```
void Solitaire::play() {
    bool winner = false;
    bool continueGame = true;
    while (!winner && continueGame){
        this->printGame();
        std::cout<<"Ingresa que hay que hacer\n    1->Mover\n    2->Mostrar siguiente de la baraja";
        std::cout<<"\n    3->Deshacer movimiento\n    4->Rehacer movimiento\n    5->Terminar juego\n";
        int option = util->getNaturalNumber( min: 1,  max: 5);
        switch (option) {
            case 1:
                moveCards();
                break;
            case 2:
                moveDisplayBaraja();
                break;
            case 3:
                try {
                    admiMoves->undo();
                } catch(const std::out_of_range& e) {
                    cout<<"No hay movimientos para retroceder"<<endl;
                    util->enterContinue();
                }
                break;
            case 4:
                try {
                    admiMoves->rendo();
                } catch(const std::out_of_range& e) {
                    cout<<"No hay movimientos para rehacer"<<endl;
```

SOLITARIO

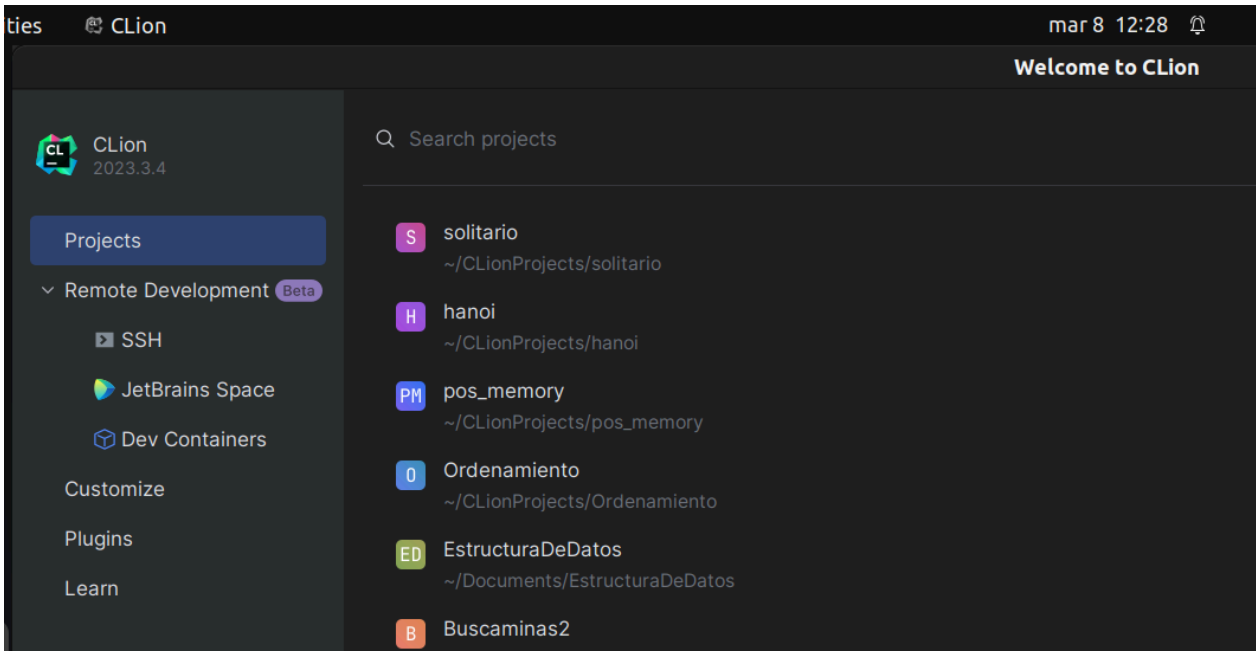
Manual técnico

08/03/2024

Yennifer Maria de León Samuc
Estructura de datos

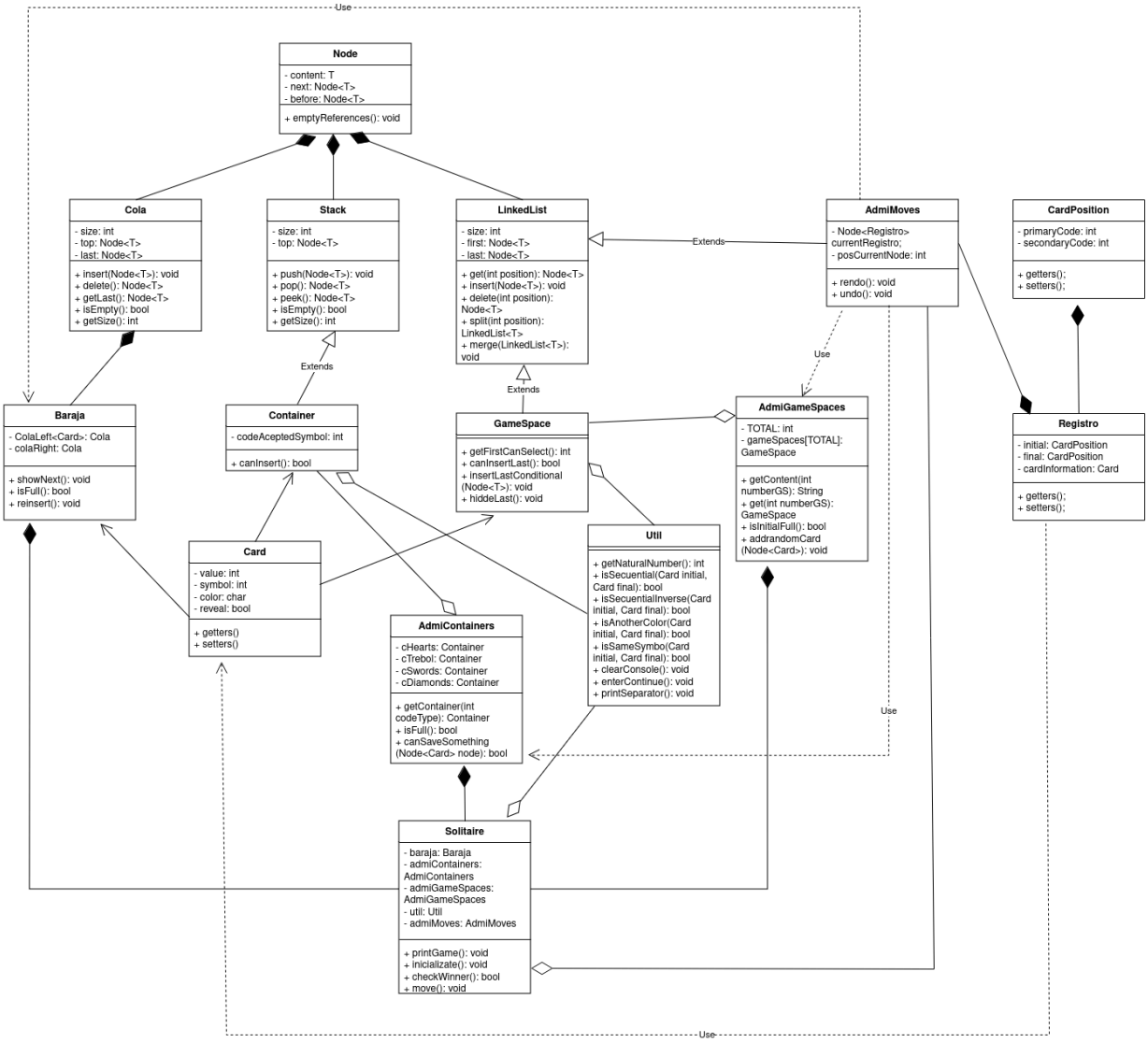
TECNOLOGÍA USADA EN EL PROYECTO

- Ide: CLion2023.3.4



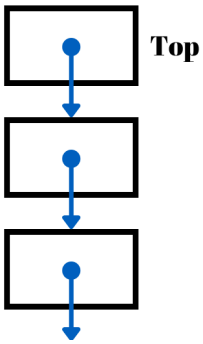
- C++ 17

DIAGRAMA DE CLASES



ESTRUCTURAS DE DATOS

En el proyecto se usaron Pilas (Stack), Colas y Listas doblemente enlazadas (véase el diagrama de clases para ver cómo interactúan con los demás elementos), se incluye una visualización gráfica de dichas estructuras, así como una explicacion breve de su funcionamiento para entender mejor el código.

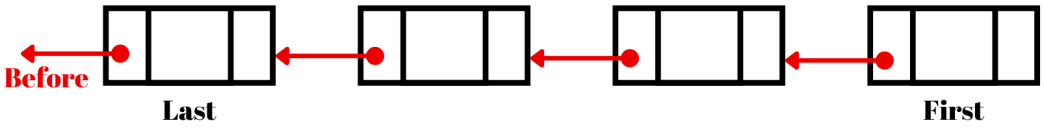


Pila (Stack)

En una pila se ingresan elementos hasta el tope y solo pueden sacarse de ahi, sigue la regla de ***"El último que entra es el primero en salir".***

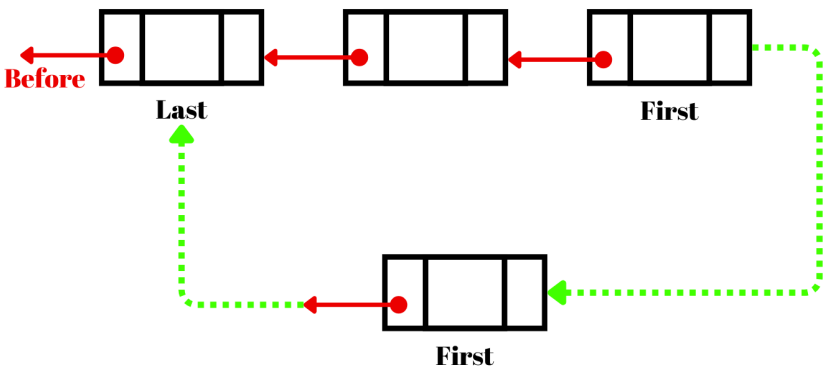
Los contenedores en el juego funcionan como pilas

Colas (Queue)



En una cola se ingresan elementos hasta el tope y solo puede sacarse de ahi el primer elemento, sigue la regla de ***"El primero que entra es el primero en salir".***

La baraja en el juego funcionan como dos colas, de la siguiente manera:

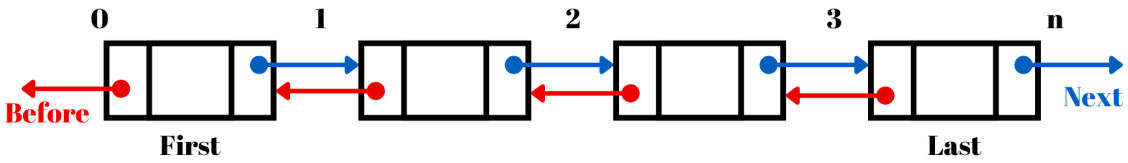


Nota: una cola siempre quedará con un solo elemento para que el juego funcione, esta es la baraja actual (a la derecha en el juego, y al cola de abajo que se muestra arriba).

Las flechas verdes es la dirección en que se van pasando los elementos en la baraja.

Listas doblemente enlazadas (Linked list)

Estas listas tienen dos enlaces hacia adelante y hacia atras, tal cual se muestra en la imagen:



En el juego sirven para los espacios de juego como tal, permiten mover varias cartas a la vez.

ALGORITMO DE FUNCIONES IMPORTANTES

I. Inicializar el juego

```
para cada simbolo hacer con su color especifico hacer hasta 13:
        crear una carta con el numero actual
        verificar si la baraja esta llena
        verificar si los espacios estan llenos
        si alguno de los dos esta lleno:
                dirigir las cartas creadas hacia el que no este lleno
        de lo contrario
                crear un numero random del 1 al 100
                si el numero esta entre 0 y 50
                        dirigir la carta a la baraja
                de lo contrario
                        dirigir la carta a los espacios
    fin para
```

II. Jugando

Hacer mientras no haya ganado o terminado el juego

mostrar la interfaz del juego

preguntar que desea hacer

en caso de:

optar por mostrar siguiente carta de la baraja

bajara -> mostrar siguiente

mover:

preguntar que desea mover:

optar por mover la carta actual de la baraja

MOVER_DE_BARAJA();

optar por mover una carta del contenedor

MOVER_DE_CONTENEDOR();

optera por mover una carta de los espacios de juego

MOVER_ESPACIOS();

retroceder movimiento:

RETROCEDER_MOVIMIENTO();

rehacer movimiento:

REHACER_MOVIMIENTO();

fin hacer