Memoria de la Práctica: PUZZLE
Programación II

Profesor de Prácticas: Montes de Oca Durán, Juan Antonio

Autor: Elena del Pilar Fernández Wyzynska GRUPO: 3

INTRODUCCIÓN:

El Problema propuesto es un puzzle, donde desde una carpeta se escoge una imagen aleatoria y el usuario decide en cuantas subdivisiones quiere que se parta la imagen tanto horizontales como verticales.
El Juego consta de otros tres botones principales que son el historial general, el historial selectivo que dado un nombre se imprimen solo el historial de esa persona, y por último salir.
Tener en cuenta que una vez iniciada la partida no se puede acceder a ninguna de las otras funcionalidades hasta que se gana y el usuario recibe 10 puntos o abandona la partida donde recibirá 0 puntos.
Al terminar la partida se podrá visualizar la imagen del puzzle completa.

DISEÑO:

IMPORTANTE:
PARA **CAMBIAR EL DIRECTORIO** DE LAS CARPETA PARA LAS **IMÁGENES ALEATORIAS** SE DEBE DE HACER EN LA **CLASE FICHERO ALEATORIO**

```
public class FicheroAleatorio {
    File randomImageFile;
    public FicheroAleatorio(){
        File Directorio = new File("Imagenes");
       File[] imageFiles = Directorio.listFiles();
       if (imageFiles != null && imageFiles.length > 0) {
            Random random = new Random();
             randomImageFile = imageFiles[random.nextInt(imageFiles.length)];

    }


}
```

Para el Diseño de la Práctica podemos englobar las clases en tres grupos principales quitados del main el cual es:
-PracticaFinal.java

- Ficheros:
  - AdicionObjectOutputStrea.java
  - JugadorObjetoFicherosAdicion.java
  - JugadorObjetoFicherosLectura.java
  - Historial.java
- Lógica:
  - FicheroAleatorio.java
  - Jugador.java
  - PosicionPiezas.java

- Interfaz Gráfica:

- SubImagen.java
- panelSubimagen.java
- lecturaDatos.java

Ficheros: Para la escritura y lectura de fichero escogí la clase **ObjectOutputStream** que serializa la clase Jugador, para ello al construir un objeto Jugador tengo que cerciorarme que tiene siempre tienen las mismas dimensiones los Atributos de Jugador.

Las clases JugadorObjetoFicherosAdicion y JugadorObjetoFicherosLectura son para la escritura y lectura de los objetos, teniendo en cuenta que JugadorObjetoFicherosAdicion es extendida por AdicionObjectOutputStrea para poder añadir objetos en un fichero y que se mantenga toda la información anterior.

La clase Historial recoge todas las anteriores para poder escribir y leer adecuadamente del fichero con sus respectivas gestiones de error. Los métodos más importantes de esta clase son:

- **Guardar Historial**: Que pasado un objeto Jugador por parámetro nos guarda en la última posición el objeto en el fichero.
- **Historial General**: Es un método que nos devuelve un String de todos los objetos jugador que también han sido pasados con un toString de la clase Jugador.

- **Historial Selectivo:** Es un método que pasado un Char Array por parámetro nos devuelve todos los objetos Jugador con ese nombre en un String.

Lógica: En esta agrupación tenemos diferentes clases que sirven para la lógica del juego.

- FicheroAleatorio: Es una clase donde al construirla escoge un archivo aleatorio del directorio puesto, tenemos dos métodos donde podemos devolver el nombre del archivo como File o como toString según nuestras necesidades.
- Jugador: La clase Jugador se almacena de manera momentánea los datos del Jugador, El nombre de máximo 15 caracteres, la fecha de la partida y la puntuación conseguida.
    - **Equals:** Es un método que nos compara en nombre del objeto Jugador con un char array pasado por parámetro y nos devuelve true si son iguales
- PosicionPiezas: Esta clase es muy importante para el funcionamiento correcto del juego aqui se almacena el numero de columnas y filas, y con ello tenemos dos arrays que uno es la posición piezas donde están ordenadas y otro es un array de la posicionactual, con todos los cambios reflejados.
  A parte de eso guardamos los puntos de corte de la imagen en dos arrays posx, posy para poder montar cada vez el puzzle de manera correcta pasando estos parámetros.
    - **SetCoordenadas**: Nos guarda en ambos arrays las posiciones de corte de cada pieza al igual que crea los arrays de posicionpiezas y posicionactual.
    - **desordenar**: método que nos desordena las piezas del puzzle
    - **cambio piezas**: Dados dos int nos cambia las información de aquellas posiciones de los arrays posx, posy, posicionactual.
    - **ordenadas**: Nos compara los dos arrays posicionactual y posicionpiezas si son iguales si lo son nos devuelve un true.

Interfaz Gráfica: En esta agrupación se encuentran aquellas clases que son para la visualización de objetos gráficos.

- SubImagen: Es una extensión de JLabel donde pasados por parámetros la imagen las dimensiones nos crea el JLabel
- panel Subimagen: La principal función de panelSubImagen es de crear todos los JLabels necesarios para la creación del puzzle con el trozo de imagen correspondiente y meterlos en el panel.
  Esta clase toma ayuda de PosicionPiezas, la cual es pasada como parámetro, para coger adecuadamente el trozo de imagen correcto así como se van creando las piezas
- lecturaDatos: Esta clase tiene una función muy definida que son las ventanas emergentes de cuando el usuario crea una nueva partida o cuando quiere acceder al Historial Selectivo.
  En el constructor si se le pasa un 0 se abrirá una pestaña emergente con el estilo de iniciar partida y si se le pasa un 1 con el estilo de Historial Selectivo

CONCLUSIÓN:

Esta práctica ha supuesto un reto a nivel de hacer funcionar todo adecuadamente y han sido muchas las horas en hacer que las cosas quedasen exactamente como uno quería, uno de mis grandes errores fue no pensar bien desde el principio todas las clases necesarias.
Las partes que más me costaron fueron sin duda la lógica de las piezas del puzzle puesto que no comprendía qué tenía que instanciar una y otra vez el puzzle para que se viera el cambio realizado.
Ha sido un proceso más bien largo donde he tenido que escarbar y comprender bien cómo funcionan las interfaces gráficas para poder implementarlo todo adecuadamente, por ejemplo los cambios de pantalla, un gran error mio fue no instanciar los JPanels con el cardLayout, y al querer cambiar de pantalla siempre me salian errores cosa que me hizo perder mucho tiempo.
Pero encuentro que la práctica es muy buen ejemplo de lo aprendido en Programación II y que te hace realmente comprender lo aprendido en clase en su profundidad.

ENLACE A VÍDEO: