OOPs in Java Project Report

MorseWise: The Code Convertor & QR Wizard

Project report

Department of Artificial Intelligence

A PROJECT REPORT

Submitted by

Batch-C

Group C-14

Shreya Arun-CB.SC.U4AIE23253

Siri Sanjana S-CB.SC.U4AIE23249

Anagha Menon-CB.SC.U4AIE23212

Varshitha Thilak Kumar-CB.SC.U4AIE23258

Supervised By:

Mrs. Sruthi Guptha

Assistant Professor Department of Artificial Intelligence

Amrita Vishwa Vidyapeetham

CERTIFICATE

This is to certify that we, the students of the Amrita School of Artificial Intelligence, have completed the "Morse Code Converter and QR code wizard" project as part of our OOPs in Java. The project work was carried out under the guidance and supervision of Mrs.Sruthi Guptha, Assistant Professor, Amrita School of Artificial Intelligence, Coimbatore. To the best of our knowledge, this work has not been submitted as the basis for any degree, diploma, associateship, fellowship, or similar award to any candidate in any university.

Date:

Mrs.Sruthi Guptha
Assistant Professor Amrita Vishwa Vidyapeetham, Coimbatore

Prof. Soman K. P
Dean
Amrita School of Artificial Intelligence
Amrita Vishwa Vidyapeetham, Coimbatore

Acknowledgment

We would like to express our sincere gratitude to our supervisor Mrs. Sruthi Assistant Professor who made this work possible. Her guidance and advice carried us through all the stages of writing the project. We would also like to give special thanks to our friends and seniors for their continuous support. We are deeply grateful for their unwavering support, expertise and dedication throughout the project. Their insights and collaborative efforts significantly enhanced the quality and outcomes of our work. Furthermore, we would like to express our gratitude to our Institution, Amrita Vishwa Vidyapeetham for providing the necessary resources and environment conductive to the successful completion of this project.

Abstract

Our project, titled “MorseWise,” presents a versatile tool for converting text to Morse code, decoding Morse code back to text, and generating QR codes for Morse-encoded messages. The application offers a user-friendly interface with multiple conversion options, including normal and custom playback speeds for Morse code transmission. Users can input text directly, upload text files for conversion, view conversion history, and save converted text to files. The MorseWise project also includes functionality for generating QR codes from Morse code messages, enhancing the accessibility and usability of Morse code in modern digital communication. With its robust features and efficient implementation, MorseWise serves as a valuable resource for Morse code enthusiasts, communication enthusiasts, and anyone seeking a reliable tool for Morse code conversion and transmission.

Table of contents

1. Introduction

Since its invention in the early 1800s, Morse code has been a respected means of communication for centuries. This technology of encoding text into sequences of dots and dashes, created by Samuel Morse and Alfred Vail, revolutionized long-distance communication and changed the course of history with its dependability and efficiency. From its modest beginnings as a telegraphy tool to its crucial role in military, naval, and amateur radio operations, Morse code has endured as a testament to inventiveness and fortitude in the face of rapid technical advancement.

Even with the development of contemporary communication technology, practitioners, educators, and enthusiasts around the world are still fascinated with Morse code. It is a language that endures and transcends boundaries of time and geography because of its simplicity and universality. But as we move closer to a digital world, the problem is how to make sure that Morse code is still applicable and understandable in modern settings. Here is where the MorseWise project shines as a shining example of accessibility and ingenuity, bridging the modern and traditional divide in Morse code communication.

In this project, we introduce MorseWise, a flexible software program that gives users smooth transmission, and translation of Morse code. With its user-friendly design and extensive feature set, MorseWise seeks to transform Morse code communication for emergency responders, students, enthusiasts, and other relevant audiences. Come along on a journey to learn the ageless art of Morse code and discover all of its applications in the linked world of today.

1.2.Problem Statement

Once a mainstay of long-distance communication, Morse code now faces obscurity in an era dominated by social media, instant messaging, and digital communication platforms. Even with its historical significance and ongoing appeal among aficionados, modern technologically advanced society finds it difficult to keep Morse code relevant and accessible. The lack of easily navigable instruments for translating, playing back, and transmitting Morse code impedes its incorporation into contemporary communication processes. This deprives lovers of Morse code, educators, emergency responders, and anybody interested in cryptography of an effective way to interact with this age-old mode of communication.
One major obstacle preventing the broad adoption and use of Morse code is the absence of a complete converter that can convert text to Morse code and vice versa, play back Morse code, convert files, and generate QR codes instantly. Current solutions frequently lack functionality, are disjointed, or necessitate laborious manual procedures, which makes it difficult to incorporate Morse code into modern communication methods.

Thus, there is an urgent need for a feature-rich, user-friendly Morse code converter that can meet the varied requirements of practitioners and enthusiasts in different fields. In order to bridge the gap between tradition and innovation and ensure the preservation and revitalization of Morse code as a relevant and accessible means of communication in the digital age, such a solution should provide strong functionality, user-friendly interfaces, and support for modern communication mediums.

1.3.Objective and Motivation

This project aims to provide an all-inclusive Java Morse Code Translator with cutting-edge features that distinguish it from other Morse code apps. The program will support a variety of special characters, numerals, and letters in both capital and lowercase. Users will be able to input text or Morse code to be translated using an intuitive interface. When an input is invalid, the program will also handle errors gracefully and display helpful error warnings. Users can upload text files to the application, which then transforms the text into Morse code or the other way around. This function keeps track of all the conversions that have been made and makes it easier to process massive volumes of data without requiring human input. The application can produce a QR code for the converted text after it has been transformed. Users can distribute the converted material in a contemporary and practical format because to this special function. The choice to store the text after conversion in a file, Users can save and retrieve their conversions for later use because to this functionality. With features including file upload and conversion, conversion history, QR code generating, and conversion saving, this Morse Code Translator is more feature-rich than others.

Due to these capabilities, users can get a comprehensive answer for their Morse code conversion needs, making it a flexible tool that goes beyond simple translation. The program's emphasis on user experience, as evidenced by its intuitive design and features, sets it apart from other Morse code translators.

Morse code, despite its outdated nature, has practical applications in assistive technology and radio communication, making this project a valuable tool for real-world use.

2.Literature Review

1.Towards a Morse code-based Non-invasive Thought-to-speech Converter- This paper researches toward a custom-built, non-invasive thought-to-speech converter that translates mental tasks into text, speech, and morse code is presented in this publication.[1]

2. Morse Code-based Communication System for Partially Paralyzed People- This study introduces a technology that allows quadriplegics with speech problems and limited finger motions to communicate with others, making their lives easier. The technology allows people to communicate with others more readily by employing Morse Code, which is simple to understand. Morse Code communication is based on dots and dashes. Text messaging and extra capabilities such as Real-time display and Text to Speech (TTS) distinguish this Morse code communication system from others, allowing for faster and more efficient communication. The system is used to message the recipient and establish short-distance conversation using the displayed board and Text to Speech.[2]

3. In order to improve the quality of their lives, this study presents a system that helps quadriplegics who have limited finger motions and speech problems connect with others. The system facilitates their communication with others by employing comprehensible Morse Code. Dots and dashes are used to build the Morse Code communication. In order to facilitate quick and effective communication, text messaging and other capabilities like text-to-speech (TTS) and real-time display have been added to different Morse code communication systems. Using the Text to Speech feature and the displayed board, the system can be utilised to create short-distance contact as well as send messages to the recipient. Pushbuttons, POT, ESP8266, Node MCU, and a mobile app are utilised to complete the objective.[3]

Although these Morse code-based communication methods are novel and helpful for people who have trouble speaking or moving, they have a few significant drawbacks. First of all, even if Morse code is straightforward, users must learn and master it, which can be difficult and time-consuming. Some users may find this learning curve too steep, especially those with significant cognitive disabilities. Second, the systems add complexity and possible points of failure due to their heavy reliance on external components like pushbuttons, POT, ESP8266, Node MCU, and mobile apps.

These parts need upkeep and technical assistance, and they might not be easily accessible, particularly for those with little experience with technology.
Furthermore, although improving communication, the text-to-speech and real-time display functions could not be responsive enough for fast-paced talks, which could cause delays and annoyance for users. Additionally, the systems' primary purpose is short-distance communication, which restricts their use in more expansive or remote environments. All things considered, these Morse code-based systems greatly enhance quadriplegics' capacity to communicate, but they also present issues with usability, technical dependence, and restricted communication range.

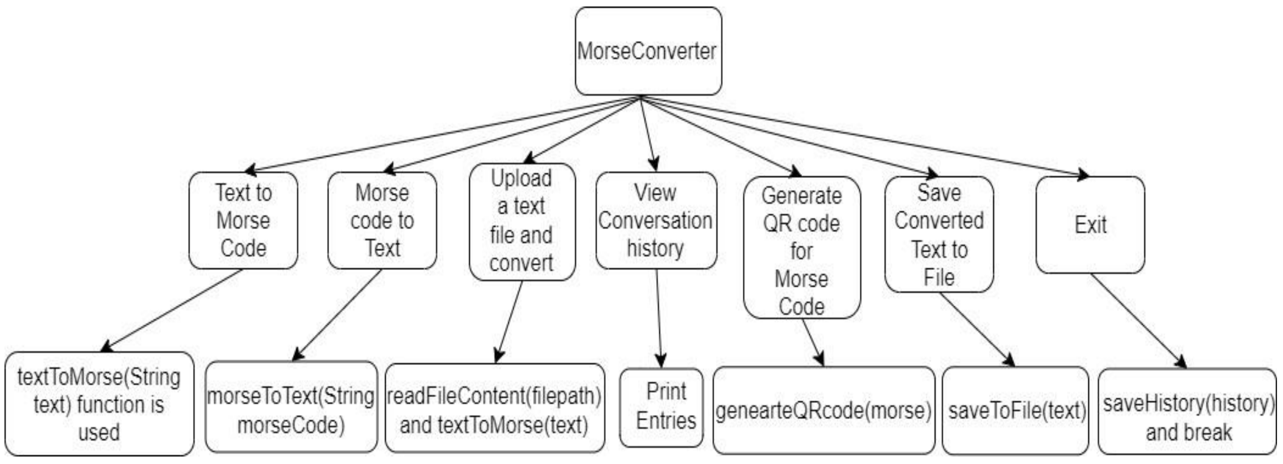
3.Advantages of our system

MorseWiser offers a comprehensive solution that has major positive effects on society. The project ensures that future generations will continue to value the cultural relevance of Morse code by conserving its heritage and fostering a respect for this antiquated mode of communication.

- Our Code provides Morse code implementation provides an extended version to approach
- QR code usage on scanning it we get the text of the converted code.
- We have implemented custom playback options where the Speed of the morse code read by the computer is managed and moreover the volume apart from the systems volume.
- The history of what the user search is also being recorded and stored. The converted text from morse code is also stored in as a txt file.

As a teaching aid, MorseWiser helps students, amateurs, and hobbyists understand and become proficient in the fundamentals of Morse code. Because of its accessibility qualities, Morse code communication is more approachable and inclusive, which increases participation and cultivates a varied community of aficionados. Additionally, MorseWiser is a communication tool that improves the effectiveness of Morse code transmission in a variety of settings, such as amateur radio operations and emergency communication.
MorseWiser connects classic Morse code with contemporary digital platforms by means of technology integration and QR code creation capabilities, making it easier to share and send messages that are encoded in Morse code.
MorseWiser stimulates experimentation and artistic expression through the use of Morse code, hence promoting creativity and innovation. Through encouraging community involvement and offering a flexible platform that can be tailored to various requirements, MorseWiser helps ensure that Morse code remains relevant and is used in the modern digital world.

4.System Design with Implementation


MorseConverter
Text to Morse Code
Morse code to Text
Upload a text file and convert
View Conversation history
Generate QR code for Morse Code
Save Converted Text to File
Exit
textToMorse(String text) function is used
morseToText(String morseCode)
readFileContent(filepath) and textToMorse(text)
Print Entries
genearteQRcode(morse)
saveToFile(text)
saveHistory(history) and break

The **MorseConverter** class is a Java program that provides functionality to convert text to Morse code, Morse code to text, generate QR codes for Morse code, and save conversion history to a file. Let's break down the code and explain its various parts:

4.1.Main Method (main):

- The **main** method is the entry point of the program.
- It starts by initializing a **Scanner** object to receive input from the user and a **List** to store conversion history.
- Then, it enters a loop where the user is presented with a menu of options.
- Depending on the user's choice, it executes the corresponding functionality.

4.2.Text to Morse Code Conversion:

- The program prompts the user to enter text to convert to Morse code.
- It then offers two options: normal playback or custom playback.
  - **Normal Playback**: Converts the text to Morse code and plays it back using default settings.
  - **Custom Playback**: Allows the user to specify custom playback settings for speed, pitch, and volume.

4.3.Morse Code to Text Conversion:

- The program prompts the user to enter Morse code to convert to text.
- It converts the Morse code to text and displays the result.

4.4.Viewing Conversion History:

- Users can view the history of all conversions made during the session.
- The program retrieves the conversion history from a file and displays it.

4.5.Generating QR Code for Morse Code:

- Users input Morse code, and the program generates a QR code representing the Morse code.
- The QR code is saved as a PNG image (**qrcode.png**).

4.6.Saving Converted Text to File:

- Users can save converted text to a file (**converted_text.txt**), which is saved on the user's desktop.

4.7.Helper Methods:

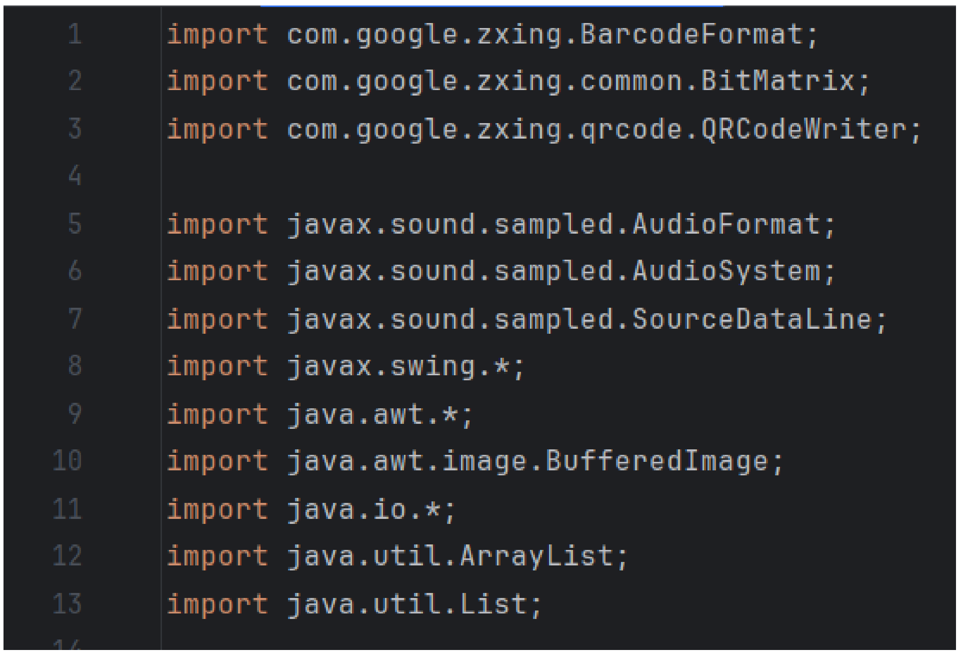
- **textToMorse**: Converts text to Morse code based on predefined mappings.
- **morseToText**: Converts Morse code to text based on predefined mappings.
- **playMorseCode**: Plays Morse code as sound, either using default settings or custom playback settings.
- **saveHistory**: Saves conversion history to a file (**history.txt**).
- **loadHistory**: Loads conversion history from a file.

4.8.QR Code Generation:

- The **generateQRCode** method generates a QR code for the provided Morse code using the Google ZXing library.
- The Morse code is encoded as text, and the QR code image is saved as a PNG file (**qrcode.png**).

5.System Implementation with code

1.1. Imports



```java
import com.google.zxing.BarcodeFormat;
import com.google.zxing.common.BitMatrix;
import com.google.zxing.qrcode.QRCodeWriter;

import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.SourceDataLine;
import javax.swing.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.*;
import java.util.ArrayList;
import java.util.List;
```

1. import com.google.zxing.BarcodeFormat;
- This imports the BarcodeFormat class from the Google Zxing library, which is used for generating and reading barcodes and QR codes.

2. import com.google.zxing.common.BitMatrix;
- This imports the BitMatrix class from the Google Zxing library, which represents a monochrome bitmap image.

3. import com.google.zxing.qrcode.QRCodeWriter;
- This imports the QRCodeWriter class from the Google Zxing library, which is used for encoding data into a QR code.

4. import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.SourceDataLine;
- These imports are from the javax.sound.sampled package, which provides classes for capturing, processing, and rendering audio data.

5. import javax.swing.*;
- This imports all the classes from the javax.swing package, which is a part of the Java Swing library, used for building graphical user interfaces (GUIs).

6. import java.awt.*;
- This imports all the classes from the java.awt package, which is a part of the Abstract Window Toolkit (AWT) library, providing low-level functionality for graphics and GUI applications.

7. import java.awt.image.BufferedImage;
- This imports the BufferedImage class from the java.awt.image package, which is used for representing and manipulating image data.

8. import java.io.*;
- This imports all the classes from the java.io package, which provides classes for input and output operations, such as reading and writing files.

9. import java.util.ArrayList;
import java.util.List;
- These imports are from the java.util package, which provides data structures and utility classes. ArrayList is an implementation of the List interface, which is a resizable-array implementation of the List interface.

6.2.Class & Method

That creates a graphical user interface (GUI) application using the Swing library. The application allows users to convert text to Morse code, Morse code to text, play Morse code audio, generate QR codes for Morse code, save converted text to a file, and view the conversion history.

1. MorseConverter class:
- This is the main class that extends the JFrame class from Swing.
- It initializes the GUI components, including text areas, buttons, and sliders.
- It contains event handlers for button clicks and other user interactions.

2. main method:
- This is the entry point of the program.
- It creates an instance of the MorseConverter class and sets the GUI frame as visible using the EventQueue.invokeLater method to ensure the GUI is created on the correct thread.

3. convertTextToMorse method:
- This method is called when the "Text to Morse" button is clicked.
- It gets the input text from the text area and converts it to Morse code using the textToMorse method.
- The resulting Morse code is displayed in the output text area, and the conversion is added to the history list.

4. convertMorseToText method:
- This method is called when the "Morse to Text" button is clicked.
- It gets the input Morse code from the text area and converts it to text using the morseToText method.
- The resulting text is displayed in the output text area, and the conversion is added to the history list.

5. saveTextToFile method:
- This method is called when the "Save to File" button is clicked.
- It gets the output text from the text area and saves it to a file named

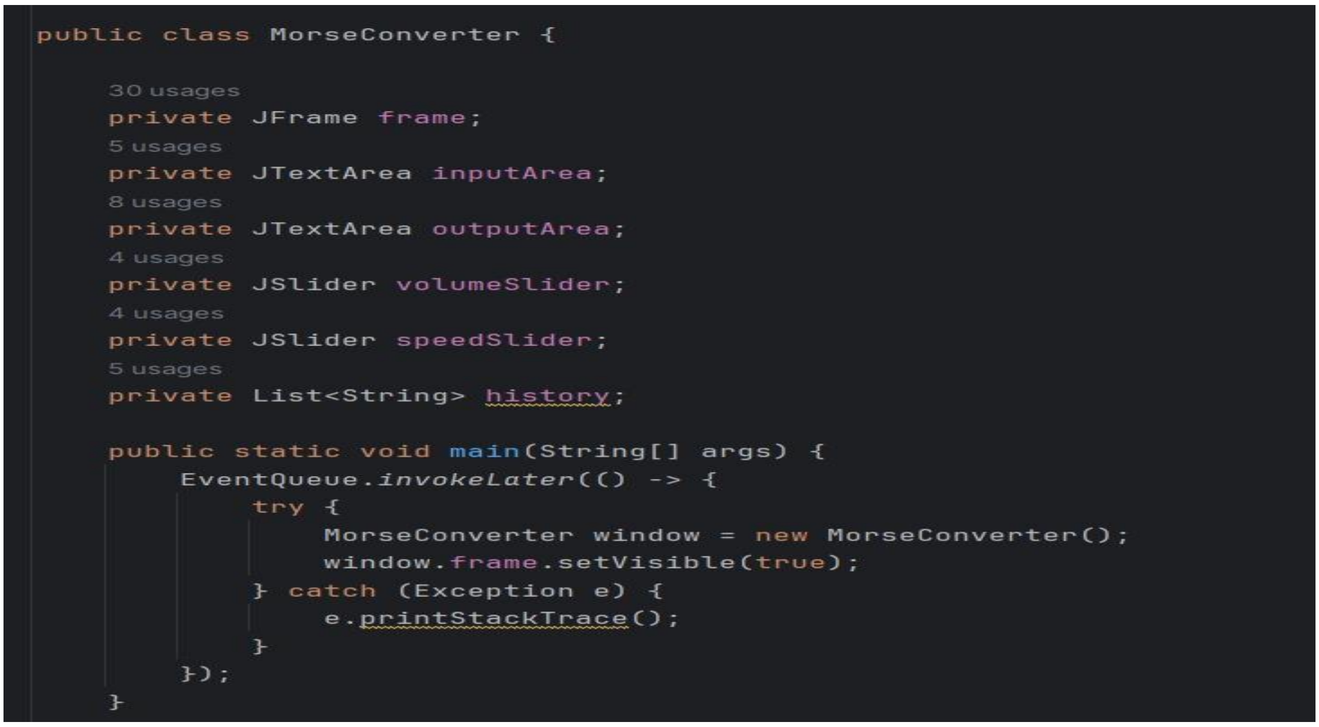
"converted_text.txt" on the user's desktop.

6. generateQRCode method:
- This method is called when the "Generate QR Code" button is clicked.
- It gets the Morse code from the output text area and generates a QR code image using the ZXing library.
- The QR code image is saved as "qrcode.png" in the current directory.

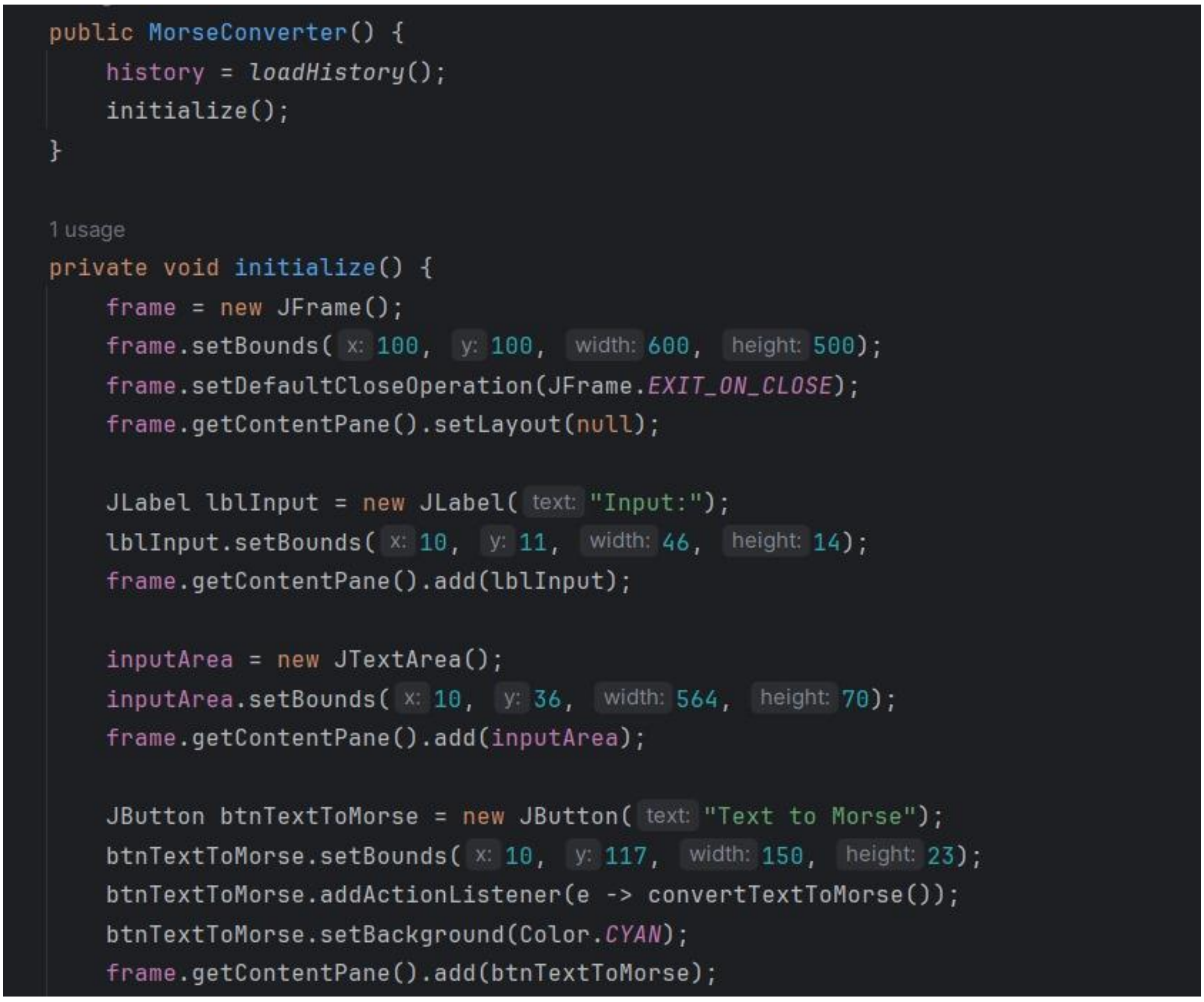
7. playMorseCode method:
- This method is called when the "Play Morse Code" button is clicked.
- It gets the Morse code from the output text area and plays the corresponding audio using the playMorseCode static method.
- The speed and volume of the audio can be adjusted using the sliders.

8. viewHistory method:
- This method is called when the "View History" button is clicked.
- It displays a dialog box containing the conversion history stored in the history list.

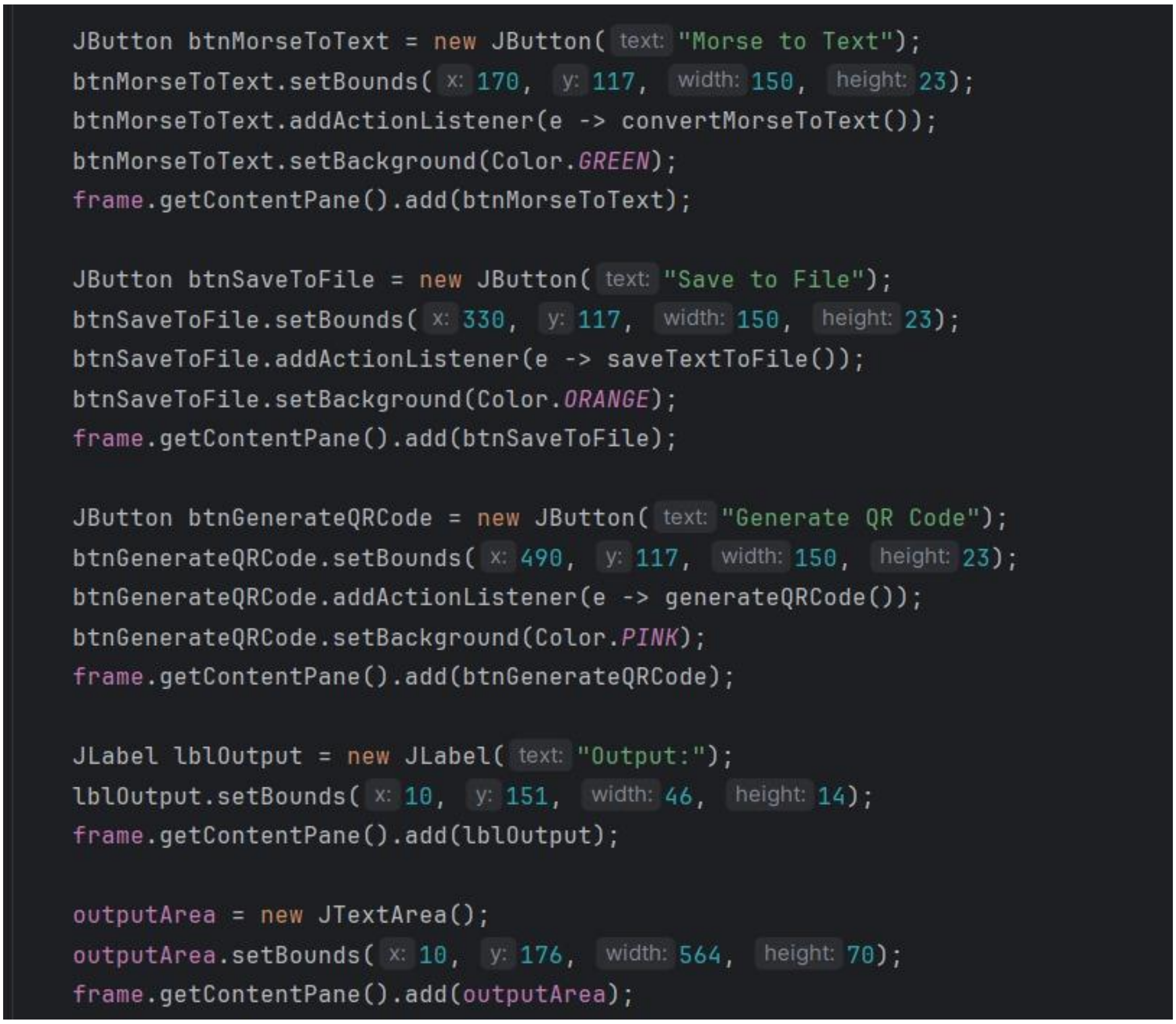
Additionally, the code includes several helper methods, such as textToMorse, morseToText, saveToFile, readFileContent, saveHistory, loadHistory, and playSound. These methods are responsible for the core functionality of converting between text and Morse code, saving/loading data to/from files, and playing audio.



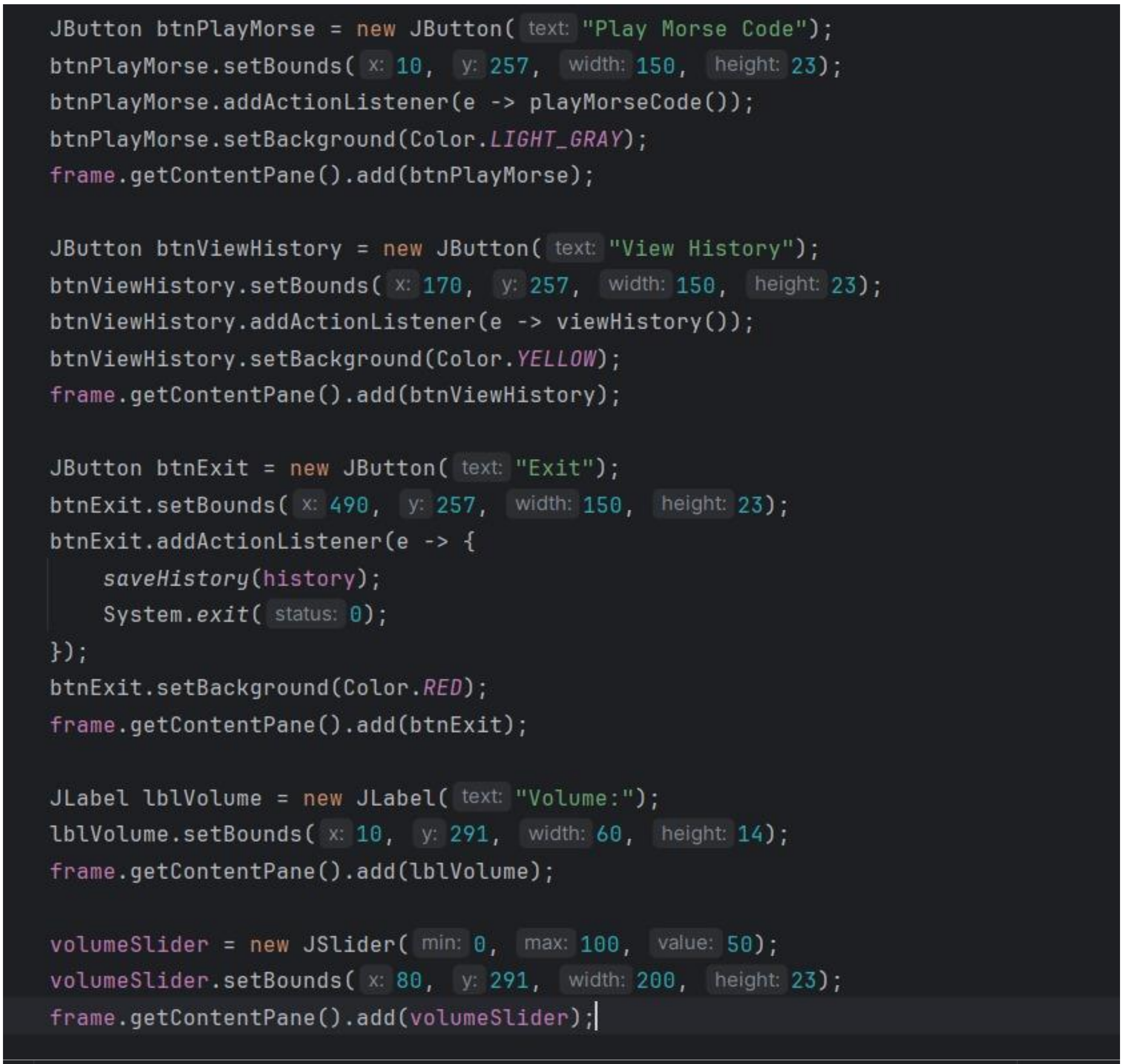
```
public class MorseConverter {

    30 usages
    private JFrame frame;
    5 usages
    private JTextArea inputArea;
    8 usages
    private JTextArea outputArea;
    4 usages
    private JSlider volumeSlider;
    4 usages
    private JSlider speedSlider;
    5 usages
    private List<String> history;

    public static void main(String[] args) {
        EventQueue.invokeLater(() -> {
            try {
                MorseConverter window = new MorseConverter();
                window.frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        });
    }
```



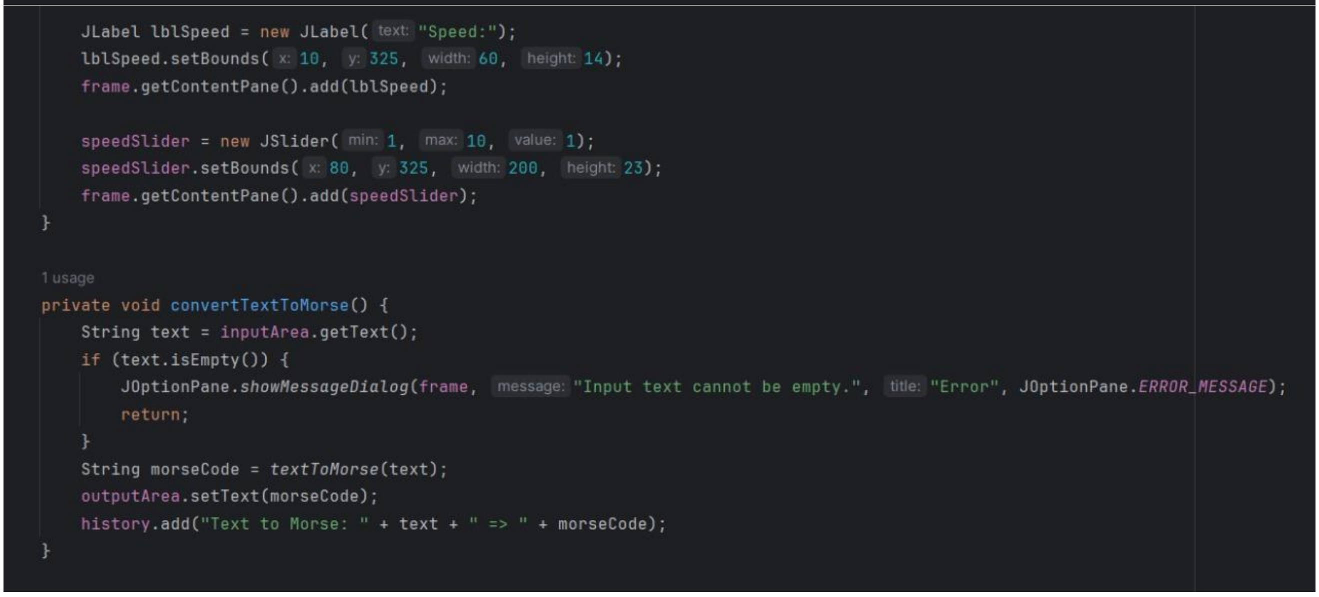
```
public MorseConverter() {
    history = loadHistory();
    initialize();
}

1 usage
private void initialize() {
    frame = new JFrame();
    frame.setBounds( x: 100,  y: 100,  width: 600,  height: 500);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().setLayout(null);

    JLabel lblInput = new JLabel( text: "Input:");
    lblInput.setBounds( x: 10,  y: 11,  width: 46,  height: 14);
    frame.getContentPane().add(lblInput);

    inputArea = new JTextArea();
    inputArea.setBounds( x: 10,  y: 36,  width: 564,  height: 70);
    frame.getContentPane().add(inputArea);

    JButton btnTextToMorse = new JButton( text: "Text to Morse");
    btnTextToMorse.setBounds( x: 10,  y: 117,  width: 150,  height: 23);
    btnTextToMorse.addActionListener(e -> convertTextToMorse());
    btnTextToMorse.setBackground(Color.CYAN);
    frame.getContentPane().add(btnTextToMorse);
```



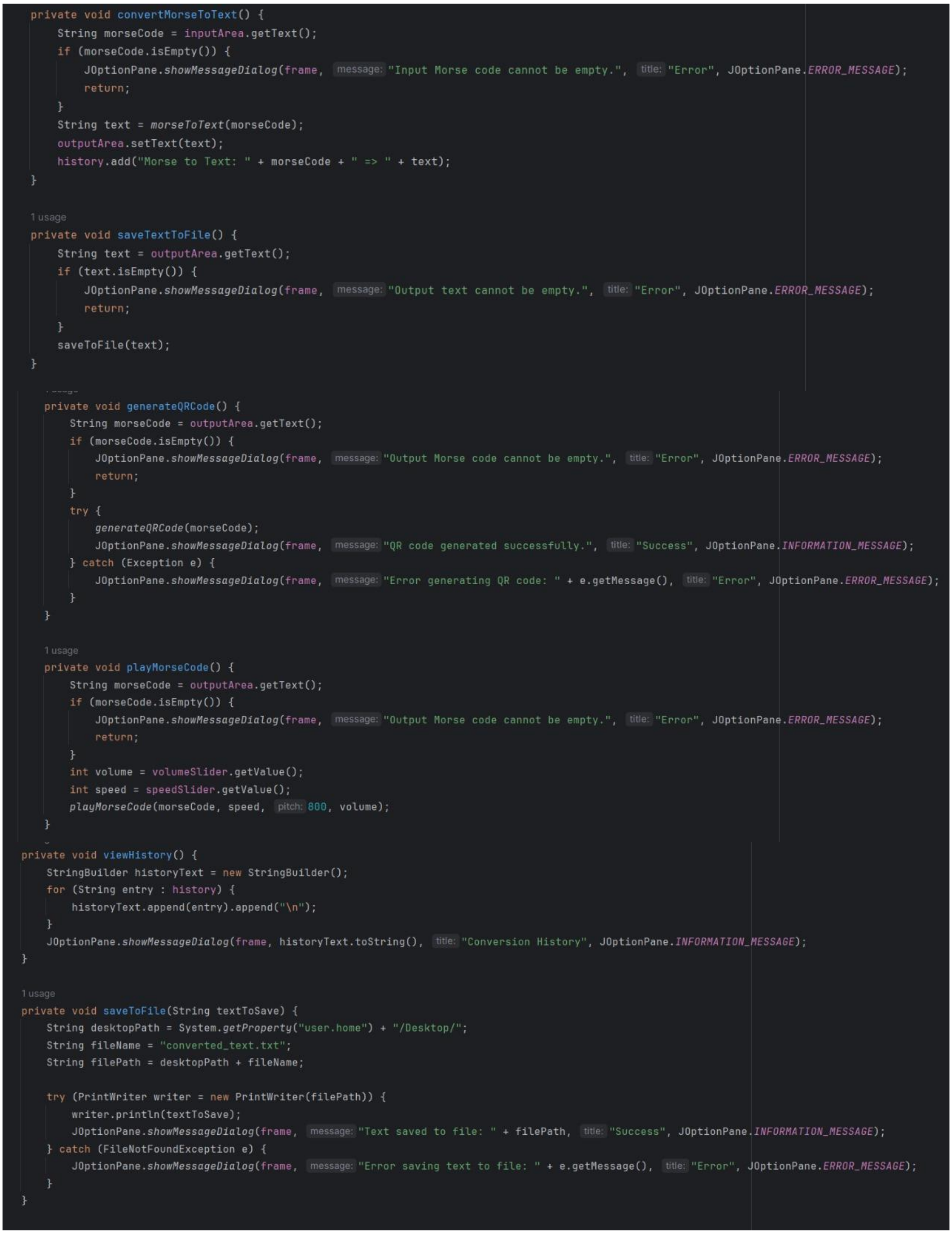
```java
JButton btnMorseToText = new JButton( text: "Morse to Text");
btnMorseToText.setBounds( x: 170, y: 117, width: 150, height: 23);
btnMorseToText.addActionListener(e -> convertMorseToText());
btnMorseToText.setBackground(Color.GREEN);
frame.getContentPane().add(btnMorseToText);

JButton btnSaveToFile = new JButton( text: "Save to File");
btnSaveToFile.setBounds( x: 330, y: 117, width: 150, height: 23);
btnSaveToFile.addActionListener(e -> saveTextToFile());
btnSaveToFile.setBackground(Color.ORANGE);
frame.getContentPane().add(btnSaveToFile);

JButton btnGenerateQRCode = new JButton( text: "Generate QR Code");
btnGenerateQRCode.setBounds( x: 490, y: 117, width: 150, height: 23);
btnGenerateQRCode.addActionListener(e -> generateQRCode());
btnGenerateQRCode.setBackground(Color.PINK);
frame.getContentPane().add(btnGenerateQRCode);

JLabel lblOutput = new JLabel( text: "Output:");
lblOutput.setBounds( x: 10, y: 151, width: 46, height: 14);
frame.getContentPane().add(lblOutput);

outputArea = new JTextArea();
outputArea.setBounds( x: 10, y: 176, width: 564, height: 70);
frame.getContentPane().add(outputArea);
```



```
JButton btnPlayMorse = new JButton( text: "Play Morse Code");
btnPlayMorse.setBounds( x: 10,  y: 257,  width: 150,  height: 23);
btnPlayMorse.addActionListener(e -> playMorseCode());
btnPlayMorse.setBackground(Color.LIGHT_GRAY);
frame.getContentPane().add(btnPlayMorse);

JButton btnViewHistory = new JButton( text: "View History");
btnViewHistory.setBounds( x: 170,  y: 257,  width: 150,  height: 23);
btnViewHistory.addActionListener(e -> viewHistory());
btnViewHistory.setBackground(Color.YELLOW);
frame.getContentPane().add(btnViewHistory);

JButton btnExit = new JButton( text: "Exit");
btnExit.setBounds( x: 490,  y: 257,  width: 150,  height: 23);
btnExit.addActionListener(e -> {
    saveHistory(history);
    System.exit( status: 0);
});
btnExit.setBackground(Color.RED);
frame.getContentPane().add(btnExit);

JLabel lblVolume = new JLabel( text: "Volume:");
lblVolume.setBounds( x: 10,  y: 291,  width: 60,  height: 14);
frame.getContentPane().add(lblVolume);

volumeSlider = new JSlider( min: 0,  max: 100,  value: 50);
volumeSlider.setBounds( x: 80,  y: 291,  width: 200,  height: 23);
frame.getContentPane().add(volumeSlider);
```



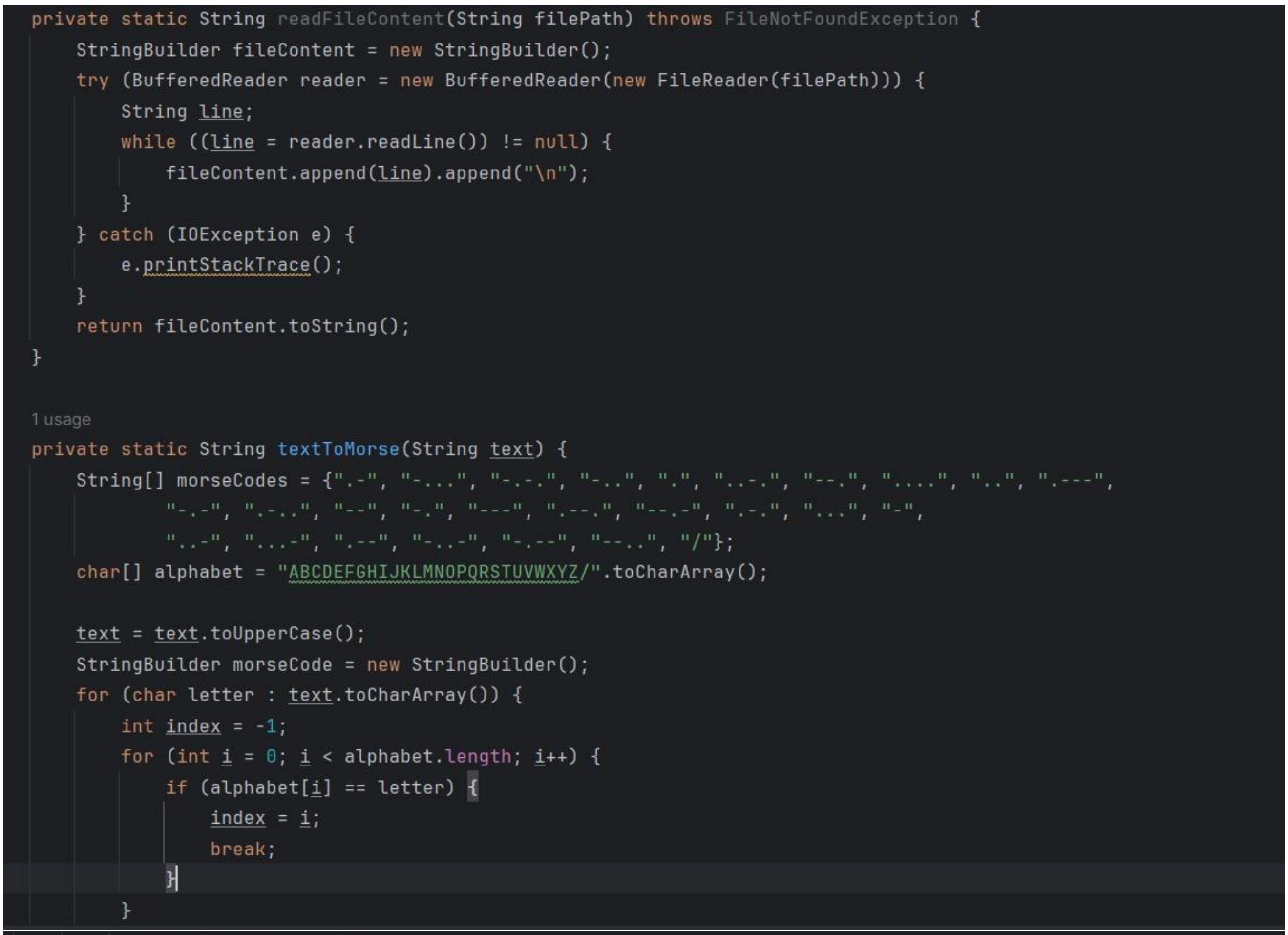
```
        JLabel lblSpeed = new JLabel( text: "Speed:");
        lblSpeed.setBounds( x: 10,  y: 325,  width: 60,  height: 14);
        frame.getContentPane().add(lblSpeed);

        speedSlider = new JSlider( min: 1,  max: 10,  value: 1);
        speedSlider.setBounds( x: 80,  y: 325,  width: 200,  height: 23);
        frame.getContentPane().add(speedSlider);
    }

    1 usage
    private void convertTextToMorse() {
        String text = inputArea.getText();
        if (text.isEmpty()) {
            JOptionPane.showMessageDialog(frame,  message: "Input text cannot be empty.",  title: "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        String morseCode = textToMorse(text);
        outputArea.setText(morseCode);
        history.add("Text to Morse: " + text + " => " + morseCode);
    }
```



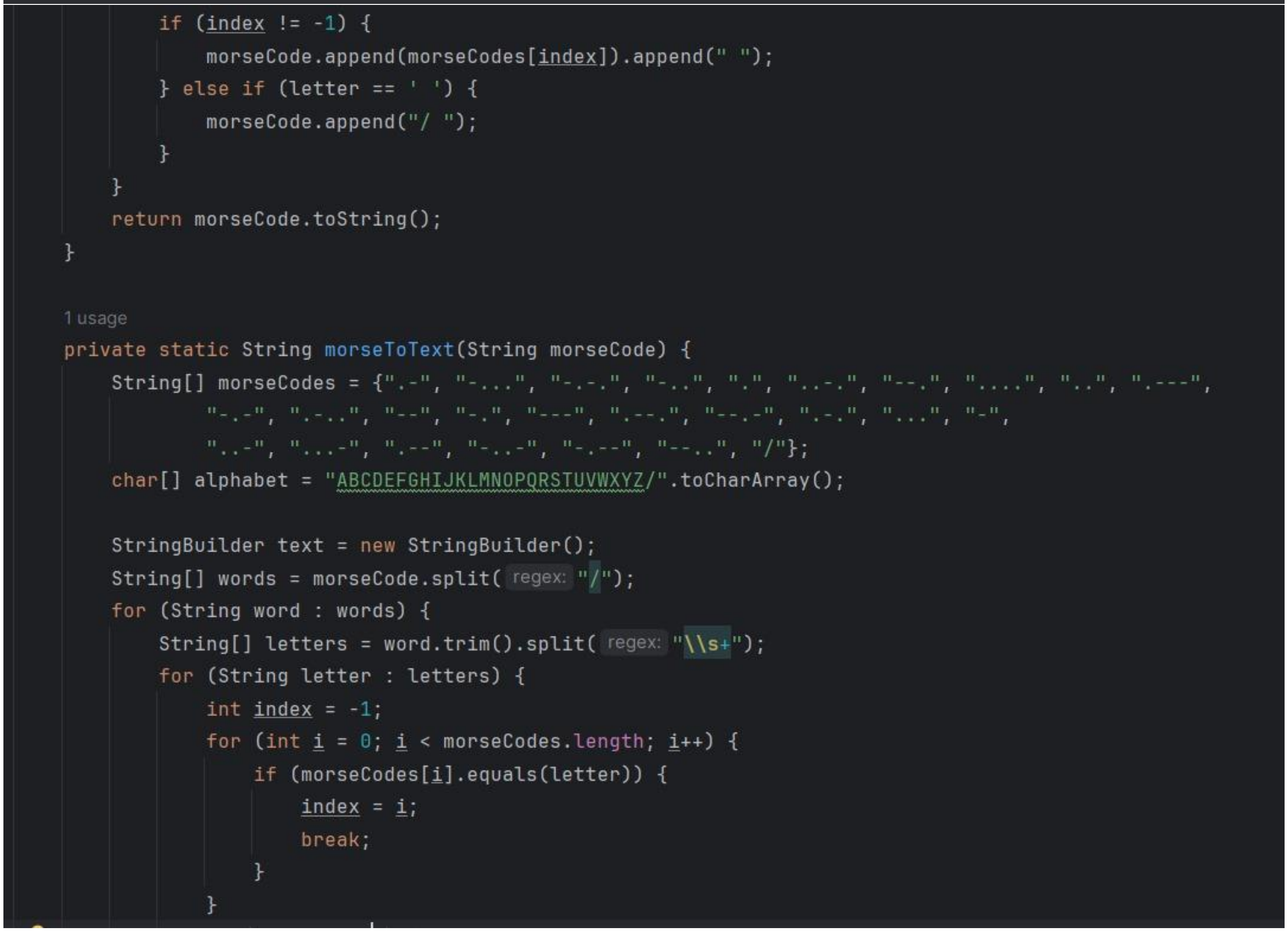
```java
private void convertMorseToText() {
    String morseCode = inputArea.getText();
    if (morseCode.isEmpty()) {
        JOptionPane.showMessageDialog(frame,  message: "Input Morse code cannot be empty.",  title: "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
    String text = morseToText(morseCode);
    outputArea.setText(text);
    history.add("Morse to Text: " + morseCode + " => " + text);
}

1 usage
private void saveTextToFile() {
    String text = outputArea.getText();
    if (text.isEmpty()) {
        JOptionPane.showMessageDialog(frame,  message: "Output text cannot be empty.",  title: "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
    saveToFile(text);
}
```

```java
private void generateQRCode() {
    String morseCode = outputArea.getText();
    if (morseCode.isEmpty()) {
        JOptionPane.showMessageDialog(frame,  message: "Output Morse code cannot be empty.",  title: "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
    try {
        generateQRCode(morseCode);
        JOptionPane.showMessageDialog(frame,  message: "QR code generated successfully.",  title: "Success", JOptionPane.INFORMATION_MESSAGE);
    } catch (Exception e) {
        JOptionPane.showMessageDialog(frame,  message: "Error generating QR code: " + e.getMessage(),  title: "Error", JOptionPane.ERROR_MESSAGE);
    }
}

1 usage
private void playMorseCode() {
    String morseCode = outputArea.getText();
    if (morseCode.isEmpty()) {
        JOptionPane.showMessageDialog(frame,  message: "Output Morse code cannot be empty.",  title: "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
    int volume = volumeSlider.getValue();
    int speed = speedSlider.getValue();
    playMorseCode(morseCode, speed,  pitch: 800, volume);
}
```

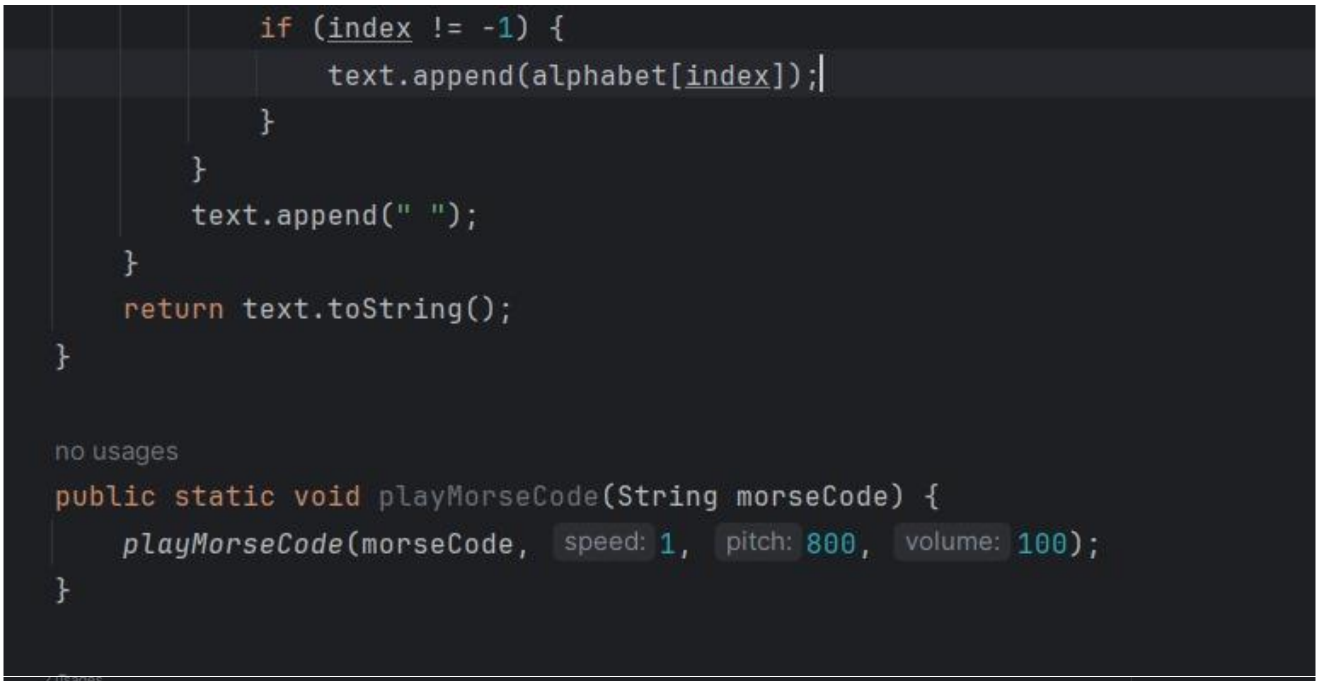
```java
private void viewHistory() {
    StringBuilder historyText = new StringBuilder();
    for (String entry : history) {
        historyText.append(entry).append("\n");
    }
    JOptionPane.showMessageDialog(frame, historyText.toString(),  title: "Conversion History", JOptionPane.INFORMATION_MESSAGE);
}

1 usage
private void saveToFile(String textToSave) {
    String desktopPath = System.getProperty("user.home") + "/Desktop/";
    String fileName = "converted_text.txt";
    String filePath = desktopPath + fileName;

    try (PrintWriter writer = new PrintWriter(filePath)) {
        writer.println(textToSave);
        JOptionPane.showMessageDialog(frame,  message: "Text saved to file: " + filePath,  title: "Success", JOptionPane.INFORMATION_MESSAGE);
    } catch (FileNotFoundException e) {
        JOptionPane.showMessageDialog(frame,  message: "Error saving text to file: " + e.getMessage(),  title: "Error", JOptionPane.ERROR_MESSAGE);
    }
}
```



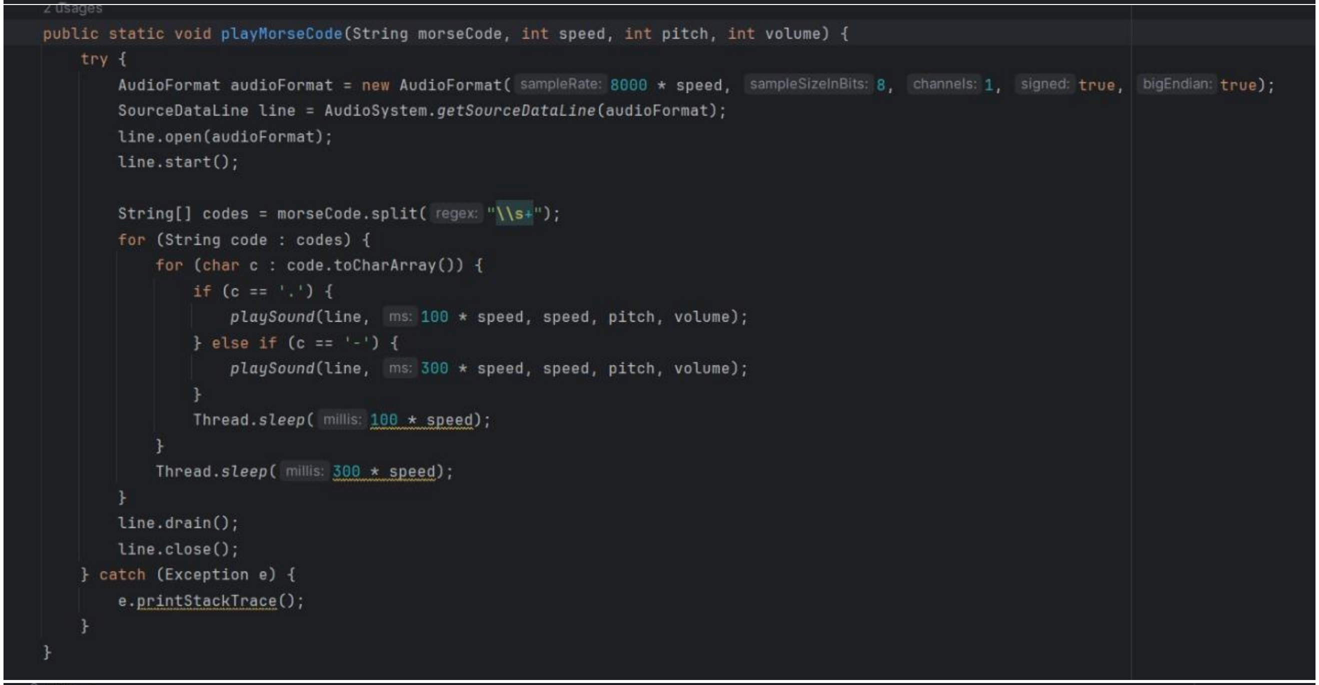
```java
private static String readFileContent(String filePath) throws FileNotFoundException {
    StringBuilder fileContent = new StringBuilder();
    try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
        String line;
        while ((line = reader.readLine()) != null) {
            fileContent.append(line).append("\n");
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
    return fileContent.toString();
}

1 usage
private static String textToMorse(String text) {
    String[] morseCodes = {".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....", "..", ".---",
            "-.-", ".-..", "--", "-.", "---", ".--.", "--.-", ".-.", "...", "-",
            "..-", "...-", ".--", "-..-", "-.--", "--..", "/"};
    char[] alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ/".toCharArray();

    text = text.toUpperCase();
    StringBuilder morseCode = new StringBuilder();
    for (char letter : text.toCharArray()) {
        int index = -1;
        for (int i = 0; i < alphabet.length; i++) {
            if (alphabet[i] == letter) {
                index = i;
                break;
            }
        }
```



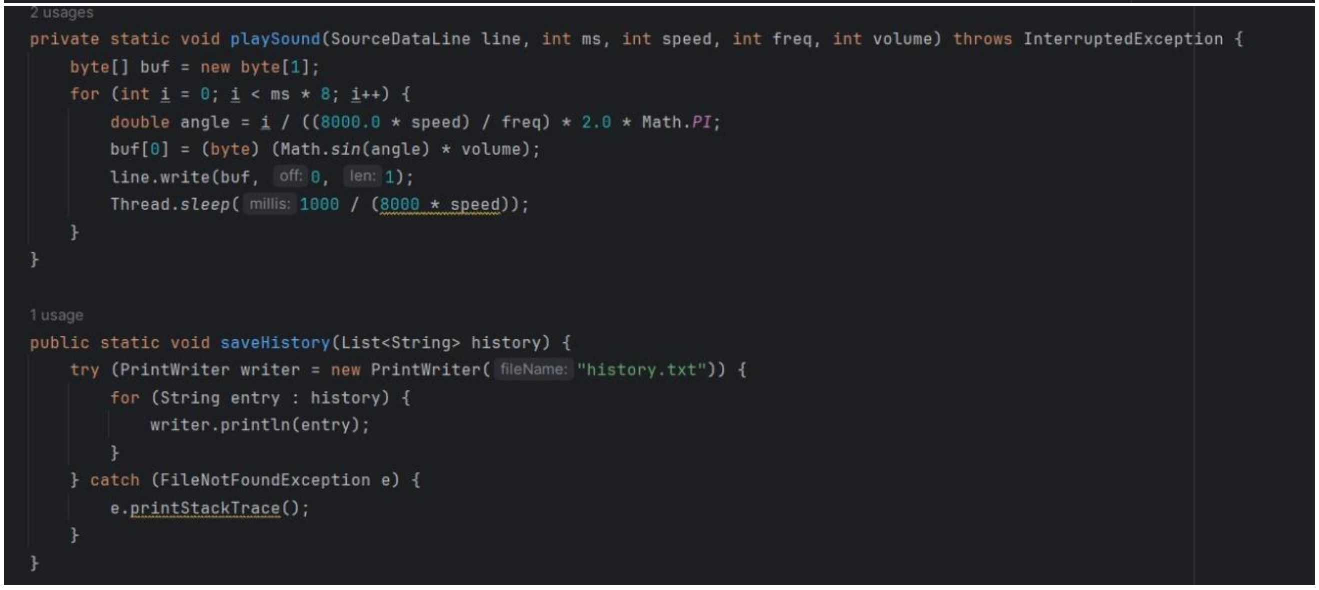
```java
        if (index != -1) {
            morseCode.append(morseCodes[index]).append(" ");
        } else if (letter == ' ') {
            morseCode.append("/ ");
        }
    }
    return morseCode.toString();
}

1 usage
private static String morseToText(String morseCode) {
    String[] morseCodes = {".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....", "..", ".---",
            "-.-", ".-..", "--", "-.", "---", ".--.", "--.-", ".-.", "...", "-",
            "..-", "...-", ".--", "-..-", "-.--", "--..", "/"};
    char[] alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ/".toCharArray();

    StringBuilder text = new StringBuilder();
    String[] words = morseCode.split( regex: "/");
    for (String word : words) {
        String[] letters = word.trim().split( regex: "\\s+");
        for (String letter : letters) {
            int index = -1;
            for (int i = 0; i < morseCodes.length; i++) {
                if (morseCodes[i].equals(letter)) {
                    index = i;
                    break;
                }
            }
```



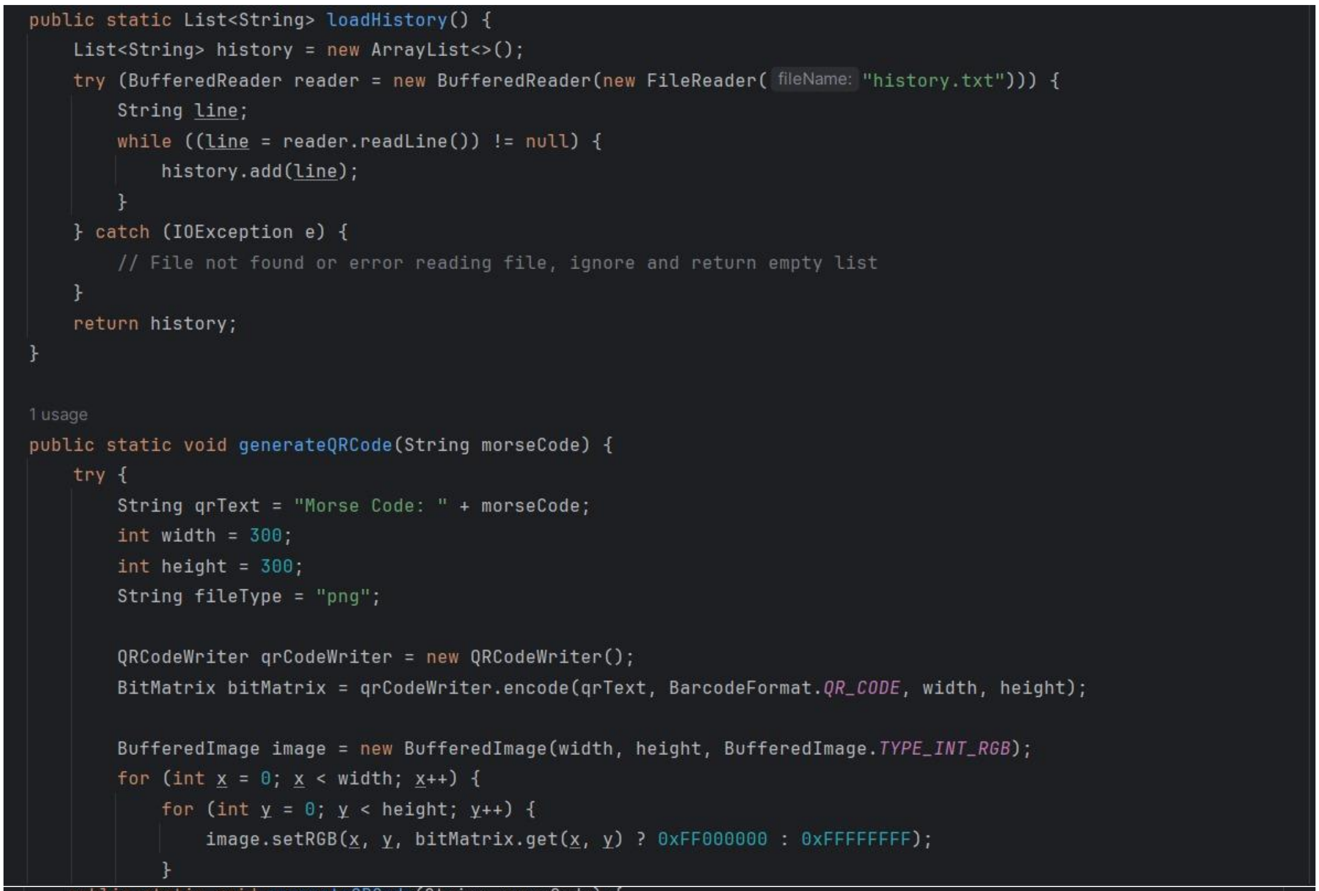
```
                if (index != -1) {
                    text.append(alphabet[index]);
                }
            }
            text.append(" ");
        }
        return text.toString();
    }

    no usages
    public static void playMorseCode(String morseCode) {
        playMorseCode(morseCode, speed: 1, pitch: 800, volume: 100);
    }
```



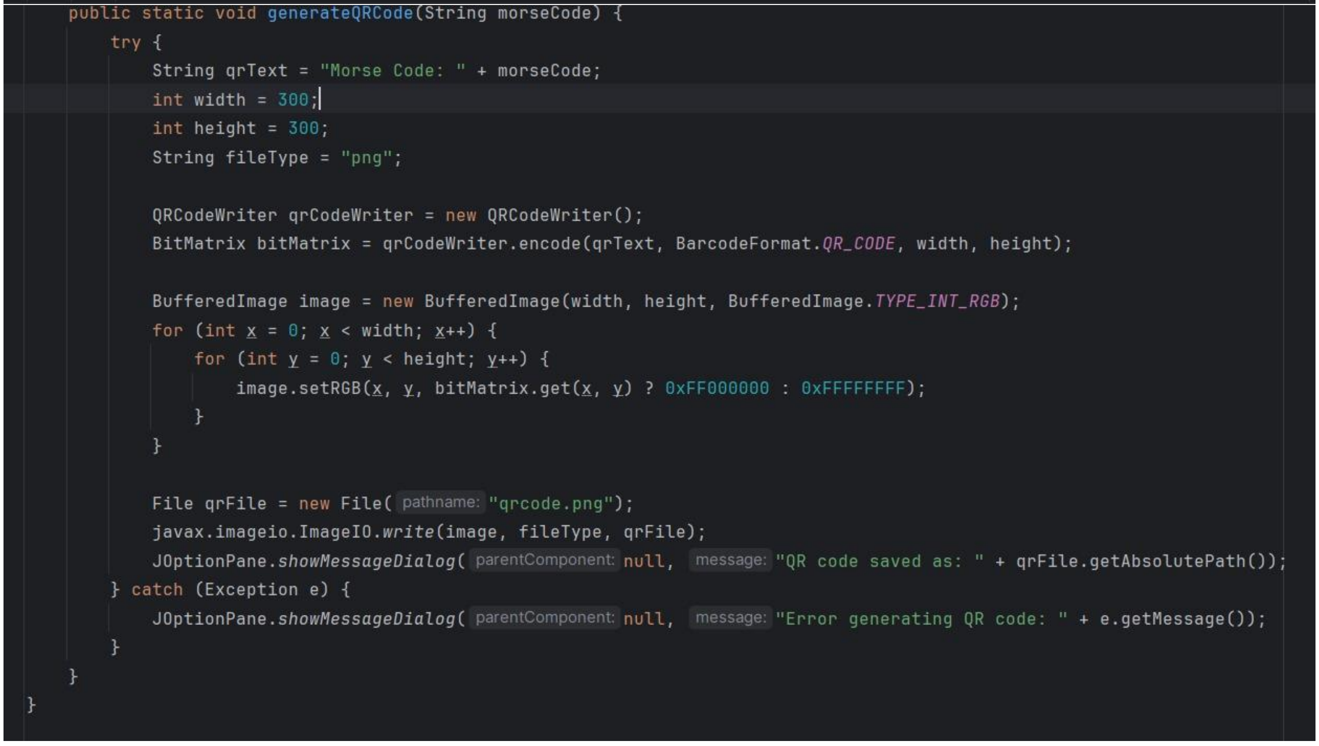
```
    2 usages
    public static void playMorseCode(String morseCode, int speed, int pitch, int volume) {
        try {
            AudioFormat audioFormat = new AudioFormat( sampleRate: 8000 * speed, sampleSizeInBits: 8, channels: 1, signed: true, bigEndian: true);
            SourceDataLine line = AudioSystem.getSourceDataLine(audioFormat);
            line.open(audioFormat);
            line.start();

            String[] codes = morseCode.split( regex: "\\s+");
            for (String code : codes) {
                for (char c : code.toCharArray()) {
                    if (c == '.') {
                        playSound(line, ms: 100 * speed, speed, pitch, volume);
                    } else if (c == '-') {
                        playSound(line, ms: 300 * speed, speed, pitch, volume);
                    }
                    Thread.sleep( millis: 100 * speed);
                }
                Thread.sleep( millis: 300 * speed);
            }
            line.drain();
            line.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```



```
2 usages
private static void playSound(SourceDataLine line, int ms, int speed, int freq, int volume) throws InterruptedException {
    byte[] buf = new byte[1];
    for (int i = 0; i < ms * 8; i++) {
        double angle = i / ((8000.0 * speed) / freq) * 2.0 * Math.PI;
        buf[0] = (byte) (Math.sin(angle) * volume);
        line.write(buf, off: 0, len: 1);
        Thread.sleep( millis: 1000 / (8000 * speed));
    }
}

1 usage
public static void saveHistory(List<String> history) {
    try (PrintWriter writer = new PrintWriter( fileName: "history.txt")) {
        for (String entry : history) {
            writer.println(entry);
        }
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
}
```



```java
public static List<String> loadHistory() {
    List<String> history = new ArrayList<>();
    try (BufferedReader reader = new BufferedReader(new FileReader( fileName: "history.txt"))) {
        String line;
        while ((line = reader.readLine()) != null) {
            history.add(line);
        }
    } catch (IOException e) {
        // File not found or error reading file, ignore and return empty list
    }
    return history;
}

1 usage
public static void generateQRCode(String morseCode) {
    try {
        String qrText = "Morse Code: " + morseCode;
        int width = 300;
        int height = 300;
        String fileType = "png";

        QRCodeWriter qrCodeWriter = new QRCodeWriter();
        BitMatrix bitMatrix = qrCodeWriter.encode(qrText, BarcodeFormat.QR_CODE, width, height);

        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        for (int x = 0; x < width; x++) {
            for (int y = 0; y < height; y++) {
                image.setRGB(x, y, bitMatrix.get(x, y) ? 0xFF000000 : 0xFFFFFFFF);
            }
```

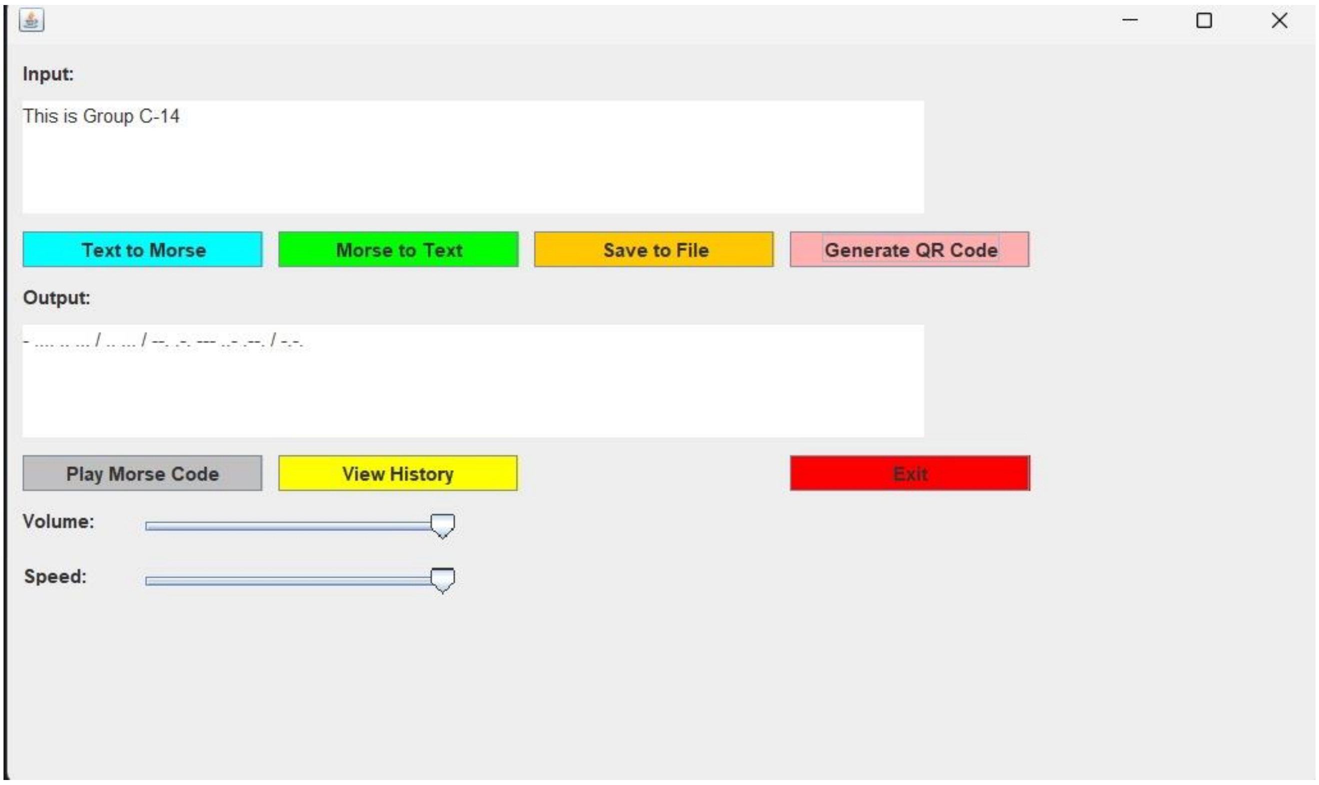


```java
public static void generateQRCode(String morseCode) {
    try {
        String qrText = "Morse Code: " + morseCode;
        int width = 300;
        int height = 300;
        String fileType = "png";

        QRCodeWriter qrCodeWriter = new QRCodeWriter();
        BitMatrix bitMatrix = qrCodeWriter.encode(qrText, BarcodeFormat.QR_CODE, width, height);

        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        for (int x = 0; x < width; x++) {
            for (int y = 0; y < height; y++) {
                image.setRGB(x, y, bitMatrix.get(x, y) ? 0xFF000000 : 0xFFFFFFFF);
            }
```

```java
        }

        File qrFile = new File( pathname: "qrcode.png");
        javax.imageio.ImageIO.write(image, fileType, qrFile);
        JOptionPane.showMessageDialog( parentComponent: null,  message: "QR code saved as: " + qrFile.getAbsolutePath());
    } catch (Exception e) {
        JOptionPane.showMessageDialog( parentComponent: null,  message: "Error generating QR code: " + e.getMessage());
    }
}
}
```

6.Results:

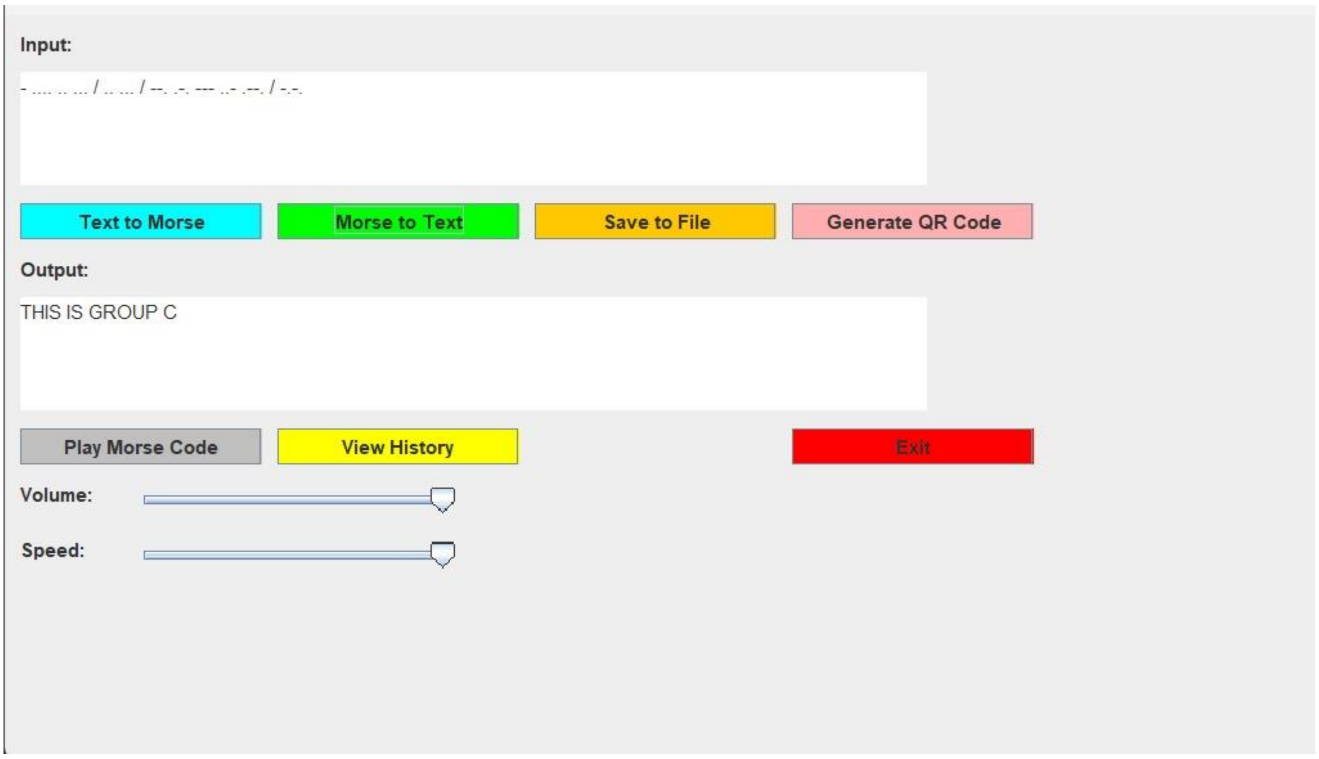
7.1.Text to morse



This is Group C-14

```
- .... .. ... / .. ... / --. .-. --- ..- .--. / -.-.
```

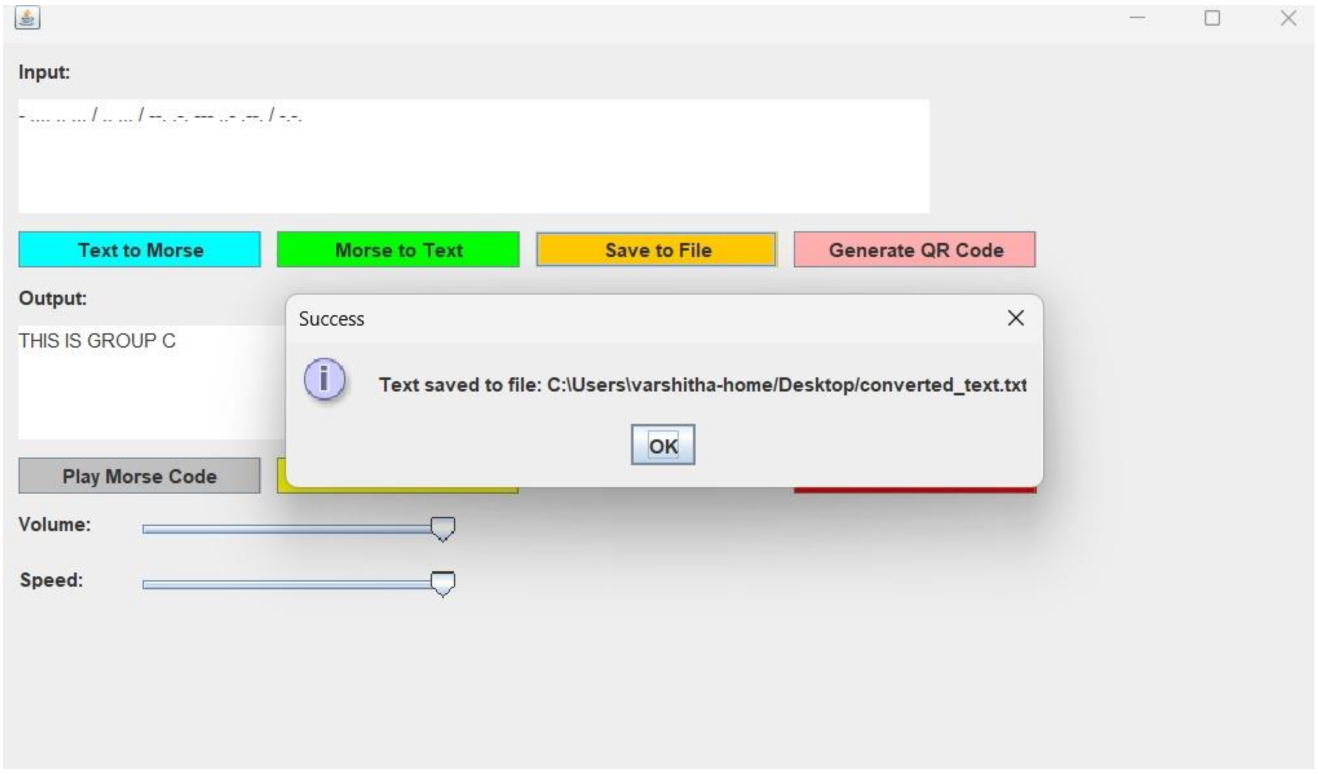
7.2.Morse to text



```
- .... .. ... / .. ... / --. .-. --- ..- .--. / -.-.
```

THIS IS GROUP C

7.3.Save the converted text



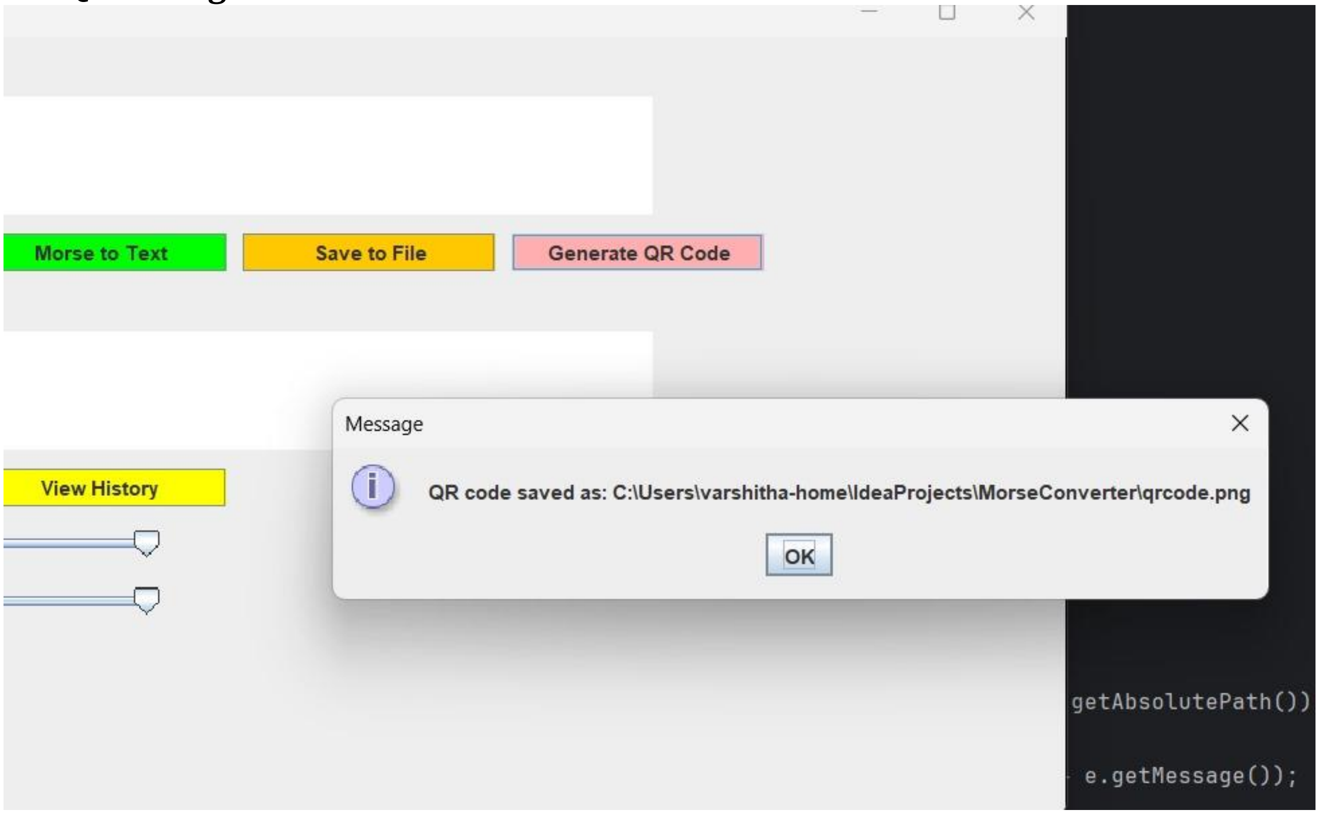
- .... / .. ... / --. .-. --- ..- .--. / -.-.

THIS IS GROUP C

Success

Text saved to file: C:\Users\varshitha-home/Desktop/converted_text.txt

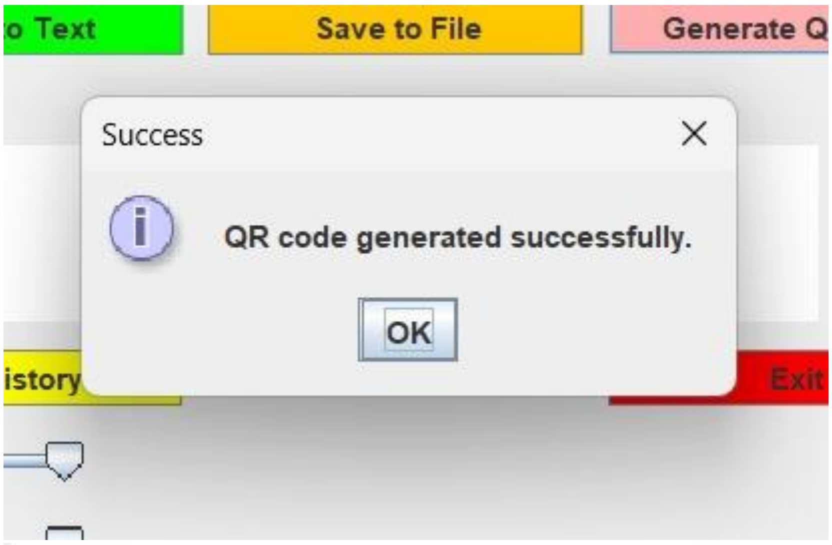
7.4.QR code generation



Message

QR code saved as: C:\Users\varshitha-home\IdeaProjects\MorseConverter\qrcode.png

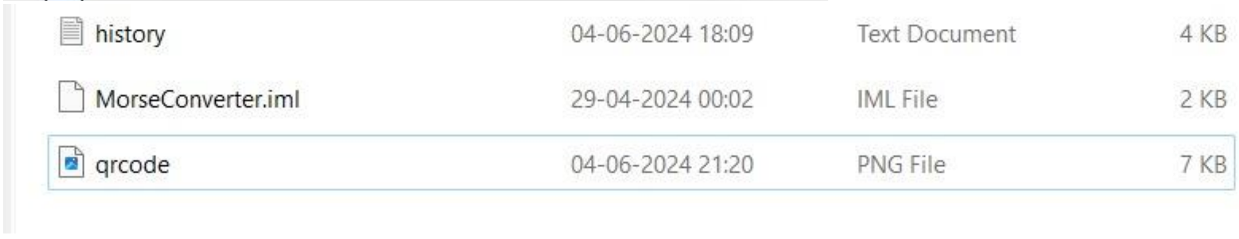
```
getAbsolutePath())
 e.getMessage());
```



Success

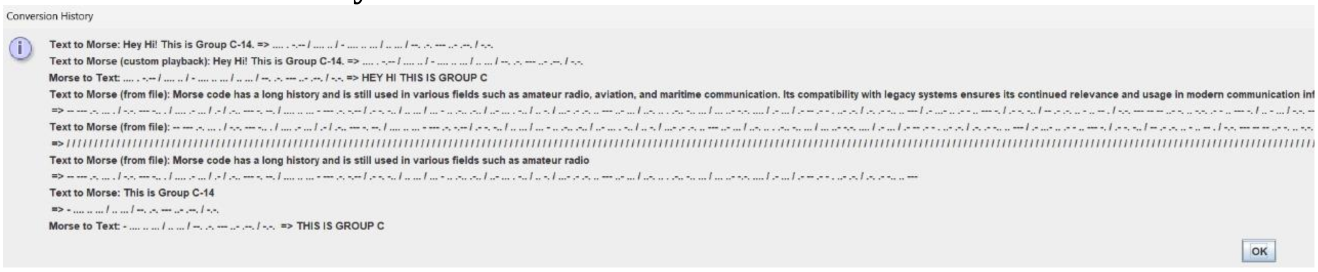
QR code generated successfully.

OK



history	04-06-2024 18:09	Text Document	4 KB
MorseConverter.iml	29-04-2024 00:02	IML File	2 KB
qrcode	04-06-2024 21:20	PNG File	7 KB

7.5.Conversion History



Conversion History

```
Text to Morse: Hey Hi! This is Group C-14. => .... . -.-- / .... .. / - .... .. ... / .. ... / --. .-. --- ..- .--. / -.-.
Text to Morse (custom playback): Hey Hi! This is Group C-14. => .... . -.-- / .... .. / - .... .. ... / .. ... / --. .-. --- ..- .--. / -.-.
Morse to Text: .... . -.-- / .... .. / - .... .. ... / .. ... / --. .-. --- ..- .--. / -.-. => HEY HI THIS IS GROUP C
Text to Morse (from file): Morse code has a long history and is still used in various fields such as amateur radio, aviation, and maritime communication. Its compatibility with legacy systems ensures its continued relevance and usage in modern communication inf
 => [illegible]
Text to Morse (from file): [illegible]
 => ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Text to Morse (from file): Morse code has a long history and is still used in various fields such as amateur radio
 => [illegible]
Text to Morse: This is Group C-14
 => - .... .. ... / .. ... / --. .-. --- ..- .--. / -.-.
Morse to Text: - .... .. ... / .. ... / --. .-. --- ..- .--. / -.-.  => THIS IS GROUP C
```

OK

7.Conclusion

The Morse Code Converter project with QR code generation is a full-featured Java application that provides a user-friendly graphical interface for converting text to Morse code, creating QR codes for Morse code, playing Morse code audio, and maintaining conversion history. The application makes use of several Java libraries, including Swing for the GUI, ZXing for QR code creation, and javax.sound for audio playback.
One of the project's primary features is the ability to convert text to Morse code and vice versa via the built-in textToMorse and morseToText functions. The application also allows users to save transformed text to a file on their desktop for future reference.

The project also has a QR code generator feature, which allows users to create a QR code image representing the Morse code output. This QR code can be used to share or store Morse code information in a compact, scannable format.
To improve the user experience, the application offers sliders for controlling the volume and speed of Morse code audio playback. Users can try out different settings to obtain the most pleasant audio experience.
Furthermore, the project keeps a conversion history, allowing users to evaluate prior conversions and QR code generation. This feature is particularly valuable for tracking and referencing previous work.

The Morse Code Converter project with QR code generation offers a user-friendly interface for text-to-Morse code conversion, QR code generation, and audio playback, making it suitable for a variety of use cases, including educational, hobby, and practical applications.

References:

[1] Nicolaou, N. and Georgiou, J., 2009. Towards a Morse code-based Non-invasive Thought-to-speech Converter. In Biomedical Engineering Systems and Technologies: International Joint Conference, BIOSTEC 2008 Funchal, Madeira, Portugal, January 28-31, 2008 Revised Selected Papers 1 (pp. 123-145). Springer Berlin Heidelberg.
[2]Begum, G., Devasena, L., AadhithYa, G., Arunachalesh, S. and Athreya, G., 2022, September. Morse Code-based Communication System for Partially Paralyzed People. In 2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA) (pp. 1301-1306). IEEE.
[3] Sachdeva, H., Misra, A., Chauhan, K. and Dave, M., 2020, February. Morse-Comm: mobile application environment for visually-impaired and neuro-muscular disabled people using morse code conversion. In 2020 7th International Conference on Signal Processing and Integrated Networks (SPIN) (pp. 809-813). IEEE.