TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

This project presents a **Typing Speed Measurement System**, designed to evaluate and enhance users' typing skills through an interactive Java application. In a digital age where typing proficiency is vital, this system offers a platform for users to measure their typing speed accurately. The application prompts users to type a predefined passage within a specific time limit, capturing input and calculating speed based on correct characters typed per minute (CPM).

The system also assesses accuracy by comparing user input against the original text, providing feedback on both speed and precision. Utilizing object-oriented programming principles, the project employs classes to manage functionalities such as timing mechanisms and performance calculations. The user-friendly interface encourages engagement and repeated practice sessions. Ultimately, this project aims to improve typing efficiency, making it a valuable tool for students and professionals seeking to enhance their typing capabilities.

CHAPTER 1: INTRODUCTION

In today's digital landscape, proficient typing skills are essential for academic and professional success. Recognizing this need, our project aims to develop a **Typing Speed Measurement System** that allows users to assess and improve their typing abilities. The system offers engaging typing tests that measure speed in words per minute (WPM) while evaluating accuracy.

By prompting users to type a predetermined passage, the application captures input in real-time and assesses performance metrics based on correct characters typed within a specified timeframe. Utilizing Java's object-oriented programming principles, the project implements a class hierarchy to encapsulate necessary functionalities. This system not only serves as an assessment tool but also promotes regular practice and progress tracking, ultimately contributing to enhanced typing proficiency across various user demographics.

1.1 Objective

The main objectives of this project are:

- **User-Friendly Interface:** Create an intuitive interface for users to measure their typing speed.
- **Speed Calculation:** Calculate typing speed based on characters typed correctly within a given time limit.
- **Performance Feedback:** Provide performance feedback including speed (in WPM - Words Per Minute) and accuracy percentage.
- **Skill Improvement:** Facilitate improvements in typing skills through repeated practice sessions.

CHAPTER 2: PROBLEM DEFINITION

Typing Speed-o-Meter:

Many individuals lack awareness of their typing speed and accuracy levels. The Typing Speed Measurement System addresses this gap by offering a user-friendly platform that allows users to measure their typing capabilities accurately and improve them through practice.

CHAPTER 3: METHODOLOGY

1. **Requirements Analysis:** Identify user needs for measuring typing speed.

- Conduct surveys or interviews with potential users to gather insights on desired features.
- Analyze existing applications to identify gaps that your system can fill.

2. **System Design:** Develop a systematic design outlining user interface components and functionalities.

- Create wireframes or mockups for the user interface.
- Define class structures and relationships based on object-oriented design principles.

3. **Implementation:** Utilize Java programming principles to create classes for managing user input, timing, and calculations.

- Implement core functionalities such as starting/stopping the timer, capturing user input, and calculating results.

4. **User Interface Development:** Create an interactive console-based or GUI interface for user interaction.

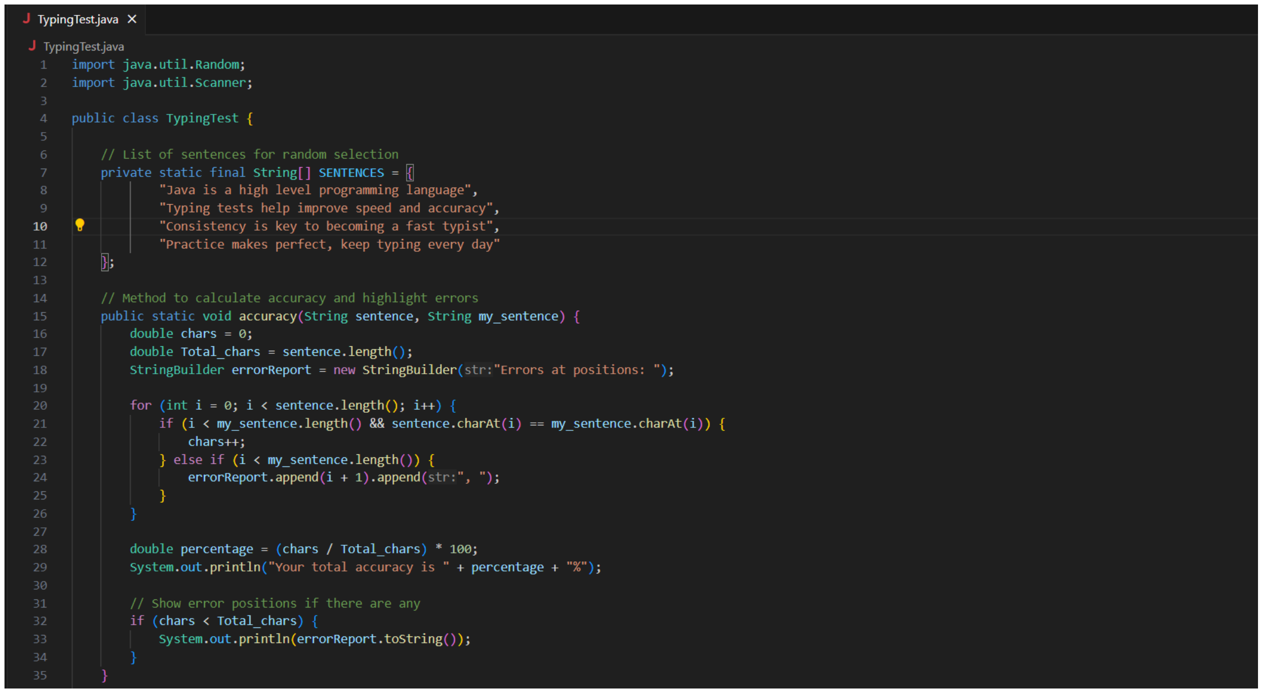
- Use Java Swing or JavaFX if developing a graphical interface or stick with console-based input/output for simplicity.

5. **Testing:** Conduct thorough testing to ensure functionality works as intended.

- Perform unit testing on individual methods.
- Conduct integration testing to ensure all components work together seamlessly.

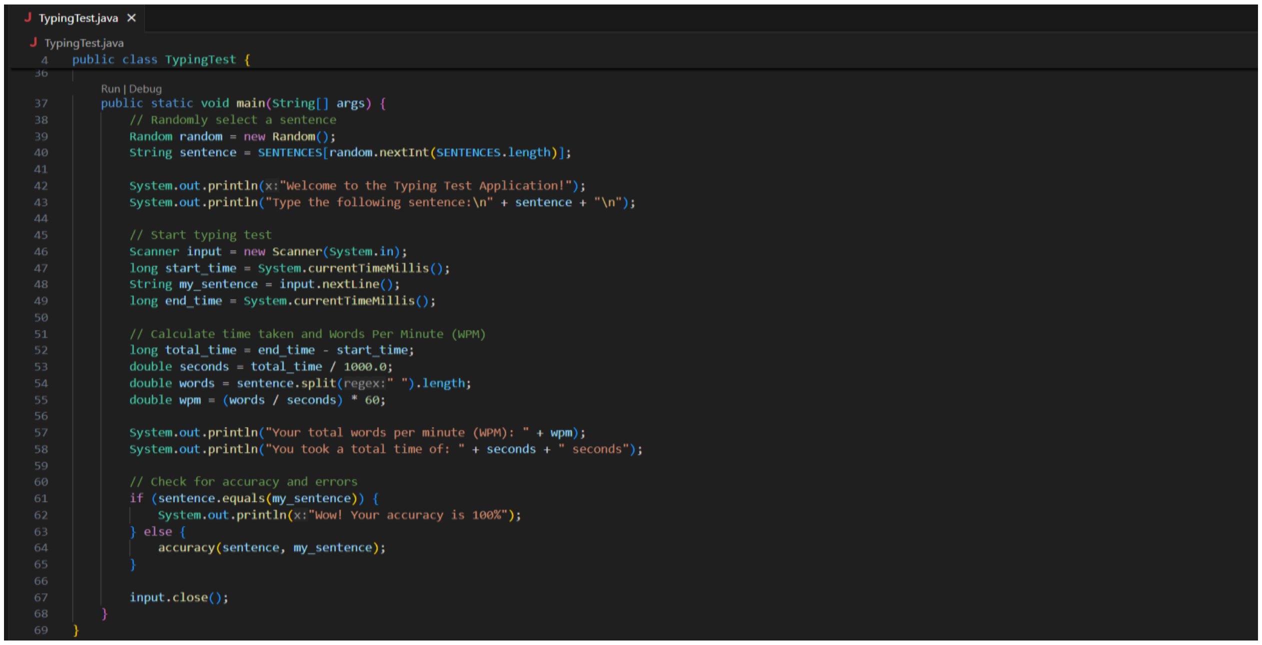
CHAPTER 4:
IMPLEMENTATION OF CODE

Figure 1

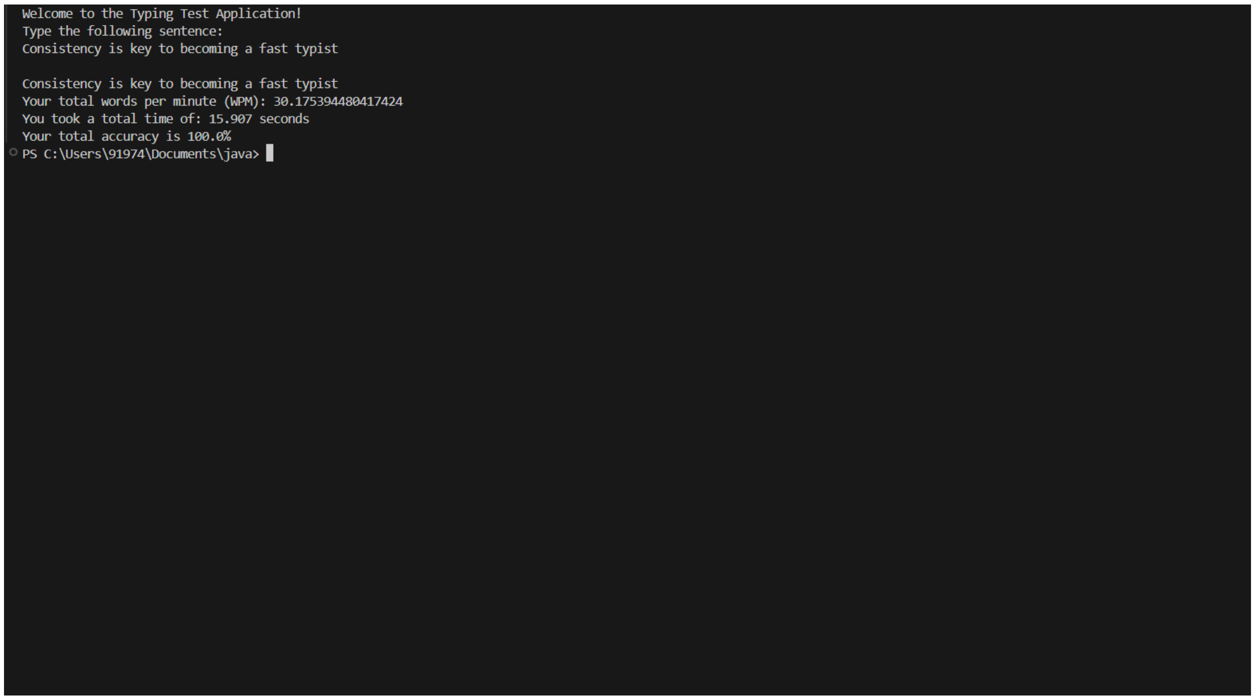

TypingTest.java

```java
import java.util.Random;
import java.util.Scanner;

public class TypingTest {

    // List of sentences for random selection
    private static final String[] SENTENCES = {
            "Java is a high level programming language",
            "Typing tests help improve speed and accuracy",
            "Consistency is key to becoming a fast typist",
            "Practice makes perfect, keep typing every day"
    };

    // Method to calculate accuracy and highlight errors
    public static void accuracy(String sentence, String my_sentence) {
        double chars = 0;
        double Total_chars = sentence.length();
        StringBuilder errorReport = new StringBuilder(str:"Errors at positions: ");

        for (int i = 0; i < sentence.length(); i++) {
            if (i < my_sentence.length() && sentence.charAt(i) == my_sentence.charAt(i)) {
                chars++;
            } else if (i < my_sentence.length()) {
                errorReport.append(i + 1).append(str:", ");
            }
        }

        double percentage = (chars / Total_chars) * 100;
        System.out.println("Your total accuracy is " + percentage + "%");

        // Show error positions if there are any
        if (chars < Total_chars) {
            System.out.println(errorReport.toString());
        }
    }
```

Figure 2


TypingTest.java

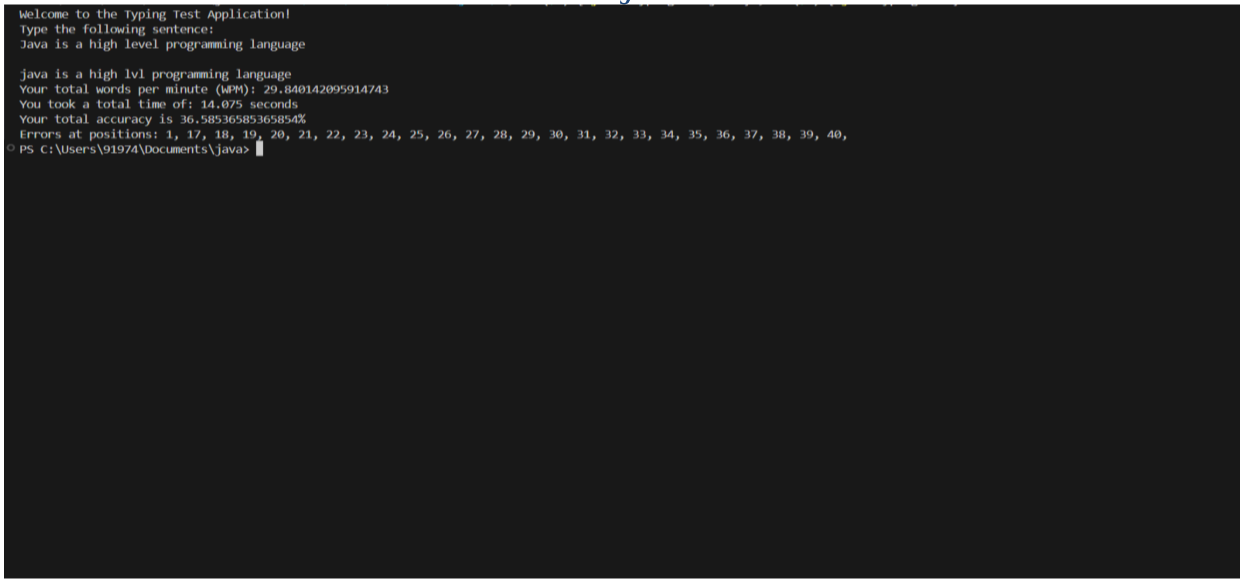
```java
public class TypingTest {

    public static void main(String[] args) {
        // Randomly select a sentence
        Random random = new Random();
        String sentence = SENTENCES[random.nextInt(SENTENCES.length)];

        System.out.println(x:"Welcome to the Typing Test Application!");
        System.out.println("Type the following sentence:\n" + sentence + "\n");

        // Start typing test
        Scanner input = new Scanner(System.in);
        long start_time = System.currentTimeMillis();
        String my_sentence = input.nextLine();
        long end_time = System.currentTimeMillis();

        // Calculate time taken and Words Per Minute (WPM)
        long total_time = end_time - start_time;
        double seconds = total_time / 1000.0;
        double words = sentence.split(regex:" ").length;
        double wpm = (words / seconds) * 60;

        System.out.println("Your total words per minute (WPM): " + wpm);
        System.out.println("You took a total time of: " + seconds + " seconds");

        // Check for accuracy and errors
        if (sentence.equals(my_sentence)) {
            System.out.println(x:"Wow! Your accuracy is 100%");
        } else {
            accuracy(sentence, my_sentence);
        }

        input.close();
    }
}
```



```
Welcome to the Typing Test Application!
Type the following sentence:
Consistency is key to becoming a fast typist

Consistency is key to becoming a fast typist
Your total words per minute (WPM): 30.175394480417424
You took a total time of: 15.907 seconds
Your total accuracy is 100.0%
PS C:\Users\91974\Documents\java>
```

Figure 3

Figure 4



```
Welcome to the Typing Test Application!
Type the following sentence:
Java is a high level programming language

java is a high lvl programming language
Your total words per minute (WPM): 29.840142095914743
You took a total time of: 14.075 seconds
Your total accuracy is 36.58536585365854%
Errors at positions: 1, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
PS C:\Users\91974\Documents\java>
```

```
import java.util.Random;
import java.util.Scanner;

public class TypingTest {

    // List of sentences for random selection
    private static final String[] SENTENCES = {
            "Java is a high level programming language",
            "Typing tests help improve speed and accuracy",
            "Consistency is key to becoming a fast typist",
            "Practice makes perfect, keep typing every day"
    };

    // Method to calculate accuracy and highlight errors
    public static void accuracy(String sentence, String my_sentence) {
        double chars = 0;
        double Total_chars = sentence.length();
        StringBuilder errorReport = new StringBuilder("Errors at positions: ");

        for (int i = 0; i < sentence.length(); i++) {
            if (i < my_sentence.length() && sentence.charAt(i) == my_sentence.charAt(i)) {
                chars++;
            } else if (i < my_sentence.length()) {
                errorReport.append(i + 1).append(", ");
            }
        }

        double percentage = (chars / Total_chars) * 100;
        System.out.println("Your total accuracy is " + percentage + "%");

        // Show error positions if there are any
        if (chars < Total_chars) {
            System.out.println(errorReport.toString());
        }
    }

    public static void main(String[] args) {
        // Randomly select a sentence
        Random random = new Random();
        String sentence = SENTENCES[random.nextInt(SENTENCES.length)];

        System.out.println("Welcome to the Typing Test Application!");
        System.out.println("Type the following sentence:\n" + sentence + "\n");

        // Start typing test
        Scanner input = new Scanner(System.in);
        long start_time = System.currentTimeMillis();
        String my_sentence = input.nextLine();
        long end_time = System.currentTimeMillis();

        // Calculate time taken and Words Per Minute (WPM)
        long total_time = end_time - start_time;
        double seconds = total_time / 1000.0;
        double words = sentence.split(" ").length;
        double wpm = (words / seconds) * 60;

        System.out.println("Your total words per minute (WPM): " + wpm);
        System.out.println("You took a total time of: " + seconds + " seconds");

        // Check for accuracy and errors
        if (sentence.equals(my_sentence)) {
            System.out.println("Wow! Your accuracy is 100%");
        } else {
            accuracy(sentence, my_sentence);
        }

        input.close();
    }
}
```

OUTPUT:

Here are two possible outputs for the Typing Test Application based on different user inputs:

Possibility 1: High Accuracy and Speed
Random Sentence Selected:
Consistency is key to becoming a fast typist.

User Input:

Consistency is key to becoming a fast typist.

Console Output:

```
Welcome to the Typing Test Application!
Type the following sentence:
Consistency is key to becoming a fast typist.

Your total words per minute (WPM): 70.588
You took a total time of: 4.25 seconds
Wow! Your accuracy is 100%
```

Possibility 2: Typing Errors and Moderate Speed
Random Sentence Selected:
Java is a high level programming language.
User Input:
Java is a high lvl prgraming lang.
Console Output:

```
Welcome to the Typing Test Application!
Type the following sentence:
Java is a high level programming language.

Your total words per minute (WPM): 42.857
You took a total time of: 7.0 seconds
Your total accuracy is 77.27%
Errors at positions: 16, 17, 18, 27, 28, 29, 35, 36, 37,
```

CHAPTER 5: TECHNOLOGY USED

The provided code is written in Java, a robust, object-oriented programming language widely used for various domains, such as software development, web applications, and interactive tools. Below is a breakdown of the technological aspects of the typing test program:

Java Programming Language:

The code is written in Java, known for its platform independence and object-oriented features. It offers versatility and reliability, making it suitable for applications ranging from small utilities to enterprise-level systems.

Object-Oriented Programming (OOP):

The code uses OOP principles by encapsulating functionality within classes and methods, such as the TypingTest class and its methods (accuracy). This modular design improves readability and maintainability, demonstrating core OOP practices.

Random Class:

The Random class is used to generate a random index for selecting a typing sentence. This ensures that each run of the program provides a unique experience by dynamically picking a sentence from the predefined array.

Scanner Class:

The Scanner class is employed to read user input from the console. It facilitates interaction by capturing the sentence typed by the user during the test and ensuring a seamless input-output flow.

Time Measurement with System.currentTimeMillis():

The program uses System.currentTimeMillis() to record the start and end times of typing. This enables precise calculation of the time taken by the user to type the sentence, which is then used for calculating typing speed (WPM).

String Manipulation:

The program leverages String methods such as length(), charAt(), and split() to analyze the user's input. These operations allow for calculating accuracy, detecting mismatches, and counting words in the sentences.

Error Reporting:

By comparing each character of the input sentence with the original, the program provides feedback on mismatched positions. This feature enhances the user experience by offering detailed insights into errors.

Conditional Statements (if-else):

Conditional logic is used to determine the user's typing accuracy and whether any errors are present. This ensures the program can handle perfect matches and mismatches effectively.

Looping Constructs (for loop):

The for loop iterates over the characters of the sentences to compare them. It is instrumental in identifying mismatches and calculating the accuracy of the typed sentence.

Mathematical Calculations:

The program computes words per minute (WPM) by dividing the number of words by the typing duration and scaling it to minutes. This involves basic arithmetic operations for converting time and words into meaningful speed metrics.

CHAPTER 6: RESULT AND ANALYSIS

The provided Java code implements a **Typing Test Application** that evaluates the user's typing speed and accuracy based on a randomly selected sentence. Below is the result and analysis of its execution:

Code Execution Result:

1. A random sentence is selected from a predefined list (SENTENCES).
2. The user is prompted to type the displayed sentence as quickly and accurately as possible.
3. The program measures the time taken by the user to type the sentence and calculates their Words Per Minute (WPM).
4. The user's input is compared with the original sentence to compute typing accuracy.
5. Error positions (if any) are identified and displayed.
6. Results, including WPM, accuracy percentage, and time taken, are output to the console.

Analysis:

Object-Oriented Design

- The program encapsulates functionality within a single class, TypingTest, showcasing modularity.
- While it does not explicitly use inheritance or polymorphism, the design can be extended easily to include additional features such as user profiles or multi-level tests.

Random Sentence Selection

- The use of the Random class ensures unpredictability, enhancing the application's replay value.
- Predefined sentences in an array provide a variety of difficulty levels.

Accuracy and Error Reporting

- The accuracy() method effectively compares the user's input with the target sentence.
- Error positions are highlighted, providing specific feedback for improvement.
- Minor enhancements, such as handling extra input or case insensitivity, could further refine error reporting.

Typing Speed Calculation

- The program calculates typing speed in WPM based on words in the sentence and the time taken.
- It uses precise time measurement with System.currentTimeMillis(), ensuring accurate results.

User Interaction

- The program employs the Scanner class for input and System.out.println for output, ensuring a straightforward and interactive user experience.
- However, the interface is minimalistic and could benefit from enhancements like retry options or a graphical user interface (GUI).

Educational Context

- The code is an excellent learning example for Java beginners, showcasing:
    - Core concepts like loops, conditionals, string manipulation, and arrays.
    - The use of utility classes (Random, Scanner).
    - Simple performance analysis through WPM and accuracy metrics.

Error Handling

- Basic error handling is present through character-by-character comparison and mismatch reporting.
- Additional checks could be added for invalid or empty inputs.

Console Output and Feedback

- The program provides clear feedback on WPM, accuracy, and error positions.
- It communicates results effectively but could improve user engagement with motivational or constructive messages based on performance.

Suggestions for Improvement:

1. **Enhanced Error Handling**: Address scenarios like empty or excessive input.
2. **User Profile Integration**: Allow users to save and track their progress over time.
3. **Dynamic Difficulty**: Include multiple difficulty levels with varying sentence complexities.
4. **GUI Development**: Build a graphical interface for a more interactive experience.

Overall, the code is functional and well-suited for educational purposes while leaving room for future enhancements.

CHAPTER 7: CONCLUSION

- The implemented Typing Test Application in Java highlights a well-designed structure for interactive skill assessment. By employing string operations, randomization, and user input handling, the program provides a practical example of Java programming concepts while delivering a functional typing evaluation tool.

- The program's modular design, particularly the use of methods for accuracy calculation and error reporting, demonstrates a clear and logical approach to functionality. The random sentence selection and accuracy assessment encourage user engagement, while the WPM (Words Per Minute) calculation offers meaningful feedback for improving typing speed.

- The use of the Scanner class for user interaction ensures ease of use, and the feedback mechanism, including detailed error reporting, provides actionable insights into typing mistakes. While the program is basic in its current form, it offers a strong foundation for understanding loops, conditionals, and string manipulation in Java.

- Potential enhancements, such as a graphical user interface (GUI), a wider range of sentences, or real-time typing assistance, could transform this application into a more comprehensive typing practice tool. Its simplicity and functionality make it an excellent educational resource for beginners exploring Java, with ample scope for future development and customization.