Práctica Final Programación II

16/06/2022

—

Christian Gil Ledesma

Enlace vídeo: https://www.youtube.com/watch?v=EMiOLZMjey0

Montes de Oca Durán, Juan Antonio

Introducción

En la siguiente práctica nos hemos propuesto como objetivo programar un juego basado en el conocido juego del siset.

La mecánica del juego es simple, se juega con una baraja francesa con todas las cartas sin comodines. Al principio del juego se reparten todas las cartas entre los jugadores y el que primero consiga desprenderse de todas sus cartas en un determinado orden resulta ganador. El orden de la colocación de las cartas empieza en 7 seguido de sus vecinos, y los vecinos de estos una vez los anteriores hayan sido colocados.

El juego se ha diseñado de tal forma que tendremos un jugador humano y tres jugadores IA. El primero en jugar será el jugador humano seguido de las tres IAs en orden. El jugador podrá interactuar con la aplicación mediante botones y podrá seleccionar las cartas que desee (y pueda) colocar en el tablero pinchando encima de estas.

Diseño

Podemos diferenciar 2 grandes bloques de clases que componen el programa:

1. ***Package interfaz***

a. PracticaFinal

i. **Descripción**: es la clase mas importante del programa. Contiene la funcionalidad principal tanto de la lógica como de la interfaz y es la que se encarga de utilizar las funcionalidades adicionales del resto de clases mediante instancias. Contiene el método Main.

ii. El **Constructor** es el que se encarga de llamar a otros subeventos de la clase para ejecutar en conjunto toda la funcionalidad del programa.

iii. El método **anyadirEventos** se encarga de gestionar tanto los eventos de los botones como del ratón.

iv. El método **configurarJFrame** gestiona la personalización del JFrame principal “ventana”.

v. El método **configurarPanelSuperior** gestiona la interfaz correspondiente a los paneles de las puntuaciones de la IA, al tablero de cartas y la mano del jugador.

vi. El método **configurarPanelInferior** gestiona la interfaz correspondiente al panel de los botones y del cuadro de texto.

vii. El método **ventanaEmergente** inicia un nuevo JFrame indicando que un jugador ha ganado

viii. El método **rePintarCaraNorte** mediante un parámetro numérico que entra como opción gestiona la interfaz del panel de las puntuaciones de la IA (situado en el “Border.Layout.NORTH” del “panelSuperior”).

ix. El método **rePintarCaraCentro** mediante un parámetro numérico que entra como opción gestiona la interfaz del panel del tablero de juego (situado en el “Border.Layout.CENTER” del “panelSuperior”).

x. El método **rePintarCaraSur** mediante un parámetro numérico que entra como opción gestiona la interfaz del panel de las cartas del jugador humano (situado en el “Border.Layout.SOUTH” del “panelSuperior”).

xi. El método **rePintarBotones** mediante un parámetro de tipo Enum que entra como opción gestiona los botones que son mostrados en el panel de botones (situado en el "Border.Layout.NORTH" del "panelInferior").

xii. El método **crearJugadores** crea las diferentes IAs y el jugador humano "Player" utilizando objetos de tipo Jugador.

xiii. El método **repartirCartas** reparte las diferentes cartas del tablero de juego entre las IAs y el jugador humano.

xiv. El método **turnoIA** realiza el turno de una IA cuyo índice es pasado por parámetro.

xv. El método **comprobarGanador** compara las cartas actuales de un jugador para ver si son cero, en cuyo caso este ha ganado.

b. TableroEtiquetasMio

i. **Descripción**: es un JPanel personalizado que incluye un array bidimensional de etiquetas de la clase "EtiquetaMia" que actúan como JLabels personalizados.

ii. El **Constructor** se encarga de inicializar los atributos de la clase y personalizar el JPanel del cual se extiende.

iii. El método **setEventoMouse** le asigna un evento de tipo "MouseListener" a una etiqueta en concreto.

iv. El método **setCartaToEtiqueta** le asigna una carta de la clase "Carta" a una etiqueta en concreto.

v. El método **cambiarImagenEtiqueta** cambia el tipo de imagen de una etiqueta en concreto a la indicada por parámetro.

vi. El método **setSeleccionable** cambia el booleano de una etiqueta en concreto para saber si puede ser seleccionada por el jugador.

c. **EtiquetaMia**

i. **Descripción**: es un JLabel personalizado que incluye una variable de tipo “Carta”.

ii. El **Constructor** se encarga de inicializar los atributos de la clase y personalizar el JLabel del cual se extiende. También asigna una imagen predeterminada según el parámetro numérico pasado.

iii. El método **cambiarImagen** Cambia la imagen de la etiqueta dependiendo del parámetro numérico pasado.

iv. El método **redimensionarImagen** redimensiona una imagen a la escala de la propia etiqueta.

d. **JButtonMio**

i. **Descripción**: es un JButton personalizado.

ii. El **Constructor** se encarga de personalizar el JButton del cual se extiende.

iii. El método **interactuable** cambia la forma de ver del botón en cuestión y desactiva el poder interactuar con este y viceversa dependiendo del valor del booleano pasado por parámetro.

2. *Package logica*

a. Tablero

i. **Descripción**: contiene un array bidimensional de cartas de la clase "Carta" que simula el tablero de juego y una variable que indica las cartas restantes de este.

ii. El **Constructor** se encarga de inicializar el array bidimensional y setear las cartas restantes a 0.

iii. El método **llenarTablero** pasa las cartas del array de la clase "Baraja" al array bidimensional del tablero "cartas".

iv. El método **sacarCarta** extrae una carta del array "cartas".

v. El método **meterCarta** añade una carta al array "cartas".

vi. El método comprobarHueco comprueba si una carta pasado por parametro puede se colocada en el array bidimensional "cartas".

vii. El método **getNumFilaPalo** devuelve un int que representa la fila en el array bidimensional de un palo.

b. Baraja

i. **Descripción**: contiene un array de cartas, unos atributos que indican el número de palos y el máximo de cartas y una variable que indica las cartas restantes.

ii. El **Constructor** inicializa el array creando cartas ordenadas según el tipo de palo y número de la carta.

iii. El método **mezclarBaraja** utiliza el algoritmo de "Fisher-Yates" para aleatorizar la posición de las cartas.

iv. El método **sacarCarta** extrae una carta del array.

v. El método **noQuedanCartas** lanza una excepción cuando la baraja se queda sin cartas.

c. **Jugador**
   i. **Descripción**: un jugador se define por un nombre y una mano, la cual es un array de cartas.
   ii. El **Constructor** inicializa los atributos (la mano con las cartas a null).
   iii. El método **anyadirCarta** añade una carta pasada por parámetro a la mano.
   iv. El método **sacarCarta** extrae una carta pasada por parámetro de la mano.

d. **Mano**
   i. **Descripción**: una mano se define por un array de cartas, una constante de máximo de cartas por jugador y una variable de cartas restantes.
   ii. El **Constructor** inicializa los atributos de la clase (el array de cartas con las posiciones a null).
   iii. El método **anyadirCarta** añade una carta pasada por parámetro al array de cartas.
   iv. El método **sacarCarta** extrae una carta pasada por parámetro del array de cartas.

e. **Carta**
   i. **Descripción**: es la representación de una carta real. Sus atributos son el número de la carta, su palo (Enum), un booleano que indica si ha sido colocada en el tablero y un String con la ruta de la imagen de la propia carta.
   ii. El **Constructor** inicializa los atributos de la clase.
   iii. El método **rellenarRutasCartas** añade el String correspondiente a la ruta de la propia carta basándose en el palo y el número de la carta.

f. **OpcionRepaintBotones**
   i. **Descripción**: un Enum con las opciones sobre que botones mostrar y como mostrarlos.

g. **Palo**
   i. **Descripción**: un Enum con los cuatro palos de las cartas.

Conclusión final

Con esta práctica he podido aprender a manejar y consolidar varios de los conceptos explicados durante el curso en lo que se refiere a componentes gráficos y cómo manipularlos.

La parte más difícil de resolver en mi caso fue cambiar mi primera versión de la práctica en la que utilizaba Graphics para mostrar la interfaz a la versión final en la que todo funciona con labels con imágenes sobre paneles.