MEMORIA TÉCNICA JUEGO DEL OSO

El juego del Oso es un juego de estrategia simple en el que dos jugadores se turnan para escribir una letra "O" o "S" en una de las casillas de un tablero. El objetivo es formar la palabra "OSO" en cualquier dirección: horizontal, vertical o diagonal. El jugador que forma la palabra "OSO" más veces gana el juego. Este documento describe la implementación del juego del Oso.

El sistema está compuesto por estos tres componentes principales:

- Cliente (JuegoClienteGUI.java): Implementa la interfaz gráfica del juego usando Java Swing.
- Servidor (ServidorJuego.java, ClienteHandler.java): Gestiona la lógica del juego y el chat y la comunicación entre los clientes.
- Juego (Juego.java, Tablero.java, Casilla.java): Gestiona el transcurso de la partida.

Clases

Clase JuegoClienteGUI: Conectar al servidor, iniciar el tablero y gestionar los turnos de juego y procesa mensajes recibidos del servidor para actualizar el estado del juego.

Atributos:

- **JButton[][] matriz_botones**: Matriz de botones que representa el tablero del juego.
- **String nickname, servidor, puerto:** Strings para guardar el nombre del cliente, el nombre del servidor y el puerto de conexión con el servidor.
- **Int filas, columnas:** Tamaño del tablero
- **Thread recibirThread:** Hilo para recibir mensajes del juego y actualizar el tablero y **etiquetas.**
- **Socket socket:** Socket con el que se comunica con el servidor.
- **DataInputStream in:** Para identificar la llegada de mensajes.
- **DataOutputStream out:** Para identificar la salida de mensajes.
- **Boolean miTurno, ganador, empate:** Booleanos que sirven para saber si es mi turno y para manejar el resultado.

Métodos:

- **Los actionPerformed de cada botón**: Gestionan las acciones a hacer tras pulsar los botones.
- **iniciarTablero:** Inicia el panel del tablero y la matriz de botones, también establece a cada botón un actionPerformed para gestionar cuando se pulsa.
- **conectarCliente:** Inicia la conexión con el servidor y el hilo para recibir mensajes del servidor y del juego.
- **mostrarAvisoFin, mostrarAvisoTurno, mostrarAvisoConfiguracion:** Crean jDialogs para cuando se finaliza la partida, cuando un cliente quiere jugar y no es su turno y cuando

se quiere realizar una conexión pero no se han establecido las características del nombre del cliente, servidor o puerto.

- **enviarJugada:** Envía un mensaje con la jugada que ha realizado a través del socket al servidor
- **procesarMensaje:** Maneja los mensajes recibidos del servidor y del juego y en función de ellos modifica el chat o la lógica del juego.

Interfaz Gráfica: componentes

- Menú de opciones de la partida con opciones desplegables
  - Conectar: realiza la conexión con el servidor del juego.
  - Configuración: el cliente establece su nombre y el servidor y puerto al que quiere conectarse.
  - Abandonar: el cliente puede abandonar el servidor al que está conectado.
- Matriz de botones que representa el tablero del juego.
- Área de texto para representar el chat y un campo de texto con un botón para enviar mensajes por el chat.
- Etiquetas para mostrar el turno del jugador y los contadores de puntos.

Clase ServidorJuego: Gestiona la recepción de conexiones de clientes manteniendo una cola de clientes para emparejarlos y comenzar partidas. Además, instancia y gestiona el juego.

Atributos:

- **Int filas, columnas:** Representa el tamaño del juego
- **Int port:** Representa el puerto de recepción.
- **Juego juego:** Es una instancia de la clase Juego para manejar el transcurso de la partida.
- **Map<String, ClienteHandler> clients:** Es un HashMap donde vamos a tener control de los clientes conectados.
- **Queue<ClienteHandler> colaClientes:** Es una cola donde se van almacenando los clientes que llegan para gestionar las partidas

Métodos:

- **iniciar:** Crea el hilo de recepción de clientes y maneja su creación.
- **nuevoCliente(ClientHandler):** Añade los clientes a la cola y mapa y maneja el comienzo de partidas
- **removerCliente(String):** Elimina el cliente.
- **enviarATodos(String):** Manda el mensaje a todos los clientes conectados.
- **procesarEntrada(String, ClientHandler):** Procesa los mensajes recibidos por el cliente.
- **main:** Crea la instancia del servidor.

Clase ClienteHandler: Maneja la comunicación con un cliente individual, escucha y procesa mensajes del cliente y envía mensajes al cliente.

Atributos:

- **Socket socket:** Es el socket con el que el servidor se comunica con el cliente
- **DataInputStream in:** Para identificar la llegada de mensajes.
- **DataOutputStream out:** Para identificar la salida de mensajes.
- **ServidorJuego servidor:** Es el servidor con el que se comunica el cliente.
- **String nickname:** Nombre del cliente.

Métodos:

- **Getters, Setters y Constructor**
- **run:** Crea el hilo y el bucle para la recepción de mensajes del cliente.
- **enviarMensaje:** Se manda un mensaje a ese cliente

Clase Juego: Gestiona la lógica del juego, verifica las condiciones de victoria y mantiene el estado del juego.

Atributos:

- **Tablero tablero:** Para llevar la partida.
- **ClientHandler j1,j2:** Son los dos jugadores que están compitiendo en la partida, con ellos se comunica.
- **Int puntos1,puntos2:** Puntuaciones de la partida
- **Int i,j:** Para manejar los movimientos del tablero.
- **String c:** Para manejar la letra que se coloca en el tablero.
- **Boolean turnoj1:** Lleva la cuenta del turno, inicialmente comienza el primer jugador conectado.

Métodos:

- **run:** Crea el hilo de la partida manejando cada jugada en cada turno y notifica a los jugadores de las jugadas, puntuaciones y resultado final.
- **pieza(int, int, String, ClientHandler):** Coloca en el tablero la jugada realizada por el cliente.

Clase Tablero y Casilla: Representa el estado del tablero del juego. Gestiona la colocación de letras y la verificación de formaciones "OSO".

<u>Flujo de Comunicación Cliente-Servidor</u>

Conexión:

1. El cliente se conecta al servidor y envía su nickname.
2. El servidor envía las dimensiones del tablero al cliente.
3. El cliente inicia el tablero gráfico basado en las dimensiones recibidas.

Juego:

1. Los jugadores se alternan para enviar jugadas al servidor.
2. El servidor procesa las jugadas y actualiza el estado del juego.
3. El servidor notifica a los clientes sobre los cambios en el tablero y el turno de juego.

Chat:

1. Los clientes pueden enviar mensajes de chat al servidor.
2. El servidor reenvía los mensajes de chat a todos los clientes conectados.

Finalización:

1. El servidor detecta el fin del juego cuando todas las casillas están llenas.
2. El servidor envía un mensaje a los clientes indicando el resultado del juego (ganador, perdedor o empate).
3. Los clientes muestran un diálogo final con el resultado del juego.

Conclusión

La implementación del juego del Oso utiliza una arquitectura cliente-servidor para permitir la interacción entre dos jugadores en una red. La interfaz gráfica del cliente proporciona una experiencia de usuario intuitiva, mientras que el servidor gestiona la lógica del juego y la comunicación entre los clientes. Este enfoque modular facilita la escalabilidad y el mantenimiento del sistema.

Preguntas

- **¿Cómo se calculan los osos de una jugada?**

  Cuando se escribe el valor del botón se rellena también la casilla de la variable tablero y se llama a la función comprobarOso(i,j) que son las posiciones de la nueva casilla rellenada. Esta función comprueba si en el tablero se ha completado un OSO con la nueva casilla rellenada y en ese caso se suma el contador de osos.

- **¿Cómo sabe el interface gráfico (GUI) la casilla en la que el jugador ha escrito una letra?**

  Creo un botón por cada casilla del tablero y añado un action listener en cada uno indicando que cuando se clicke se compruebe la jugada.

- **¿Se ha logrado reutilizar el código encargado de pintar el tablero en el GUI o por el contrario el código se ha copiado y pegado en cada cliente gráfico?**

  En mi caso he creado un tablero nuevo en cada GUI de cliente, lo que he hecho ha sido notificar al jugador que casilla ha marcado el otro jugador para modificarla.

- **¿Cómo logra el interface gráfico (GUI) actualizar el tablero cuando le llegan los mensajes del servidor?**

Seleccionando el botón correspondiente a la casilla que se va a cambiar y modificando su contenido.

- **¿Qué tipo de stream (data u object) se ha utilizado en las comunicaciones? ¿Cómo se ha serializado el mensaje enviado por el socket?**

  Strings con un prefijo que indica la acción que desea realizar. En caso de que se quisieran pasar más valores se parsea el string y se obtienen los valores deseados.

- **¿Cómo se ha implementado la funcionalidad de "abandono"?**

  Cuando un cliente decide abandonar se notifica al servidor y éste lo desconecta de su lista de clientes

- **¿Cómo sabe el servidor qué jugador ha realizado la jugada que ha recibido?**

  Porque se le pasa un String con el prefijo /JUGADA por el socket y podemos saber a que cliente corresponde ese socket.

- **Si dos jugadas llegan a la vez al servidor, ¿cómo se ha asegurado que no se comience a realizar una hasta la otra finalice?**

  Se ha sincronizado la zona de código dónde se manipula el tablero para que no puedan modificarlo dos jugadores simultáneamente.

- **¿Qué diferencias existe entre el servidor que permite jugar una sola partida y el servidor que permite jugar múltiples partidas?**

- **¿Cómo se gestionan los turnos de los jugadores?**

  Con un booleano que indica si el turno es del jugador 1, en caso contrario será turno del jugador 2.