University of Science and Technology Chittagong

Department of Computer Science and Engineering

Batch – 43(A)

Lab Project : Word Scramble Game

Course Name: Object Oriented Programming

Course Code: CSE 124

Submitted by

Name: Sidratul Muntaha

Roll No: 24070112

Submitted to

Debabrata Mallick

Lecturer, Department of Computer Science & Engineering , University of Science and Technology Chittagong

Date of Submission: 22 / 05 /2025

Remarks:

Signature of Course Teacher:

Table Of Contents:

Word Scramble Game

List Of Figures

Introduction

WordScramble: A Java-Based Word Puzzle Game

WordScramble is a Java-based interactive word puzzle game designed to offer an engaging and educational experience for players. Traditional console-based games often lack structure and maintainability, making them difficult to upgrade or scale.

To overcome these limitations, WordScramble is built using **Object-Oriented Programming (OOP)** principles, ensuring a modular and extensible codebase. The system models essential components such as players, words, timer, and scoring logic through classes and objects. It features a **Swing-based GUI**, **real-time countdown timer**, **player-vs-player mode**, and **score tracking**, providing a fun yet structured gameplay experience.

This project demonstrates how OOP concepts can be applied to create well-structured Java applications while combining learning and entertainment in one platform.

1.Proposal

1.1 Objective

The goal of this project is to create a Java Swing-based Word Scramble Game that allows two players to unscramble words within a time limit. It combines logic, language, and time-based competition to enhance vocabulary skills in a fun way.

1.2 Scope

The game supports 3 difficulty levels (Easy, Medium, Hard), tracks scores and times, supports multiplayer (2 players), and uses GUI elements like text fields, labels, buttons, and timers to provide interactive gameplay.

1.3 Problem Statement

While many word games exist, few provide a timed, level-based, GUI-supported experience for competitive two-player interaction. This project addresses that gap using Java's GUI toolkit Swing.

1.4 Methodology

- Java Swing is used to build the GUI.
- Game logic uses OOP principles and javax.swing.Timer for countdown.
- Words are scrambled using Collections.shuffle().
- Code is structured around a single JFrame-based class for simplicity and clarity.

2. Implementation

2.1 Class Diagram

Class Diagram

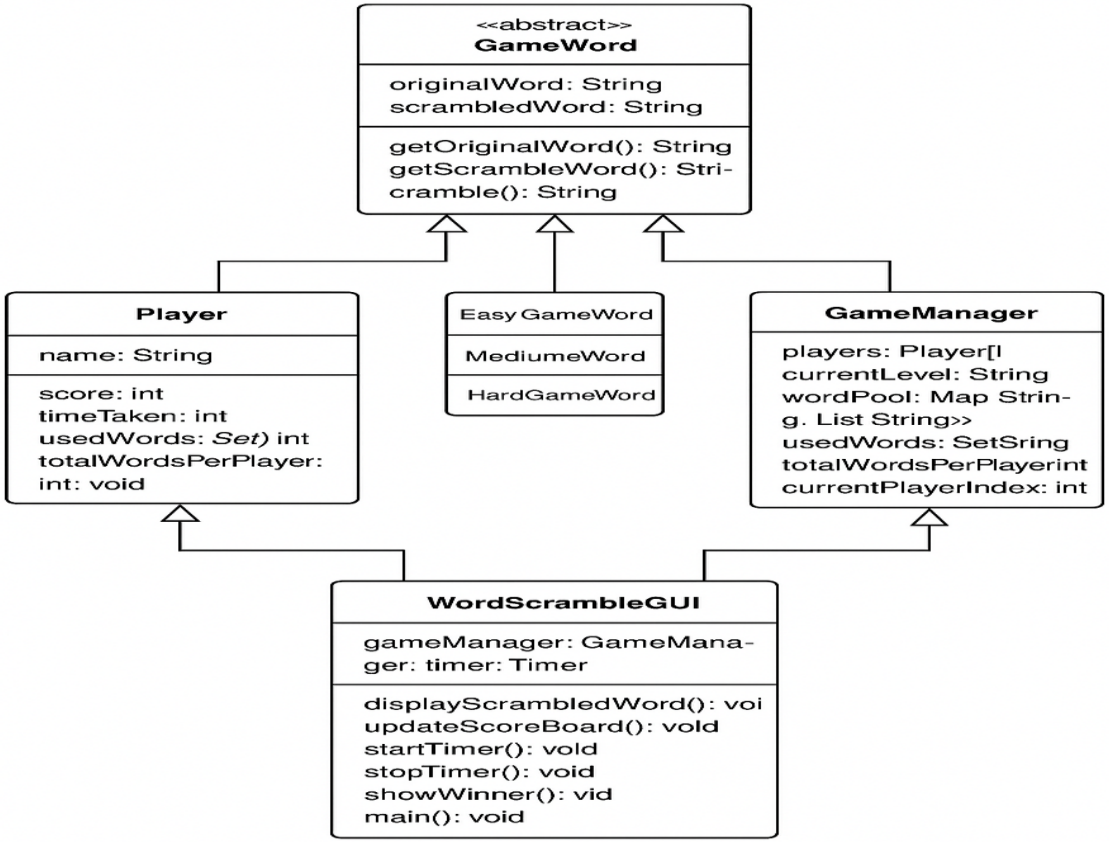

Fig.1 Class diagram of word scramble game

2.2 Parent Class

There is no explicit parent class in inheritance, but WordScrambleGUI extends JFrame, which acts as a GUI container class from Java Swing.

```
public class WordScrambleGUI extends JFrame {

}
```

Explanation:

- **WordScrambleGUI** is a custom class that extends **JFrame**, making it a top-level window in a Swing application.
- By extending JFrame, WordScrambleGUI inherits all the functionalities of a window, such as the ability to add components (buttons, labels, text fields), handle window events, and control window properties like size and visibility.
- This setup is standard in Swing applications where a custom class is created to define the GUI's layout and behavior.

2.3 Child Class

No custom child classes are created from WordScrambleGUI, but internally the game uses composition to manage elements like Timer, JComboBox, and JLabel.

2.4 Code Snippets

- **Word Scrambling:**

```
private String scramble(String word) {
  List<String> letters = Arrays.asList(word.split(""));
  Collections.shuffle(letters);
StringBuilder sb = new StringBuilder();
   for (String letter : letters) {
  sb.append(letter);
```

```
    }

    return sb.toString();

}
```

Explanation:

- Randomly rearranges the letters of the input word to create a scrambled version.
- **Process:**
  - word.split("") splits the word into an array of its constituent letters.
  - Arrays.asList(...) converts the array into a List for easier manipulation.
  - Collections.shuffle(letters) randomly shuffles the elements in the list.
  - A StringBuilder is used to efficiently concatenate the shuffled letters back into a single string.
- This method is likely called when presenting a new scrambled word to the player in the game.

- **b. Timer Logic:**

```
countdownTimer = new Timer(1000, e -> {

    timeLeft--;

    timerLabel.setText("Time: " + timeLeft);

    if (timeLeft <= 0) {

        countdownTimer.stop();

        resultLabel.setText("Time's up! Word was: " + currentWord);

        inputField.setEnabled(false);

        checkButton.setEnabled(false);

        wordsGuessed[currentPlayer]++;
```

```
        switchPlayer();

        loadNextWord();

    }

});
```

Explanation:

- Timer(1000, e -> { ... }) creates a Swing timer that triggers an action every 1000 milliseconds (1 second).
- timeLeft-- decrements the remaining time.
- timerLabel.setText(...) updates the GUI to reflect the new time.
- When timeLeft reaches zero:
    - The timer stops to prevent further decrements.
    - The result label informs the player that time is up and reveals the correct word.
    - Input fields and buttons are disabled to prevent further interaction.
    - The current player's guessed word count is incremented.
    - The game switches to the next player and loads a new word.

- **c. Check Answer:**

```
private void checkAnswer(ActionEvent e) {

    String guess = inputField.getText().trim();

    if (guess.equalsIgnoreCase(currentWord)) {

        resultLabel.setText("Correct!");

        scores[currentPlayer]++;

        times[currentPlayer] += (getInitialTime() - timeLeft);
```

```
        scoreLabels[currentPlayer].setText(playerNames[currentPlayer] + " Score: " + scores[currentPlayer]);

        countdownTimer.stop();

        inputField.setEnabled(false);

        checkButton.setEnabled(false);

        wordsGuessed[currentPlayer]++;

        switchPlayer();

        loadNextWord();

    } else {

        resultLabel.setText("Wrong! Try again.");

    }

}
```

Explanation :

- Retrieves the player's guess from the input field and trims any leading/trailing whitespace.
- Compares the guess to the currentWord, ignoring case differences.
- If the guess is correct:
    - Updates the result label to indicate success.
    - Increments the current player's score.
    - Adds the time taken to solve the word to the player's total time.
    - Updates the player's score label in the GUI.
    - Stops the countdown timer.
    - Disables the input field and check button to prevent further input.
    - Increments the count of words guessed by the current player.
    - Switches to the next player and loads a new word.
- If the guess is incorrect:
    - Updates the result label to prompt the player to try again.

2.5 Sample Output:

Example Game Flow:

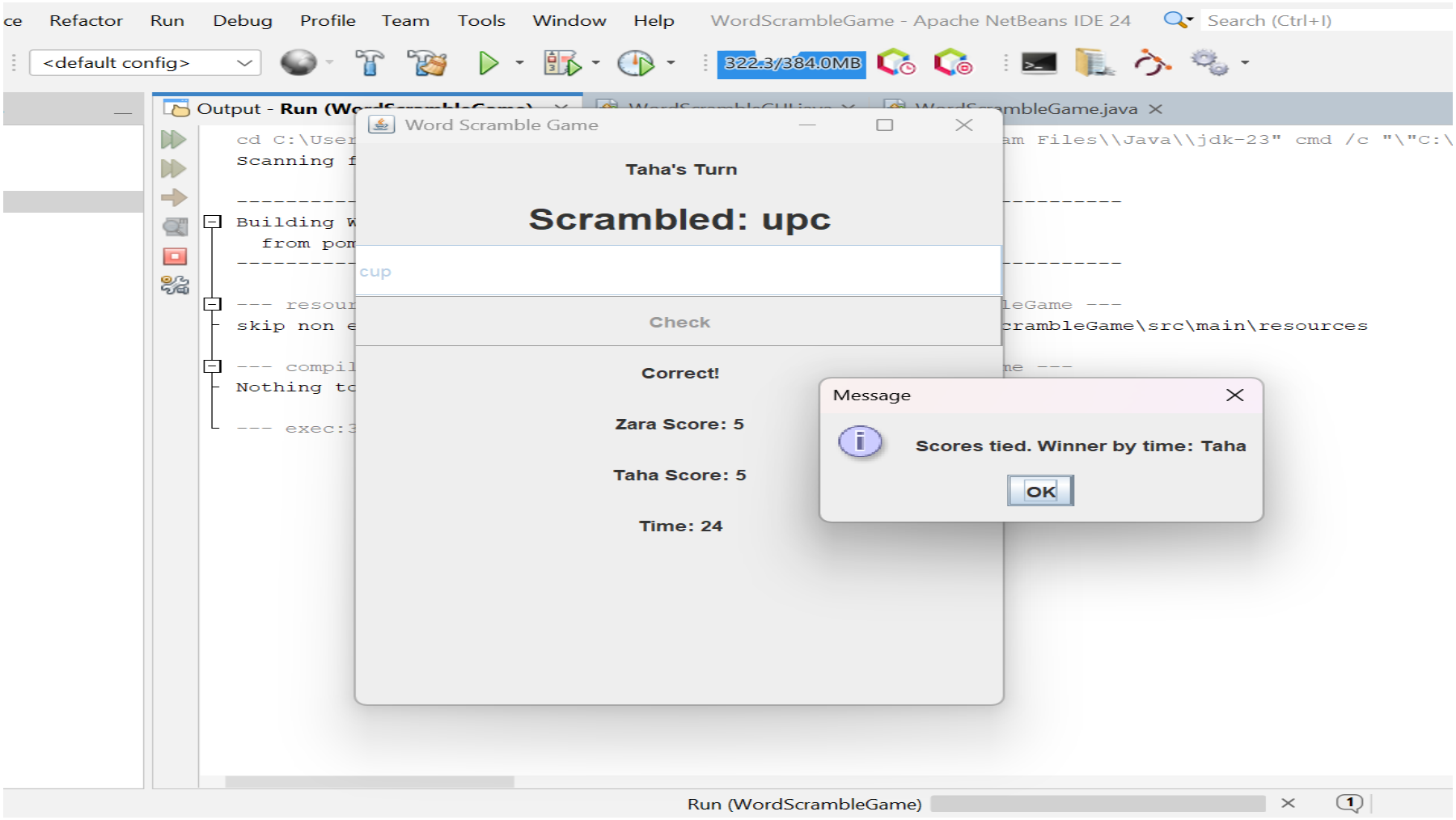

Fig.2 Code Output of Word Scramble Game

Output Explanation: Tie & Winner

- Both players have the **same score**, the game compares their **total answer time**. The player with the **less total time** is declared the winner. The final output shows the scores and announces the winner based on score and speed.

3. Remaining Work

- Add more than 2 player
- Add background sound

4. Conclusion

- The **WordScramble Game** project reflects how core Java OOP concepts can be used to build an engaging and maintainable desktop application. By using inheritance, abstraction, and polymorphism, the code remains modular and scalable.
  With a simple user interface, time-based challenge, and score tracking, the game offers both educational value and entertainment. This project not only strengthened understanding of Java programming but also demonstrated the importance of structured design in software development.

5. Future Work

Based on the current foundation of the WordScramble game, several enhancements can be planned for the future:

- **Multi-player system:** To allow more than 2 player.
- Save high score.

6. References

- **Oracle Java Swing Documentation**
  Comprehensive guide to building graphical user interfaces (GUIs) using Swing components in Java.
  Trail: Creating a GUI With Swing
- **GeeksforGeeks Java Timer Tutorial**
  Detailed explanation of the java.util.Timer class, including how to schedule tasks in Java.
  Java.util.Timer Class in Java
- **Stack Overflow for UI Event Handling**
  Community-driven discussions and solutions related to Java event handling.
  Newest 'java+event-handling' Questions
- **OpenAI ChatGPT for Structuring and Design Help**
  Prompt Engineering Best Practices for ChatGPT
- **Java Code Snippets (Custom Implementation)**
  https://docs.informatica.com/integration-cloud/data-integration/current-version/transformations/java-transformation/developing-the-java-code/creating-java-code-snippets.html
- **W3Schools Java Tutorial**
  https://www.w3schools.com/java/

7. Appendix

- GitHub Link: https://github.com/Mun-taha