

OOPs in Java Project Report

MorseWise: The Code Converter & 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:

Dr. Sruthi Guptha

Assistant Professor Department of Artificial Intelligence

Amrita Vishwa Vidyapeetham

Acknowledgment

We would like to express our sincere gratitude to our supervisor MS. Shruthi 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 conducive 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

<i>Sno</i>	<i>Topic</i>	<i>Page no.</i>
<i>1</i>	<i>Introduction</i>	<i>5</i>
<i>2</i>	<i>Problem statement</i>	<i>6</i>
<i>3</i>	<i>Objective and Motivation</i>	<i>7</i>
<i>4</i>	<i>System Design and Explanation</i>	<i>8</i>
<i>5</i>	<i>Advantages</i>	<i>11</i>
<i>6</i>	<i>System implementation with code</i>	<i>12</i>
<i>7</i>	<i>Results</i>	<i>21</i>

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.

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.

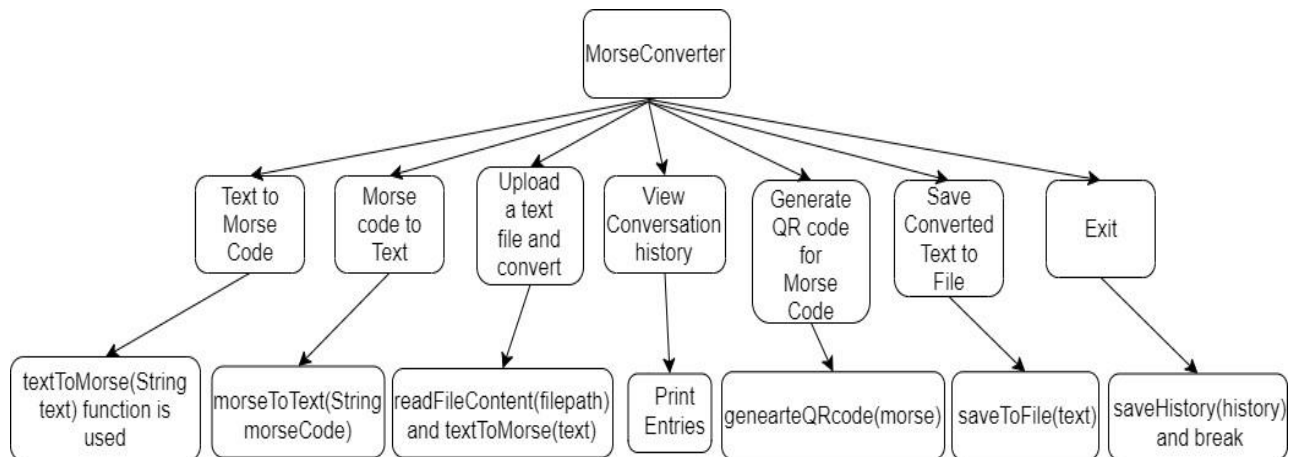
3.Objective and Motivation

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.

4. System Design with Implementation



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. File Upload and Conversion:

- Users can upload a text file and choose between normal or custom playback for conversion.
- The program reads the file content, performs the chosen conversion, and displays the result.

4.5. 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.

4.6. 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**).

4.7. Saving Converted Text to File:

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

4.8. 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.

4.9. 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**).

4.10. Error Handling:

- The program handles exceptions such as file not found, IO exceptions, and general exceptions gracefully by displaying appropriate error messages.

Summary:

- The **MorseWise** program provides a user-friendly interface for converting text to Morse code, Morse code to text, generating QR codes, and managing conversion history.
- It offers both normal and custom playback options, giving users flexibility in the conversion process.
- File upload functionality enhances usability, allowing users to convert text from files.
- The program ensures data integrity by saving conversion history to a file for future reference.

5. 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.

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.

6. System Implementation with code

6.1. Imports

```
1  import com.google.zxing.BarcodeFormat;
2  import com.google.zxing.common.BitMatrix;
3  import com.google.zxing.qrcode.QRCodeWriter;
4  import javax.sound.sampled.AudioFormat;
5  import javax.sound.sampled.AudioSystem;
6  import javax.sound.sampled.SourceDataLine;
7  import java.awt.image.BufferedImage;
8  import java.io.*;
9  import java.util.ArrayList;
10 import java.util.List;
11 import java.util.Scanner;
```

These are import statements in Java, which are used to import classes or packages from external libraries into your Java program:

1. **import com.google.zxing.BarcodeFormat:**

This imports the 'BarcodeFormat' class from the 'com.google.zxing' package. 'com.google.zxing' is a package from the ZXing (Zebra Crossing) library, which is a popular open-source barcode image processing library in Java

2. **import com.google.zxing.common.BitMatrix:**

This imports the 'BitMatrix' class from the 'com.google.zxing.common' package. 'BitMatrix' is a class in the ZXing library that represents a 2D matrix of bits, typically used for storing binary data for generating barcode images.

3. **import com.google.zxing.qrcode.QRCodeWriter:**

- This imports the 'QRCodeWriter' class from the 'com.google.zxing.qrcode' package. 'QRCodeWriter' is a class in the ZXing library specifically used for generating QR codes.

4. **import javax.sound.sampled.AudioFormat:**

- This imports the 'AudioFormat' class from the 'javax.sound.sampled' package. 'javax.sound.sampled' is a package in the Java Sound API, which provides functionality for working with audio data in Java programs. The

'AudioFormat' class represents an audio data format, including sample rate, sample size, channels, and encoding type.

5. import javax.sound.sampled.AudioSystem:

- This imports the 'AudioSystem' class from the 'javax.sound.sampled' package. 'AudioSystem' is a class in the Java Sound API that provides methods for accessing and controlling the audio system resources.

6. import javax.sound.sampled.SourceDataLine:

- This imports the 'SourceDataLine' class from the 'javax.sound.sampled' package. 'SourceDataLine' is a class in the Java Sound API representing a data line through which audio data can be played back. It is used for writing audio data to an output device, such as speakers or headphones.

7. import java.awt.image.BufferedImage:

- This imports the 'BufferedImage' class from the 'java.awt.image' package. 'BufferedImage' is a class in the Java API representing an image with an accessible buffer of image data. It is commonly used for working with image data in Java programs.

8. import java.io.*:

- This imports all classes in the 'java.io' package, which is the package in Java that provides classes for input and output operations. The '*' wildcard character is used to import all classes within the package.

9. import java.util.ArrayList:

- This imports the 'ArrayList' class from the 'java.util' package. 'ArrayList' is a class in Java representing a dynamic array that can grow or shrink in size dynamically. It is commonly used to store collections of objects.

10. import java.util.List:

- This imports the 'List' interface from the 'java.util' package. 'List' is an interface in Java representing an ordered collection of elements. It is the root interface of the Java Collections Framework's list hierarchy.

11. import java.util.Scanner:

- This imports the 'Scanner' class from the 'java.util' package. 'Scanner' is a class in Java used for obtaining user input from the console or other input

streams. It provides methods for parsing primitive types and strings from input streams.

6.2.Class Overview

Main Class:

Name: MorseConverter

Methods:

1.main(String[] args):

This is the main entry point of the program. It presents a menu to the user and allows them to choose various options for Morse code conversion, playback, file operations, viewing history, and generating QR codes.

2.saveToFile(String textToSave):

Saves the provided text to a file named "converted_text.txt" on the user's desktop.

3.readFileContent(String filePath):

Reads the content of a text file specified by the provided file path and returns it as a string.

4.textToMorse(String text):

Converts the given text to Morse code. It translates each character of the input text to its corresponding Morse code representation.

5.morseToText(String morseCode):

Converts the given Morse code to text. It translates each Morse code sequence back into its corresponding alphabetical character.

6.playMorseCode(String morseCode):

Plays the Morse code as audio. It uses a standard playback speed, pitch, and volume.

7.playMorseCode(String morseCode, int speed, int pitch, int volume):

Plays the Morse code as audio with custom playback options specified by the provided speed, pitch, and volume.

8.playSound(SourceDataLine line, int ms, int speed, int freq, int volume):

Plays a sound for a specified duration, frequency, and volume using the provided source data line.

9.saveHistory(List<String> history):

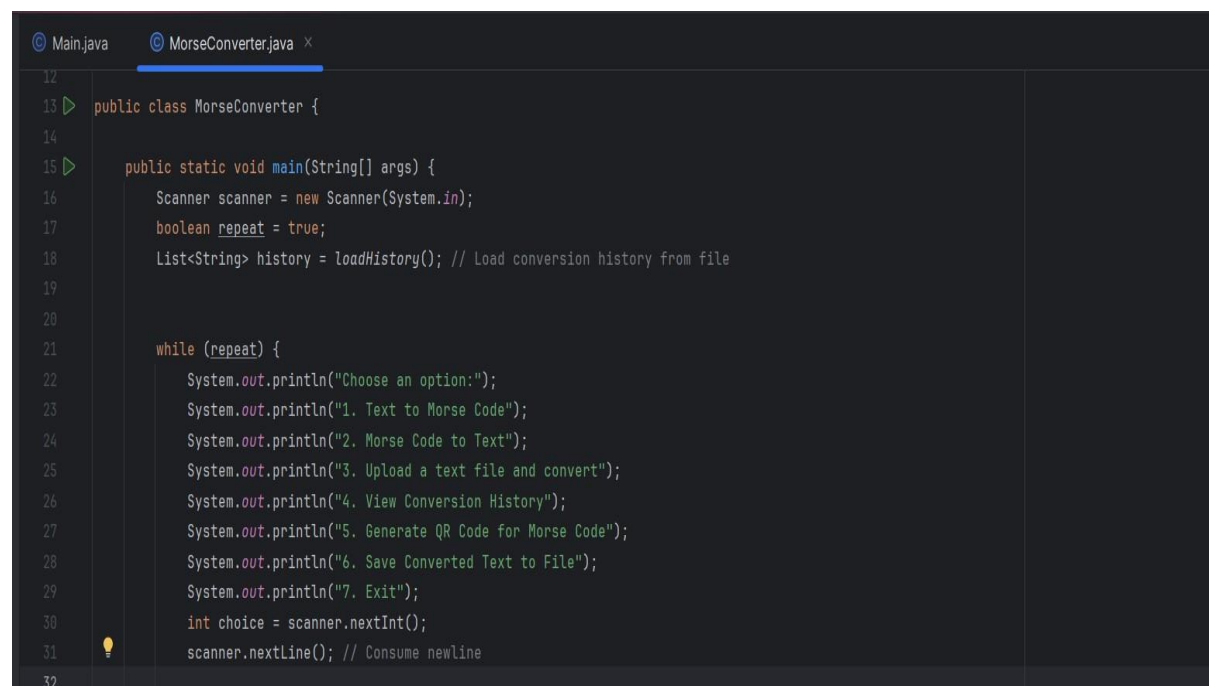
Saves the conversion history to a file named "history.txt".

10.loadHistory():

Loads the conversion history from the "history.txt" file and returns it as a list of strings.

11.generateQRCode(String morseCode):

Generates a QR code image representing the provided Morse code. The QR code contains the text "Morse Code: " followed by the Morse code string. The generated QR code image is saved as "qrcode.png".

6.2.1 Main Class:

```
12
13 public class MorseConverter {
14
15     public static void main(String[] args) {
16         Scanner scanner = new Scanner(System.in);
17         boolean repeat = true;
18         List<String> history = loadHistory(); // Load conversion history from file
19
20
21         while (repeat) {
22             System.out.println("Choose an option:");
23             System.out.println("1. Text to Morse Code");
24             System.out.println("2. Morse Code to Text");
25             System.out.println("3. Upload a text file and convert");
26             System.out.println("4. View Conversion History");
27             System.out.println("5. Generate QR Code for Morse Code");
28             System.out.println("6. Save Converted Text to File");
29             System.out.println("7. Exit");
30             int choice = scanner.nextInt();
31             scanner.nextLine(); // Consume newline
32
```



```

Main.java  MorseConverter.java x
32
33     switch (choice) {
34     case 1:
35         System.out.println("Enter the text to convert to Morse Code:");
36         String textToConvert = scanner.nextLine();
37         System.out.println("Choose the conversion option:");
38         System.out.println("1. Normal playback");
39         System.out.println("2. Custom playback");
40         int playbackOption = scanner.nextInt();
41         scanner.nextLine(); // Consume newline
42         if (playbackOption == 1) {
43             String morseCode = textToMorse(textToConvert);
44             System.out.println("Morse Code: " + morseCode);
45             playMorseCode(morseCode);
46             history.add("Text to Morse: " + textToConvert + " => " + morseCode);
47         } else if (playbackOption == 2) {
48             System.out.println("Enter speed (1-10): ");
49             int speed = scanner.nextInt();
50             System.out.println("Enter pitch (100-1000): ");
51             int pitch = scanner.nextInt();
52             System.out.println("Enter volume (1-100): ");
53             int volume = scanner.nextInt();
54             String morseCode = textToMorse(textToConvert);
55             System.out.println("Morse Code: " + morseCode);
56             playMorseCode(morseCode, speed, pitch, volume);
57             history.add("Text to Morse (custom playback): " + textToConvert + " => " + morseCode);
58         } else {
59             System.out.println("Invalid playback option.");
60         }
61         break;

```

```

Main.java  MorseConverter.java x
59     case 2:
60         System.out.println("Enter the Morse Code to convert to text:");
61         String morseToConvert = scanner.nextLine();
62         String text = morseToText(morseToConvert);
63         System.out.println("Text: " + text);
64         history.add("Morse to Text: " + morseToConvert + " => " + text);
65         break;

```

```

Main.java  MorseConverter.java x
66     case 3:
67         System.out.println("Enter the path to the text file:");
68         String filePath = scanner.nextLine();
69         try {
70             String fileContent = readFileContent(filePath);
71             System.out.println("File Content:\n" + fileContent);
72             System.out.println("Choose the conversion option:");
73             System.out.println("1. Normal playback");
74             System.out.println("2. Custom playback");
75             int conversionOption = scanner.nextInt();
76             scanner.nextLine(); // Consume newline
77             if (conversionOption == 1) {
78                 String morseCodeFromFile = textToMorse(fileContent);
79                 System.out.println("Morse Code: " + morseCodeFromFile);
80                 playMorseCode(morseCodeFromFile);
81                 history.add("Text to Morse (from file): " + fileContent + " => " + morseCodeFromFile);
82             } else if (conversionOption == 2) {
83                 System.out.println("Enter speed (1-10): ");
84                 int speed = scanner.nextInt();
85                 System.out.println("Enter pitch (100-1000): ");
86                 int pitch = scanner.nextInt();
87                 System.out.println("Enter volume (1-100): ");
88                 int volume = scanner.nextInt();
89                 String morseCodeFromFile = textToMorse(fileContent);
90                 System.out.println("Morse Code: " + morseCodeFromFile);
91                 playMorseCode(morseCodeFromFile, speed, pitch, volume);
92                 history.add("Text to Morse (custom playback, from file): " + fileContent + " => " + morseCodeFromFile);
93             } else {
94                 System.out.println("Invalid conversion option.");
95             }

```



```

        } catch (FileNotFoundException e) {
            System.out.println("File not found.");
        }
        break;
    case 4:
        System.out.println("Conversion History:");
        for (String entry : history) {
            System.out.println(entry);
        }
        break;
    case 5:
        System.out.println("Enter the Morse Code to generate QR code:");
        String morseCodeForQR = scanner.nextLine();
        generateQRCode(morseCodeForQR);
        break;

    case 6:
        System.out.println("Enter the text to save to file:");
        String textToSave = scanner.nextLine();
        saveToFile(textToSave);
        break;

    case 7:
        repeat = false;
        saveHistory(history); // Save conversion history to file before exiting
        break;
    default:
        System.out.println("Invalid option. Please choose again.");
        break;
    }
}

scanner.close();
}

```

```

private static 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);
        System.out.println("Text saved to file: " + filePath);
    } catch (FileNotFoundException e) {
        System.out.println("Error saving text to file: " + e.getMessage());
    }
}

```



```

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();
    }
}

```

```

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;
}

```

```

public static void generateQRCode(String morseCode) {
    try {
        // Encode the Morse code as text
        String qrText = "Morse Code: " + morseCode;
        // Set up QR code parameters
        int width = 300;
        int height = 300;
        String fileType = "png";
        // Create QR code writer
        QRCodeWriter qrCodeWriter = new QRCodeWriter();
        // Encode the QR code
        BitMatrix bitMatrix = qrCodeWriter.encode(qrText, BarcodeFormat.QR_CODE, width, height);
        // Convert BitMatrix to BufferedImage
        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);
            }
        }
        // Save QR code as PNG image
        File qrFile = new File( pathname: "qrcode.png");
        javax.imageio.ImageIO.write(image, fileType, qrFile);
        // Display QR code location
        System.out.println("QR code saved as: " + qrFile.getAbsolutePath());
    } catch (Exception e) {
        System.out.println("Error generating QR code: " + e.getMessage());
    }
}

```

7.Results

```
Choose an option:
1. Text to Morse Code
2. Morse Code to Text
3. Upload a text file and convert
4. View Conversion History
5. Generate QR Code for Morse Code
6. Save Converted Text to File
7. Exit
```

```
Choose an option:
1. Text to Morse Code
2. Morse Code to Text
3. Upload a text file and convert
4. View Conversion History
5. Generate QR Code for Morse Code
6. Save Converted Text to File
7. Exit
```

```
1
```

```
Enter the text to convert to Morse Code:
```

```
Hey Hi! This is Group C-14.
```

```
Choose the conversion option:
```

1. Normal playback
2. Custom playback

```
1
```

```
Morse Code: .... . -.- / .... .. / - .... .. ... / .. ... / -- .-. --- ... .-. / -.-.
```

```
Choose an option:
```

1. Text to Morse Code
2. Morse Code to Text
3. Upload a text file and convert
4. View Conversion History
5. Generate QR Code for Morse Code
6. Save Converted Text to File
7. Exit

```
1
```

```
Enter the text to convert to Morse Code:
```

```
Hey Hi! This is Group C-14.
```

```
Choose the conversion option:
```

1. Normal playback
2. Custom playback

```
2
```

```
Enter speed (1-10):
```

```
10
```

```
Enter pitch (100-1000):
```

```
200
```

```
Enter volume (1-100):
```

```
90
```

```
Morse Code: .... . -.- / .... .. / - .... .. ... / .. ... / -- .-. --- ... .-. / -.-.
```

Choose an option:

1. Text to Morse Code
2. Morse Code to Text
3. Upload a text file and convert
4. View Conversion History
5. Generate QR Code for Morse Code
6. Save Converted Text to File
7. Exit

Enter the Morse Code to convert to text:

Text: HEY HI THIS IS GROUP C

Choose an option:

1. Text to Morse Code
2. Morse Code to Text
3. Upload a text file and convert
4. View Conversion History
5. Generate QR Code for Morse Code
6. Save Converted Text to File
7. Exit

5

Enter the path to the text file:

C:\Users\varshitha-home\Desktop\grpc-14.txt

File Content:

Choose the conversion option:

1. Normal playback
2. Custom playback

1

Choose an option:

1. Text to Morse Code
2. Morse Code to Text
3. Upload a text file and convert
4. View Conversion History
5. Generate QR Code for Morse Code
6. Save Converted Text to File
7. Exit

Choose an option:

1. Text to Morse Code
2. Morse Code to Text
3. Upload a text file and convert
4. View Conversion History
5. Generate QR Code for Morse Code
6. Save Converted Text to File
7. Exit

4

Conversion History:

Text to Morse: Hey Hi! This is Group C-14. => , -.- / / - / / --. .-. --- ... ---. / -.-.

Choose an option:

1. Text to Morse Code
2. Morse Code to Text
3. Upload a text file and convert
4. View Conversion History
5. Generate QR Code for Morse Code
6. Save Converted Text to File
7. Exit

6

Enter the text to save to file:

Morse code has a long history and is still used in various fields such as amateur radio

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

Choose an option:

1. Text to Morse Code
2. Morse Code to Text
3. Upload a text file and convert
4. View Conversion History
5. Generate QR Code for Morse Code
6. Save Converted Text to File
7. Exit

3

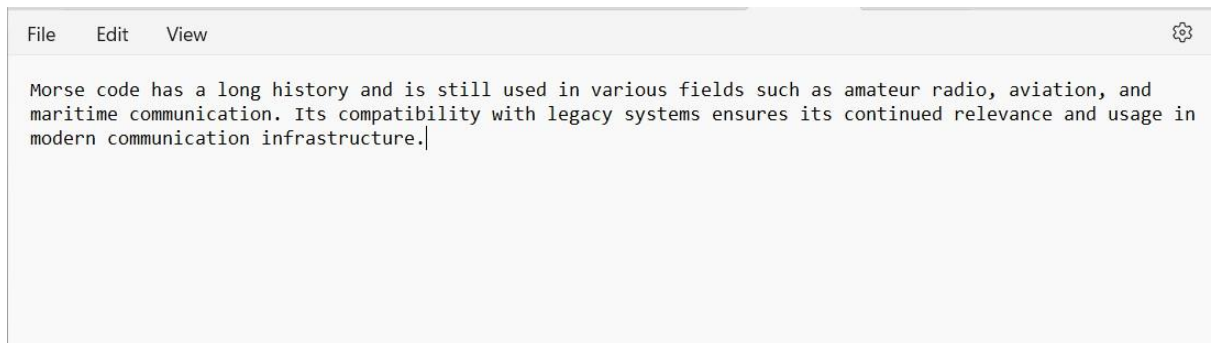
Enter the path to the text file:

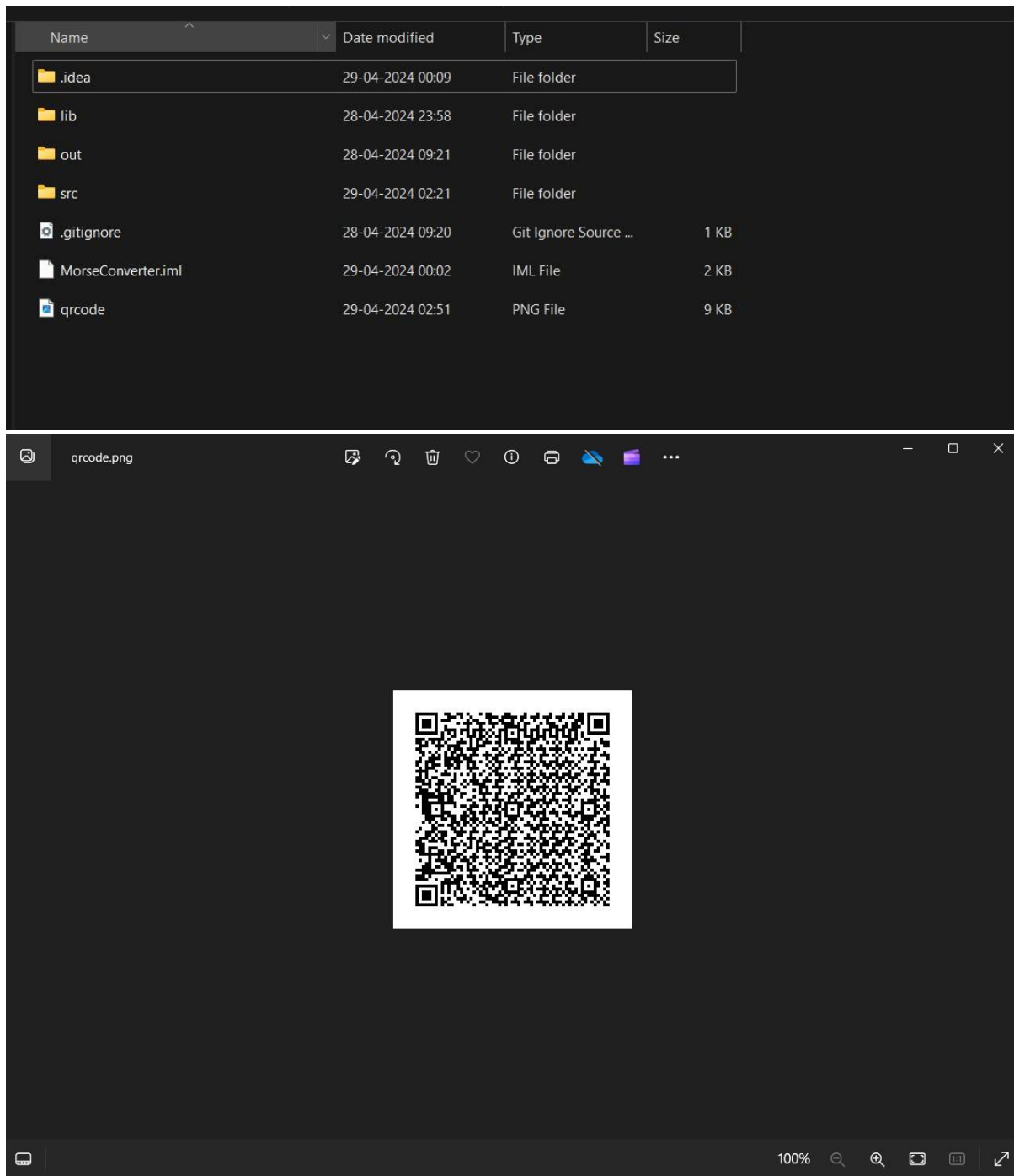
```
C:\Users\varshitha-home\Desktop\converted_text.txt
```

File Content:

Morse code has a long history and is still used in various fields such as amateur radio

OOPS report





```
Choose an option:
1. Text to Morse Code
2. Morse Code to Text
3. Upload a text file and convert
4. View Conversion History
5. Generate QR Code for Morse Code
6. Save Converted Text to File
7. Exit
7

Process finished with exit code 0
```