Universidad Nacional de General Sarmiento

Licenciatura en Sistemas

Tecnicatura en Informática

Trabajo Práctico nº1
Ahorcado Swing

Programación III

1º semestre - 2022

Docente: Daniel Bertaccini

Integrantes: Alan Fernandez
Marcos Diaz
Soledad Ramirez

Com: 02

Introducción:

El objetivo del trabajo práctico es realizar una implementación del clásico juego “El Ahorcado”, el cual consiste en adivinar una palabra determinada considerando diferentes letras que expresa el jugador. Cuando la letra está presente en la palabra, se reemplaza la misma en los lugares correspondientes. El jugador gana si logra adivinar la palabra antes de cometer el máximo de errores permitidos, los cuales se cuentan cuando la letra expresada no se encuentra en la palabra.

En nuestra versión del juego, el usuario cuenta con cinco intentos posibles. Si comete más de cinco errores, pierde el juego. Si logra adivinar la palabra antes, es ganador.

Se utilizó el plugin de Eclipse Windowbuilder para construir la interfaz visual del juego.

Desarrollo del juego:

Para la construcción del juego utilizamos una arquitectura multicapa, donde se desarrolla la vista, el manejo de eventos y la lógica de negocios por separado.

Paquete Vista:

En el main se crea un objeto **Ventana,** el cual construye una página de inicio donde se incluyen las diferentes opciones de juego: jugar al juego clásico, jugar con palabras en inglés, jugar en modo difícil con palabras con mayor cantidad de letras y la opción de ingresar una palabra para que la computadora intente adivinarla. Utilizamos para esto los botones que facilita **Windowsbuilder.**

Una vez seleccionada la opción de juego, se visualiza una nueva ventana al crear un objeto que denominamos **Marco**. Este objeto contiene toda la información visual y los controles con los que jugará el usuario:

- labels que muestran la información de la palabra a adivinar, la palabra a adivinar (que se muestra en principio como una sucesión de guiones), la indicación del textarea donde el usuario puede ingresar la letra elegida, un sector donde se muestran todas las letras que va ingresando el usuario y los intentos restantes.
- botones que ayudan en el desarrollo del juego: **enviar** validará cada letra ingresada, **pista** facilitará visualizar una letra correcta de la palabra, con el juego terminado se muestran en pantalla botones para jugar nuevamente o ir al menú principal.

Si la opción elegida es **jugar al revés,** se creará otro objeto llamado **MarcoReves** el cual contiene otras labels y botones donde se visualiza la palabra elegida por el usuario, la letra elegida por la máquina, una lista de las letras elegidas durante la partida y si está incluida la misma en la palabra considerada, además de los intentos restantes. Se dispone de un botón **siguiente** el cual hace que la computadora elija una nueva letra, un botón **jugar otra vez** y **menú principal**.

Paquete Modelo:

Dentro de este paquete creamos la clase **ChequeaLetra** el cual cuenta con la lógica que maneja el procesamiento de las letras ingresada por el usuario.

Se inicializa este objeto cada vez que mostramos en pantalla el marco y se crean dentro de él un String **palabr**a, que será la palabra elegida aleatoriamente y dos arrays de char, el primero (**arrayConLetras**) con los caracteres de la palabra y el segundo (**arrayConGuiones**) con la cantidad de guiones que corresponde a la cantidad de letras de la palabra.

En el método **validarLetra()** se toma una letra ingresada y se pregunta si existe en la palabra, si está presente, se reemplaza en arrayConGuiones el guión o guiones correspondientes a la letra. En el caso de que no se encuentre en la palabra se baja en uno los intentos restantes.

El método **agregarLetraIngresadaALista()** agrega la letra a un array de letras ingresadas que se mostrará por pantalla.

El método **validarLetraRepetida()** devuelve true si la letra ingresada ya se encuentra en el array de letras ingresadas.

El método **actualizarPantalla()** actualiza el **arrayConLetras** con las letras ingresadas que son correctas y se muestran en pantalla.

El método **eleccionDeLetraDePc()** es un método para el modo **juegoAlReves** (en el cual la que adivina es la computadora). Primero se elige entre las vocales, luego dentro de las consonantes mas comunes y luego dentro de la totalidad de las letras.

Tambien se incluyen los getters y setters que se utilizan en los métodos.

Paquete Controller:

Incluimos dentro de éste las clases que controlan los diferentes eventos que tienen lugar durante una partida del juego y lo comunican con la lógica de negocio.

- Clase **ControladorInicial**: se crea cuando se inicializa un **marco** al presionar el boton **jugar**, y construye una nueva clase **ChequeaLetra** el cual maneja el desarrollo de una partida del juego. Actualiza el número de intentos restantes al inicio de la partida.
- Clase **ControladorDeLetras**: maneja el evento del botón **enviar**, se inicializa cuando se crea el **marco,** en el cual con el clic del usuario se chequea que la letra ingresada sea válida. Se implementa el método **validarLetraRepetida(),** también el método **validarLetra()** que reemplaza las letras en la palabra a

adivinar, **reiniciarJuego()** que inicia un nuevo juego cuando se pierde la partida por utilizar todos los intentos posibles, también se valida si el jugador pierde o gana, se implementa el método **actualizarPantalla()** (que reemplaza los guiones de la palabra por las letras correctas).

Estos métodos en general, actualizan los elementos visuales del marco: si las letras ingresadas se encuentran en las palabras, se muestran en la pantalla. Además se muestra una lista de letras ingresadas y si el jugador pierde se muestra una frase informativa y la palabra a adivinar y si gana se muestra otra frase informativa. Al momento de terminar la partida se muestran dos botones para iniciar un nuevo juego o acceder al menú principal.

- Clase **ControladorJuegoAlReves:** dibuja los guiones de la palabra elegida por el usuario en la pantalla cuando se presiona el botón **juego al revés**, también lanza un **JOptionPane** que recibe la palabra del usuario a adivinar por la computadora.
- Clase **ControladorJugarOtraVez:** controla el comportamiento del botón **jugar otra vez** que reinicia el juego teniendo en cuenta si el modo de juego es el clásico o el modo al revés.
- Clase **ControladorLetraAlReves**: se inicializa cuando se crea un **MarcoReves** (marco del modo **juego al revés**) el cual controla el botón **siguiente.** Se implementan diferentes métodos de **chequeaLetra** tales como **validarLetraRepetida(), eleccionDeLetraDePc(), validarLetra(),** que procesan las letras elegidas aleatoriamente por la computadora y comparan y dibujan las letras incluidas en la palabra elegida. Además actualiza la vista en base al desarrollo del juego.
- Clase **ControladorPista:** controla el comportamiento del botón pista, el cual va a mostrar una letra presente en la palabra. Se chequea las letras que aún no han sido adivinadas de la palabra considerada y de ellas se muestra una aleatoriamente. Se utilizan los métodos de **ChequeaLetra validarLetra(), actualizarPantalla(**), y se valida si el jugador ganó ya que decidimos que es posible adivinar la última letra con una pista.
- **ControladorVentana** es una clase que maneja el botón **menu principal,** la cual dirige a la pantalla de inicio.

Conclusiones:

Con este trabajo práctico pudimos notar las ventajas de desarrollar con una arquitectura multicapa, donde las diferentes funciones del programa se encuentran en diferentes capas y se comunican de manera controlada. Al ir complejizando el juego, fue más simple poder modificar cada capa sin afectar las demás, por lo que el desarrollo se vuelve más eficiente.