



Portada

Título: Desarrollo de Wordle en Java

Apellidos, Nombre: Pop, Daniel Andrei

Módulo: Desarrollo de Aplicaciones Web

Fecha de entrega: 21 de marzo de 2025

1. Introducción

En este trabajo se ha desarrollado una versión simple del juego *Wordle* utilizando el lenguaje Java. La aplicación pone en práctica conceptos fundamentales de la programación orientada a objetos, tales como la división en clases, la encapsulación, y el manejo de ficheros para la lectura y escritura de información.

Para la realización del juego se ha utilizado un fichero de texto denominado **palabras.txt**, en el que cada línea contiene una palabra de 5 letras. Este formato ha sido elegido por su sencillez y por facilitar la manipulación y validación de la información.

El código se ha estructurado en las siguientes clases:

Main: Punto de entrada del programa, donde se cargan las palabras y se inicializa el juego.

WordleGame: Encargada de la lógica principal del juego, la gestión de intentos y la interacción con el usuario.

WordleFeedback: Responsable de generar el feedback visual mediante códigos ANSI, mostrando letras en colores según la coincidencia con la palabra secreta.

WordleFileManager: Gestiona la lectura de las palabras desde el fichero y el guardado del historial de partidas.



1.1. Contenido

Descripción del juego:

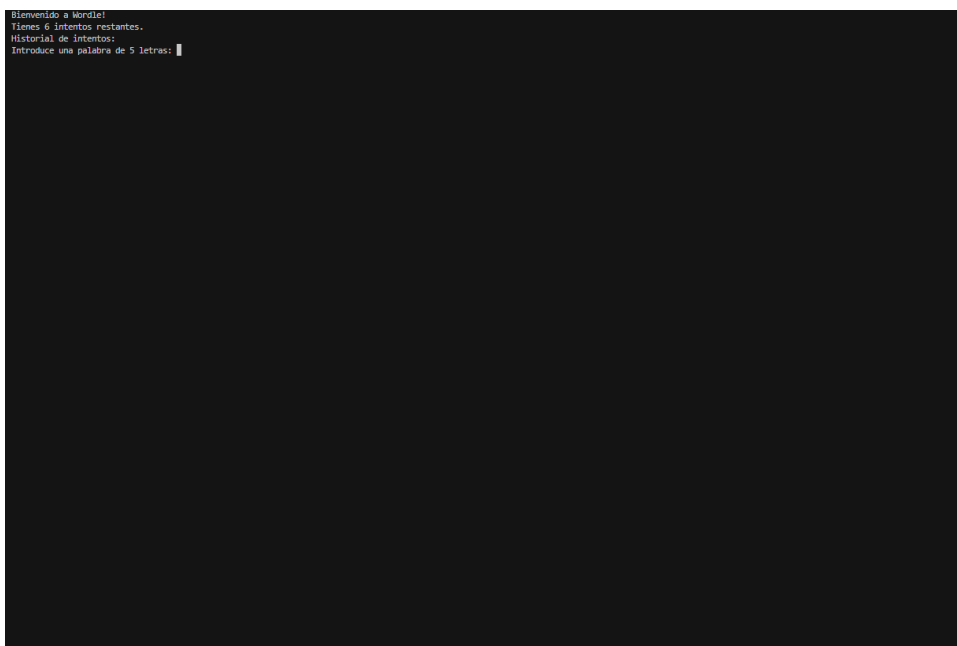
Wordle es un juego en el que el jugador debe adivinar una palabra de 5 letras en un máximo de 6 intentos. Tras cada intento se ofrece retroalimentación visual:

- **Verde:** La letra está en la posición correcta.
- **Amarillo:** La letra está en la palabra pero en otra posición.
- **Sin color:** La letra no está en la palabra.

Capturas de pantalla y explicación de una partida:

1. Inicio del juego:

1.1. Se muestra el mensaje de bienvenida y el número de intentos restantes.



```
Bienvenido a Wordle!  
Tienes 6 intentos restantes.  
Historial de intentos:  
Introduce una palabra de 5 letras: |
```

2. Durante la partida:

2.1. Se muestra el historial de intentos y se solicita al usuario que introduzca una palabra de 5 letras.



2.2. Se valida la entrada y, en caso de error (por ejemplo, si la palabra no tiene 5 letras), se muestra un mensaje adecuado.

```
Bienvenido a Wordle!
Tienes 6 intentos restantes.
Historial de intentos:
Introduce una palabra de 5 letras: abeto
ABETO
Tienes 5 intentos restantes.
Historial de intentos:
abeto
Introduce una palabra de 5 letras: actor
ACTOR
Tienes 4 intentos restantes.
Historial de intentos:
abeto
actor
Introduce una palabra de 5 letras: agudo
AGUDO
Tienes 3 intentos restantes.
Historial de intentos:
abeto
actor
agudo
Introduce una palabra de 5 letras: albas
ALBAS
Tienes 2 intentos restantes.
Historial de intentos:
abeto
actor
agudo
albas
Introduce una palabra de 5 letras: alado
ALADO
Tienes 1 intentos restantes.
Historial de intentos:
abeto
actor
agudo
albas
alado
Introduce una palabra de 5 letras: aguas
AGUAS
```

2.3. Se han acabado los intentos. La palabra secreta era: Antón

3. Finalización exitosa:

3.1. Una vez adivinada la palabra, se muestra un mensaje de felicitación junto con el historial completo de intentos.



```
Bienvenido a Wordle!  
Tienes 6 intentos restantes.  
Historial de intentos:  
Introduce una palabra de 5 letras: abeto  
ABETO  
Tienes 5 intentos restantes.  
Historial de intentos:  
abeto  
Introduce una palabra de 5 letras: altar  
ALTAR  
Tienes 4 intentos restantes.  
Historial de intentos:  
abeto  
altar  
Introduce una palabra de 5 letras: albas  
ALBAS  
Tienes 3 intentos restantes.  
Historial de intentos:  
abeto  
altar  
albas  
Introduce una palabra de 5 letras: aguas  
¡Felicidades! Has adivinado la palabra correcta: aguas
```

3.2.

Conclusión

Como mejora para este proyecto se propone la incorporación de **niveles de dificultad**. Con esta funcionalidad, el usuario podría seleccionar entre diferentes categorías o longitudes de palabras, lo cual ampliaría la jugabilidad del programa. Para implementarla se podrían realizar las siguientes acciones:

- Modificar la clase **WordleGame** para aceptar un parámetro de dificultad.
- Ampliar o crear nuevos ficheros de palabras según las categorías (por ejemplo, animales, objetos, etc.).
- Ajustar la lógica de validación y feedback para adaptarse a palabras de diferentes longitudes.

Esta mejora no solo enriquecería la experiencia del usuario, sino que también implicaría profundizar en conceptos de diseño modular y manejo avanzado de ficheros.



1.1.1.1. Anexo

```
import java.io.*;

import java.util.*;

public class Main {

    public static void main(String[] args) {

        // Cargar las palabras desde el fichero "palabras.txt"

        String[] words = WordleFileManager.loadWords("palabras.txt");

        if(words.length == 0) {

            System.out.println("No se han cargado palabras. Verifica el
fichero 'palabras.txt'.");

            return;

        }

        // Crear la instancia del juego con las palabras cargadas

        WordleGame game = new WordleGame(words);

        // Iniciar la partida

        game.start();

    }

}

class WordleGame {

    private final int MAX_TRIES = 6;

    private final int WORD_LENGTH = 5;

    private String[] fileWords;

    private String secretWord;
```



```
private int remainingAttempts;

private ArrayList<String> triesHistory;

public WordleGame(String[] fileWords) {

    this.fileWords = fileWords;

    this.secretWord = selectRandomWord(fileWords);

    this.remainingAttempts = MAX_TRIES;

    this.triesHistory = new ArrayList<>();

}

public void start() {

    Scanner scanner = new Scanner(System.in);

    System.out.println("Bienvenido a Wordle!");

    while (remainingAttempts > 0) {

        System.out.println("Tienes " + remainingAttempts + "
intentos restantes.");

        showTriesHistory();

        String guess = getUserInput(scanner);

        triesHistory.add(guess);

        // Comprobar si el jugador adivinó la palabra secreta

        if (guess.equalsIgnoreCase(secretWord)) {

            System.out.println(";Felicitades! Has adivinado la
palabra correcta: " + secretWord);

            break;

        } else {
```



```
        remainingAttempts--;

        System.out.println(WordleFeedback.feedBackString(guess,
secretWord));

    }

}

        if    (remainingAttempts    ==    0    &&
!triesHistory.contains(secretWord)) {

        System.out.println("Se han acabado los intentos. La palabra
secreta era: " + secretWord);

    }

    scanner.close();

}

private void showTriesHistory() {

    System.out.println("Historial de intentos:");

    for (String tryWord : triesHistory) {

        System.out.println(tryWord);

    }

}

private String selectRandomWord(String[] words) {

    Random random = new Random();

    return words[random.nextInt(words.length)];

}
```



```
private String getUserInput(Scanner scanner) {  
  
    String input;  
  
    while (true) {  
  
        System.out.print("Introduce una palabra de " + WORD_LENGTH  
+ " letras: ");  
  
        input = scanner.nextLine().trim();  
  
        if (input.length() == WORD_LENGTH) {  
  
            break;  
  
        } else {  
  
            System.out.println("La palabra debe tener " +  
WORD_LENGTH + " letras.");  
  
        }  
  
    }  
  
    return input;  
  
}  
  
}  
  
class WordleFeedback {  
  
    private static final int WORD_LENGTH = 5;  
  
    public static final String ANSI_RESET = "\u001B[0m";  
  
    public static final String ANSI_RED = "\u001B[31m";  
  
    public static final String ANSI_GREEN = "\u001B[32m";  
  
    public static final String ANSI_YELLOW = "\u001B[33m";  

```




```
private static String applyColor(String letter, String color) {  
  
    return color + letter.toUpperCase() + ANSI_RESET;  
  
}  
  
public static String feedBackString(String guess, String  
secretWord) {  
  
    StringBuilder feedback = new StringBuilder();  
  
    for (int i = 0; i < WORD_LENGTH; i++) {  
  
        String letter = String.valueOf(guess.charAt(i));  
  
        if (secretWord.charAt(i) == guess.charAt(i)) {  
  
            feedback.append(applyColor(letter, ANSI_GREEN));  
  
        }  
  
        else if (secretWord.contains(letter)) {  
  
            feedback.append(applyColor(letter, ANSI_YELLOW));  
  
        } else {  
  
            feedback.append(letter.toUpperCase());  
  
        }  
  
    }  
  
    return feedback.toString();  
  
}  
}
```



```
class WordleFileManager {

    // Método para cargar palabras desde un fichero de texto

    public static String[] loadWords(String fileName) {

        ArrayList<String> wordsList = new ArrayList<>();

        try (BufferedReader br = new BufferedReader(new
FileReader(fileName))) {

            String line;

            while ((line = br.readLine()) != null) {

                if (!line.trim().isEmpty()) {

                    wordsList.add(line.trim());

                }

            }

        } catch (IOException e) {

            System.err.println("Error al leer el fichero: " +
e.getMessage());

        }

        return wordsList.toArray(new String[0]);

    }

    public static void saveGame(String gameOutput, String fileName) {

        try (BufferedWriter bw = new BufferedWriter(new
FileWriter(fileName, true))) {

            bw.write(gameOutput);

            bw.newLine();

        } catch (IOException e) {
```



```
        System.err.println("Error al escribir el fichero: " +  
e.getMessage());  
    }  
}  
}
```