



MINI PROYECTO 1: I KNOW THAT WORD

ANÁLISIS DE CLASES

FileManager

- ❖ Responsabilidades:
 1. Guarda el conjunto de palabras disponibles para el juego.
 2. Selecciona las palabras para cada ronda de juego.
 3. Leer desde un archivo de texto, las frases que se pueden usar en el juego.
- ❖ Funciones:
 - `lectureFile(String archivo)`

```
public ArrayList<String> lectureFile(String archivo) {  
    ArrayList<String> lista = new ArrayList<~>();  
  
    String PATH = "src/myProject/archivos/" + archivo;  
  
    try {  
        fileReader = new FileReader(PATH);
```

GUIGridBagLayout

- ❖ Responsabilidades:
 1. Grafica cada una de las fases del juego.
 2. Grafica si pierde o gana.
 3. Grafica los resultados de cada ronda.
 4. Grafica las palabras designadas para los niveles y las respuestas del jugador.
- ❖ Funciones:
 - `initGridBagLayout()`

```
*/  
private void initGridBagLayout() {  
    //Set up JFrame Container's Layout  
    this.getContentPane().setLayout(new GridBagLayout());  
    GridBagConstraints constraints = new GridBagConstraints();  
    this.setUndecorated(true);  
    //Create Listener Object and Control Object  
    palabras = new Palabras();  
    escucha = new Escucha();
```

- `resultados()`

```
public void resultados() {  
    if (elModelo.superonivel()) {  
        if (nivelActual == 10) {  
            resultado.setText("Has superado el nivel máximo, podrás repetirlo si deseas");  
            estadisticas.setText("Porcentaje aciertos: " + elModelo.porcentaje + "%");  
            reiniciarNivel.setVisible(true);  
            nivelMaximoSuperado = nivelActual;  
        } else {  
            resultado.setText("Has pasado este nivel, podrás avanzar al siguiente nivel");  
        }  
    } else {  
        resultado.setText("Has fallado este nivel, intenta mejorar tu respuesta");  
    }  
}
```

Jugador

❖ Responsabilidades:

1. Definir si el jugador está registrado o no.
2. Registrar el jugador en usuarios.txt
3. Establecer en qué nivel se encuentra el jugador.
4. Actualiza el progreso del jugador.

❖ Funciones:

➤ `estaRegistrado(String nombre)`

```
public boolean estaRegistrado(String nombre) {  
    if (listaUsuarios.contains(nombre)) {  
        puesto = listaUsuarios.indexOf(nombre);  
        return true;  
    } else {  
        return false;  
    }  
}
```

➤ `getNivel()`

```
public int getNivel() {  
    nivel = Integer.parseInt(listaNiveles.get(puesto));  
    return nivel;  
}
```

➤ `getNombre()`

```
public String getNombre() { return usuario; }
```

➤ `nombreVacio(String nombre)`

```
public boolean nombreVacio(String nombre) {  
    if (nombre == null || nombre == "" || nombre.length() < 3) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

➤ `registrarJugador(String nombre, int nivelSuperado)`

```
public void registrarJugador(String nombre, int nivelSuperado) {  
    String user = nombre;  
    int Level = nivelSuperado;  
  
    try {  
        fileWriter = new FileWriter( fileName: "src/myProject/archivos/Usuarios.txt", append: true );/  
        output = new BufferedWriter(fileWriter);  
        output.newLine();  
        output.write(nombre);  
    }
```

➤ `actualizarUsuario(String nombre, int nivelSuperado)`

```
public void actualizarUsuario(String nombre, int nivelSuperado) {  
  
    int indexAGuardar = 0;  
    String valorActualizar;  
  
    indexAGuardar = listaUsuarios.indexOf(nombre);
```

ModelKnowThatWord

❖ Responsabilidades:

1. Dar valores iniciales a todos los atributos.
2. Determinar el número de palabras por nivel.
3. Determinar que palabras usar para cada ronda.
4. Generar el listado de palabras para la ronda.
5. Calcular el porcentaje de aciertos que tuvo el jugador en la ronda.

❖ Funciones:

➤ `determinarPalabrasNivel(int nivelAJugar)`

```
public int determinarPalabrasNivel(int nivelAJugar) {
    switch (nivelAJugar) {
        case 1:
            palabrasNivel = 10;
            palabrasMemorizar = 5;
            aciertosNecesarios = 70;
            break;
        case 2:
            palabrasNivel = 14;
            palabrasMemorizar = 7;
            aciertosNecesarios = 70;
            break;
        case 3:
    }
}
```

➤ `getArrayPalabrasNivel()`

```
public ArrayList<String> getArrayPalabrasNivel() {

    for (int i = 0; i < palabrasNivel; i++) {
        int auxiliar = (int) (Math.random() * diccionario.size());

        if (!arrayPalabrasNivel.contains(diccionario.get(auxiliar))) {
            arrayPalabrasNivel.add(diccionario.get(auxiliar));
        } else {
            i = i - 1;
        }
    }
}
```

➤ `getArrayPalabrasAMemorizar()`

```
public ArrayList<String> getArrayPalabrasMemorizar() {

    for (int i = 0; i < palabrasNivel / 2; i++) {
        int auxiliar = (int) (Math.random() * arrayPalabrasNivel.size());
        if (!arrayPalabrasMemorizar.contains(arrayPalabrasNivel.get(auxiliar))) {
            arrayPalabrasMemorizar.add(arrayPalabrasNivel.get(auxiliar));
        } else {
            i = i - 1;
        }
    }
}
```

➤ `palabraEstaEnNivel(String palabra)`

```
public boolean palabraEstaEnNivel(String palabra) {

    if (arrayPalabrasMemorizar.contains(palabra)) {
        return true;
    } else{
        return false;
    }
}
```

➤ `calcularPorcentaje()`

```
    public int calcularPorcentaje() {  
  
        porcentaje = (aciertos * 100) / palabrasNivel;  
  
        return porcentaje;  
    }
```

➤ superoNivel()

```
public boolean superoNivel() {  
    if (porcentaje >= aciertosNecesarios) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

Palabras

❖ Responsabilidades:

5. Es el contacto con la lista de palabras para los niveles.
6. Da las palabras que encuentra en el archivo de Palabras.txt

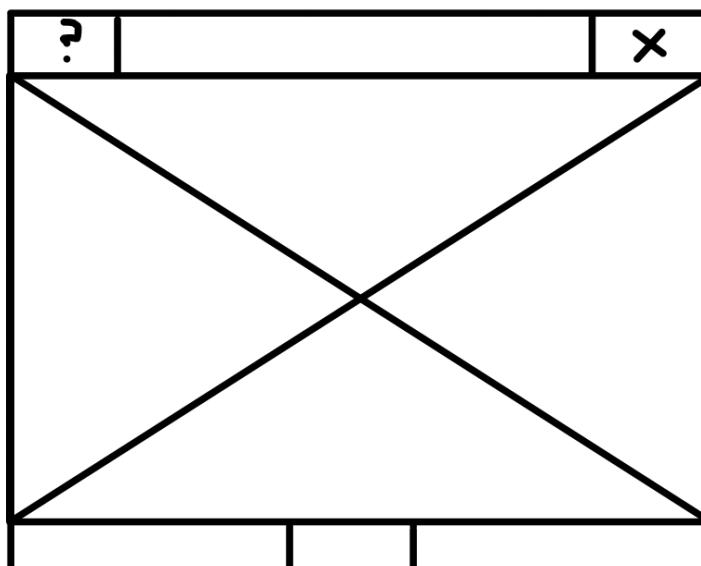
❖ Funciones:

➤ getPalabras()

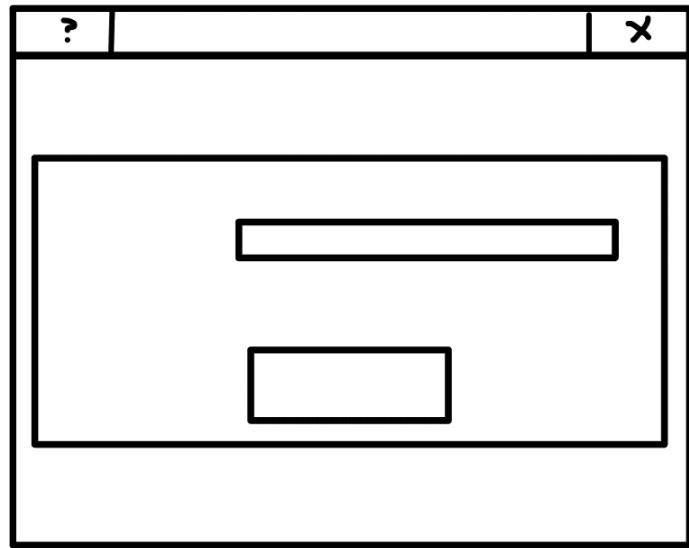
```
public ArrayList<String> getPalabras(){  
    return diccionario;  
}
```

BOCETOS DE INTERFAZ

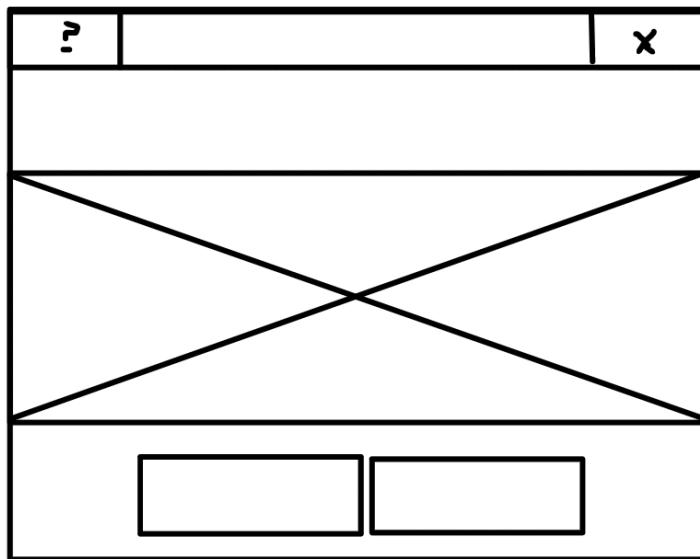
❖ Pantalla de inicio



❖ Nombre de usuario



❖ Interfaz de nivel



❖ Resultados

