

Universidad De San Carlos de Guatemala
Centro Universitario de Occidente CUNOC



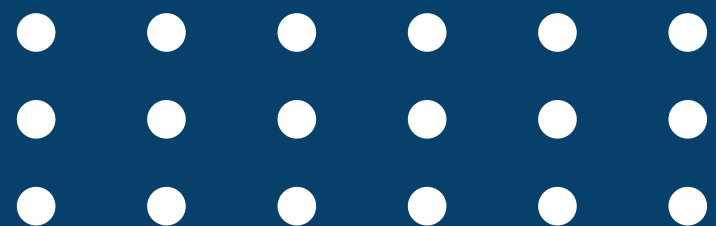
MANUAL DE TÉCNICO

202031773 Enmer Oswaldo Sandoval Mazariegos

Nodo

```
Juego.h  Nodo.h x
1  //
2  // Created by laptop on 8/03/2024.
3  //
4
5  #ifndef PRACTICA1EDD_NODO_H
6  #define PRACTICA1EDD_NODO_H
7
8
9  #include "Carta.h"
10
11  class Nodo {
12  public:
13      Carta* carta;
14      Nodo* siguiente;
15
16      Nodo(Carta* carta);
17  };
18
19
20  #endif //PRACTICA1EDD_NODO_H
21
```

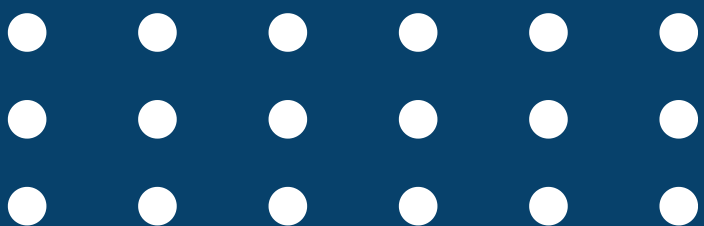
```
Juego.h  Nodo.h  Nodo.cpp x
1  //
2  // Created by laptop on 8/03/2024.
3  //
4
5  #include "Nodo.h"
6
7  Nodo::Nodo(Carta *carta) : carta(cartas), siguiente(nullptr) {}
```



Clase carta

```
Juego.h  Juego.cpp  Carta.h x
11 public:
12     int numero;
13     Palo palo;
14     char color;
15     bool estado;
16
17     Carta(int numero, Palo palo, char color, bool estado);
18
19     int getNumero();
20     Palo getPalo();
21     char getColor();
22     bool getEstado();
23
24     void setNumero(int numero);
25     void setPalo(Palo palo);
26     void setColor(char color);
27     void setEstado(bool estado);
28 };
29
30
31 #endif //PRACTICA1EDD_CARTA_H
32
```

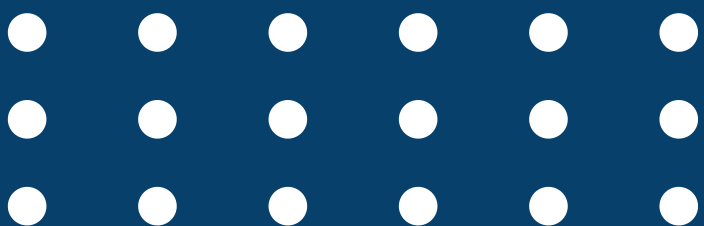
```
Juego.h  Juego.cpp  Carta.h  Carta.cpp x
1 //
2 // Created by laptop on 8/03/2024.
3 //
4
5 #include "Carta.h"
6
7 Carta::Carta(int numero, Palo palo, char color, bool estado) {
8     this->numero = numero;
9     this->palo = palo;
10    this->color = color;
11    this->estado = estado;
12 }
13
14 void Carta::setNumero(int numero) {
15     this->numero = numero;
16 }
17
18 void Carta::setPalo(Palo palo) {
19     this->palo = palo;
20 }
21
22 void Carta::setColor(char color) {
23     this->color = color;
24 }
25
26 void Carta::setEstado(bool estado) {
27     this->estado = estado;
28 }
```



Pila

```
Debug | Practica1EDD | 
Juego.h  Juego.cpp  Carta.h  Carta.cpp  Pila.cpp x
5  > #include ...
8
9  ↩ Pila::Nodo::Nodo(Carta* carta) : carta(carta), siguiente(nullptr) {}
10
11 ↩ Pila::Pila() : cima(nullptr) {}
12
13 ↩ void Pila::push(Carta* carta) {
14     Nodo* nodo = new Nodo(carta);
15     nodo->siguiente = cima;
16     cima = nodo;
17 }
18
19 ↩ void Pila::pop() {
20     if (cima != nullptr) {
21         Nodo* nodoAEliminar = cima;
22         cima = cima->siguiente;
23         delete nodoAEliminar;
24     }
25 }
26
27 ↩ Carta* Pila::top() {
28     if (cima != nullptr) {
29         return cima->carta;
30     } else {
31         return nullptr;
32     }
33 }
```

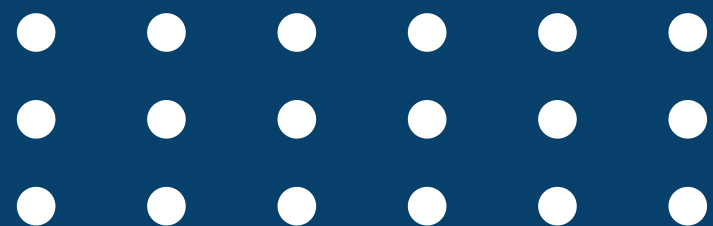
```
Juego.h  Juego.cpp  Carta.h  Carta.cpp  Pila.h x
6  #define PRACTICA1EDD_PILA_H
7
8  > #include ...
12
13 class Pila {
14 public:
15     class Nodo {
16         Carta* carta;
17         Nodo* siguiente;
18
19         ↩ Nodo(Carta* carta);
20     };
21
22     Nodo* cima;
23
24 ↩ Pila();
25
26 ↩ void push(Carta* carta);
27 ↩ void pop();
28 ↩ Carta* top();
29 ↩ bool estaVacía();
30 };
31
```



Colas

```
Juego.h  Juego.cpp  Carta.h  Carta.cpp  Pila.cpp  ColasPrincipales.h  ColasPrincipales.cpp
11
12 class ColasPrincipales {
13 public:
14     class Nodo {
15
16     public:
17         Nodo(Carta* carta);
18
19         Nodo* siguiente;
20         Carta* carta;
21     };
22
23     Nodo* frente;
24     Nodo* final;
25
26     ColasPrincipales();
27
28     void encolar(Carta* carta);
29     void desencolar();
30     Carta* peek();
31     bool estaVacia();
32 };
33
34 #endif //PRACTICA1EDD_COLASPRINCIPALES_H
35
```

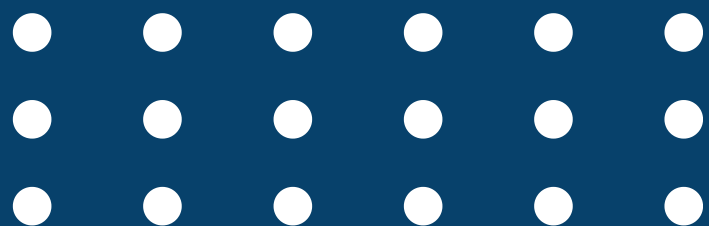
```
Juego.h  Juego.cpp  Carta.h  Carta.cpp  Pila.cpp  ColasPrincipales.h  ColasPrincipales.cpp
11
12 void ColasPrincipales::encolar(Carta *carta) {
13     Nodo* nodo = new Nodo(carta);
14     if(estaVacia()){
15         frente = final = nodo;
16     } else {
17         final->siguiente = nodo;
18         final = nodo;
19     }
20 }
21
22 void ColasPrincipales::desencolar() {
23     if(!estaVacia()){
24         Nodo* nodoAEliminar = frente;
25         frente = frente->siguiente;
26         if(frente == nullptr){
27             final = nullptr;
28         }
29         delete nodoAEliminar;
30     }
31 }
32
33 Carta* ColasPrincipales::peek() {
34     if (!estaVacia()){
35         return frente->carta;
36     } else{
37         return nullptr;
38     }
39 }
```



Lista Doblemente enlazada

```
cpp Carta.h Carta.cpp Pila.cpp ColasPrincipales.h ColasPrincipales.cpp ListaDoble.cpp x
31 void ListaDoble::eliminarInicio() {
32     if(inicio != nullptr){
33         Nodo* nodoAEliminar = inicio;
34         inicio = inicio->siguiente;
35         if(inicio == nullptr){
36             fin = nullptr;
37         } else {
38             inicio->anterior = nullptr;
39         }
40         delete nodoAEliminar;
41     }
42 }
43
44 void ListaDoble::eliminarFinal() {
45     if(fin != nullptr){
46         Nodo* nodoAEliminar = fin;
47         fin = fin->anterior;
48         if(fin == nullptr){
49             inicio = nullptr;
50         } else {
51             fin->siguiente = nullptr;
52         }
53         delete nodoAEliminar;
54     }
55 }
56
57 void ListaDoble::eliminar(Pila* pila){
```

```
cpp Carta.h Carta.cpp Pila.cpp ColasPrincipales.h ColasPrincipales.cpp ListaDoble.cpp x
7 ListaDoble::Nodo::Nodo(Pila* pila) : pila(pila), siguiente(nullptr), anterior(nullptr) {}
8
9 void ListaDoble::insertarInicio(Pila *pila) {
10     Nodo* nodo = new Nodo(pila);
11     if(inicio == nullptr){
12         inicio = fin = nodo;
13     } else {
14         nodo->siguiente = inicio;
15         inicio->anterior = nodo;
16         inicio = nodo;
17     }
18 }
19
20 void ListaDoble::insertFinal(Pila *pila) {
21     Nodo* nodo = new Nodo(pila);
22     if(inicio == nullptr){
23         inicio = fin = nodo;
24     } else {
25         nodo->anterior = fin;
26         fin->siguiente = nodo;
27         fin = nodo;
28     }
29 }
30
31 void ListaDoble::eliminarInicio() {
32     if(inicio != nullptr){
33         Nodo* nodoAEliminar = inicio;
34         inicio = inicio->siguiente;
35         if(inicio == nullptr){
36             fin = nullptr;
37         } else {
38             inicio->anterior = nullptr;
39         }
40         delete nodoAEliminar;
41     }
42 }
43
44 void ListaDoble::eliminarFinal() {
45     if(fin != nullptr){
46         Nodo* nodoAEliminar = fin;
47         fin = fin->anterior;
48         if(fin == nullptr){
49             inicio = nullptr;
50         } else {
51             fin->siguiente = nullptr;
52         }
53         delete nodoAEliminar;
54     }
55 }
56
57 void ListaDoble::eliminar(Pila* pila){
```



Lista Doblemente enlazada

```
go.cpp Carta.h Carta.cpp Pila.cpp ColasPrincipales.h ColasPrincipales.cpp ListaDoble.h x
ents\Practica1EDD 4
8
9 #include "Pila.h"
10
11 class ListaDoble {
12 public:
13     class Nodo {
14     public:
15         Pila* pila;
16
17         Nodo(Pila* pila);
18
19         Nodo* siguiente;
20         Nodo* anterior;
21     };
22     Nodo* inicio;
23     Nodo* fin;
24
25
26     ListaDoble() : inicio(nullptr), fin(nullptr) {}
27     void insertarInicio(Pila* pila);
28     void insertFinal(Pila* pila);
29     void eliminarInicio();
30     void eliminarFinal();
31     void eliminar(Pila* pila);
32 };
33
```

