**“WORD GUESSING GAME”**

A Java Project Report

ON

Submitted in Partial Fulfillment of the Requirements

For the award of the Degree of

**Bachelor of Technology**

**in**

**Electronics & Computer Engineering (ECM)**

By

**B. Athmiya Varshini 21311A1937**

Under the Guidance / Supervision of

**Mrs. N SWAPNA**

**Assistant Professor**

#### 

**Department of Electronics & Computer Engineering**

**Sreenidhi Institute of Science & Technology (Autonomous)**

**2022-2023**

**DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING**

**SREENIDHI INSTITUTE OF SCIENCE & TECHNOLOGY (AUTONOMOUS)**

****

**CERTIFICATE**

This is to certify that the Python Project entitled “**WORD GUESSING GAME”,** submitted by **B. AthmiyaVarshini,**bearing Roll No’s **21311A1937**towards partial fulfillment for the award of Bachelor’s Degree in Electronics & Computer Engineering from Sreenidhi Institute of Science & Technology, Ghatkesar, Hyderabad, is a record of bonafide work done by him. The results embodied in the work are not submitted to any other University or Institute for award of any degree or diploma.

**Mrs. N Swapna Dr.D. MOHAN (PhD)**

Assistant Professor Professor&HOD ECM

**DECLARATION**

This is to certify that the work reported in the present Java Project titled **“WORD GUESSING GAME”** is a record work done by my team in the **Department of Electronics and Computer Engineering, Sreenidhi Institute of Science and Technology, Yamnampet, Ghatkesar.**

The report is based on the project work done entirely by our team and not copied from any other source.

**BANDA ATHMIYA VARSHINI-21311A1937**

**ACKNOWLEDGMENT**

I convey my sincere thanks to**Prof. C V TOMY, Director** and **Dr.T.Ch. SIVA REDDY**, Principal, Sreenidhi Institute of Science and Technology, Ghatkesar for providing resources to complete this seminar.

I am very thankful to **Dr. D. MOHAN,** Head of ECM Department, Sreenidhi Institute of Science and Technology, Ghatkesar for providing an initiative to this project and giving valuable timely suggestions over our seminar work and for their kind co-operation in the completion of the seminar.

I convey my sincere thanks to **Mrs. N Swapna,** Assistant Professor and all the faculties of ECM Department,Sreenidhi Institute of Science and Technology, for their continuous help, co-operation, and support to complete this project.

Finally, I extend my sense of gratitude to almighty, my parents, all my friends, teaching and non-teaching staff, who directly or indirectly helped me in this endeavor.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENTS** | **PAGE NO** |
| 1 | Description | 5 |
| 2 | Program | 5 |
| 3 | Output | 11 |
| 4 | Explanation | 12 |
| 5 | Steps for Execution the Program | 13 |
| 6 | Limitations | 13 |
| 7 | Future Enhancements | 15 |
|  |  | 15 |
| 8 | References |  |
| 9 | Conclusion | 15 |

**DESCRIPTION**

In this game, there is a list of words present, out of which our interpreter will choose 1 random word. The user first has to input their names and then, will be asked to guess any alphabet. If the random word contains that alphabet, it will be shown as the output(with correct placement) else the program will ask you to guess another alphabet. The user will be given 12 turns(which can be changed accordingly) to guess the complete word. The Word Guessing Game is a Java-based interactive game that challenges players to guess a hidden word by providing guesses for individual letters. The game utilizes basic concepts of object-oriented programming and incorporates a graphical user interface (GUI) to enhance the user experience.

**Key Features:**

**Word Selectio**n: The game randomly selects a word from a predefined list or database. The selected word remains hidden from the player's view.

**GUI Interface:** The game incorporates a graphical user interface that displays the current state of the word being guessed, including the correctly guessed letters and any incorrect guesses made by the player.

**Letter Input:** Players can input their guesses for individual letters using the GUI. The game validates the input to ensure it is a single letter and checks if the letter exists in the hidden word.

**Feedback and Scoring:** After each guess, the game provides feedback on whether the guessed letter is correct or incorrect. It updates the display accordingly, revealing the correct letter's positions in the word and penalizing incorrect guesses.

**Game Progress Tracking:** The game keeps track of the player's progress, including the number of remaining attempts, letters already guessed, and the number of correct guesses. It may also display a hangman figure to indicate the remaining attempts visually.

**Win/Lose Conditions:** The game determines whether the player has won or lost based on specific conditions. A win occurs when the player correctly guesses all the letters in the word before running out of attempts. A loss occurs when the player exhausts all attempts without guessing the word correctly.

**Game Replay:** After each game, the player can choose to play again, starting with a new word to guess.

**PROGRAM**

import java.util.Scanner;

public class WordGuessingGame

{

private static final String[] WORDS = { "apple", "banana", "cherry", "grape", "orange" };

private static final int MAX\_ATTEMPTS = 6;

private String word;

private StringBuilder guessedWord;

private int attemptsLeft;

public void initializeGame()

{

int randomIndex = (int) (Math.random() \* WORDS.length);

word = WORDS[randomIndex];

// Initialize guessedWord with underscores

guessedWord = new StringBuilder();

for (int i = 0; i < word.length(); i++)

{

guessedWord.append("\_");

}

attemptsLeft = MAX\_ATTEMPTS;

}

public void playGame()

{

Scanner scanner = new Scanner(System.in);

while (attemptsLeft > 0)

{

System.out.println("Word: " + guessedWord);

System.out.println("Attempts Left: " + attemptsLeft);

System.out.print("Guess a letter: ");

char guess = scanner.next().charAt(0);

if (checkGuess(guess))

{

System.out.println("Correct guess!")

} else

{

System.out.println("Incorrect guess!");

attemptsLeft--;

}

if (guessedWord.toString().equals(word))

{

System.out.println("Congratulations! You guessed the word correctly: " + word);

return;

}

}

System.out.println("Game over! You ran out of attempts. The word was: " + word);

}

private boolean checkGuess(char guess) {

boolean correctGuess = false;

for (int i = 0; i < word.length(); i++)

{

if (word.charAt(i) == guess)

{

guessedWord.setCharAt(i, guess);

correctGuess = true;

}

}

return correctGuess;

}

public static void main(String[] args) {

WordGuessingGame game = new WordGuessingGame();

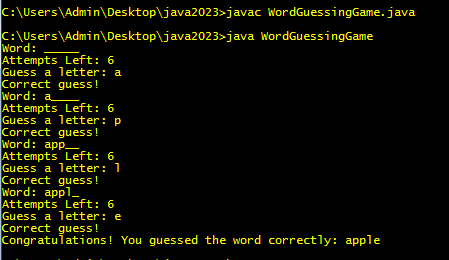
game.initializeGame();

game.playGame();

}

}

**OUTPUT**



**EXPLAINATION**

**1.**The program starts by importing the necessary package (java.util.Scanner) for user input.

**2.**The WordGuessingGame class is defined, which represents the main game logic.

**3.**The program declares and initializes two constants: WORDS (an array of words to be guessed) and MAX\_ATTEMPTS (the maximum number of attempts allowed).

**4.**The initialize Game method initializes the game by selecting a random word from the WORDS array, setting the initial guessed word as underscores, and setting the number of attempts left to MAX\_ATTEMPTS.

**5.**The play Game method is where the actual gameplay happens. It uses a Scanner object to read user input and continues the game until the player either guesses the word correctly or runs out of attempts.

**6.**Within the play Game method, the current state of the game (guessed word and attempts left) is displayed to the player. The player is prompted to enter a letter guess.

**7.**The check Guess method checks whether the guessed letter is present in the word. It updates the guessedWord by replacing underscores with the correctly guessed letter, and returns true if the guess is correct, or false if it is incorrect.

**8.**if the player makes a correct guess, the program displays a message indicating so. If the guess is incorrect, the attempts left are decremented by 1.

**9.**After each guess, the program checks if the player has guessed the entire word correctly. If so, it displays a congratulatory message and ends the game.

**10.**If the player runs out of attempts without guessing the word correctly, a "game over" message is displayed along with the correct word.

**11.**The main method creates an instance of WordGuessingGame, initializes the game, and starts the game play by calling the play Game method.

**12.**That's a high-level explanation of the Java program for the Word Guessing Game. Feel free to ask if you have any specific questions about any part of the code!

**STEPS FOR EXECUTING THE PROGRAM**

The program is saved in the following process

1. Open the notepad in your laptop or Desktop.
2. Don’t forget to download the IDE from the online which used to execute the code.
3. Then start writing the code in the Notepad.
4. Then save the file with .java extension.
5. Open the IDE the file and execute in the program then you see the errors and work on it.
6. **NOTE:** Write the program very carefully try to write it without errors and do the follow the steps of program that are required in the Java.

**FUTURE ENHANCEMENT**

* The future enhancement of this project can be more block like 6\*6 and more
* We can even add time limit for a match.
* We can even add more than 2 players.
* We can add a solo player game so the users can a game with computer.

**REFERENCE**

The reference is taken from the text “geeksforgeeks”

**CONCLUSION**

This game can play in offline only and it is a multi-player ,where group of friends can challenge themselves and who guesses the correct word the times is the winner. In conclusion, developing a word guessing game project in Java can be an engaging and fun way to apply your programming skills. Throughout the project, you learned about various concepts and techniques, including random word selection, user input processing, hint generation, win/lose conditions, and game loop implementation.

By following the outlined steps, you created a game that allows players to guess a hidden word within a limited number of attempts. The game provides feedback to help players narrow down their guesses and determines whether they win or lose based on their performance.

Remember that this is just a starting point, and there are endless possibilities for expanding and improving the game. You can enhance it by adding more features such as scoring, difficulty levels, time constraints, multiplayer functionality, or a graphical user interface (GUI) for a more immersive experience.

Overall, developing a word guessing game in Java not only enhances your programming skills but also provides an opportunity to create an enjoyable and interactive experience for users. It's a great way to demonstrate your understanding of Java programming concepts while building a fun and challenging game.