



**Sede  
Santo Domingo**

**UNIVERSIDAD DE LAS FUERZAS ARMADAS-ESPE  
SEDE SANTO DOMINGO DE LOS TSÁCHILAS**

**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN - DCCO-SS**

**CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN**



<b>PERIODO</b>	:	202351 Octubre 2023 - Marzo 2024
<b>ASIGNATURA</b>	:	Estructura de datos
<b>TEMA</b>	:	Informe laboratorio 2 pilas - colas - Listas
<b>ESTUDIANTES</b>	:	Juan Jimenez
<b>NIVEL-PARALELO - NRC</b>	:	Tercer Semestre - 16137
<b>DOCENTE</b>	:	Ing. Javier Cevallos
<b>FECHA DE ENTREGA</b>	:	16-12-2023

**Introducción:**

En el presente informe se revisan los temas que se usaron para la elaboración del laboratorio 2 de la primera unidad el cual consiste en la elaboración de un programa con el uso de las pilas, colas y listas enlazadas con las cuales se tenían que usar sí o sí en el programa.

Los desarrolladores utilizan los arrays y las variantes de listas enlazadas para construir una gran variedad de estructuras de datos complejas. Esta página explora dos de esas estructuras: las Pilas, las Colas . Cuando presentemos los algoritmos lo haremos únicamente en código Java por motivos de brevedad.(*Listas, pilas y colas*, 2021).

## Objetivos

### Objetivo general

Realizar un informe sobre las pilas, colas y listas enlazadas en el cual toca detallar sobre los temas para tener un mejor entendimiento del tema para en el futuro saber para darle el uso adecuado en los programas que se vayan a realizar.

### Objetivos específico

- elaboración de un código para mostrar cómo es la codificación de los diferentes temas para ver cómo funcionan.
- Comprender el concepto y la aplicación de pilas, colas y listas en la programación y la resolución de problemas.
- Familiarizarse con las operaciones y el funcionamiento de pilas, colas y listas, incluyendo la inserción, eliminación y recorrido de elementos.

## Desarrollo

### Listas enlazadas:

En ciencias de la computación, una lista enlazada es una de las estructuras de datos fundamentales, y puede ser usada para implementar otras estructuras de datos. Consiste en una secuencia de nodos, en los que se guardan campos de datos arbitrarios y una o dos referencias, enlaces o punteros al nodo anterior o posterior. El principal beneficio de las listas enlazadas respecto a los vectores convencionales es que el orden de los elementos enlazados puede ser diferente al orden de almacenamiento en la memoria o el disco, permitiendo que el orden de recorrido de la lista sea diferente al de almacenamiento.(DeltaPC, 2019)

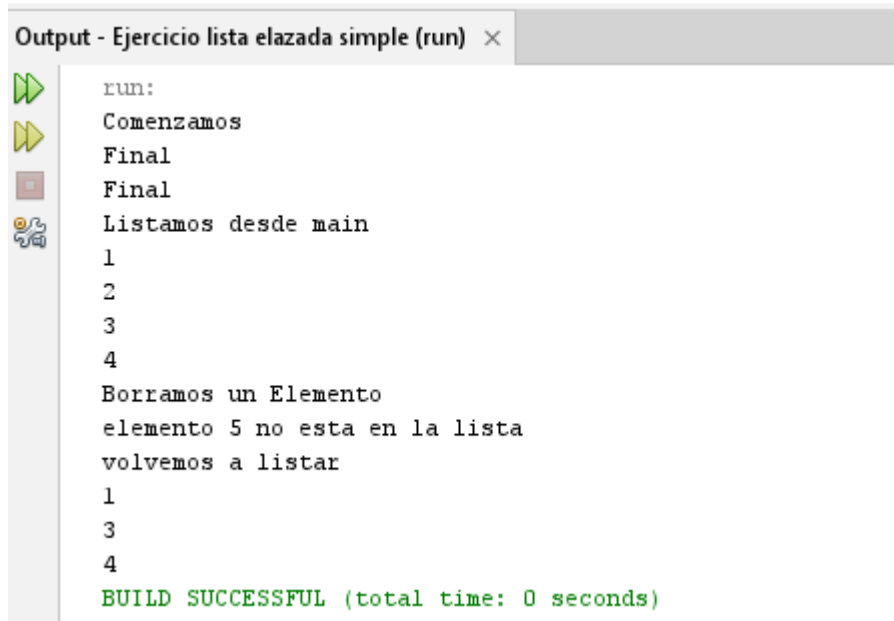
Código del programa ejemplo listas enlazadas donde se explica en las imágenes las funciones:

```
Start Page x EjercicioListaElazadaSimple.java x
Source History
1 package ejercicio.lista.elazada.simple;
2 /**
3  *
4  * @author jrome
5  */
6 public class EjercicioListaElazadaSimple {
7
8     public static void main(String args[]) {
9         //Proceso que ocurre en el programa para el ejemplo de listas enlazadas
10        System.out.println("Comenzamos");
11        //se llama a la clase lista simple
12        listaSimple nl=new listaSimple();
13        //se ingresan los datos a la lista
14        nl.insertarPrimero(2);
15        nl.insertarPrimero(1);
16        nl.insertarFinal(3);
17        nl.insertarFinal(4);
18        //se lista los datos
19        System.out.println("Listamos desde main");
20        nl.listar();
21        //se borra el elemento 2 y 5 de la lista
22        System.out.println("Borramos un Elemento");
23        nl.borrar(2);
24        nl.borrar(5);
25        //se lista desde el main para ver los datos restantes
26        System.out.println("volvemos a listar");
27        nl.listar();
28
29    }
30 }
31
32
33 //clase nodo que es donde se ingresan los datos
34 class Nodo{
35     private int elemento;
36     private Nodo siguiente;
37     //constructor de la clase nodo que inicializa los valores de la misma
38     public Nodo (int elem, Nodo sig){
39         elemento = elem;
40         siguiente = sig;
41     }
42     public int getElemento(){
43         return elemento;
44     }
45     public Nodo getSig(){
46         return siguiente;
47     }
48     public void setElemento(int elem){
49         elemento = elem;
50     }
51     public void setSig(Nodo sig){
52         siguiente = sig;
53     }
54
55 }
56
57
```

```
Start Page x EjercicioListaEnlazadaSimple.java x
Source History
57
58 //clase listasimple donde estan los metodos para las diferentes funciones de la lista
59 class listaSimple{
60     private Nodo primero;
61     private int numElem;
62 public listaSimple(){
63     primero = null;
64     numElem = 0;
65 }
66
67 //metodo para insertar datos a la lista
68 public void insertarPrimero (int elemento){
69     Nodo nuevo = new Nodo (elemento, primero);
70     primero = nuevo;
71     numElem++;
72 }
73
74 //metodo para insertar al final de la lista
75 public void insertarFinal (int elemento){
76     Nodo nuevo = new Nodo(elemento, null);
77     if (primero == null){
78         primero = nuevo;
79     }
80     else {
81         Nodo actual = primero;
82         while (actual.getSig() != null){
83             actual = actual.getSig();
84         }
85         actual.setSig(nuevo);
86         numElem++;
87     }
88     System.out.println("Final");
89 }
90
```

```
Start Page x EjercicioListaEnlazadaSimple.java x
Source History
91 //metodo para borrar algun dato de la lista
92 public void borrar (int elem){
93     if (primero == null)
94         System.out.println("lista vacia");
95     else
96         if (primero.getElemto() == elem){
97             primero = primero.getSig();
98             numElem--;
99         }
100         else {
101             Nodo actual = primero;
102             while (actual.getSig() != null && actual.getSig().getElemto() != elem)
103                 actual = actual.getSig();
104             if (actual.getSig() == null)
105                 System.out.println ("elemento "+elem+" no esta en la lista");
106             else{
107                 actual.setSig(actual.getSig().getSig());
108                 numElem--;
109             }
110         }
111 }
112
113 //metodo para mostrar los datos de la lista
114 public void listar(){
115     Nodo actual = primero;
116     while (actual.getSig() != null){
117         System.out.println(actual.getElemto());
118         actual = actual.getSig();
119     }
120     System.out.println(actual.getElemto());
121 }
122 }
123
124
125
```

Ejecución del programa ejemplo lista enlazadas:

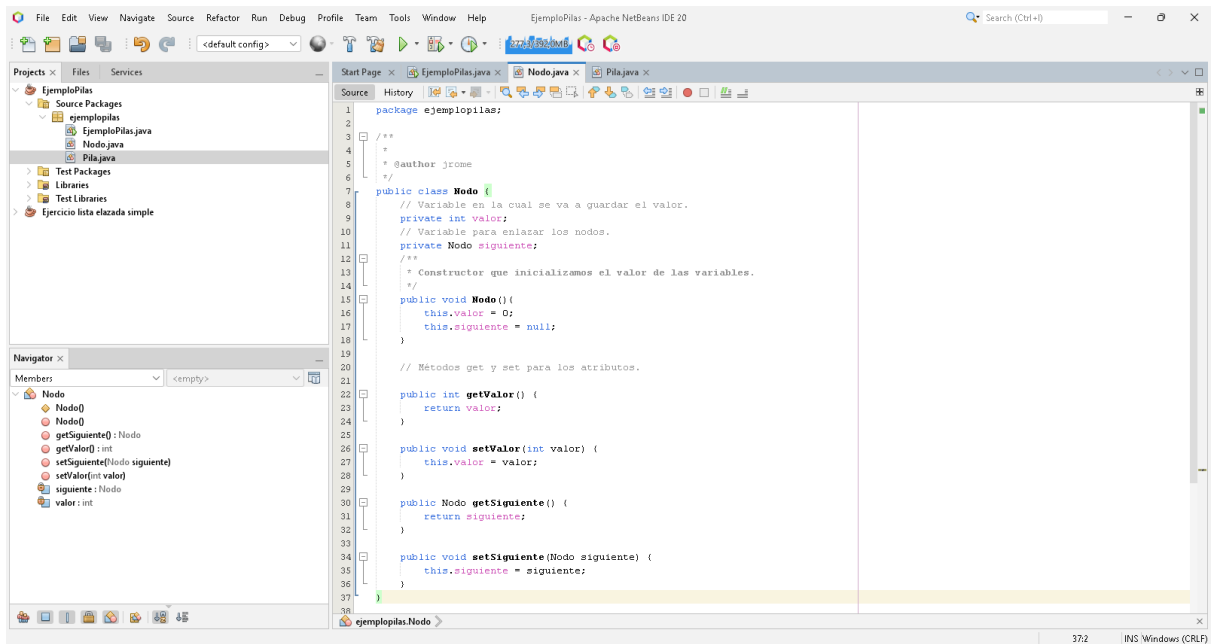
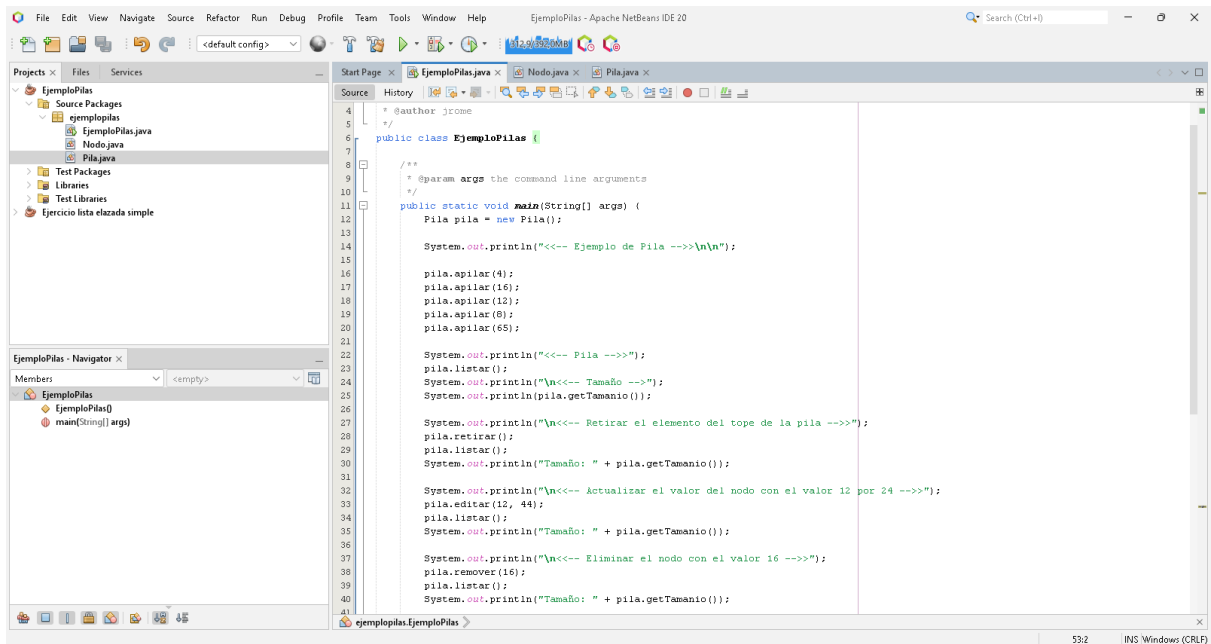


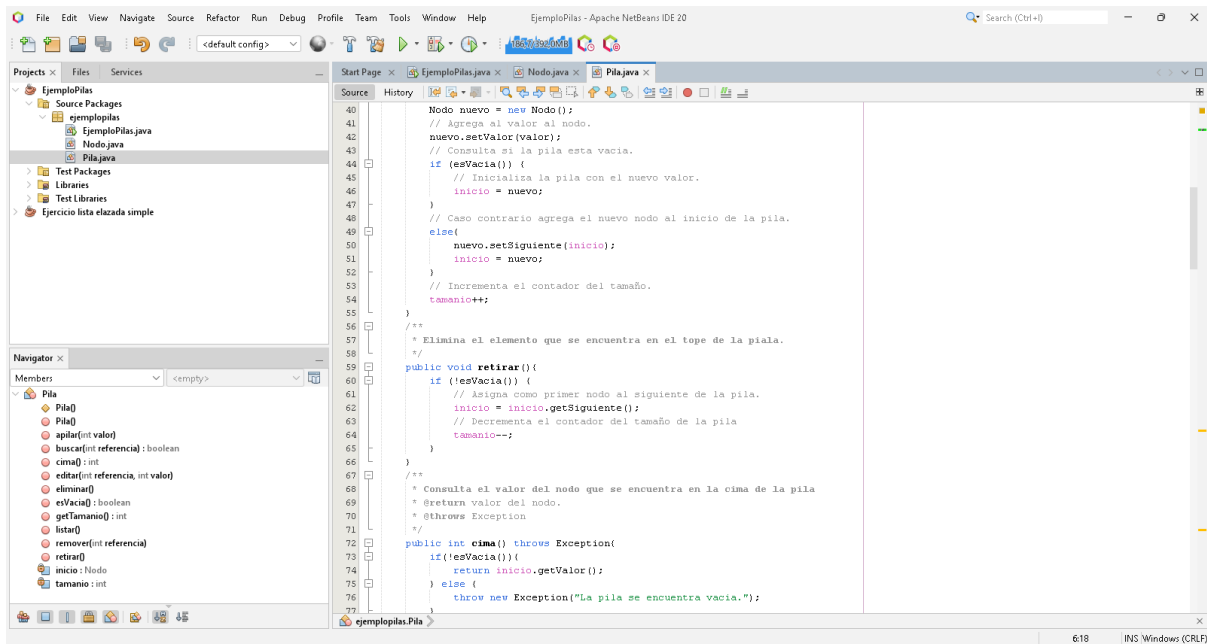
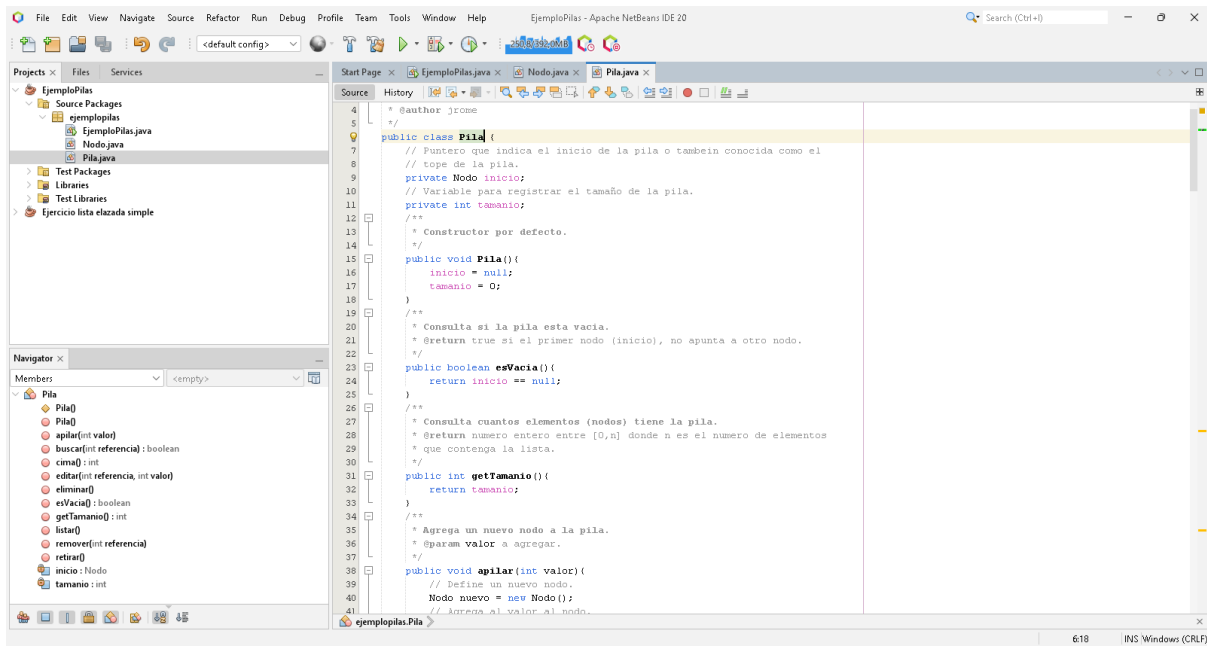
```
run:
Comenzamos
Final
Final
Listamos desde main
1
2
3
4
Borramos un Elemento
elemento 5 no esta en la lista
volvemos a listar
1
3
4
BUILD SUCCESSFUL (total time: 0 seconds)
```

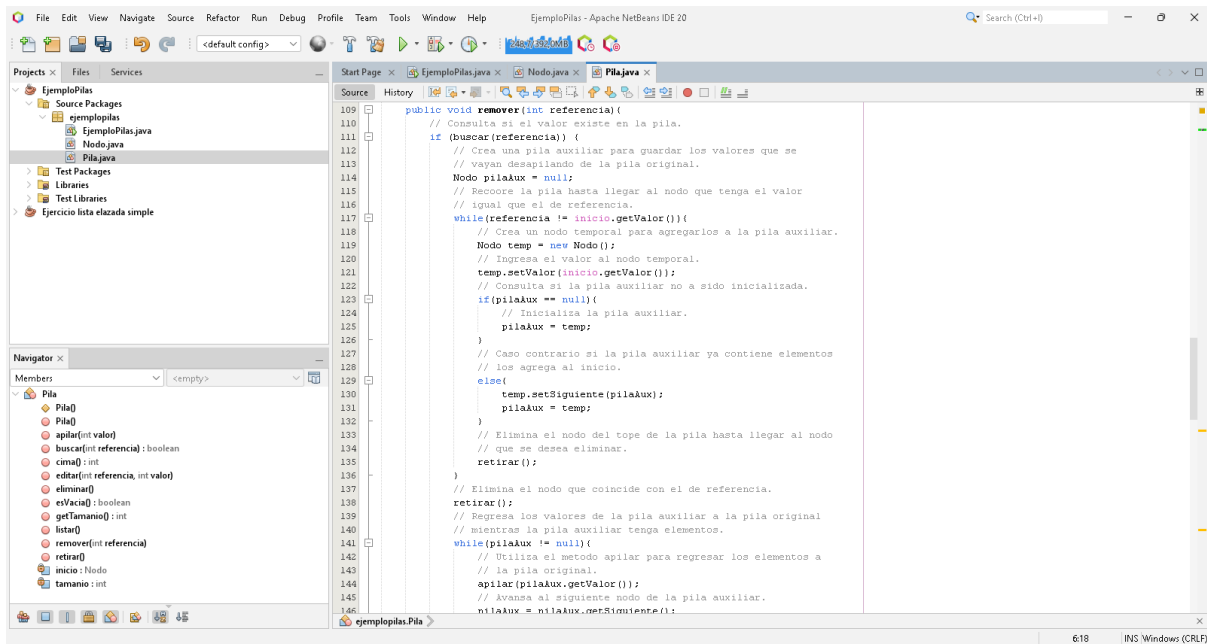
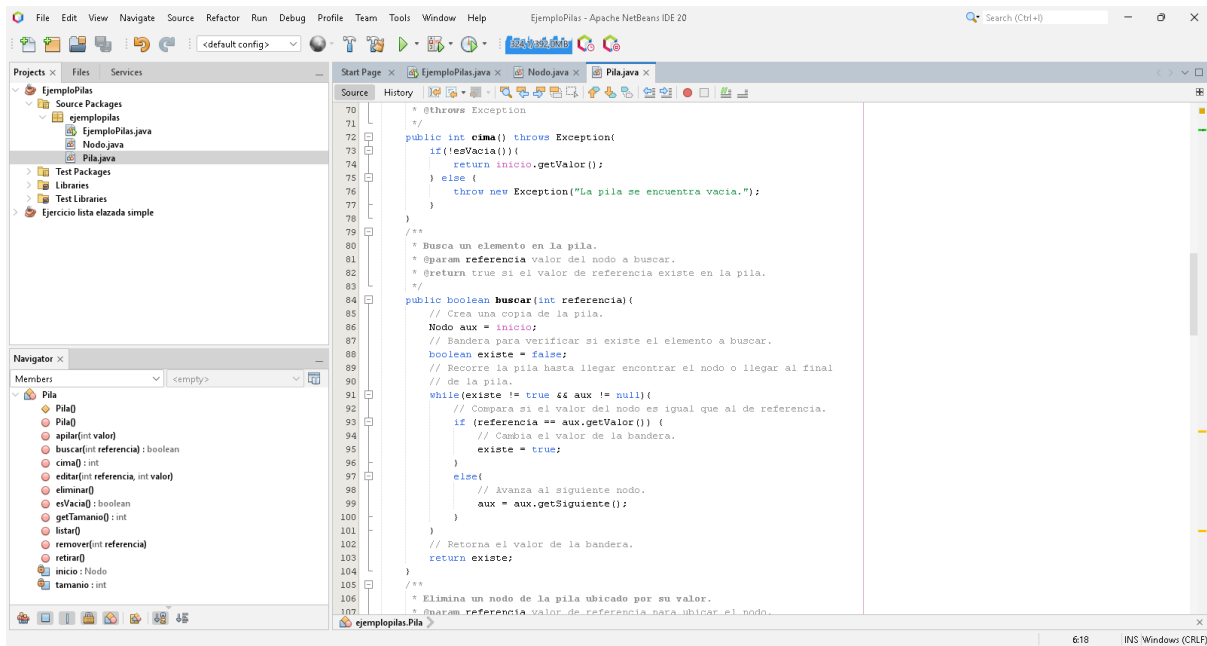
Pilas:

Stack en Java generalmente significa la clase de Collection Framework que implementa la interfaz List. Funciona según el principio de la estructura de datos Stack, que se utiliza para organizar uno de los tipos de memoria. También podría ser parte de la memoria para guardar datos. En este artículo, prestaremos atención en primer lugar a la clase Stack , consideraremos sus métodos y daremos ejemplos. Pero también hablaremos sobre una estructura de datos como Stack y para qué se utiliza.(Squirrels, 2023).

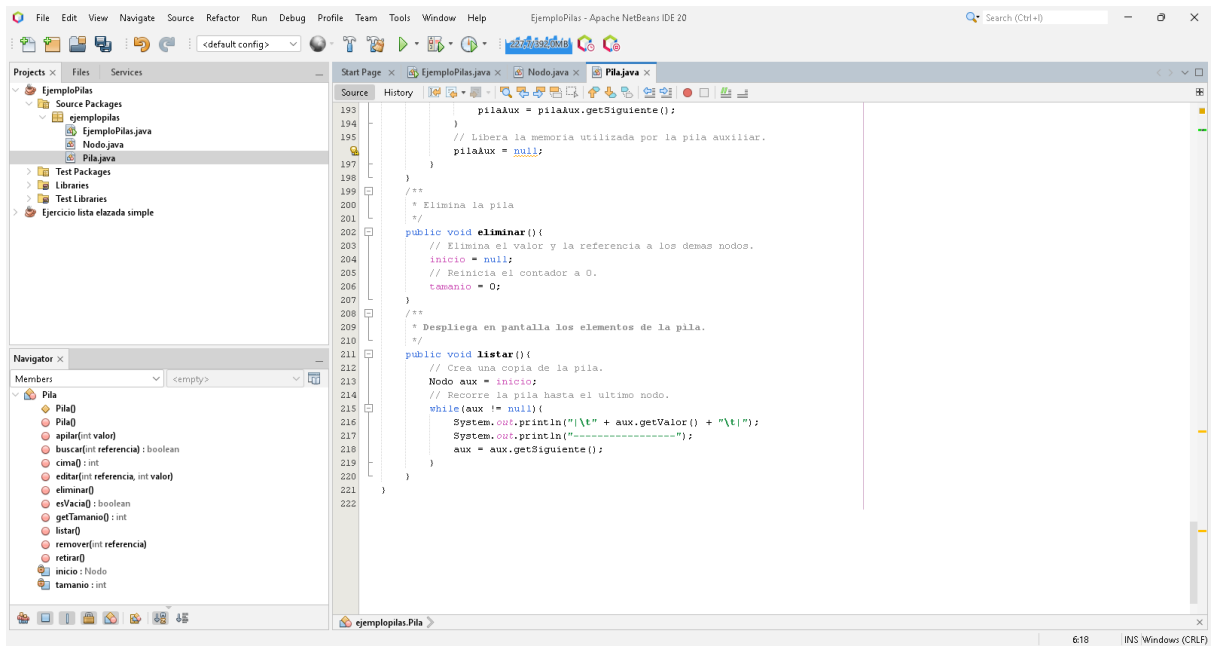
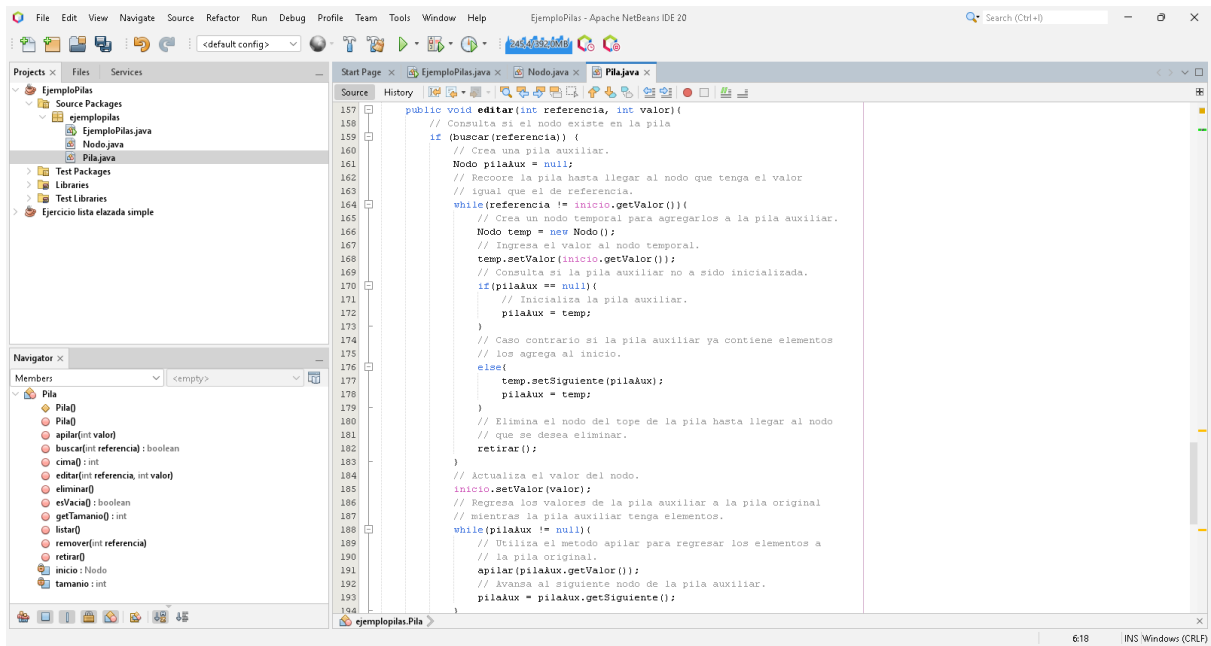
Código ejemplo pilas:











Ejecución del programa ejemplo pilas:

```
Output - EjemploPilas (run) ×
run:
<<-- Ejemplo de Pila -->>

<<-- Pila -->>
| 65 |
-----
| 8 |
-----
| 12 |
-----
| 16 |
-----
| 4 |
-----

<<-- Tamaño -->
5

<<-- Retirar el elemento del tope de la pila -->>
| 8 |
-----
| 12 |
-----
| 16 |
-----
| 4 |
-----

Tamaño: 4

<<-- Actualizar el valor del nodo con el valor 12 por 24 -->>
| 8 |
-----
| 44 |
-----
| 16 |
-----
| 4 |
-----
```

```
Output - EjemploPilas (run) ×
| 16 |
-----
| 4 |
-----
Tamaño: 4

<<-- Actualizar el valor del nodo con el valor 12 por 24 -->>
| 8 |
-----
| 44 |
-----
| 16 |
-----
| 4 |
-----

Tamaño: 4

<<-- Eliminar el nodo con el valor 16 -->>
| 8 |
-----
| 44 |
-----
| 4 |
-----

Tamaño: 3

<<-- Consulta si existe el valor 65 -->>
false

<<-- Elimina la pila -->>

<<-- Consulta si la pila esta vacia -->>
true

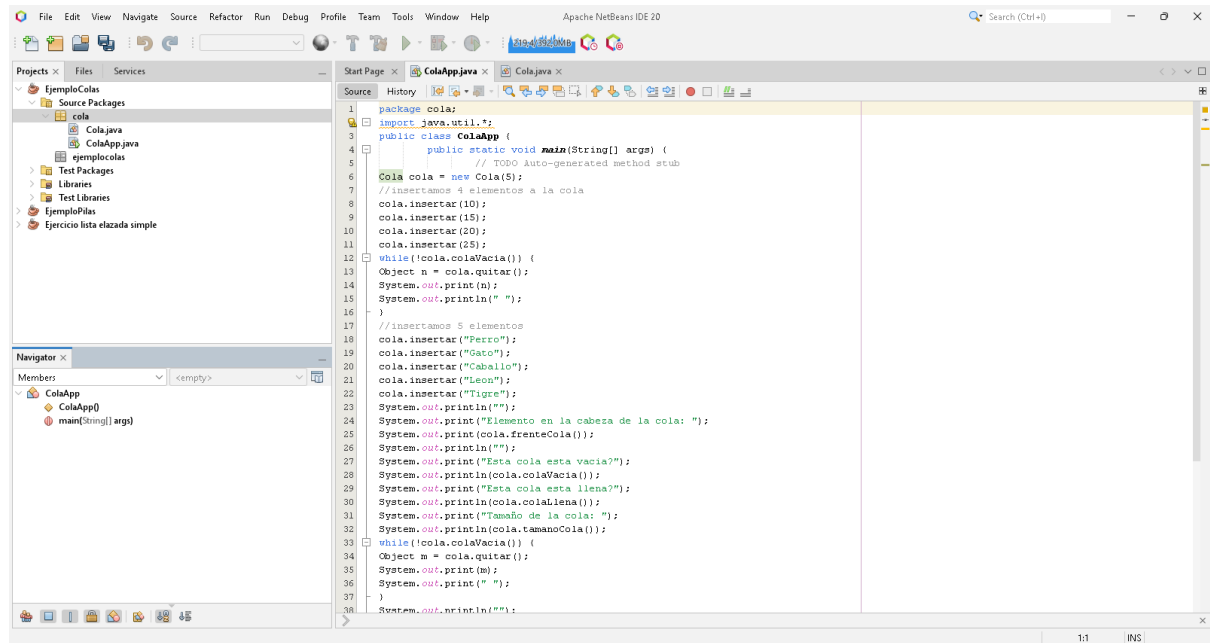
<<-- Fin de ejemplo pila -->>
BUILD SUCCESSFUL (total time: 0 seconds)
```

Colas:

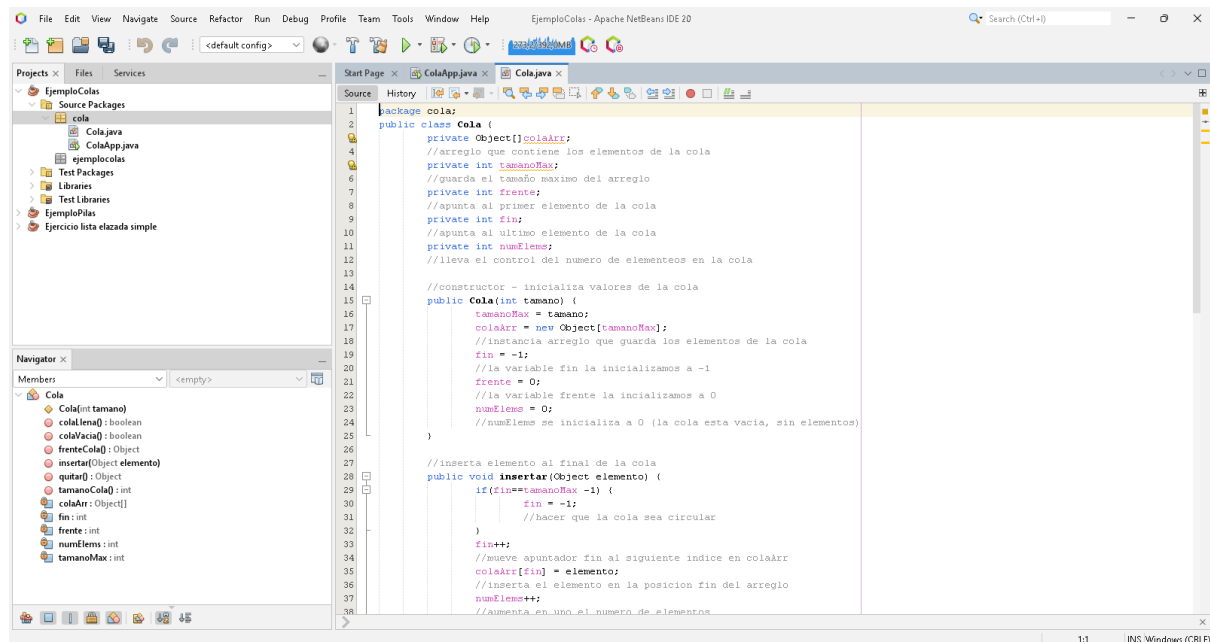
Una cola es una estructura de datos abstractos lineales con el orden particular de realizar operaciones: primero en entrar, primero en salir (FIFO). Eso significa que puede agregar un elemento (o ponerlo en la cola) solo al final de la estructura y tomar un elemento (quitarlo de la cola o eliminarlo de la cola) solo desde el principio. Puede imaginar la estructura de datos de la cola muy fácilmente. Parece una

cola o una fila de clientes en la vida real. El cliente que llegó primero, también será atendido primero. Si tiene cuatro personas en fila en McDonalds o en otro lugar, el primero en hacer fila será el primero en llegar a la tienda. Si llega el nuevo cliente, será el 5º en la fila para conseguir hamburguesas. (Squirrels, 2023)

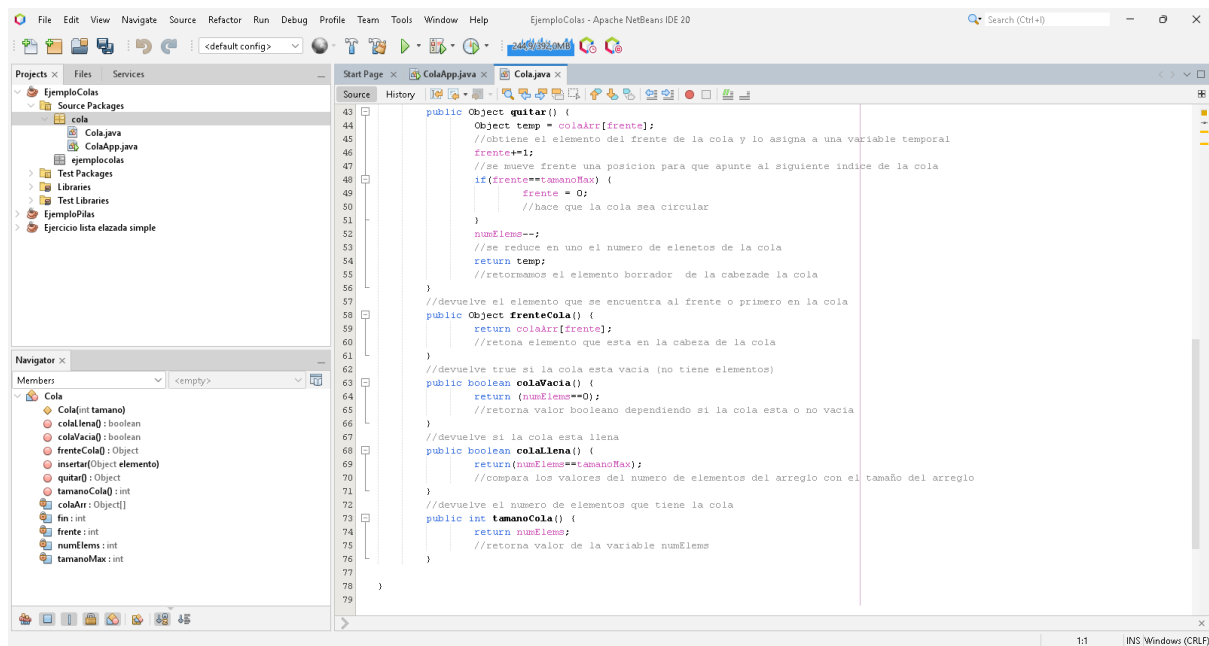
Código ejemplo colas:



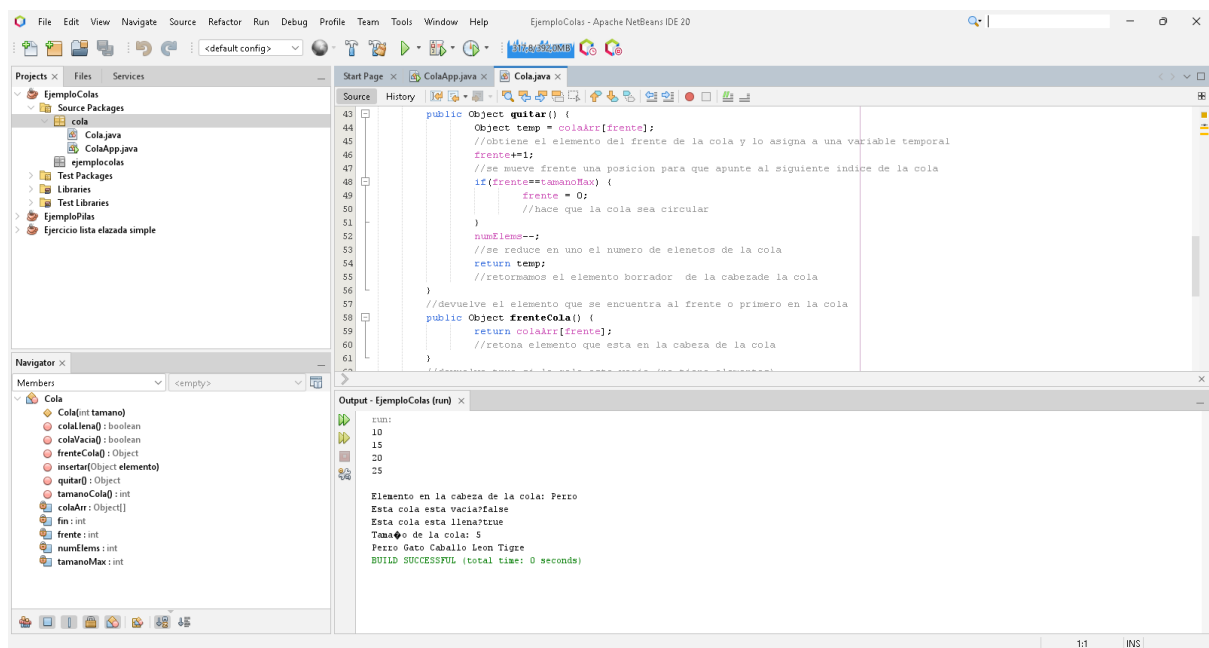
```
1 package cola;
2 import java.util.*;
3 public class ColaApp {
4     public static void main(String[] args) {
5         // TODO auto-generated method stub
6         Cola cola = new Cola(5);
7         //insertamos 4 elementos a la cola
8         cola.insertar(10);
9         cola.insertar(15);
10        cola.insertar(20);
11        cola.insertar(25);
12        while(!cola.colaVacia()) {
13            Object n = cola.quitar();
14            System.out.println(n);
15            System.out.println(" ");
16        }
17        //insertamos 5 elementos
18        cola.insertar("Perrito");
19        cola.insertar("Gato");
20        cola.insertar("Caballito");
21        cola.insertar("Leon");
22        cola.insertar("Tigre");
23        System.out.println("");
24        System.out.print("Elemento en la cabeza de la cola: ");
25        System.out.print(cola.frenteCola());
26        System.out.println("");
27        System.out.print("Esta cola esta vacia?");
28        System.out.print(cola.colaVacia());
29        System.out.print("Esta cola esta llena?");
30        System.out.print(cola.colaLlena());
31        System.out.print("Tamaño de la cola: ");
32        System.out.print(cola.tamanoCola());
33        while(!cola.colaVacia()) {
34            Object m = cola.quitar();
35            System.out.print(m);
36            System.out.print(" ");
37        }
38        System.out.println("");
39    }
40 }
```



```
1 package cola;
2 public class Cola {
3     private Object[] colaArr;
4     //arreglo que contiene los elementos de la cola
5     private int tamanoMax;
6     //guarda el tamaño maximo del arreglo
7     private int frente;
8     //apunta al primer elemento de la cola
9     private int fin;
10    //apunta al ultimo elemento de la cola
11    private int numElem;
12    //lleva el control del numero de elementos en la cola
13
14    //constructor - inicializa valores de la cola
15    public Cola(int tamano) {
16        tamanoMax = tamano;
17        colaArr = new Object[tamanoMax];
18        //instancia arreglo que guarda los elementos de la cola
19        fin = -1;
20        //la variable fin la inicializamos a -1
21        frente = 0;
22        //la variable frente la inicializamos a 0
23        numElem = 0;
24        //numElem se inicializa a 0 (la cola esta vacia, sin elementos)
25    }
26
27    //inserta elemento al final de la cola
28    public void insertar(Object elemento) {
29        if(fin==tamanoMax-1) {
30            fin = -1;
31            //hacer que la cola sea circular
32        }
33        fin++;
34        //mueve apuntador fin al siguiente indice en colaArr
35        colaArr[fin] = elemento;
36        //inserta el elemento en la posicion fin del arreglo
37        numElem++;
38        //aumenta en uno el numero de elementos
39    }
40 }
```



Ejecución del programa:



## Conclusiones

- Las pilas, colas y listas son estructuras de datos fundamentales en informática, cada una con sus propias características y aplicaciones específicas.
- El entendimiento profundo de estas estructuras de datos es esencial para el desarrollo de algoritmos eficientes y la resolución de problemas de manera óptima.
- La correcta implementación y utilización de pilas, colas y listas puede contribuir significativamente a la optimización de procesos y recursos en el desarrollo de software.

## Recomendaciones

- Realizar ejercicios prácticos que involucren la manipulación de pilas, colas y listas para afianzar los conceptos teóricos.
- Explorar aplicaciones reales de estas estructuras de datos en el desarrollo de software, como la gestión de tareas, la simulación de procesos y la gestión de recursos.
- Buscar recursos adicionales, como tutoriales, ejemplos de código y problemas de programación, para seguir fortaleciendo el entendimiento y habilidades en el uso de pilas, colas y listas.

## Bibliografía:

DeltaPC. (2019, enero 28). *Java Listas enlazadas Simples*. Delta PC; Delta PC Informatica y Electrónica S.L. <https://www.deltapci.com/java-listas-enlazadas-simples/>

*Listas, pilas y colas*. (2021, mayo 14). Google.com.  
<https://sites.google.com/site/programacionbasicajava/listas-pilas-y-colas>

Squirrels, J. (2023a, julio 21). *Java Queue Interface y sus implementaciones*. CodeGym.  
<https://codegym.cc/es/groups/posts/es.307.java-queue-interface-y-sus-implementaciones>

Squirrels, J. (2023b, julio 21). *Pila de Java*. CodeGym.  
<https://codegym.cc/es/groups/posts/es.828.pila-de-java>