



Ingeniería en Computadores
Algoritmos y Estructuras de Datos I

I Proyecto

Connect Dots

Profesor:

Leonardo Araya Martínez

Integrantes:

Darío Garro Moya

Brayan Alpízar Elizondo

Esteban Campos Abarca

II Semestre 2023

Fecha de entrega: 30/09/2023

Descripción del Problema

En "Connect Dots", los jugadores pueden unir dos puntos adyacentes en la malla mediante una línea en su turno. Cuando un jugador completa un cuadrado, ese cuadrado se identifica para indicar que le pertenece a ese jugador. El jugador también acumula puntos por cada cuadrado. Una vez se hizo la jugada continua con el siguiente jugador. El servidor central registra el estado del juego, incluyendo la ubicación de las líneas, los cuadrados cerrados y la puntuación de cada jugador. Los jugadores pueden ver la pantalla de juego actualizada cada pocos segundos, lo que les permite tomar decisiones estratégicas basadas en la evolución del tablero. El objetivo del juego es acumular la mayor cantidad de puntos al cerrar cuadrados y planificar movimientos estratégicos para evitar que cierren cuadrados antes de los otros jugadores.

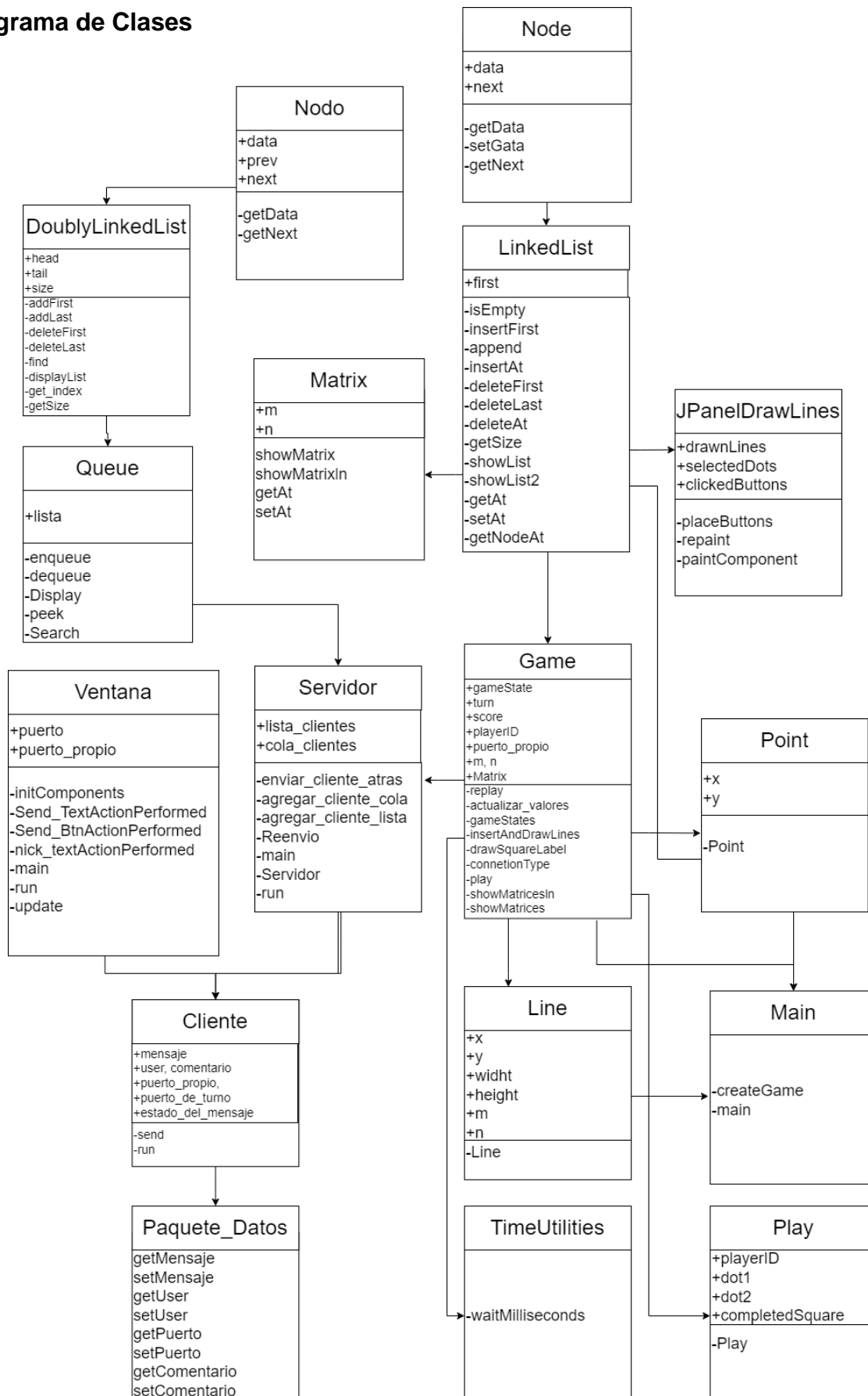
La comunicación entre clientes y servidor se realiza en formato JSON. La estructura de JSON puede contener información sobre las acciones de los jugadores, el estado del juego y las actualizaciones de la pantalla. El servidor lleva un registro del estado del juego, asigna cuadrados cerrados a los jugadores y gestiona las acciones de estos. Además, habrá un control físico con el cual se podrá controlar todo el juego, y con el cual se controlará donde se posicionan las líneas. El juego está diseñado para ser altamente escalable y puede admitir un gran número de jugadores simultáneamente. Los jugadores pueden unirse o salir del juego en cualquier momento sin afectar la estabilidad del servidor ni la experiencia de juego de otros jugadores.

Descripción de las estructuras de datos desarrolladas

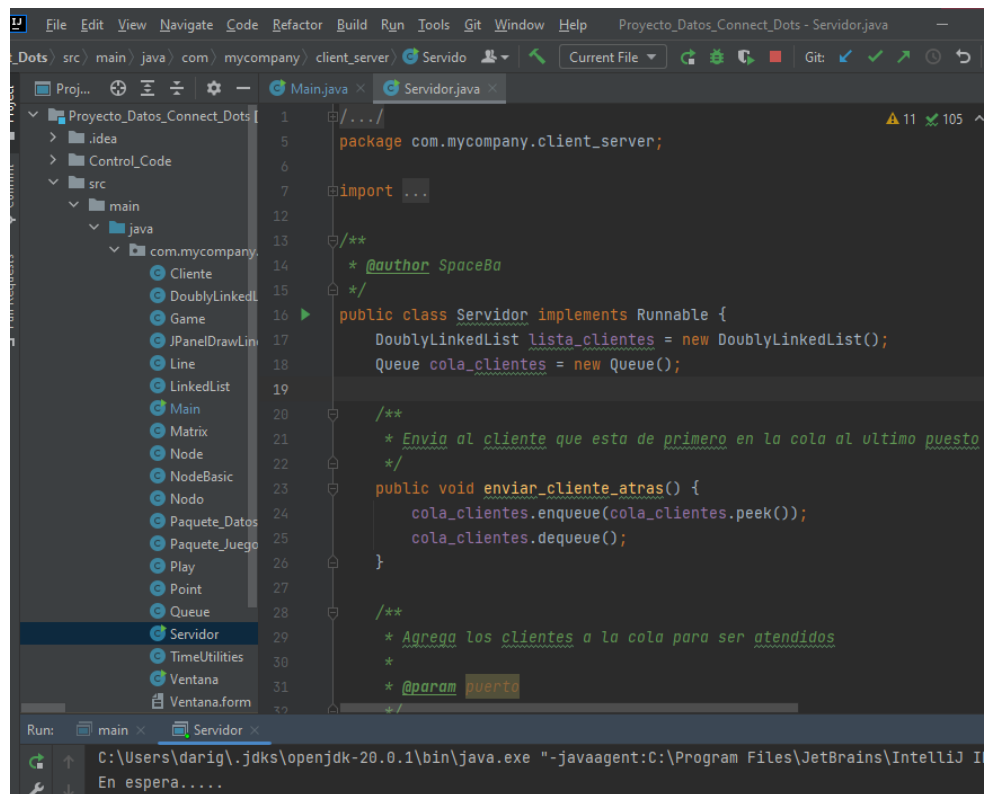
Como parte del proyecto se utilizó una lista doblemente enlazada, la cual ayuda al poderla implementar como parte del servidor el cual conecta a todos los clientes que estén activos, esta lista permite la navegación hacia atrás en la lista, aparte de navegar hacia adelante, siendo que cada nodo tiene dos referencias, el siguiente y el previo, esta misma es una variación de una lista simple, la cual también utilizamos las cuales a diferencia de la doblemente enlazada la simple solo permite avanzar hacia el siguiente nodo, sin posibilidad de regresar al anterior, esta lista la utilizamos para la creación de la matriz, la creación de puntos en la interfaz, y la creación de , los cuales son clases del proyecto esenciales para la jugabilidad y la formación de la matriz en la interfaz.

Al utilizar los dos tipos de listas se tuvo que hacer dos tipos de nodos, uno para la doblemente enlazada, el cual tiene dos referencias, el siguiente nodo, y el anterior. El segundo nodo, utilizado en la lista simple, solo una referencia hacia el siguiente nodo. Aparte de usar una matriz hecha a base de listas simples, una donde se contengan otras listas simples que funcionan como las filas de la matriz, la cual se dibuja en la interfaz donde se aplica la lógica interna del videojuego. Así también se usó una Queue utilizada para el servidor del juego, el cual deja acceder solo al último nodo. La cual cuando entra un nuevo jugador, lo deja a lo último de la cola, haciendo que no pueda accionar hasta que este sea el primero en la cola, una vez que el primero de la cola termina la jugada continúa el siguiente y lo devuelve al final de la cola.

Diagrama de Clases

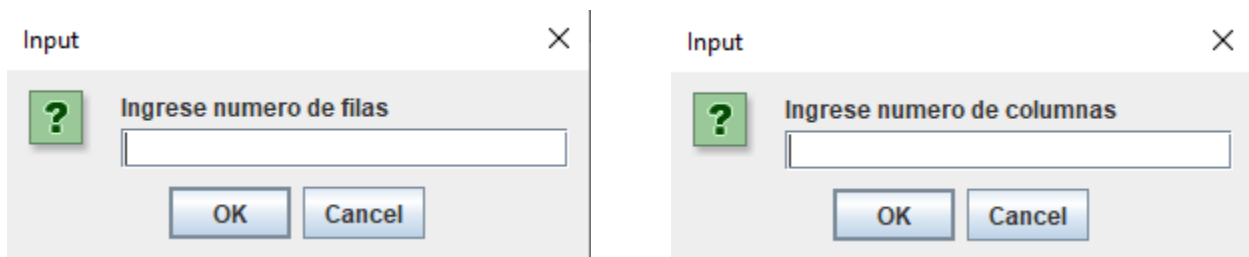


Ejemplos del programa

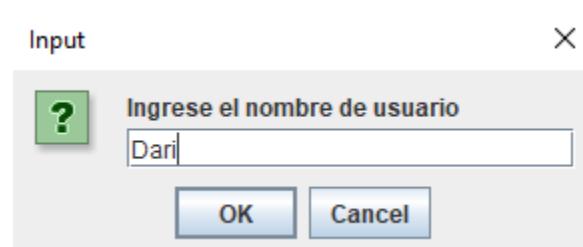


```
1  package com.mycompany.client_server;
2
3  import ...
4
5  /**
6   * @author SpaceBa
7   */
8  public class Servidor implements Runnable {
9      DoublyLinkedList lista_clientes = new DoublyLinkedList();
10     Queue cola_clientes = new Queue();
11
12     /**
13      * Envia al cliente que esta de primero en la cola al ultimo puesto
14      */
15     public void enviar_cliente_atras() {
16         cola_clientes.enqueue(cola_clientes.peek());
17         cola_clientes.dequeue();
18     }
19
20     /**
21      * Agrega los clientes a la cola para ser atendidos
22      *
23      * @param puerta
24      */
25     ...
26 }
```

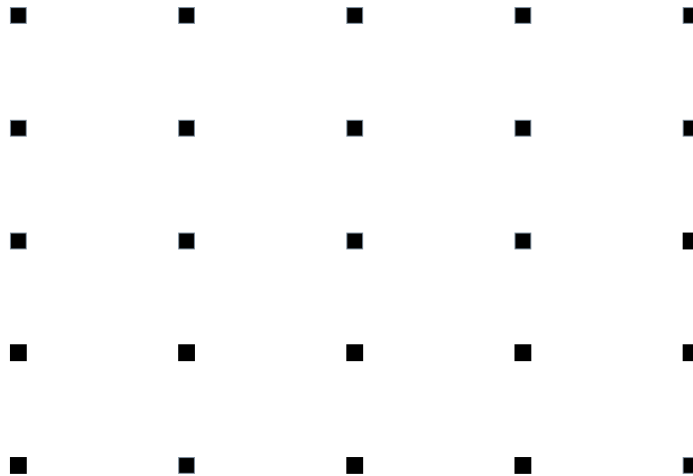
Se corre la clase Servidor, la cual se queda en espera, hasta que se conecte al menos un cliente



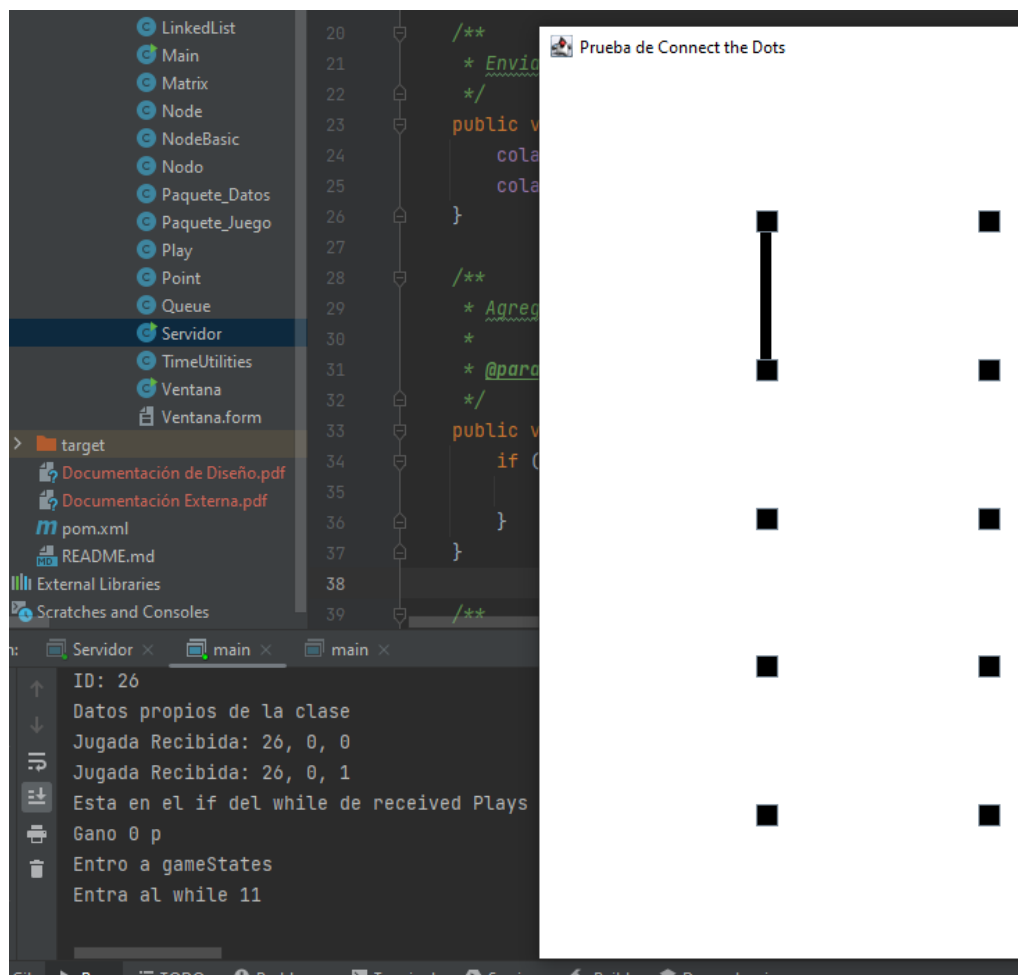
Al correr la clase main, se crea un cliente y una ventana con el juego, esta manda la información del puerto y el nombre del jugador que esté jugando en esa ventana. Se pregunta primero de que tamaño quiere la matriz, el número de filas y el número de columnas, este adaptándose a la ventana para mostrar todos los cuadros.



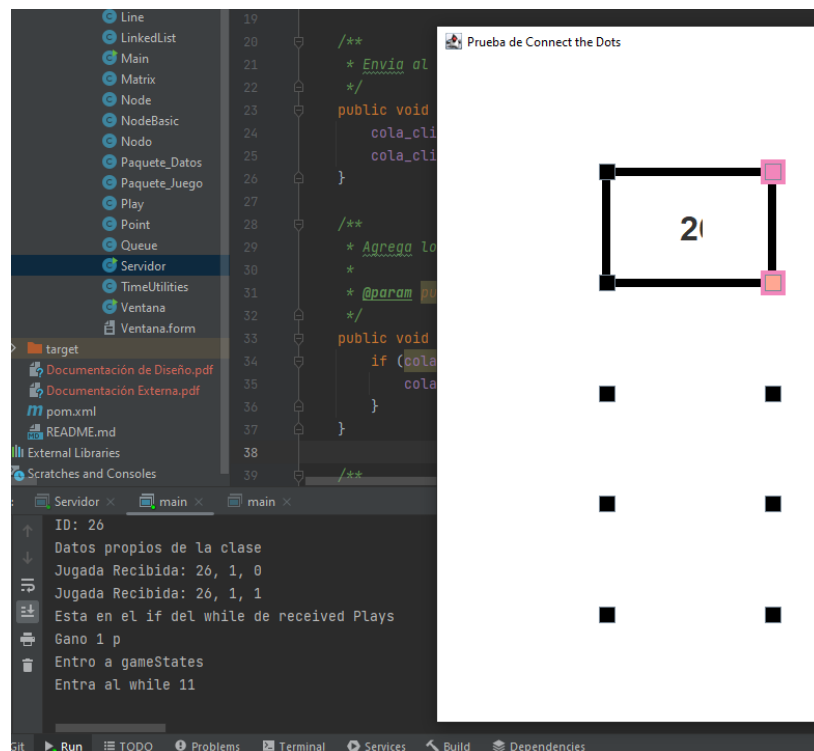
Después de elegir el tamaño de la matriz te pregunta un nombre de usuario para poder usarlo al jugar



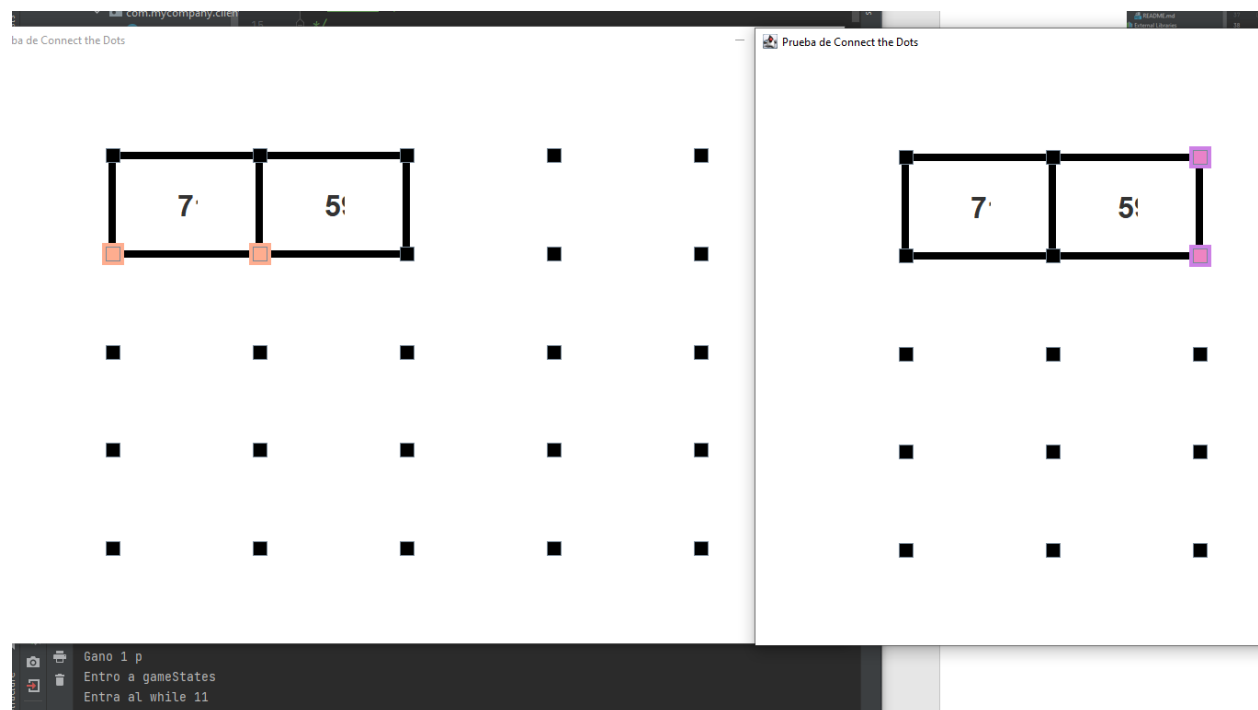
Vemos como se dibuja la matriz con el tamaño que hemos escogido.



Cuando hago una línea en la ventana se manda la información de esto mismo, la creación de una línea y si se completó un cuadrado, si no se completa, gana 0 puntos. Aparte de proporcionar la información de que puntos se presionaron de la matriz.



Cuando se completa un cuadrado se informa que el jugador gano puntos, aparte de identificar al jugador que completó el cuadrado en la interfaz.



Aquí se muestran 2 jugadores distintos, cuando uno de ellos crea una línea, se actualiza la misma línea creada en cada uno de los jugadores y las muestra.