PROYECTO PROGRAMACIÓN AVANZADA

Ander Monreal Ayanz

Empezaré por la explicación del juego del OSO,

El juego en cuestión se basa en turnos, donde los jugadores escriben una letra "O" o "S" en uno de los cuadrados disponibles en el tablero. El objetivo principal es formar la palabra "OSO", y el jugador que logre formarla más veces será el ganador.

Cuando un jugador consigue formar la palabra "OSO", las letras utilizadas en esa formación se tachan y ese jugador tiene la oportunidad de seguir jugando colocando una letra adicional en otro cuadrado vacío. Al principio del juego, los jugadores se turnan para colocar sus letras, y es difícil cometer un error que permita al otro jugador anotar puntos. Sin embargo, a medida que los cuadrados del tablero se van llenando, las opciones para evitar la formación de palabras se reducen, lo que puede llevar a una secuencia de "osos" consecutivos.

Es importante mencionar que el jugador que comienza tiene una pequeña desventaja en comparación con el segundo jugador, por lo que a menudo se decide quién comenzará mediante un sorteo al principio. Si se juegan varias partidas consecutivas, los jugadores se turnarán para iniciar cada partida. El juego llega a su fin cuando todos los cuadrados del tablero se han llenado, y el tamaño del tablero puede variar dependiendo de la duración que se desee para el juego.

En la versión original del juego, que se juega con papel y lápiz, existen dos formas de jugar: puntuando solamente los "OSO" escritos en forma horizontal y vertical en la cuadrícula, o también puntuando los escritos en diagonal. La segunda opción es un poco más desafiante y requiere mayor atención para evitar cometer errores, ya que una vez que se ha obtenido la palabra "oso", es responsabilidad del jugador tachar las letras correspondientes. Sin embargo, en esta versión del juego que estás aprendiendo, el ordenador se encargará de tachar los "OSO" conseguidos.

Este proyecto ha pasado por varias versiones, infinidad de cambios, clases movidas de paquete en paquete, hasta al final llegar a un código en el que se ajusta a ciertos puntos de la entrega.

Esta entrega contiene un paquete prueba en el que como su nombre indica, finalmente fue mi última prueba en la que quise implementar al servidor una pequeña interfaz donde el usuario podría elegir el tamaño del tablero, pero por falta de tiempo y acumulacion de muchos examenes y entregas me ha sido imposible terminar esta parte.

- Clase Servidores.java:

Esta clase contiene un MAIN, es la encargada en iniciar el servidor, al ejecutarse creará dos objetos uno de tipo ServerChat y el otro de tipo ServerGame, ambos objetos extienden de la clase Thread por lo que representan dos hilos que actuarán simultáneamente, por lo que siguiente iteración después de crear los objetos será iniciarlos con el método .start( );.

No contiene ningún método ni variables.

- Clase ServerChat.java:

Esta clase extiende de la clase Thread, y se encarga de representar uno de los hilos creados por Servidores.java, su función es llevar el servidor de un chat en el que varios clientes puedan establecer comunicación cada uno desde su máquina y sin ningún problema. Al aplicar el método run( ) sobre esta, se creará un socket en un determinado puerto (diferente al de la clase ServerGame.java) y acto seguido, hasta que no sea interrumpido o el número de clientes conectados sea superior a 2, irá aceptando clientes (máximo 2) y creando un objeto clienteThread que más adelante hablaremos de dicha clase, pero básicamente se encarga de crear dos hilos más, uno por cliente, con el fin de llevar el control de cada cliente.

Tiene dos variables:

- final int port; → Se encarga de establecer una conexión en el puerto dado por esta variable.

- final List<ClientThreadChat> clientes = new LinkedList<>(); → Se encarga de controlar las conexiones de los clientes guardandolos en una lista.

Contiene el método run( ), ya he explicado de que se encarga este método run, extendido de la clase Thread.

- Clase ServerGame.java:

Esta clase extiende de la clase Thread, y se encarga de representar el otro hilo creado por Servidores.java, su función es llevar el servidor y controlar una partida del juego especificado en la segunda página, de tal forma en la que dos jugadores puedan jugar hasta en diferentes máquinas. Al aplicar el método run( ) sobre esta, se creará un socket en un determinado puerto (distinto al de la clase ServerChat.java) y acto seguido, aceptará la conexión de los clientes y creará dos objetos clientThread uno para cada jugador que más adelante comentaremos dicha clase, pero básicamente se encarga de crear hilos de esta forma este servidor podrá atender a ambos clientes a la vez y llevará un poco las riendas de la partida.

Tiene tres variables:
- Juego partida → Un objeto de la clase Juego.java, donde está el control de la partida.
- final int port → Se encarga de establecer una conexión en el puerto dado por esta variable.
- final List<ClientThreadChat> clientes = new LinkedList<>(); → Se encarga de controlar las conexiones de los clientes guardandolos en una lista.

Contiene el método run( ), ya he explicado de que se encarga este método run, extendido de la clase Thread.

- Clase ClientThreadJuego.java y ClientThreadGame.java:

Resumo estas clases en una ya que tienen funciones similares, la del juego acorde con la partida y la del chat acorde al chat, ambas son las encargadas de recibir un mensaje de un cliente y enviarselo a todos de forma muy resumida, de esta forma conseguimos la comunicación entre clientes, antes

de efectuar el mandado de mensajes, se encarga de sincronizar a los clientes de esta forma evitaremos errores.

Variables:

Ambas tienen un:

- final Socket socket;
- DataOutputStream out → Encargado de mandar la información por el stream
- final List<ClientThread> clientes → Lista con los clientes

Cabe resaltar que la del juego contiene también una variable Juego

Los métodos también son similares:

- Synchronized public void sendMsg (String msg) → Sincronizados los clientes manda el mensaje por la variable out.
- run ( ), encargado de iniciar la ejecución del hilo como ya sabemos

- Clase Cliente.java:

Es una clase ejecutora, el constructor de la clase recibe un objeto de configuración y realiza varias tareas, como establecer el nombre del cliente, crear sockets para el chat y el juego, obtener los flujos de entrada y salida de los sockets y recibir el tamaño del tablero. El código crea hilos para el chat y el juego, los cuales se encargarán de manejar la comunicación y la lógica correspondiente. Estos hilos se inician mediante el método start().

El código también incluye métodos para configurar la configuración del cliente, establecer el tablero, enviar mensajes al juego y enviar mensajes al chat. Por último, el método main() crea una instancia del menú y la muestra en la interfaz gráfica.

En resumen, este código Java representa la implementación de la clase Cliente en un programa de chat y juego, donde se establecen las conexiones, se gestionan los mensajes y se controla la interacción entre el cliente y el servidor.

- Clase Menu.java:

La clase Menu es una interfaz gráfica de usuario que muestra un menú con varias opciones para el juego. La clase tiene varios atributos, como un objeto Cliente, un objeto Tablero y un objeto ConfiguracionCliente. El constructor de la clase Menu inicializa la interfaz gráfica y desactiva el botón "Nueva Partida" inicialmente, de esta forma hasta que no haya una configuración, el programa no podrá iniciarse.

- Clase ConfiguracionCliente.java:

Es una interfaz gráfica que se ejecuta una vez la haya activado el menú, su función es coger la configuración del cliente, host, puertos y nombre.

- Clase EleccionLetra.java:

Es una interfaz gráfica que se encarga de mostrar si elegir la S o la O.

- Clases HiloChat.java y HiloGame.java:

Son los hilos que usará el cliente para poder conectarse tanto al servidor del chat como al del juego.

Esto es un poco resumido todas las clases rápidamente