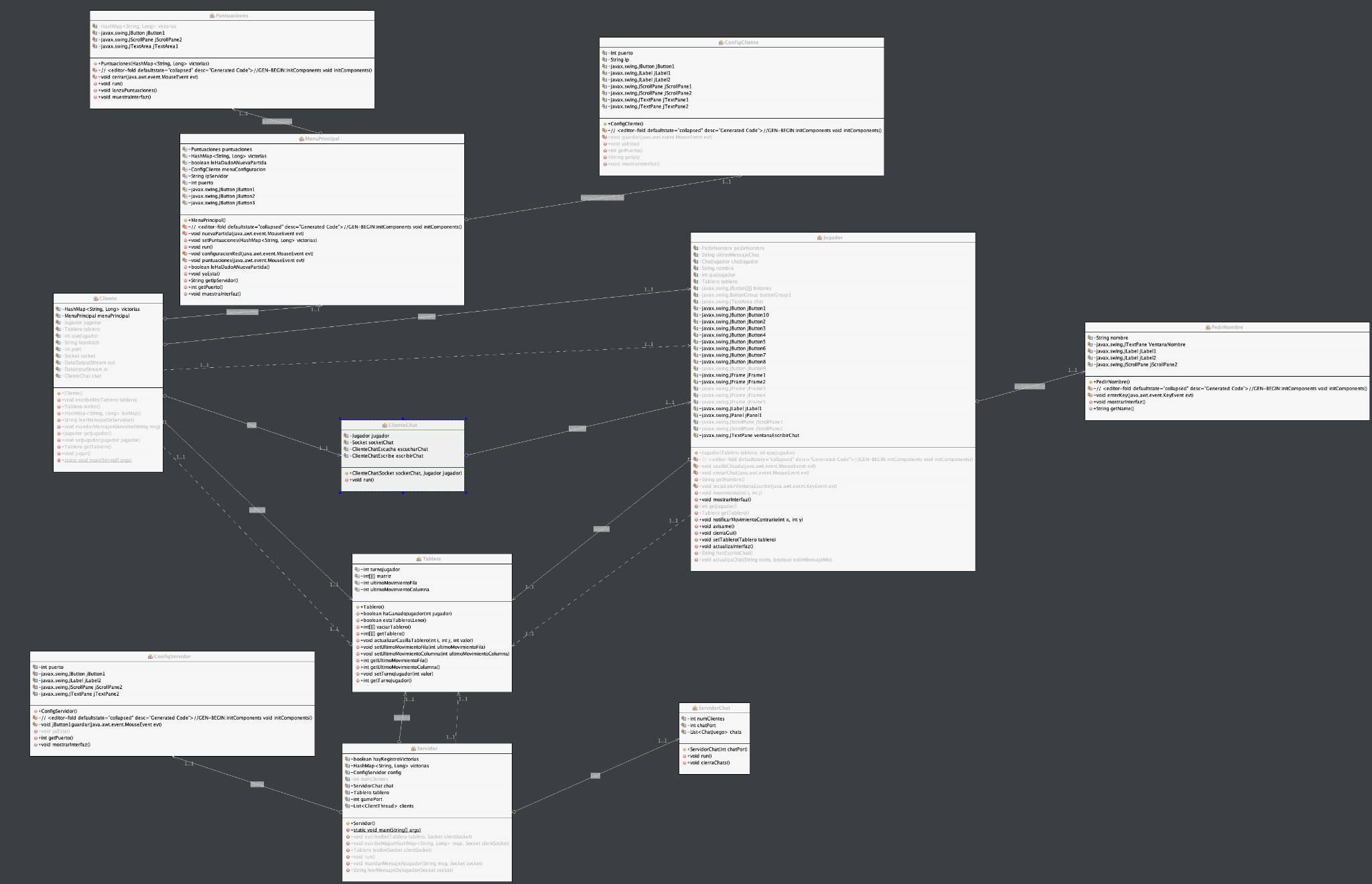
|  |
| --- |
| Universidad Pública de Navarra, Programación Avanzada |
| Tres en raya |
| Práctica individual |
| Álvaro Larraya Conejo |

# Introducción

## Diagrama de clases



(Entrego también el png del diagrama)

## Explicación de la ejecución

Las dos clases ejecutables son Cliente y Servidor. A Servidor hay que introducirle el puerto desde el que va a escuchar. Una vez ejecutado el servidor y elegido el puerto se ejecuta el cliente. Aparecerá un menú en el que habrá varias opciones, puntuaciones no me ha dado tiempo a que funcionase del todo, pero las demás funcionalidades están todas implementadas correctamente. En la pestaña de configuración se puede elegir el puerto e ip del servidor al cual nos queremos conectar. Una vez le demos a nueva partida veremos el tablero y el chat, aunque no podremos hacer nada hasta que no se conecte el otro jugador. Una vez los dos conectados empieza el primer jugador que se haya conectado y los dos podremos ver los mensajes que se introduzcan desde el chat.

Las otras dos clases más importantes son Tablero y Jugador que implementan las funcionalidades necesarias para jugar la partida, además Jugador es la clase que contiene la gui que ve el cliente durante la partida. He hecho dos clases más para el correcto funcionamiento del chat que son ServidorChat y ClienteChat, las cuales se comunican igual que Cliente y Servidor pero por otro puerto (el puerto de Servidor +1). Las demás clases son auxiliares a las ya mencionadas y sirven para imprimir por pantalla interfaces graficas que no sean la principal como el menu principal, la gui para pedir el nombre del jugador, la gui para el menu de configuracion o la gui para las puntuaciones.

# Clases y métodos

Crea una tabla igual que esta por cada clase que creas que es importante en tu aplicación.

|  |  |
| --- | --- |
| **Clase:** Servidor | |
| **Campo** | **Descripción** |
| hayRegistroVictorias | Sirve para saber si se ha jugado alguna vez en este servidor |
| victorias | Es una variable que se inicializa con el fichero registro de resultados en anteriores partidas, teniendo así acceso fácil a victorias por jugador |
| config | Es el menu de configuracion |
| numClientes | Cuántos clientes se han conectado hasta el momento |
| chat | Es inicializada con una nueva instancia de ServidorChat desde donde se realizarán las comunicaciones con ClienteChat |
| tablero | Contiene el turno del jugador y el estado de la partida |
| gamePort | Puerto desde el que se está escuchando |
| clients | Lista de hilos desde los que se atiende a los clientes |
| **Método** | **Descripción** |
| Servidor | Constructor desde el que se instancia todas las variables y se pone en marcha el hilo de Servidor |
| escribeBin | Sirve para pasar el tablero por el socket que nos den como parametro de entrada |
| escribeMapa | Manda el mapa de victorias por el socket |
| leeBin | Lee desde el socket un tablero y lo devuelve |
| run | Se obtiene el socket de los dos clientes y se transfieren sus datos a la clase ClientThread la cual atiende a los clientes |
| mandarMensajeAJugador | Manda un texto normal al socket que se le dice |
| leerMensajeDeJugador | Lee desde el socket y devuelve el texto que haya leido |

|  |  |
| --- | --- |
| **Clase:** Cliente | |
| **Campo** | **Descripción** |
| victorias | Es una variable que se inicializa con el fichero registro de resultados en anteriores partidas, teniendo así acceso fácil a victorias por jugador |
| menuPrincipal | instancia de la clase MenuPrincipal que permite proyectar el menu y recoger los datos necesarios |
| jugador | Instancia de la clase Jugador |
| tablero | Instancia de la clase Tablero |
| queJugador | Entero que nos permite saber si este es el jugador que primero se ha conectado (1) o si el es segundo (2) |
| hostAddr | Es la ip del servidor al que nos vamos a conectar |
| socket | Para conectarnos |
| port | El puerto al que nos vamos a conectar |
| out | Nos permite mandar mensajes por el socket |
| in | Nos permite recibir mensajes del socket |
| chat | Instancia de ClienteChat desde el que vamos a mandar mensajes de chat |
| **Método** | **Descripción** |
| Cliente | Constructor desde el cual inicializamos las variables, ejecutamos el hilo del chat y el de la partida |
| escribeBin | Sirve para pasar el tablero por el socket que nos den como parametro de entrada |
| leeBin | Lee desde el socket un tablero y lo devuelve |
| leeMap | Recibe desde el socket el mapa de victorias y lo devuelve |
| leerMensajeDeServidor | Manda un texto normal al socket que se le dice |
| mandarMensajeAServidor | Manda un texto normal al socket que se le dice |
| jugar | Este metodo sincroniza la llegada de mensajes del servidor con el movimiento del jugador y manda su nombre al servidor si ha ganado |

|  |  |
| --- | --- |
| **Clase:** Jugador | |
| **Campo** | **Descripción** |
| pedirNombre | Sirve para mostrar la interfaz en la que el usuario introduce su nombre |
| ultimoMensajeChat | Como dice el nombre es el ultimo mensaje que se ha enviado por el chat |
| chatJugador | Es la subclase desde la que se actualiza el chat |
| nombre | Es el nombre del jugador |
| queJugador | Si es 1 es que es el primero en conectarse, sino es el 2 |
| tablero | instancia de la clase Tablero |
| botones | array de botones que forman la interfaz del tablero |
| **Método** | **Descripción** |
| Jugador | constructor que inicializa variables |
| casillaClicada | si es el turno del jugador identifica el botón pulsado y actualiza la interfaz grafica |
| enviarChat | Al pulsar el botón enviar coge el texto que está en el jpanel para escribir y lo pone el chat, vaciando el jpanel |
| teclaEnterVentanaEscribir | hace lo mismo que enviarChat pero al pulsar la tecla enter al terminar de escribir |
| movimiento | actualiza el tablero y el turno del jugador |
| mostrarInterfaz | ejecuta la gui |
| avisame | devuelve el control al hilo que le ha llamado cuando el usuario al cual le toca jugar hace su movimiento |
| cierraGui | cierra la gui |
| hanEscritoChat | devuelve el control al hilo que le ha llamado cuando algun usuario escribe en el chat y le devuelve que se ha escrito |
| actualizaChat |  |

|  |  |
| --- | --- |
| **Clase:** Tablero | |
| **Campo** | **Descripción** |
| turnoJugador | a que jugador le toca jugar |
| matriz | representa el tablero, esta llena de ceros, unos y doses. Un cero es que no hay ficha, 1 es la ficha del jugador 1 y 2 es la ficha del jugador 2 |
| ultimoMovimientoFila | fila de la ultima ficha que se ha posicionado en el tablero |
| ultimoMovimientoColumna | columna de la ultima ficha que se ha posicionado en el tablero |
| **Método** | **Descripción** |
| Tablero | constructor que instancia las variables |
| haGanadoJugador | comprueba si hay un tres en raya del jugador que se le pasa como parametro de entrada |
| estaTableroLLeno | busca un hueco en el tablero, si lo hay devuelve cierto sino falso |
| vaciarTablero | quita todas las fichas que haya en el tablero |
| actualizarCasillaTablero | actualiza la matriz |

# Trabajo futuro y Conclusiones

Lo último que hice antes de entregar el trabajo fue intentar implementar las puntuaciones, pero no me dió tiempo, me hubiera gustado poder terminarlo, pero todo lo demás va bastante bien. Se podría mejorar mandando un mensaje de “has ganado”, “empate” o lo que sea por gui en vez de por terminal, tampoco me dió tiempo. Estaría bien poner un label para que el usuario sepa en todo momento cuando es o no su turno.

Este trabajo lo he organizado iterativamente. Al principio solo implementé el tres en raya en local, luego con interfaz gráfica, posteriormente online y finalmente añadí el chat, menú, interfaces para pedir el nombre, el menú de configuración… También he añadido los jar aunque los hice con mucha prisa y no me dió tiempo a probarlos del todo. Espero que este trabajo sea de tu agrado.