# Programación

# Trabajo de Enfoque

Realizado por: Daniel Roman Garcia

# 1. Introducción

Este documento describe el proceso de desarrollo del juego Wordle en Java, en el cual se selecciona una palabra al azar desde un archivo .txt ubicado en la misma carpeta que el juego.

```java
public class wordle {
//Como buscar archivo externo con informacion correcta
    public static void main(String[] args) {
        List<String> words = loadWordsFromFile("palabras.txt");
        if (words.isEmpty()) {
            System.out.println("Error: No se pudieron cargar las palabras.");
            return;
        }
        WordleGame game = new WordleGame(words);
        game.start();
    }

    private static List<String> loadWordsFromFile(String fileName) {
        try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
            return br.lines()
                    .filter(line -> line.length() == 5)
                    .map(String::toUpperCase)
                    .toList();
        } catch (IOException e) {
            System.out.println("Error al leer el archivo: " + e.getMessage());
            return Collections.emptyList();
        }
    }
}
```

Inicialmente, surgieron varios errores al intentar leer la palabra desde el archivo. Para solucionarlo, se implementaron mejoras en el código, asegurando que todas las palabras se conviertan a mayúsculas y que tengan exactamente cinco letras.

Además, al utilizar try y catch para manejar excepciones, NetBeans recomendó incluir IOException e para garantizar una correcta lectura del archivo y manejar posibles errores de entrada y salida de datos.

```java
private String selectRandomWord(List<String> words) {// Elegir aleatoriamente palabra del archivo
    Random random = new Random();
    return words.get(random.nextInt(words.size()));
}
```

Ya hemos visto cómo se captura la palabra. Para mejorar la experiencia del juego, se ha decidido añadir un factor aleatorio que permite seleccionar una palabra diferente en la mayoría de las partidas, asegurando mayor variedad y dinamismo.

# 2. Como funciona Wordle

Segun los requisitos especificados, el Wordle debería mostrarnos que letras han sido falladas mediante colores, identificar y validar que es una palabra de 5 letras y nos dice cuántos intentos nos quedan, además se ha decidido quitar letras del alfabeto cada vez que se falle como pista al jugador.

## 2.1. Variables

Se inicializaron las diferentes variables fijas, como se muestra en la imagen. Además, se utilizó un entorno privado para mejorar la seguridad del código.

Para la correcta inicialización de las variables dentro de la clase, se empleó la palabra clave this, lo que permite diferenciar las variables de instancia y garantizar su correcto funcionamiento en las operaciones requeridas.

```
//Declaracion de Variables fijas
    private static final int MAX_TRIES = 6;
    private static final int WORD_LENGTH = 5;
    private final List<String> fileWords;
    private final String secretWord;
    private int remainingAttempts;
    private final List<String> triesHistory;
    private final Scanner scanner;
    private final Set<Character> availableLetters;

    public WordleGame(List<String> words) {
        this.fileWords = new ArrayList<>(words);
        this.secretWord = selectRandomWord(this.fileWords);
        this.remainingAttempts = MAX_TRIES;
        this.triesHistory = new ArrayList<>();
        this.scanner = new Scanner(System.in);
        this.availableLetters = new HashSet<>();

        // Inicializar availableLetters con todas las letras del alfabeto
        for (char c = 'A'; c <= 'Z'; c++) {
            availableLetters.add(c);
        }
```

## 2.2. Inicio

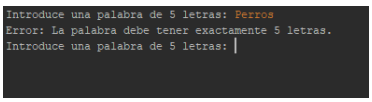
En el momento que inicializamos el proyecto nos recibe con un saludo además de la cantidad de intentos que le quedan además de las letras que puedes utilizar. En este instante pedimos la palabra de 5 letras.

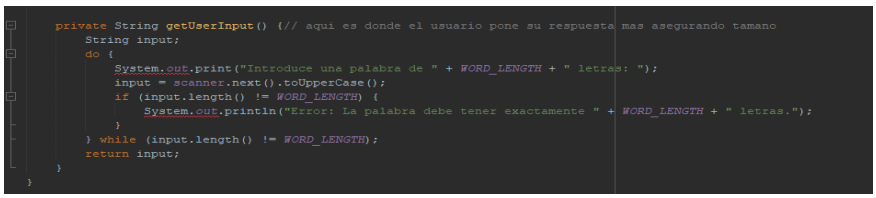
```
run:
 Bienvenido a Wordle

Te quedan 6 intentos

Letras disponibles: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Introduce una palabra de 5 letras:
```

```
public void start() {
    System.out.println(" Bienvenido a Wordle ");
    while (remainingAttempts > 0) {
        showTriesHistory();
        showAvailableLetters(); // Mostrar letras disponibles antes del intento

        String userGuess = getUserInput();
        triesHistory.add(userGuess);

        updateAvailableLetters(userGuess); // Actualizar letras disponibles

        if (userGuess.equals(secretWord)) {
            System.out.println("\n¡Felicidades! Has adivinado la palabra: " + se
            return;
        }

        System.out.println(WordleFeedback.feedBackString(userGuess, secretWord));
        remainingAttempts--;
    }
    System.out.println("\nHas perdido. La palabra secreta era: " + secretWord);
}
```

## 2.3. Feedback

Al introducir la palabra puede surgir 2 respuestas, la primera si introducimos una palabra de diferente número de letras.

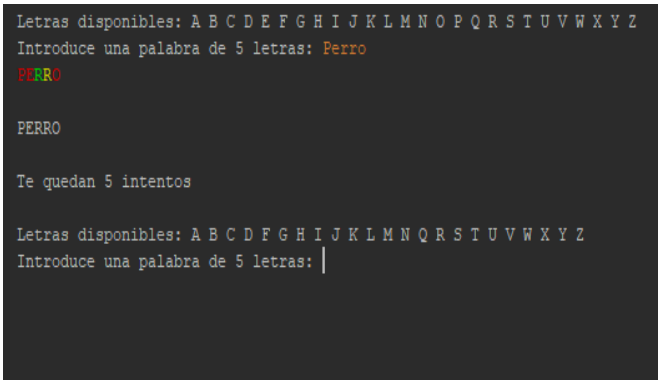


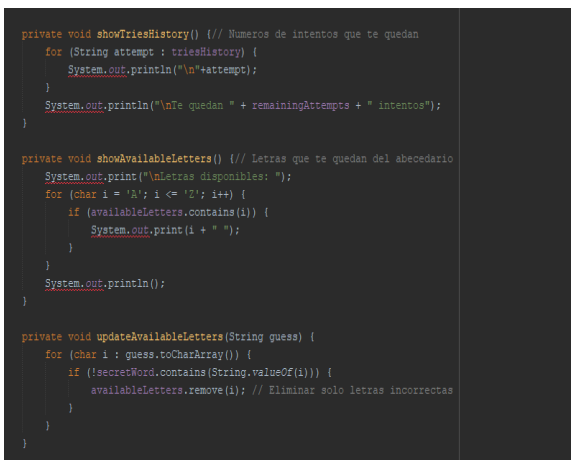



```
private String getUserInput() {// aqui es donde el usuario pone su respuesta mas asegurando tamano
    String input;
    do {
        System.out.print("Introduce una palabra de " + WORD_LENGTH + " letras: ");
        input = scanner.next().toUpperCase();
        if (input.length() != WORD_LENGTH) {
            System.out.println("Error: La palabra debe tener exactamente " + WORD_LENGTH + " letras.");
        }
    } while (input.length() != WORD_LENGTH);
    return input;
}
}
```

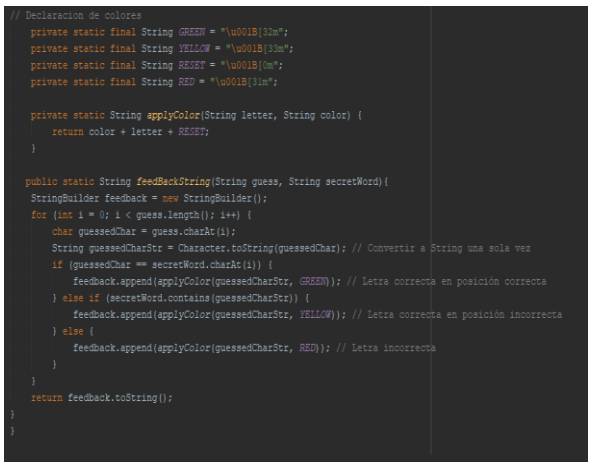
Si la palabra es de 5 letras, el mensaje cambiara dando Cuatro feedbacks, la primera nos dara la palabra que hemos intentado separada por colores como se describe en los requisitos.

La segunda va a mostrar el historial de las palabras que se han usado anteriormente. La tercera mostrara la cantidad de intentos restantes. Y por último se quitarán las letras del alfabeto que se hayan fallado y las que el jugador debe usar.



```
private void showTriesHistory() {// Numeros de intentos que te quedan
    for (String attempt : triesHistory) {
        System.out.println("\n"+attempt);
    }
    System.out.println("\nTe quedan " + remainingAttempts + " intentos");
}

private void showAvailableLetters() {// Letras que te quedan del abecedario
    System.out.print("\nLetras disponibles: ");
    for (char i = 'A'; i <= 'Z'; i++) {
        if (availableLetters.contains(i)) {
            System.out.print(i + " ");
        }
    }
    System.out.println();
}

private void updateAvailableLetters(String guess) {
    for (char i : guess.toCharArray()) {
        if (!secretWord.contains(String.valueOf(i))) {
            availableLetters.remove(i); // Eliminar solo letras incorrectas
        }
    }
}
```

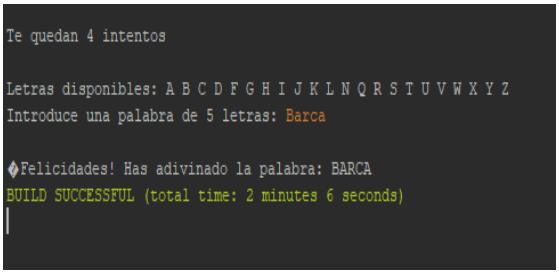


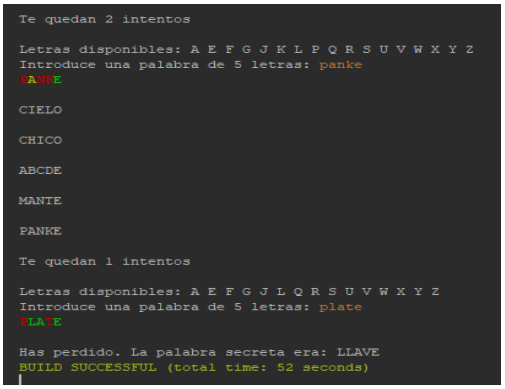
```
// Declaracion de colores
    private static final String GREEN = "\u001B[32m";
    private static final String YELLOW = "\u001B[33m";
    private static final String RESET = "\u001B[0m";
    private static final String RED = "\u001B[31m";

    private static String applyColor(String letter, String color) {
        return color + letter + RESET;
    }

   public static String feedBackString(String guess, String secretWord){
    StringBuilder feedback = new StringBuilder();
    for (int i = 0; i < guess.length(); i++) {
        char guessedChar = guess.charAt(i);
        String guessedCharStr = Character.toString(guessedChar); // Convertir a String una sola vez
        if (guessedChar == secretWord.charAt(i)) {
            feedback.append(applyColor(guessedCharStr, GREEN)); // Letra correcta en posición correcta
        } else if (secretWord.contains(guessedCharStr)) {
            feedback.append(applyColor(guessedCharStr, YELLOW)); // Letra correcta en posición incorrecta
        } else {
            feedback.append(applyColor(guessedCharStr, RED)); // Letra incorrecta
        }
    }
    return feedback.toString();
}
}
```

## 2.4. Mensaje final y cierre

Según la respuesta dada si es correcta se mostrará un mensaje como se puede ver en la siguiente imagen.

Después el proyecto se cerrará terminando el proceso. Si la respuesta hace que llegues al final de los maximos intentos este mostrar un mensaje en el cual se notifica que el jugador ha perdido y cuál era la palabra secreta, despues de esto el proyecto se cerrara




# 3. Conclusión

La creación de este proyecto fue sencilla, ya que se siguieron los requerimientos establecidos. Sin embargo, la implementación del archivo externo requirió un proceso extenso de prueba y testeo. En cuanto a la introducción de colores, considero que es un aspecto mejorable. Intenté utilizar background color, pero la implementación resultó compleja y presentaba una alta tasa de errores. Se implemento el sistema de eliminación de letras inicializando el alfabeto y eliminando letras falladas.

# 4. Bibliografía

Strategio. (2022, agosto 10). Build a Wordle clone in Java. Medium. https://medium.com/strategio/build-a-wordle-clone-in-java-c7b7b924fb8d

Bro Code. (2022, octubre 9). YouTube. https://www.youtube.com/watch?v=EdWuxZVBEMA

Apache Software Foundation. (2025). Apache NetBeans (versión 25) [Software]. https://netbeans.apache.org/

Horstmann, C. S. (2019). Core Java Volume I: Fundamentals (11th ed.). Pearson Education.

# 5. Anexo

En este anexo se puede acceder a la carpeta entera desde el siguiente enlace https://github.com/DRGDam/Programacion-Enfoque. Aquí se encontrará todo el proyecto, pudiendo ejecutarse en un IDE desde la carpeta src y ejecutar el archivo wordle.java.

También se dejará el código escitro por si no tiene acceso a Internet, aunque se recomienda revisar en IDE.

## Codigo

```
public class wordle {

//Como buscar archivo externo con informacion correcta

    public static void main(String[] args) {

        List<String> words = loadWordsFromFile("palabras.txt");

        if (words.isEmpty()) {

            System.out.println("Error: No se pudieron cargar las palabras.");

            return;

        }

        WordleGame game = new WordleGame(words);

        game.start();

    }

    private static List<String> loadWordsFromFile(String fileName) {

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

            return br.lines()
```

```
                .filter(line -> line.length() == 5)
                .map(String::toUpperCase)
                .toList();
        } catch (IOException e) {
            System.out.println("Error al leer el archivo: " + e.getMessage());
            return Collections.emptyList();
        }} }
class WordleGame {
//Declaracion de Variables fijas
    private static final int MAX_TRIES = 6;
    private static final int WORD_LENGTH = 5;
    private final List<String> fileWords;
    private final String secretWord;
    private int remainingAttempts;
    private final List<String> triesHistory;
    private final Scanner scanner;
    private final Set<Character> availableLetters;

    public WordleGame(List<String> words) {
        this.fileWords = new ArrayList<>(words);
        this.secretWord = selectRandomWord(this.fileWords);
        this.remainingAttempts = MAX_TRIES;
        this.triesHistory = new ArrayList<>();
        this.scanner = new Scanner(System.in);
        this.availableLetters = new HashSet<>();
```

```
        // Inicializar availableLetters con todas las letras del alfabeto
        for (char c = 'A'; c <= 'Z'; c++) {
            availableLetters.add(c);
    }}
     private String selectRandomWord(List<String> words) {// Elegir aleatoriamente palabra del
archivo
        Random random = new Random();
        return words.get(random.nextInt(words.size()));
     }
     public void start() {
        System.out.println(" Bienvenido a Wordle ");
        while (remainingAttempts > 0) {
            showTriesHistory();
            showAvailableLetters(); // Mostrar letras disponibles antes del intento
            String userGuess = getUserInput();
            triesHistory.add(userGuess);
            updateAvailableLetters(userGuess); // Actualizar letras disponibles
            if (userGuess.equals(secretWord)) {
                System.out.println("\n¡Felicidades! Has adivinado la palabra: " + secretWord);
                return;
            }

            System.out.println(WordleFeedback.feedBackString(userGuess, secretWord));
            remainingAttempts--;
```

```
        }
        System.out.println("\nHas perdido. La palabra secreta era: " + secretWord);
    }
    private void showTriesHistory() {// Numeros de intentos que te quedan
        for (String attempt : triesHistory) {
            System.out.println("\n"+attempt);
        }
        System.out.println("\nTe quedan " + remainingAttempts + " intentos");
    }
    private void showAvailableLetters() {// Letras que te quedan del abecedario
        System.out.print("\nLetras disponibles: ");
        for (char i = 'A'; i <= 'Z'; i++) {
            if (availableLetters.contains(i)) {
                System.out.print(i + " ");
         }  }
        System.out.println();
    }
    private void updateAvailableLetters(String guess) {
        for (char i : guess.toCharArray()) {
            if (!secretWord.contains(String.valueOf(i))) {
                availableLetters.remove(i); // Eliminar solo letras incorrectas
              } }}
    private String getUserInput() {// aqui es donde el usuario pone su respuesta mas
asegurando tamano
        String input;
```

```
            do {
                System.out.print("Introduce una palabra de " + WORD_LENGTH + " letras: ");
                input = scanner.next().toUpperCase();
                if (input.length() != WORD_LENGTH) {
                    System.out.println("Error: La palabra debe tener exactamente " + WORD_LENGTH
+ " letras.");
                }
            } while (input.length() != WORD_LENGTH);
            return input;
        }}
    class WordleFeedback {
    // Declaracion de colores
        private static final String GREEN = "\u001B[32m";
        private static final String YELLOW = "\u001B[33m";
        private static final String RESET = "\u001B[0m";
        private static final String RED = "\u001B[31m";
        private static String applyColor(String letter, String color) {
            return color + letter + RESET;  }
         public static String feedBackString(String guess, String secretWord){
        StringBuilder feedback = new StringBuilder();
        for (int i = 0; i < guess.length(); i++) {
            char guessedChar = guess.charAt(i);
            String guessedCharStr = Character.toString(guessedChar); // Convertir a String una
sola vez
            if (guessedChar == secretWord.charAt(i)) {
```

```
                feedback.append(applyColor(guessedCharStr, GREEN)); // Letra correcta en posición
correcta

            } else if (secretWord.contains(guessedCharStr)) {

                feedback.append(applyColor(guessedCharStr, YELLOW)); // Letra correcta en
posición incorrecta

            } else {

                feedback.append(applyColor(guessedCharStr, RED)); // Letra incorrecta

          }  }

          return feedback.toString();

      }}
```