Instituto tecnológico de Costa Rica

Programación orientada a objetos

Introducción del diseño al software

Isaac Herrera Monge

2019160455

29/11/2024

II Semestre 2024

William Mata Rodríguez

Table of Contents

Futoshiki

Futoshiki es un juego de rompecabezas basado en tableros, también conocido bajo el nombre de Desigual. Se puede jugar en una tabla cuadrada que tiene un tamaño fijo dado (4x4 por ejemplo. El propósito del juego es descubrir los dígitos ocultos dentro de las celdas del tablero; cada celda se llena con un dígito entre 1 y el tamaño de las tablas. En cada fila y columna cada dígito aparece exactamente una vez; por lo tanto, cuando se revelan, los dígitos del tablero forman un llamado cuadrado latino.

Temas investigados

Detallaremos los temas que se tuvo que investigar por aparte para el desarrollo del proyecto, estos temas no se vieron necesariamente en el curso o en clase y se reforzó o se investigó por aparte

Implementación de Jframe externos

La idea principal del proyecto fue crear un Jframe que cargara el tablero y funcionara independiente del tamaño de la matriz, esto por que quise implementarlo todo en una sola clase, sin embargo esto resultó ser bastante difícil al inicio pues el editor de froms predeterminado de IntelliJ El IDE utilizado para el proyecto) no acepta que un programa use su editor al mismo tiempo que manualmente estás dibujando por medio del código, esto representó un gran problema que sinceramente se pudo solucionar fácil, pero no me gustaba la idea de crear 6 vistas idénticas para cada una de las dimensiones indicadas. Tras investigar encontré que podía crear una clase que extendiera de Jframe y me permitía importarlo en el editor. Un ejemplo fue usado de https://chortle.ccsu.edu/java5/notes/chap56/ch56_11.html y se siguió las recomendaciones de https://stackoverflow.com/questions/15867148/why-do-we-need-to-extend-jframe-in-a-swing-application

GridLayouts

El gridLayout es uno de los tantos organizadores de elementos en Java, en este caso ese layout me quedaba perfecto pues crea una matrix de n por m y lo acomoda de tal manera que conserve el aspecto y acomode los elementos. Los botones para el juego fueron creados dentro de este Layout permitiendo

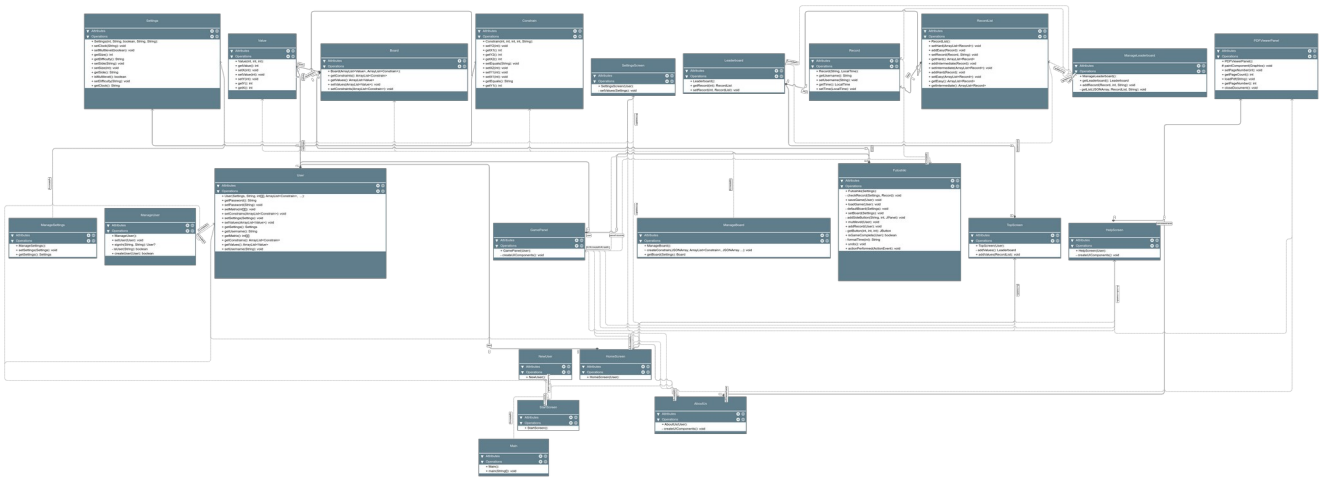
que no tengamos que preocuparnos tanto por la ubicación o el acomodo. Para esto consultamos la documentación de Oracle https://docs.oracle.com/javase/tutorial/uiswing/layout/grid.html

PDF Reader

Con la ayuda de la librería commons-logging, fontbox y PDFBox somos capaces de cargar el PDF en las ventanas de About Us y Help. Aquí desplegaremos el PDF con el manual de usuario y este PDF, con la intención de no tener que colocar la misma presentación, la información sobre la librería se puede acceder a través de https://pdfbox.apache.org/

Solución

Modelo UML



Puede acceder al diagrama interactivo en el siguiente enlace https://www.yworks.com/yed-live/?file=https://gist.githubusercontent.com/Isaacoun100/9122765884b5c45acd51ef15a3ef146c/raw/96baa41b8a21554a997c3fde199b29683c461536/Imported%20Document

Implementación del MVC

Igual que en el proyecto pasado, se implementó el MVC para el desarrollo del proyecto, en la carpeta Futoshiki/src/view podemos ver las vistas que son con las que el usuario interactúa, sin embargo, para poder acceder y recibir esa información tenemos el controlador Futoshiki/src/controller que es el encargado de administrar los datos y por último tenemos el modelo, que contiene los datos y la lógica

Se implementó durante todo el proyecto, particularmente podemos hablar de la clase Futoshiki/src/view/Futoshiki.java que contiene la lógica de la vista en sí, es decir, al capa de interacción con el usuario, pero cuando por ejemplo queremos dibujar el tablero o las condiciones de la partida entonces se usa Futoshiki/src/controller/ManageBoard.java para poder acceder a las condiciones del tablero, sin embargo, ManageBoard acceder a su vez a Futoshiki/src/model/Board.java y Futoshiki/src/bin/boards.json para saber las dimensiones del tablero, si es con temporizador o cronómetor, si es multinivel, etcétera.

Conclusiones

Problemas encontrados y soluciones

- El problema principal fue el desarrollo del tablero, más allá de sólo crear una matrix lo difícil fue encontrar la manera de hacer que el tablero fuera responsivo al tamaño de la matriz, es decir, que pudiera escalar de nxn al tamaño que fuera necesario. El problema no radicaba en que los botones fueran accesibles en la matriz, ya que por ejemplo, si creo un botón en una matriz 10x10, cómo lo referencio? Cómo lo renombro?. Reconozco que hubo testarudez de parte mía en querer hacer la solución de esta manera pues incrementó la complejidad del proyecto. Sin embargo resultó ser una solución mejor, rápida y elegante.
    - La clave estuvo en entender cómo Java maneja su queue de memoria, en otros lenguajes como C esto sería mucho más sencillo pues se accede a memoria, sin embargo con Java no es el caso, o por lo menos eso creí, aprendí de un comportamiento de cómo Java trabaja con puntero detrás de escena pero que el usuario no ve, esto lo logré creando un ArrayList que contenía los botones cuando se creaban, en sí el Array no contiene más que una referencia a los botones y permite entonces que al reverenciarlo permita su acceso y modificación pero a través del array.

Aprendizajes Obtenidos

Cuando aprendí a usar Java lo hice através de Netbeans, sin embargo, cuando llegué a la universidad conocí de IntelliJ, una herramienta que resulta ser superior. Cuando utilicé Java en la universidad fue para hacer servidores, sprintgboot, APIs, sockets, en fin, no había interfaces involucradas (Backend), no estaba preparado para el funcionamiento de interfaces en IntelliJ. Pese a que ahorita lo manejo con facilidad por que entendí su funcionamiento, al inicio fue bastante frustrante no poder modificar la interfaz como yo quería, esto resultó dando frutos pues era la implementación más natural al proyecto, en definitiva tomó mucho tiempo diseñar el código en sí antes de implementarlo

Lista de revisión del proyecto

Concepto	Puntos Originales	Avance	Puntos obtenidos	Análisis
Opción Jugar (despliegue del juego según configuración)	12	100		
Botón Iniciar Juego	10	100		
Crear Top 10	8	100		
Botón Borrar Jugada	5	100		
Botón Rehacer Jugada	5	100		
Botón Borrar Juego	3	100		
Botón Terminar Juego	2	100		
Botón Guardar Juego	5	100		
Botón Cargar Juego (incluye el despliegue del mismo)	9	100		
Opción Top 10	5	100		
Opción Configurar Datos del 1 al 5	5	100		
Dato 6 (jugador)	9	100		
Ayuda (manual de usuario)	5	100		
Cronómetro tiempo real	5	100		
Temporizador tiempo real	5	100		
Implementación juego multinivel	10	100		
Total	100	100		
Partes desarrolladas adicionalmente				

Bibliografía

Usage of JFrames. (s. f.). https://chortle.ccsu.edu/java5/notes/chap56/ch56_11.html. https://chortle.ccsu.edu/java5/notes/chap56/ch56_11.html

How to Use GridLayout (The Java™ Tutorials > Creating a GUI With Swing > Laying Out Components Within a Container). (s. f.). https://docs.oracle.com/javase/tutorial/uiswing/layout/grid.html