Descripción del Problema: Connect Dots

Connect Dots es un emocionante juego multiplayer diseñado para involucrar a múltiples jugadores. El objetivo principal del juego es permitir que los jugadores se conecten entre sí en una malla de puntos mediante líneas trazadas en sus turnos. Los jugadores tienen la oportunidad de cerrar cuadrados al unir los puntos de la malla con líneas, lo que les otorga puntos por cada cuadrado cerrado. Además, cada cuadrado cerrado lleva la identificación del jugador que lo cerró. El objetivo final del juego es cerrar la mayor cantidad de cuadrados posible y, al final de la partida, el jugador que haya cerrado más cuadrados se declara el ganador.

Reglas del Juego

Malla de Puntos: El juego se desarrolla en una malla de puntos, que actúa como el tablero de juego.

Turnos de los Jugadores: Los jugadores participan en el juego por turnos, lo que significa que cada jugador tiene la oportunidad de tomar decisiones en su turno.

Cerrar Cuadrados: Cuando un jugador logra unir los puntos de la malla con líneas y cierra un cuadrado en su turno, se le permite continuar jugando hasta que no logre cerrar un cuadrado en su turno.

Puntos: Todos los cuadrados del juego tienen el mismo valor en términos de puntos. Cada vez que un jugador cierra un cuadrado, se le otorgan puntos correspondientes.

Objetivo: El objetivo principal del juego es cerrar cuadrados para acumular la mayor cantidad de puntos posible y superar a los demás jugadores.

Ganador: Al concluir la partida, el jugador que haya cerrado la mayor cantidad de cuadrados se declara el ganador.

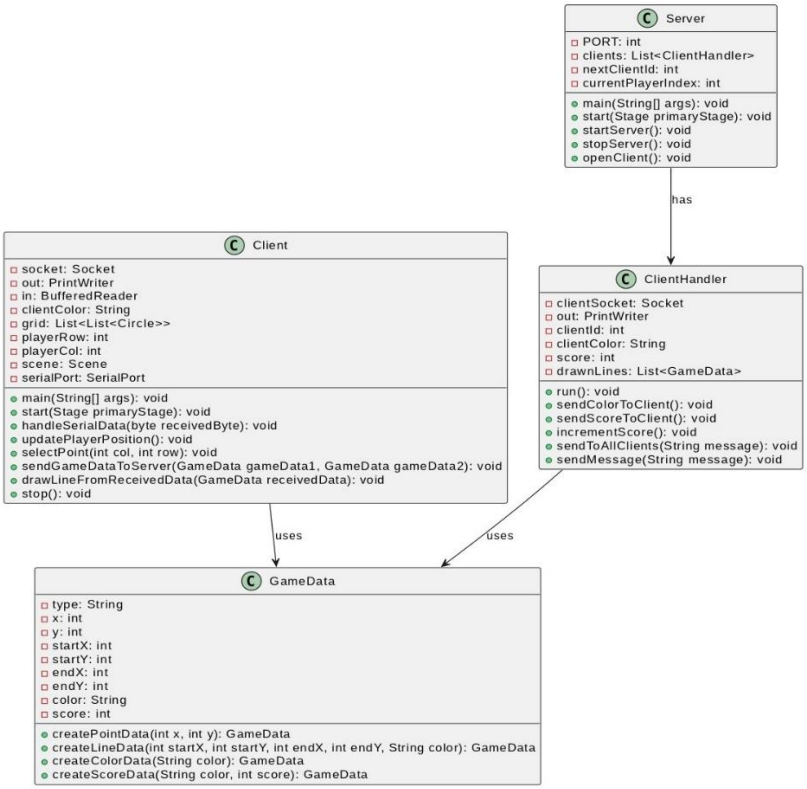
Componentes del Juego:

Servidor Central: Existe un servidor central implementado en Java que actúa como el núcleo del juego. Este servidor se encarga de gestionar las partidas y supervisar el flujo del juego.

Clientes: Cada jugador participa en el juego a través de una aplicación cliente que se ejecuta en su propia computadora. Estos clientes establecen una conexión con el servidor y envían acciones para interactuar con el juego. Además, reciben actualizaciones del servidor y las representan gráficamente en la pantalla.

Comunicación en Formato JSON: Toda la comunicación entre los clientes y el servidor se realiza utilizando el formato, lo que facilita la transmisión de datos estructurados de manera eficiente entre las partes.

Diagramas de Clases



Descripción de las estructuras de datos desarrolladas

grid: List<List<Circle>>:

- Es una estructura de lista bidimensional utilizada en la clase **Client** para representar la cuadrícula de puntos (círculos) en la interfaz gráfica del juego. La lista contiene listas de objetos **Circle** (círculos) que representan los puntos en la cuadrícula.

Desafío: Sincronización y Concurrencia en la Arquitectura Cliente-Servidor Tradicional

a. Listado de requerimientos del sistema:

Los requerimientos del sistema para el juego "Connect Dots" pueden definirse de la siguiente manera:

1. **Requerimientos Funcionales:**
    - El sistema debe permitir a múltiples jugadores conectarse al servidor.
    - Los jugadores deben poder seleccionar puntos en la cuadrícula para dibujar líneas.
    - El sistema debe gestionar turnos entre jugadores.

- Los jugadores deben poder cerrar cuadrados para obtener puntos.
- El sistema debe llevar un registro de los puntos y cuadrados cerrados por cada jugador.
- El servidor debe mantener el control completo del juego y enviar actualizaciones a los clientes.
- Los clientes deben mostrar la interfaz gráfica del juego y enviar acciones al servidor.
- El sistema debe ser capaz de iniciar y finalizar partidas.

2. **Requerimientos No Funcionales:**
    - La comunicación entre el servidor y los clientes debe utilizar el formato JSON.
    - El sistema debe ser escalable para admitir un número variable de jugadores.
    - La interfaz gráfica del juego debe ser intuitiva y fácil de usar.
    - El sistema debe ser eficiente en términos de consumo de ancho de banda y recursos.
    - Se debe garantizar la seguridad en la comunicación para prevenir trampas.

b. Elaboración de opciones de solución al problema:

A continuación, se presentan dos opciones de solución posibles para el juego "Connect Dots", junto con diagramas conceptuales simplificados que ilustran cada enfoque:

1. **Arquitectura Cliente-Servidor Tradicional:**
    - Descripción: En esta opción, se implementa una arquitectura cliente-servidor tradicional donde el servidor centraliza la lógica del juego y la comunicación entre jugadores.
    - Diagrama Conceptual: Se muestra un servidor central que gestiona múltiples clientes conectados. El servidor controla la lógica del juego y envía actualizaciones a los clientes.
2. **Arquitectura Juego Distribuido:**

Descripción: En esta opción, se implementa una arquitectura de juego distribuido donde cada cliente ejecuta su propia lógica de juego y se comunica con otros clientes para sincronizar el estado del juego.

Diagrama Conceptual: Se muestran varios clientes conectados que interactúan directamente entre sí para sincronizar el juego. El servidor actúa como un coordinador inicial.

c. Valoración de opciones de solución:

La valoración de las opciones de solución se realizaría en función de los siguientes criterios:

- **Escalabilidad:** El enfoque de arquitectura de juego distribuido es más escalable, ya que permite un mayor número de jugadores sin sobrecargar el servidor central.
- **Eficiencia:** La arquitectura cliente-servidor tradicional puede ser más eficiente en términos de ancho de banda y recursos del servidor, ya que centraliza la lógica del juego. Sin embargo, la arquitectura distribuida ofrece una distribución de carga entre los clientes.
- **Seguridad:** Ambas opciones deben implementar medidas de seguridad para prevenir trampas. Esto debe ser evaluado en detalle.
- **Facilidad de desarrollo:** La arquitectura cliente-servidor tradicional puede ser más fácil de desarrollar y mantener, ya que el servidor controla la lógica del juego. La arquitectura distribuida es más compleja en términos de sincronización entre clientes.
- **Experiencia del usuario:** Ambas opciones deben proporcionar una experiencia de usuario satisfactoria, por lo que la interfaz gráfica y la capacidad de juego son criterios importantes.
- **Mantenibilidad:** La facilidad de mantenimiento a largo plazo debe considerarse, incluyendo actualizaciones y modificaciones futuras.

d. Selección de la propuesta final:

La opción seleccionada como propuesta final es la **Arquitectura Cliente-Servidor Tradicional**. A continuación, se argumenta la elección:

Razones para la Selección:

1. **Centralización de la Lógica del Juego:** La arquitectura cliente-servidor tradicional centraliza la lógica del juego en el servidor, lo que facilita el control y la administración del juego. Esto es especialmente útil en un juego como "Connect Dots", donde se requiere un control estricto de los turnos, la puntuación y la detección de cuadrados cerrados.
2. **Seguridad:** La centralización de la lógica del juego en el servidor permite implementar medidas de seguridad más efectivas para prevenir trampas y asegurar la integridad del juego. El servidor puede validar y autorizar las acciones de los jugadores de manera más confiable.
3. **Escalabilidad Controlada:** Aunque la arquitectura distribuida puede ser más escalable, la arquitectura cliente-servidor tradicional es más adecuada para un número razonable de jugadores, lo que se ajusta a los objetivos del proyecto.
4. **Facilidad de Desarrollo y Mantenimiento:** La opción cliente-servidor tradicional es más fácil de desarrollar y mantener a largo plazo. Facilita la identificación y corrección de errores, así como la implementación de futuras actualizaciones.
5. **Experiencia del Usuario:** La centralización de la lógica del juego simplifica la experiencia del usuario, ya que los clientes no necesitan gestionar tanta lógica compleja y sincronización entre sí.

e. Diseño de la Alternativa Seleccionada:

El diseño final seleccionado se basa en la arquitectura cliente-servidor tradicional. A continuación, se describe el diseño en términos de:

- **Protocolo de Comunicación:** Se utilizará el protocolo TCP/IP para la comunicación entre clientes y servidor, asegurando una comunicación confiable y ordenada.
- **Diagrama de Bloques del Sistema:** Se elaborará un diagrama de bloques del sistema que incluye componentes clave como el servidor central, múltiples clientes, la interfaz gráfica del juego y la lógica del juego.
- **Diagrama UML:** Se crearán diagramas UML, incluyendo diagramas de clases y secuencia para representar la estructura de clases del juego y la interacción entre componentes durante el juego.

f. Validación del Diseño:

La validación del diseño se llevará a cabo mediante pruebas exhaustivas del sistema. Se verificará que el diseño cumpla con los requerimientos establecidos, garantice la salud y seguridad pública, minimice el costo total de la vida, mantenga el carbono neto cero y considere aspectos relacionados con recursos, culturales, sociales y ambientales según sea necesario.

Las pruebas incluirán:

- Pruebas de funcionalidad para asegurarse de que todas las funciones del juego funcionen correctamente, incluyendo la selección de puntos, cierre de cuadrados, registro de puntajes y control de turnos.
- Pruebas de seguridad para validar la integridad del juego y prevenir trampas.
- Pruebas de rendimiento para evaluar la eficiencia del sistema en términos de uso de recursos y consumo de ancho de banda.
- Pruebas de usabilidad para garantizar una experiencia de usuario satisfactoria.
- Pruebas de escalabilidad para evaluar el comportamiento del sistema con múltiples jugadores.
- Pruebas de mantenibilidad para asegurar que el sistema pueda actualizarse y modificarse de manera efectiva en el futuro.