|  |
| --- |
| **Ex.No: 8 DATE: 27-09-23 STRING HANDLING** |

1) Write a program to implement Hangman game in Java, a word-guessing game where one player thinks of a Secret word, and another player (the computer i.e., Guesser or user) tries to guess it by stating one letter at a time.If the guessed letter is in the secret word, it's revealed in the display; otherwise the number of attempt is reduced by one. The player usually has a limited number of incorrect guesses allowed before they lose the game.The game ends when the player successfully guesses the word (win) or when they run out of allowed incorrect guesses (lose).The game typically has the following components:[CO:2 BL:6]

Input: The guesser or user tries to guess the secret word by stating each of the character in the secret word.

Output: A representation of the secret word is displayed with underscores for each letter that hasn't been guessed yet. For example, if the secret word is "java" and the player has guessed 'a', the display would be "\_ a \_ a".

Method: Guessing

Guessing: Fix the maximum attempts. Create a guessedLetter Boolean array of secret word length size to verify the guessed character of secret word.

Process the guess

i. Check input is a valid input (i.e a character)Check if the guessed letter is in the secret word

ii. Update guessedLetters array

iii. Check if the entire word has been guessed

iv. Increment attempts if the guess is incorrect

v. Print game messagesWin/Lose Conditions

vi. The player receives feedback on their guesses, including whether the guessed letter is in the word and the current state of the word display.

vii. Replay: After the game ends, the players may choose to play again with a new secret word. Increase the complexity of the program by having more than one secretWord and choose it randomly. Give clues about the secretWord and reduce the points accordingly.

Code:

import java.time.LocalDateTime;  
import java.time.format.DateTimeFormatter;  
import java.util.Scanner;  
import java.util.Random;  
  
public class hangMan3524 {  
 // Array of secret words to choose from  
 private static final String[] secretWords = {"java", "python", "javascript", "programming", "computer"};  
  
 // Clues corresponding to secret words  
 private static final String[] clues = {"A popular programming language", "A versatile scripting language", "The language of the web", "The act of writing code", "A device used for computation"};  
  
 private static final int maxAttempts = 5;  
 private static final int baseScore = 10;  
  
 // Score awarded per correct guess  
 private static final int correctGuessScore = 1;  
  
 // Score penalty for using a clue  
 private static final int penaltyScore = 2;  
  
 public static void main(String[] args) {  
 // Code for getting the current date and time  
 DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");  
 LocalDateTime now = LocalDateTime.now();  
 System.out.println("2021503524 " + "Mugundh J B " + dtf.format(now));  
  
 Scanner in = new Scanner(System.in);  
 boolean playAgain = true;  
  
 // Initialize the player's total score  
 int totalScore = 0;  
 int timesPlayed = 0;  
  
 while (playAgain) {  
 // Select a random secret word from the array  
 String secretWord = getRandomSecretWord();  
 timesPlayed++;  
 // Create an array to track guessed letters  
 char[] guessedLetters = new char[secretWord.length()];  
  
 // Initialize the number of attempts allowed  
 int attempts = maxAttempts;  
  
 // Flags to track game state  
 boolean wordGuessed = false;  
 boolean usedClue = false;  
  
 // Display instructions to game  
 System.out.println("Welcome to Hangman!");  
 System.out.println("You have " + maxAttempts + " attempts to guess the word.");  
  
 while (attempts > 0 && !wordGuessed) {  
 // Display the current state of the word  
 displayWord(secretWord, guessedLetters);  
 System.out.println("Attempts left: " + attempts);  
  
 // Get the player's guess  
 char guess = getGuess(in);  
  
 if (isGuessCorrect(secretWord, guessedLetters, guess)) {  
 System.out.println("Good guess!");  
 } else {  
 System.out.println("Incorrect guess!");  
 attempts--;  
  
 // Offer a clue if the player has not used one before  
 if (!usedClue) {  
 System.out.print("Do you want a clue? (yes/no): ");  
 String clueInput = in.next().toLowerCase();  
 if (clueInput.equals("yes")) {  
 // Provide a clue and apply a penalty to the score  
 System.out.println("Here's your clue: " + getClue(secretWord));  
 totalScore -= penaltyScore;  
 usedClue = true;  
 }  
 }  
 }  
  
 // Check if the entire word has been guessed  
 wordGuessed = isWordGuessed(guessedLetters);  
 }  
  
 // Calculate and update the score for the round  
 int score = calculateScore(secretWord, guessedLetters, usedClue);  
 if(score > 0)  
 totalScore += score;  
  
 if (wordGuessed) {  
 // Display the word and a congratulations message  
 displayWord(secretWord, guessedLetters);  
 System.out.println("\nCongratulations! You've guessed the word.");  
 } else {  
 // Display an out-of-attempts message with the correct word  
 System.out.println("\nOut of attempts. The word was: " + secretWord);  
 }  
 // Display the player's score for the current round and the total score  
 System.out.println("Your score for this round: " + score + " / 10");  
 if(score > 0 && totalScore > score)  
 System.out.println("Your total score: " + totalScore + " / " + timesPlayed \* 10);  
 else  
 System.out.println("Your total score: " + (totalScore + 2) + " / " + timesPlayed \* 10);  
  
 // Ask if the player wants to play again  
 System.out.print("\nDo you want to play again? (yes/no): ");  
 String playAgainInput = in.next().toLowerCase();  
 playAgain = playAgainInput.equals("yes");  
  
 }  
 // Display a thank you message when the player finishes playing  
 System.out.println("\nThanks for playing Hangman!");  
 }  
 // Method to select a random secret word from the array  
 private static String getRandomSecretWord() {  
 Random rand = new Random();  
 int randIdx = rand.nextInt(secretWords.length);  
 return secretWords[randIdx];  
 }  
 // Method to get the clue corresponding to a secret word  
 private static String getClue(String secretWord) {  
 for (int i = 0; i < secretWords.length; i++) {  
 if (secretWords[i].equals(secretWord)) {  
 return clues[i];  
 }  
 }  
 return "";  
 }  
 // Method to display the current state of the word with hidden and guessed letters  
 private static void displayWord(String secretWord, char[] guessedLetters) {  
 for (int i = 0; i < secretWord.length(); i++) {  
 char letter = secretWord.charAt(i);  
 if (contains(guessedLetters, letter)) {  
 System.out.print(letter);  
 } else {  
 System.out.print("\_");  
 }  
 System.out.print(" ");  
 }  
 System.out.println();  
 }  
  
 // Method to check if a letter is in an array  
 private static boolean contains(char[] array, char letter) {  
 for (char c : array) {  
 if (c == letter) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 // Method to get a valid single-letter guess from the player  
 private static char getGuess(Scanner in) {  
 char guess;  
 while (true) {  
 System.out.print("Guess a letter: ");  
 String input = in.next();  
 if (input.length() == 1 && Character.isLetter(input.charAt(0))) {  
 guess = input.charAt(0);  
 break;  
 } else {  
 System.out.println("Invalid input. Please enter a single letter.");  
 }  
 }  
 return guess;  
 }  
  
 // Method to check if a guess is correct and update the guessed letters array  
 private static boolean isGuessCorrect(String secretWord, char[] guessedLetters, char guess) {  
 boolean correctGuess = false;  
 for (int i = 0; i < secretWord.length(); i++) {  
 if (secretWord.charAt(i) == guess) {  
 guessedLetters[i] = guess;  
 correctGuess = true;  
 }  
 }  
 return correctGuess;  
 }  
  
 // Method to check if the entire word has been guessed  
 private static boolean isWordGuessed(char[] guessedLetters) {  
 for (char c : guessedLetters) {  
 if (c == 0) {  
 return false;  
 }  
 }  
 return true;  
 }  
  
 // Method to calculate the score for the round  
 private static int calculateScore(String secretWord, char[] guessedLetters, boolean usedClue) {  
 if (isWordGuessed(guessedLetters)) {  
 int score = baseScore - (usedClue ? penaltyScore : 0);  
 return Math.max(score, 0);  
 } else {  
 int correctGuessCount = 0;  
 for (char c : guessedLetters) {  
 if (c != 0) {  
 correctGuessCount++;  
 }  
 }  
 int roundScore = correctGuessCount \* correctGuessScore - (usedClue ? penaltyScore : 0);  
 // Ensure that the round score is not less than 0  
 roundScore = Math.max(roundScore, 0);  
 // Calculate the total score by multiplying the round score by the maximum possible score (10)  
 int totalScore = roundScore \* 10;  
 return totalScore;  
 }  
 }  
  
}

Output:





