance() {  
 return SynthesizerHolder.*instance*;  
 }  
  
 */\*\*  
 \* Converts the given message into speech.  
 \*  
 \* @param message The message to be spoken.  
 \*/* public void speak(String message) {  
 voice.speak(message);  
 }  
  
 @Override  
 public void close() {  
 voice.deallocate();  
 }  
}

**Файл EventMenu.java:**

package com.nickmegistone.event;  
  
public interface EventMenu {  
  
 void selected(int index);  
}

**Файл Vocalia.java:**

package com.nickmegistone.vocaliamaven;  
  
import com.nickmegistone.form.CreatorsForm;  
import com.nickmegistone.form.Form;  
import com.nickmegistone.form.InitForm;  
import com.nickmegistone.form.LicensesForm;  
import org.jetbrains.annotations.NotNull;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.MouseEvent;  
  
import static com.nickmegistone.apputils.AppUtils.\*;  
  
public class Vocalia extends javax.swing.JFrame {  
  
 private int mouseX, mouseY;  
 private boolean isFullScreen = false;  
 private final InitForm initForm;  
 private final CreatorsForm creatorsForm;  
 private final LicensesForm licensesForm;  
  
 public Vocalia() {  
 initComponents();  
 setBackground(*MAIN\_BACKGROUND\_COLOR*);  
 initForm = new InitForm();  
 creatorsForm = new CreatorsForm();  
 licensesForm = new LicensesForm();  
 menu1.initMenu(index -> {  
 switch (index) {  
 case 0 -> showForm(initForm);  
 case 1 -> showForm(new Form(index)); // *TODO: add help form* case 2 -> showForm(creatorsForm);  
 case 9 -> showForm(licensesForm);  
 default -> showForm(new Form(index));  
 }  
 });  
 showForm(initForm);  
 menu1.setAllTemporarilyOffExcept(0);  
 }  
  
 private void showForm(Component com) {  
 body.removeAll();  
 body.add(com);  
 body.revalidate();  
 body.repaint();  
 }  
  
 // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents  
 private void initComponents() {  
  
 roundPanel1 = new com.nickmegistone.swing.RoundPanel();  
 menu1 = new com.nickmegistone.component.Menu();  
 body = new com.nickmegistone.swing.RoundPanel();  
  
 setDefaultCloseOperation(javax.swing.WindowConstants.*EXIT\_ON\_CLOSE*);  
 setTitle("Vocalia");  
 setBackground(new java.awt.Color(24, 24, 24));  
 setCursor(new java.awt.Cursor(java.awt.Cursor.*DEFAULT\_CURSOR*));  
 setMinimumSize(new java.awt.Dimension(940, 540));  
 setPreferredSize(new java.awt.Dimension(*WIDTH*, *HEIGHT*));  
  
 roundPanel1.setBackground(*LIGHT\_BACKGROUND\_COLOR*);  
 roundPanel1.setPreferredSize(new java.awt.Dimension(*FRAME\_WIDTH*, *FRAME\_HEIGHT*));  
 roundPanel1.addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {  
 public void mouseDragged(java.awt.event.MouseEvent evt) {  
 roundPanel1MouseDragged(evt);  
 }  
 });  
 roundPanel1.addMouseListener(new java.awt.event.MouseAdapter() {  
 public void mouseClicked(java.awt.event.MouseEvent evt) {  
 roundPanel1MouseClicked(evt);  
 }  
 public void mousePressed(java.awt.event.MouseEvent evt) {  
 roundPanel1MousePressed(evt);  
 }  
 });  
  
 menu1.setMinimumSize(new java.awt.Dimension(0, 0));  
 menu1.setPreferredSize(new java.awt.Dimension(256, 680));  
  
 body.setBackground(*DARK\_BACKGROUND\_COLOR*);  
 body.setPreferredSize(new java.awt.Dimension(998, 680));  
 body.setLayout(new java.awt.BorderLayout());  
  
 javax.swing.GroupLayout roundPanel1Layout = new javax.swing.GroupLayout(roundPanel1);  
 roundPanel1.setLayout(roundPanel1Layout);  
 roundPanel1Layout.setHorizontalGroup(  
 roundPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(roundPanel1Layout.createSequentialGroup()  
 .addGap(10, 10, 10)  
 .addComponent(menu1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGap(10, 10, 10)  
 .addComponent(body, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 0, Short.*MAX\_VALUE*)  
 .addContainerGap())  
 );  
 roundPanel1Layout.setVerticalGroup(  
 roundPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(roundPanel1Layout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(roundPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(menu1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 0, Short.*MAX\_VALUE*)  
 .addComponent(body, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 0, Short.*MAX\_VALUE*))  
 .addGap(10, 10, 10))  
 );  
  
 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  
 getContentPane().setLayout(layout);  
 layout.setHorizontalGroup(  
 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(roundPanel1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 282, Short.*MAX\_VALUE*)  
 );  
 layout.setVerticalGroup(  
 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(roundPanel1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 16, Short.*MAX\_VALUE*)  
 );  
  
 pack();  
 }// </editor-fold>//GEN-END:initComponents  
  
 private void roundPanel1MouseDragged(@NotNull MouseEvent evt) {//GEN-FIRST:event\_roundPanel1MouseDragged  
 System.*out*.println(evt);  
 setLocation(evt.getXOnScreen() - mouseX, evt.getYOnScreen() - mouseY);  
 }//GEN-LAST:event\_roundPanel1MouseDragged  
  
 private void roundPanel1MousePressed(@NotNull MouseEvent evt) {//GEN-FIRST:event\_roundPanel1MousePressed  
 System.*out*.println(evt);  
 mouseX = evt.getX();  
 mouseY = evt.getY();  
 }//GEN-LAST:event\_roundPanel1MousePressed  
  
 private void roundPanel1MouseClicked(@NotNull MouseEvent evt) {//GEN-FIRST:event\_roundPanel1MouseClicked  
 System.*out*.println(evt);  
 if (evt.getClickCount() > 1) {  
 GraphicsDevice device = GraphicsEnvironment.*getLocalGraphicsEnvironment*().getDefaultScreenDevice();  
 if (isFullScreen) {  
 // Exit full screen mode  
 device.setFullScreenWindow(null);  
 isFullScreen = false;  
 } else {  
 // Enter full screen mode  
 device.setFullScreenWindow(this);  
 isFullScreen = true;  
 }  
 }  
 }//GEN-LAST:event\_roundPanel1MouseClicked  
  
 public static void main(String[] args) {  
 /\* Set the Nimbus look and feel \*/  
 //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  
 /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  
 \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html  
 \*/  
 try {  
 for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.*getInstalledLookAndFeels*()) {  
 if ("Nimbus".equals(info.getName())) {  
 javax.swing.UIManager.*setLookAndFeel*(info.getClassName());  
 break;  
 }  
 }  
 } catch (ClassNotFoundException | InstantiationException | IllegalAccessException |  
 UnsupportedLookAndFeelException ex) {  
 java.util.logging.Logger.*getLogger*(Vocalia.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 }  
 //</editor-fold>  
  
 /\* Create and display the form \*/  
 java.awt.EventQueue.*invokeLater*(() -> {  
 Vocalia vocalia = new Vocalia();  
 vocalia.setIconImage(new ImageIcon(System.*getProperty*("user.dir") + "/src/main/java/com/nickmegistone/resources/logo.png").getImage());  
 vocalia.setVisible(true);  
 });  
 }  
  
 // Variables declaration - do not modify//GEN-BEGIN:variables  
 private javax.swing.JPanel body;  
 private com.nickmegistone.component.Menu menu1;  
 private com.nickmegistone.swing.RoundPanel roundPanel1;  
 // End of variables declaration//GEN-END:variables  
}

**Файл AppUtils.java:**

package com.nickmegistone.apputils;  
  
import org.jetbrains.annotations.NotNull;  
  
import java.awt.\*;  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.net.URI;  
import java.net.URISyntaxException;  
import java.net.URLDecoder;  
import java.nio.charset.StandardCharsets;  
  
public final class AppUtils {  
  
 public static final int *MENU\_SLEEP\_MILLIS* = 1400;  
 public static final int *FRAME\_WIDTH* = 1280;  
 public static final int *FRAME\_HEIGHT* = 720;  
 public static final int *THUMB\_SIZE* = 80;  
 public static final int *INTERNET\_TIMEOUT* = 1500;  
 public static final Color *SEARCH\_ENABLED\_COLOR* = new Color(0, 102, 102);  
 public static final Color *SEARCH\_DISABLED\_COLOR* = new java.awt.Color(255, 102, 0);  
 public static final Color *SCROLLBAR\_COLOR* = new Color(130, 130, 130);  
 public static final Color *MAIN\_BACKGROUND\_COLOR* = new Color(51, 51, 51);  
 public static final Color *LIGHT\_BACKGROUND\_COLOR* = new Color(102, 102, 102);  
 public static final Color *DARK\_BACKGROUND\_COLOR* = new Color(24, 24, 24);  
 public static final Color *AVATAR\_BORDER\_TEXT\_COLOR* = new Color(224, 224, 224);  
 public static final String *SYNTHESIZER\_IS\_SPEAKING* = "Synthesizer is speaking...";  
 public static final String *SEND* = "Send...";  
 public static final String *SEARCH\_WHEN\_CLICKED* = "";  
 public static final String *NO\_INTERNET\_CONNECTION\_SEARCH* = String.*format*("No internet connection, retrying after %,.1f seconds...", *INTERNET\_TIMEOUT* / 1000.0);  
  
 */\*\*  
 \* Retrieves the content of a URL address.  
 \*  
 \* @param urlAddress The URL address from which to retrieve the content.  
 \* @return The content of the URL as a string.  
 \*/* public static @NotNull String getUrlContent(String urlAddress) {  
 StringBuilder response = new StringBuilder();  
 try {  
 try (BufferedReader bufferedReader = new BufferedReader(  
 new InputStreamReader(  
 new URI(  
 urlAddress  
 ).toURL().openConnection().getInputStream()  
 )  
 )) {  
 String line;  
 while ((line = bufferedReader.readLine()) != null) {  
 response.append(line).append("\n");  
 }  
 }  
 } catch (IOException | URISyntaxException e) {  
 throw new RuntimeException(e);  
 }  
 return URLDecoder.*decode*(response.toString(), StandardCharsets.*UTF\_8*);  
 }  
}