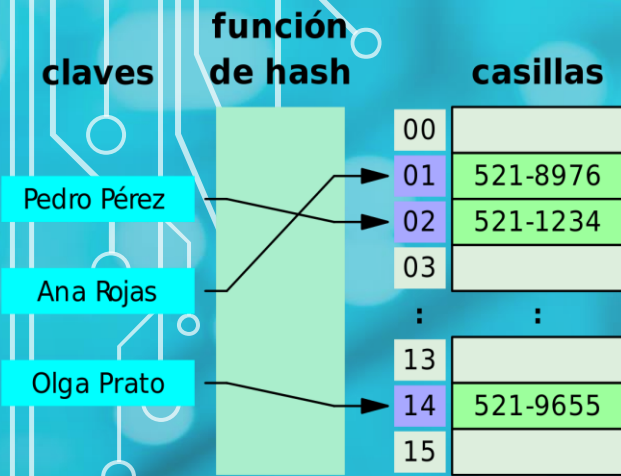


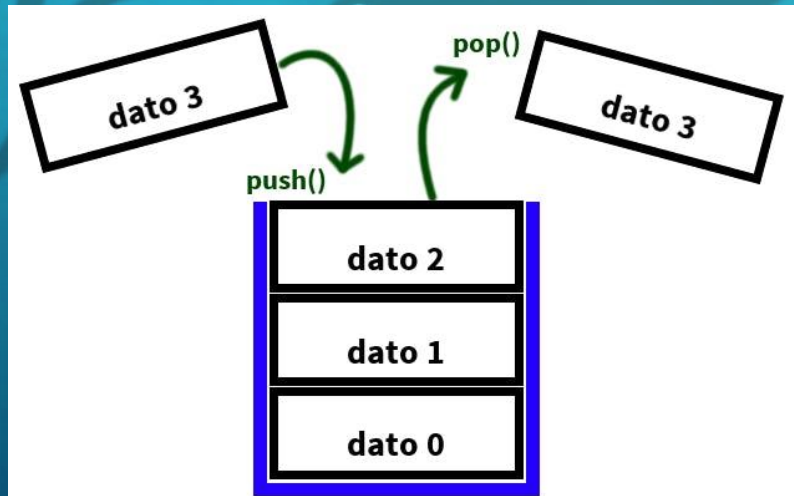
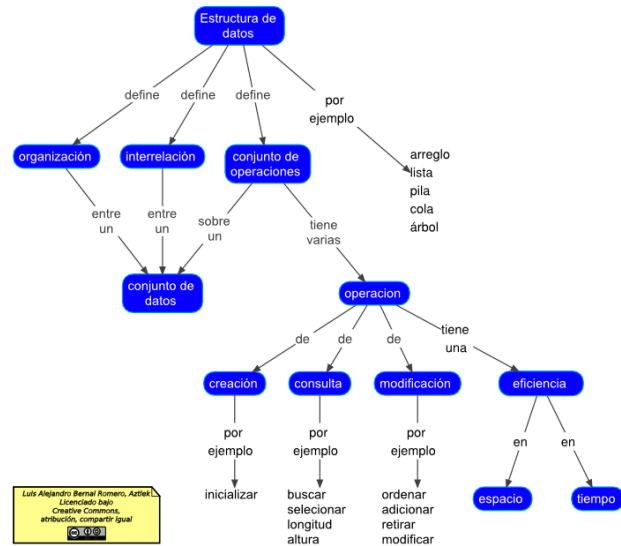
# ESTRUCTURA DE DATOS

SERGIO ANDRES MENDOZA ALVARADO

[https://github.com/SergioAndresMendozaAlvarado/ESTRUCUTRA-DE-DATOS/blob/main/Hito%204/LAB1/Practica\\_Final\\_H4/CODIGO/main.java](https://github.com/SergioAndresMendozaAlvarado/ESTRUCUTRA-DE-DATOS/blob/main/Hito%204/LAB1/Practica_Final_H4/CODIGO/main.java)



## Estructura de Datos

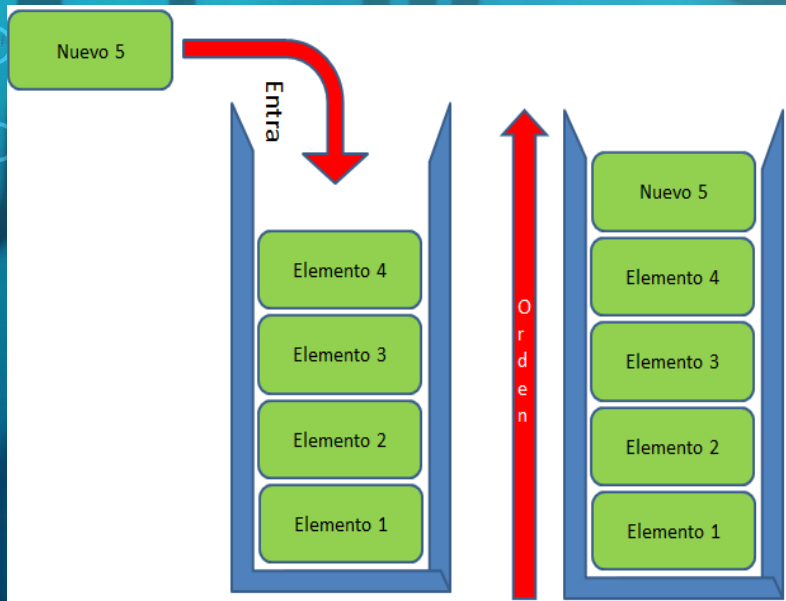
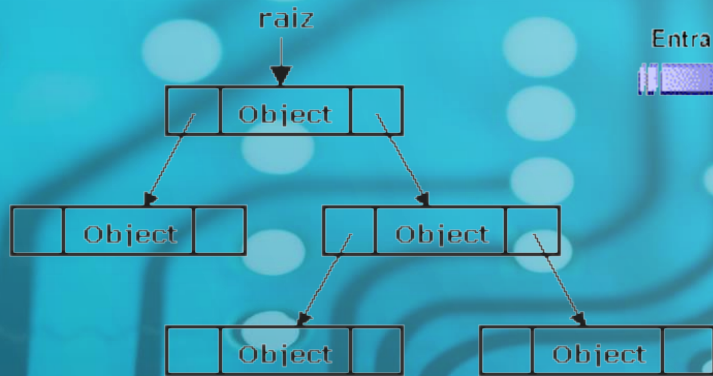


¿A QUE NOS REFERIMOS CUANDO HABLAMOS DE ESTRUCTURA DE DATO?

- Es una forma de organizar y almacenar datos en un orden específico, para facilitar su manipulación y procesamiento. Las estructuras de datos se utilizan en una amplia variedad de aplicaciones de software, desde bases de datos hasta algoritmos de programación y representaciones gráficas.

```
List<String> ejemploLista = new ArrayList<String>();
ejemploLista.add("Juan");
ejemploLista.add("Pedro");
ejemploLista.add("José");
ejemploLista.add("María");
ejemploLista.add("Sofía");
```

```
System.out.println(ejemploLista.subList(0, 2));
```



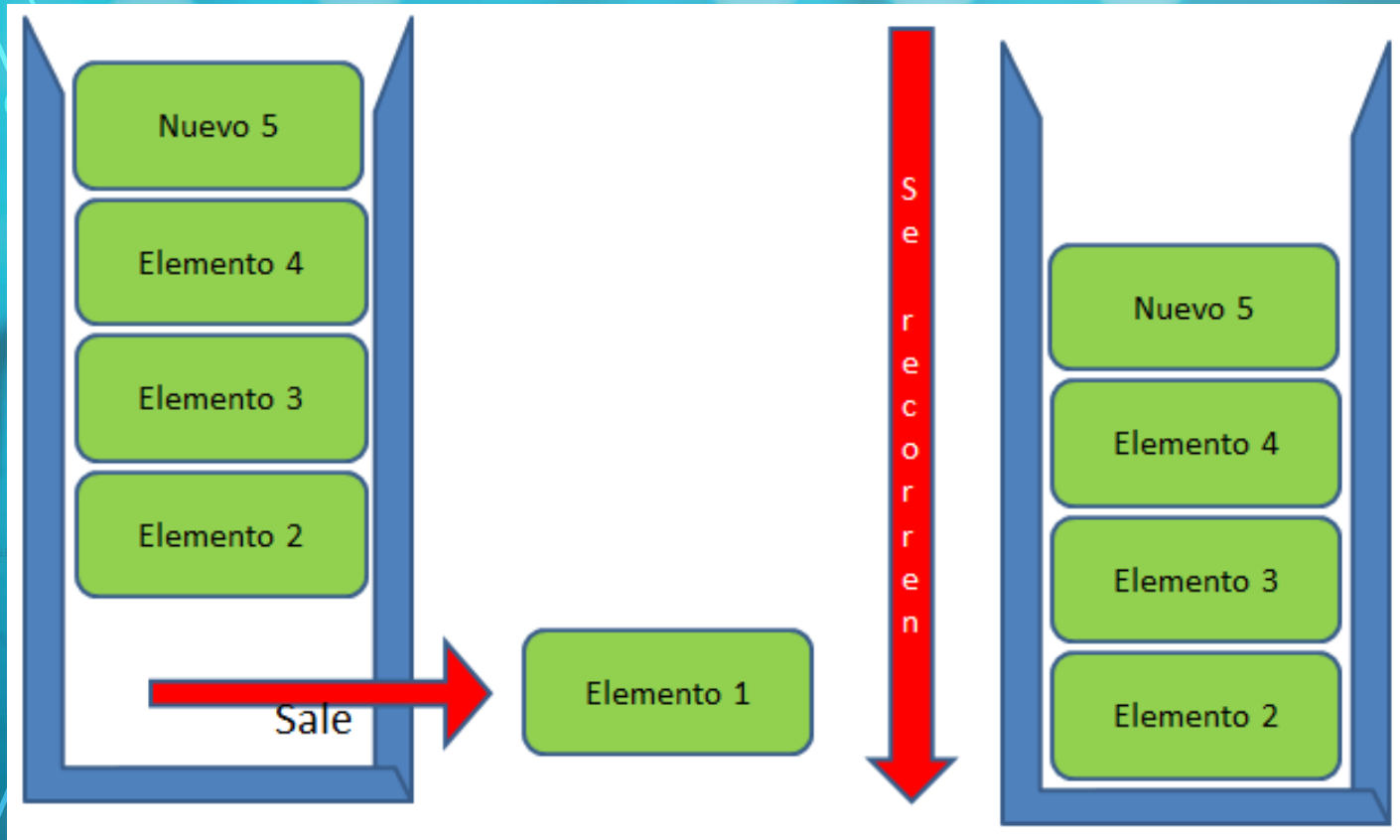
# ¿QUÉ TIPO DE ESTRUCTURA DE DATOS EXISTEN?

- Listas: son estructuras de datos que se utilizan para almacenar una colección de elementos ordenados. Las listas pueden ser simples o dobles, y pueden ser implementadas como matrices o listas enlazadas.
- Pilas: son estructuras de datos que se utilizan para almacenar elementos en un orden específico. La operación de inserción se llama push, y la operación de eliminación se llama pop. Las pilas se utilizan para procesar tareas en orden inverso al que se recibieron.
- Colas: son estructuras de datos que se utilizan para almacenar elementos en un orden específico. La operación de inserción se llama enqueue, y la operación de eliminación se llama dequeue. Las colas se utilizan para procesar tareas en el orden en que se recibieron.
- Árboles: son estructuras de datos que se utilizan para representar relaciones jerárquicas entre los datos. Los árboles pueden ser de búsqueda, de expresión, de decisión, entre otros.

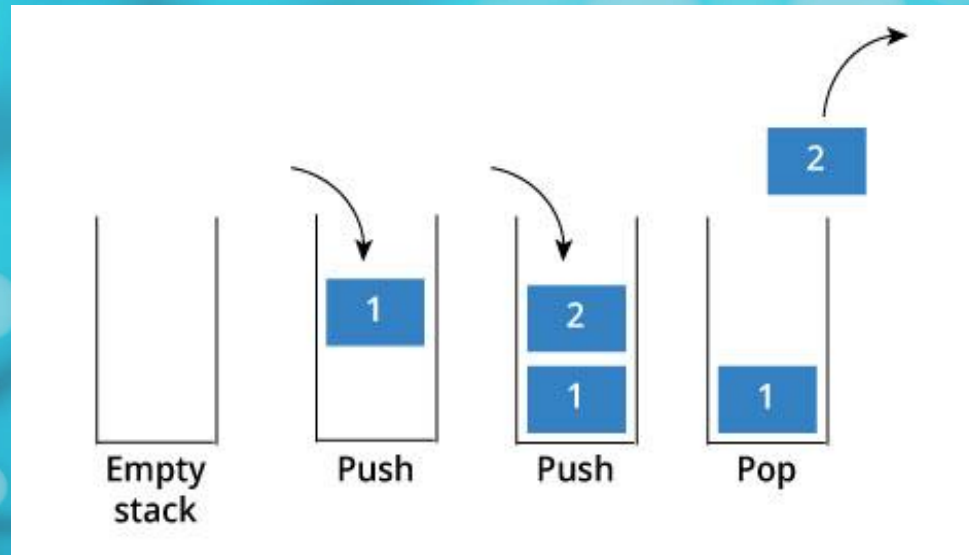


# ¿QUE SIGNIFICA FIFO?

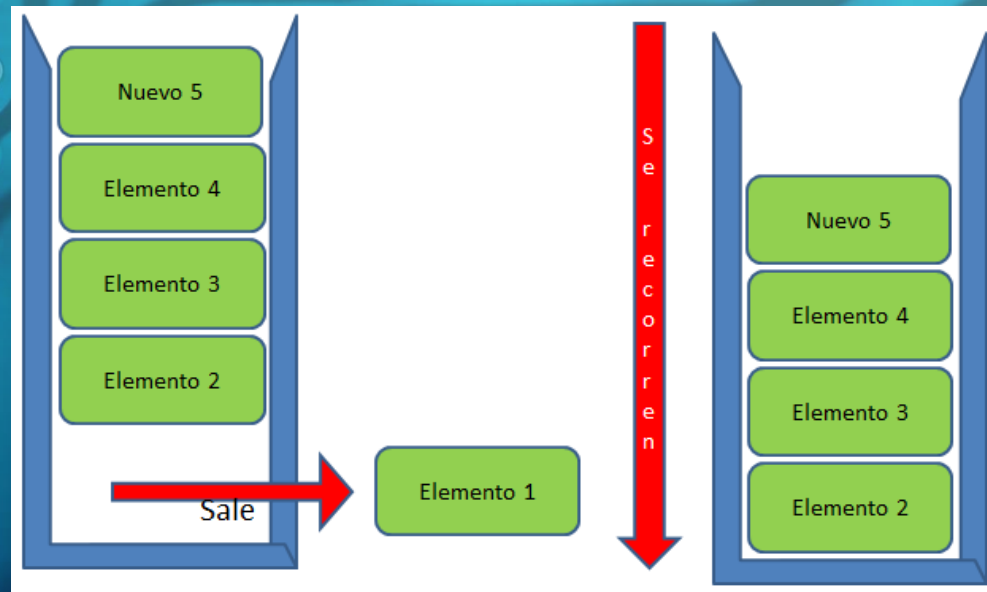
- First Input First Output
- La traducción sería: primero en entrar primero en salir.
- Refleja muy bien lo que es el funcionamiento de una cola ya que funciona de esa manera, el primer dato en entrar es el primero en salir, tomemos en cuenta un ejemplo de una fila, el primero en llegar es el primero en ser atendido y el primero en irse, así es el funcionamiento de una cola



## PILA



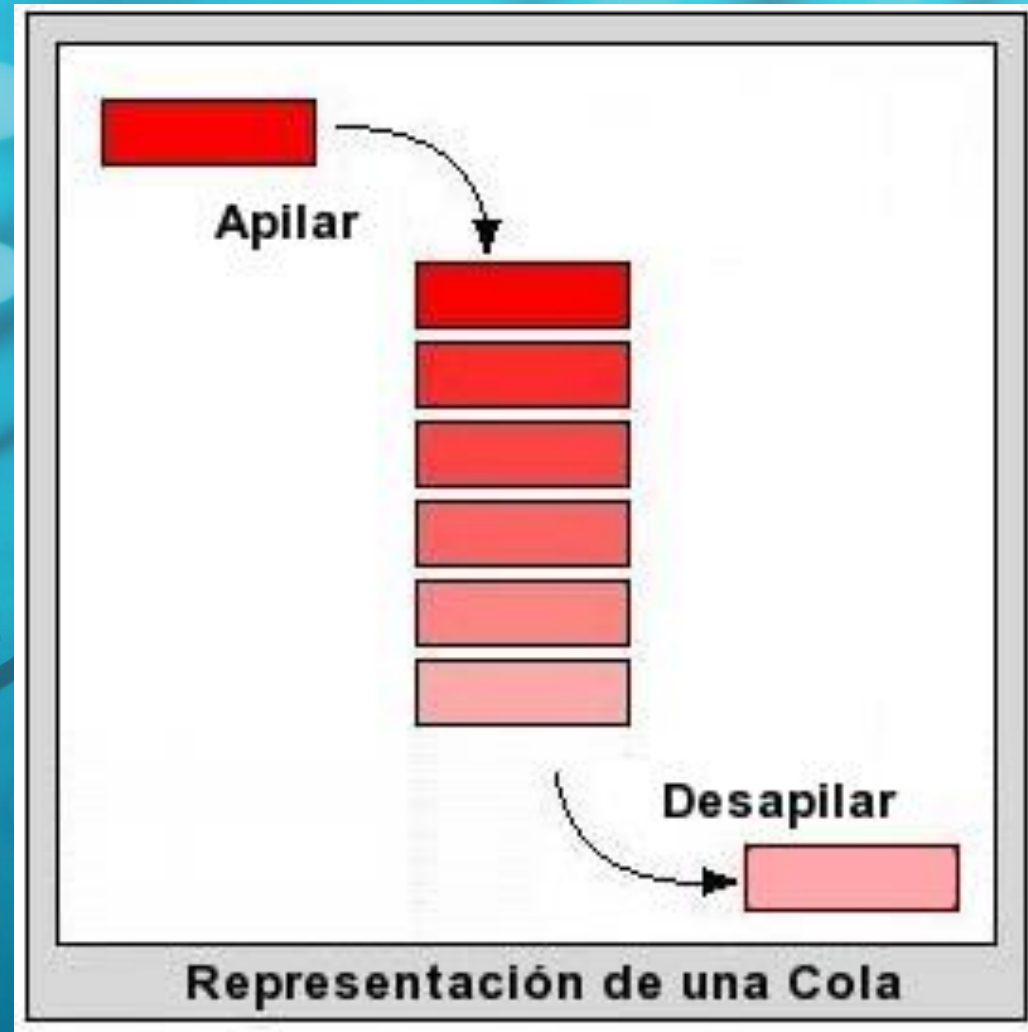
## Cola



## ¿MUESTRA LA DIFERENCIA ENTRE LIFO Y FIFO?

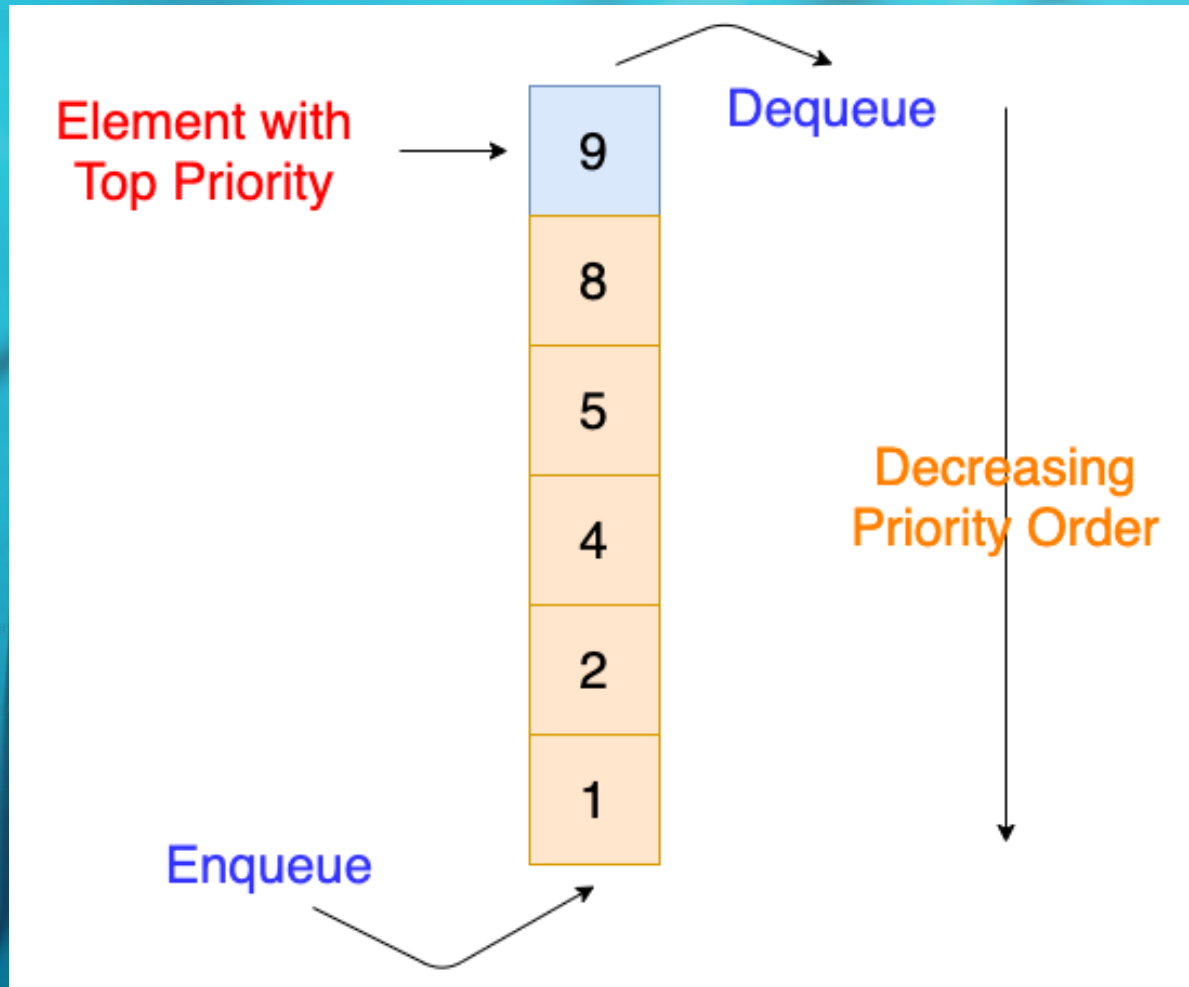
- La diferencia las grande es:
- FIRST INPUT FIRST OUTPUT
- LAST INPUT FIRST OUTPUT
- La traducción de FIFO es: primero en entrar primero en salir, lo que refleja el funcionamiento de una cola, mientras que LIFO traducido es: ultimo en entrar primero en salir, lo cual refleja el funcionamiento de una pila.

# ¿QUÉ ES UNA COLA?



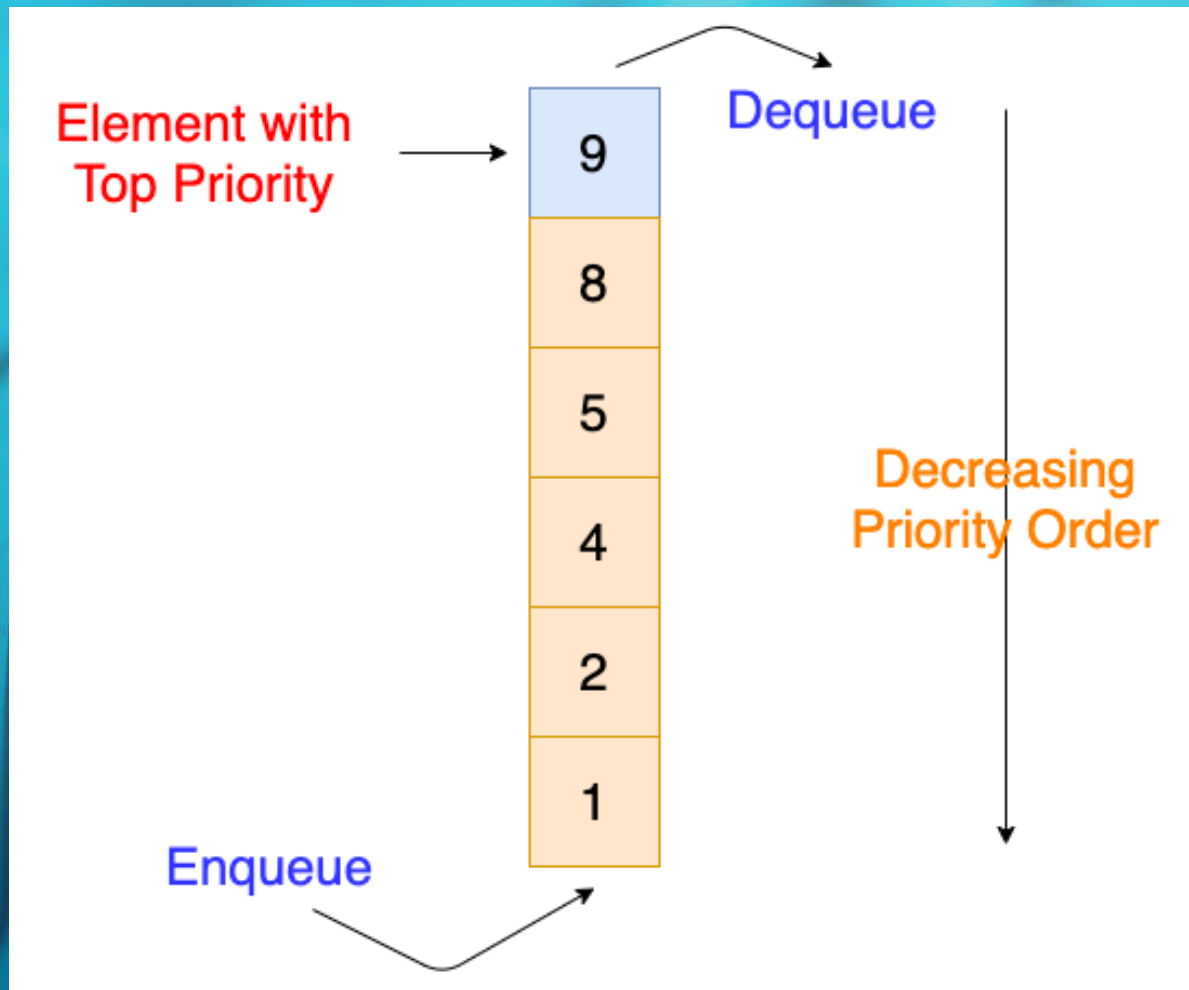
- La cola se gestiona mediante el principio conocido como "FIFO" (First In, First Out), lo que significa que el primer elemento en ser agregado a la cola será el primero en ser eliminado. A menudo, se denomina al elemento agregado primero como el "frente" de la cola y al elemento agregado más recientemente como el "final" o "cola" de la misma.
- La operación principal en una cola es la inserción de elementos, que se realiza en el extremo final de la cola y se conoce como "enqueue". Asimismo, la eliminación de elementos se realiza en el extremo frontal y se conoce como "dequeue". Esto garantiza que los elementos se procesen en el orden en que fueron agregados.

# ¿QUÉ ES TOPE EN UNA PILA?



- En Java, Queue es una interfaz que representa una cola en el lenguaje de programación. La interfaz Queue se encuentra en el paquete `java.util` y define los métodos comunes para trabajar con colas.
- Una Queue en Java es similar a una cola en estructura de datos, siguiendo el principio FIFO (First In, First Out). Esto significa que los elementos se insertan al final de la cola y se eliminan desde el frente de la misma.
- La interfaz Queue en Java proporciona métodos como `add()` o `offer()` para agregar elementos a la cola, `remove()` o `poll()` para eliminar y devolver el elemento frontal, y `peek()` para obtener el elemento frontal sin eliminarlo.

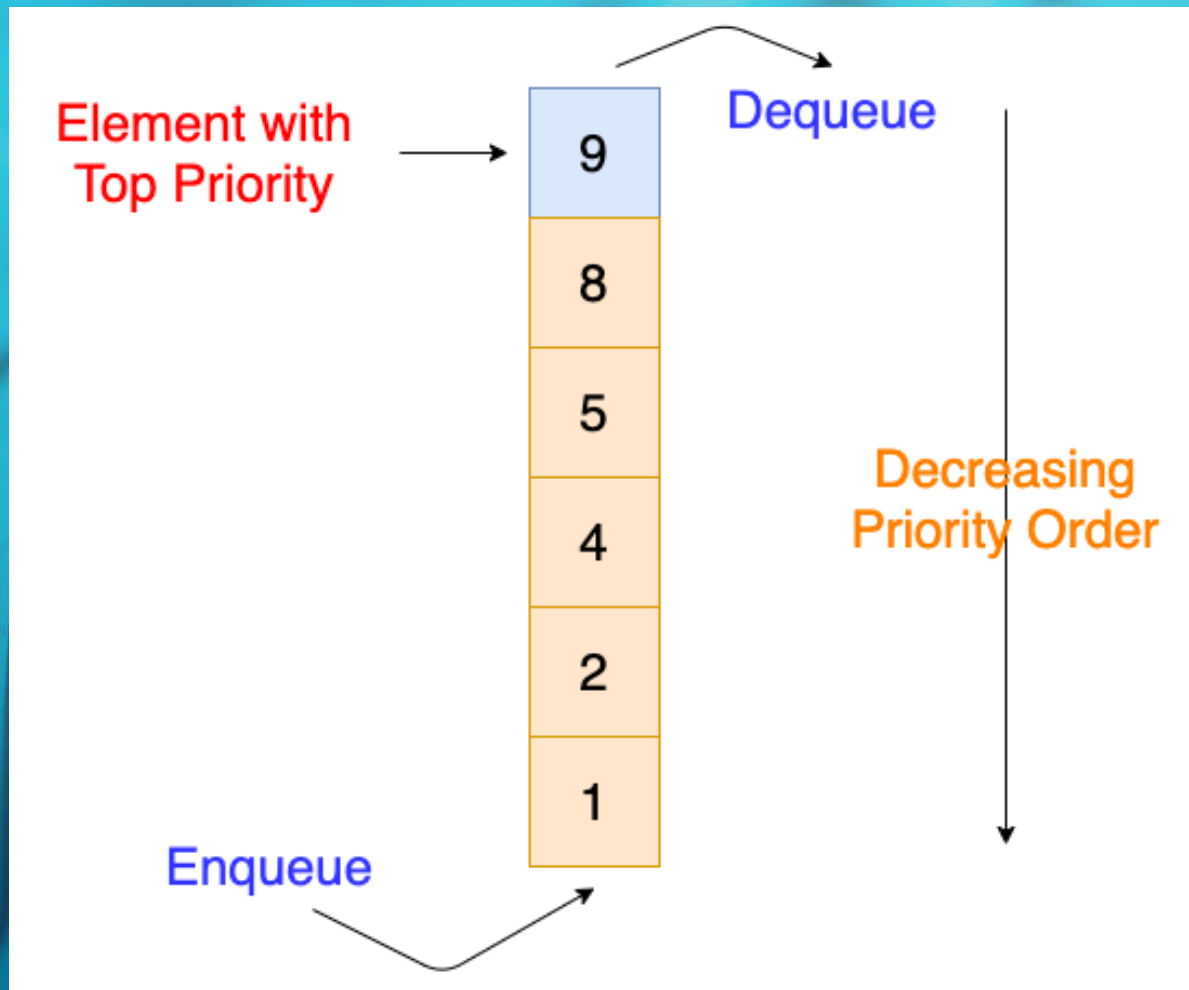




## ¿QUÉ ES INI O REAR EN UNA COLA?

- "INI" hace referencia al extremo trasero o de la cola, es decir, al punto de entrada donde se agregan nuevos elementos. Es el lugar por donde se realiza la operación de "enqueue" o inserción de elementos en la cola.





## ¿QUÉ ES INI O REAR EN UNA COLA?

- "FRONT": Se refiere al extremo inicial de la cola, similar a "INI". Representaría el punto de entrada de la cola donde se agregan nuevos elementos.

# ¿AQUE SE REFIEREN LOS MÉTODOS ES\_VACIO(), ES\_LLENO(), EN UNA PILA?

SON MÉTODOS DE TIPO BOOLEANO, Y RETORNAN VALORES DE TIPO BOOLEANO.

EL MÉTODO ESVACIO(), UTILIZA UN IF QUE PREGUNTA A LA COLA SI HAY VALORES O NO, EL MÉTODO ESLLENO(), UTILIZA UN IF QUE PREGUNTA A LA PILA SI ESTA LLENA, Y SI ESTA LLENA AUNQUE SE INGRESEN MAS DATOS, SIMPLEMENTE DESAPARECERAN.

8 usages

```
public boolean EsVacio(){  
    if(this.tope == 0){  
        return true;  
    }else{  
        return false;  
    }  
}
```

```
public boolean EsLleno(){  
    if(this.tope == this.max){  
        return true;  
    }else{  
        return false;  
    }  
}
```

# ¿QUÉ SON LOS MÉTODOS ESTÁTICOS EN JAVA?

Un método estático, es muy diferente al método estatico Main, en este método no podemos correr código, solamente podemos desarrollar la lógica o el procedimiento que realizara.

En cambio el método estatico Main puedes correr código, entonces ¿para que sirven los métodos estáticos no main si no pueden correr codigo?

Si que pueden correr código cuando se les instancia en el método Main, véamoslo como una clase con menos complejidad que realizara un procedimiento según nuestros requerimientos.

```
suma( a: 4, b: 7); //INSTANCIA
}
```

1 usage

```
public static void suma(int a, int b){ //METODO ESTATICO

    double r;
    r = a+b;
    System.out.println("La suma total es: "+r);

}
```

Run: Main x

C:\Users\PC\.jdk\openjdk-19.0.2\bin\java.exe "-javaagent  
La suma total es: 11.0

```
no usages
public class Main {
    no usages
    public static void main(String[] args){

        suma( a: 4, b: 7); //INSTANCIA
```



ColaClientes		
m	ColaClientes ()	
f	ini	int
f	clientes	Cliente []
f	max	int
f	fin	int
m	esVacia()	boolean
m	Adicionar (Cliente)	void
m	esLlena()	boolean
m	vaciar(ColaClientes)	void
m	nroElem ()	int
m	eliminar ()	Cliente
m	mostrar()	void

A través de un gráfico, muestre los métodos mínimos que debería de tener una COLA

Los métodos mínimos que debe tener una cola se ve en el siguiente grafico, como podemos son 7, aun pueden existir mas métodos que ayuden a nuestra necesidad, pero estos son los principales que nos ayudaran a hacer el uso de una pila en Java.

Clase

Cola

## EJERCICIO N° 11

- Crear la clase Cliente
- Crear la clase ColaDeClientes
- Crear la clase Main

Met. estatico

```
1 package ColaClientes;
2
3 public class Main {
4     public static void main(String[] args){
5
6         ColaClientes cola_client = new ColaClientes();
7
8         Cliente c1 = new Cliente( nombre: "Andres", apellidos: "Mendoza", edad: 19, pais: "Bolivia", genero: "M", tipo: "Gold");
9         Cliente c2 = new Cliente( nombre: "Jose", apellidos: "Condori", edad: 71, pais: "Chile", genero: "M", tipo: "Gold");
10        Cliente c3 = new Cliente( nombre: "Saul", apellidos: "Alanoca", edad: 21, pais: "Argentina", genero: "M", tipo: "Silver");
11        Cliente c4 = new Cliente( nombre: "Luis", apellidos: "Choque", edad: 22, pais: "Bolivia", genero: "M", tipo: "Silver");
12        Cliente c5 = new Cliente( nombre: "Marcos", apellidos: "Quispe", edad: 60, pais: "Chile", genero: "M", tipo: "Gold");
13
14    }
```

```
public class Cliente {
    3 usages
    private String nombre;
    3 usages
    private String apellidos;
    3 usages
    private int edad;
    3 usages
    private String pais;
    3 usages
    private String genero;
    3 usages
    private String tipo;

    10 usages
    public Cliente(String nombre, String apellidos, int edad, String pais, String genero, String tipo){
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.edad = edad;
        this.pais = pais;
        this.genero = genero;
        this.tipo = tipo;
    }

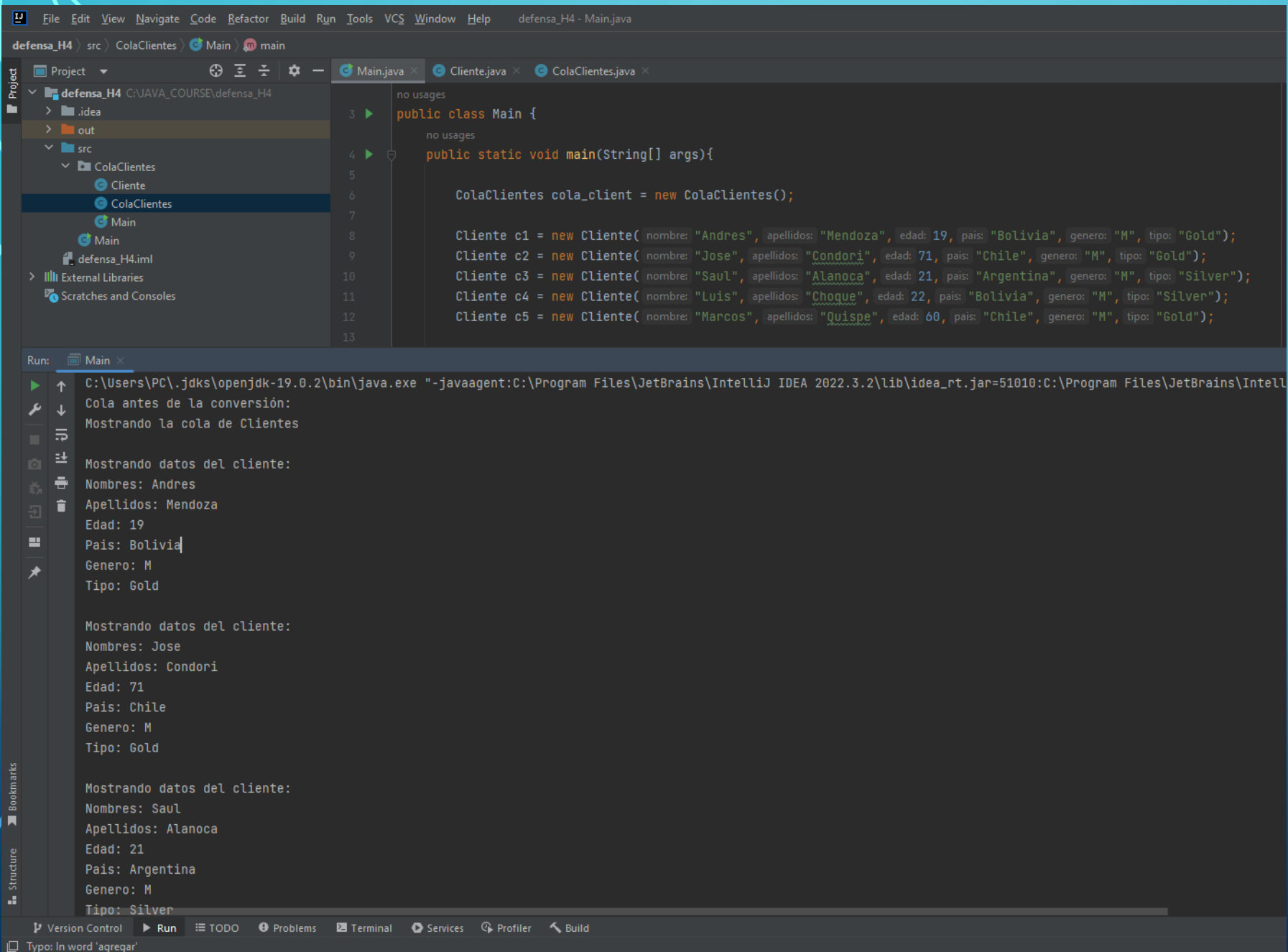
    2 usages
    public String getNombre() {
        return nombre;
    }

    no usages
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    1 usage
}
```

```
1 package ColaClientes;
2
3 public class ColaClientes{
4
5     3 usages
    private int max;
6     8 usages
    private int fin;
7     7 usages
    private int ini;
8     3 usages
    private Cliente[] clientes;

9
10
11     7 usages
    public ColaClientes() {
12
13         this.max = 100;
14         this.fin = 0;
15         this.ini = 0;
16
17         this.clientes = new Cliente[this.max + 1];
18
19     }
20 }
```



The screenshot shows the IntelliJ IDEA IDE with a project named 'defensa\_H4'. The 'Main.java' file is open, displaying the following code:

```
public class Main {  
    public static void main(String[] args){  
  
        ColaClientes cola_client = new ColaClientes();  
  
        Cliente c1 = new Cliente( nombre: "Andres", apellidos: "Mendoza", edad: 19, pais: "Bolivia", genero: "M", tipo: "Gold");  
        Cliente c2 = new Cliente( nombre: "Jose", apellidos: "Condori", edad: 71, pais: "Chile", genero: "M", tipo: "Gold");  
        Cliente c3 = new Cliente( nombre: "Saul", apellidos: "Alanoca", edad: 21, pais: "Argentina", genero: "M", tipo: "Silver");  
        Cliente c4 = new Cliente( nombre: "Luis", apellidos: "Choque", edad: 22, pais: "Bolivia", genero: "M", tipo: "Silver");  
        Cliente c5 = new Cliente( nombre: "Marcos", apellidos: "Quispe", edad: 60, pais: "Chile", genero: "M", tipo: "Gold");  
    }  
}
```

The Run console at the bottom shows the execution output:

```
C:\Users\PC\.jdk\openjdk-19.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.3.2\lib\idea_rt.jar=51010:C:\Program Files\JetBrains\IntelliJ IDEA 2022.3.2\bin" -Dfile.encoding=UTF-8  
Cola antes de la conversión:  
Mostrando la cola de Clientes  
  
Mostrando datos del cliente:  
Nombres: Andres  
Apellidos: Mendoza  
Edad: 19  
Pais: Bolivia  
Genero: M  
Tipo: Gold  
  
Mostrando datos del cliente:  
Nombres: Jose  
Apellidos: Condori  
Edad: 71  
Pais: Chile  
Genero: M  
Tipo: Gold  
  
Mostrando datos del cliente:  
Nombres: Saul  
Apellidos: Alanoca  
Edad: 21  
Pais: Argentina  
Genero: M  
Tipo: Silver
```

## EJERCICIO N° 12

- Crear una cola con 5 clientes.
- En la clase MAIN deberán estar los 5 clientes.
- Mostrar todos los datos de la cola de clientes



The screenshot shows an IDE with three tabs: Main.java, Cliente.java, and ColaClientes.java. The Main.java tab is active, displaying the following code:

```
no usages
public static void main(String[] args){

    ColaClientes cola_client = new ColaClientes();

    Cliente c1 = new Cliente( nombre: "Andres", apellidos: "Mendoza", edad: 19, pais: "Bolivia", genero: "M", tipo: "Gold");
    Cliente c2 = new Cliente( nombre: "Jose", apellidos: "Condori", edad: 71, pais: "Chile", genero: "M", tipo: "Gold");
    Cliente c3 = new Cliente( nombre: "Saul", apellidos: "Alanoca", edad: 21, pais: "Argentina", genero: "M", tipo: "Silver");
    Cliente c4 = new Cliente( nombre: "Luis", apellidos: "Choque", edad: 22, pais: "Bolivia", genero: "M", tipo: "Silver");
    Cliente c5 = new Cliente( nombre: "Marcos", apellidos: "Quispe", edad: 60, pais: "Chile", genero: "M", tipo: "Gold");
}
```

The Run window at the bottom shows the output of the program, displaying the details of each client created:

```
Mostrando datos del cliente:
Nombres: Andres
Apellidos: Mendoza
Edad: 19
Pais: Bolivia
Genero: M
Tipo: VIP

Mostrando datos del cliente:
Nombres: Jose
Apellidos: Condori
Edad: 71
Pais: Chile
Genero: M
Tipo: Gold

Mostrando datos del cliente:
Nombres: Saul
Apellidos: Alanoca
Edad: 21
Pais: Argentina
Genero: M
Tipo: Silver

Mostrando datos del cliente:
Nombres: Luis
Apellidos: Choque
```

## EJERCICIO N° 13

- En el mes de diciembre a todos los clientes de Bolivia se les dará una promoción en cuanto a precios en viajes a nivel nacional.
- A todos los clientes que sean de nacionalidad boliviana y además el tipo de cliente GOLD, convertir a estos clientes en VIP
  - Es decir si es de Bolivia y es GOLD deberá ser ahora un cliente VIP
- El método estático dentro de la clase MAIN recibe 3 atributos
  - La cola de clientes
  - El tipo de cliente
  - La nacionalidad del cliente.

## EJERCICIO N° 13

```
no usages
public static void convertirClientesVIP(ColaClientes cola, String tipoCliente, String nacionalidad) {

    ColaClientes aux = new ColaClientes();

    while (!cola.esVacia()) {
        Cliente cliente = cola.eliminar();

        if (cliente.getPais().equals(nacionalidad) && cliente.getTipo().equals(tipoCliente)) {
            cliente.setTipo("VIP");
        }

        aux.Adicionar(cliente);
    }

    cola.vaciar(aux);
}
```

- En el mes de diciembre a todos los clientes de Bolivia se les dará una promoción en cuanto a precios en viajes a nivel nacional.
- A todos los clientes que sean de nacionalidad boliviana y además el tipo de cliente GOLD, convertir a estos clientes en VIP
- Es decir si es de Bolivia y es GOLD deberá ser ahora un cliente VIP
- El método estático dentro de la clase MAIN recibe 3 atributos
- La cola de clientes
- El tipo de cliente
- La nacionalidad del cliente.

The screenshot shows an IDE with three tabs: `Main.java`, `Cliente.java`, and `ColaClientes.java`. The `Main.java` tab is active, displaying the following code:

```
41  
42  
43 //convertirClientesVIP cola_client, "Gold", "Bolivia");  
44  
45 moverMayoresDe60AlInicio(cola_client, edad: 60);  
46  
47 //agregarSaulAlInicio(cola_client, cola_client2);  
48  
49  
50 System.out.println("\nCola después de la conversión:");  
51 cola_client.mostrar();  
52 // System.out.println("\nMostrando la 2da cola");  
53 // cola_client2.mostrar();
```

The Run window at the bottom shows the output of the program:

```
Cola después de la conversión:  
Mostrando la cola de Clientes  
  
Mostrando datos del cliente:  
Nombres: Jose  
Apellidos: Condori  
Edad: 71  
Pais: Chile  
Genero: M  
Tipo: Gold  
  
Mostrando datos del cliente:  
Nombres: Andres  
Apellidos: Mendoza  
Edad: 19  
Pais: Bolivia  
Genero: M  
Tipo: Gold  
  
Mostrando datos del cliente:  
Nombres: Saul  
Apellidos: Alanoca  
Edad: 21  
Pais: Argentina  
Genero: M  
Tipo: Silver
```

## EJERCICIO N° 14

- Mover al inicio todos los clientes mayores a 60 años.
- Es decir si el cliente es mayor a 60 deberá de moverlo al inicio de la cola.
- El método recibe 2 parámetros
  - La Cola de Clientes
  - El valor(int) de la edad.



```

1 usage
public static void moverMayoresDe60AlInicio(ColaClientes cola,int edad) {

    ColaClientes colaAux = new ColaClientes();
    ColaClientes colaAux2 = new ColaClientes();

    while(!cola.esVacia()) {
        Cliente cliente = cola.eliminar();

        if (cliente.getEdad() > edad) {

            colaAux.Adicionar(cliente);

        }else {

            colaAux2.Adicionar(cliente);

        }
    }
    while (!colaAux.esVacia()) {
        cola.Adicionar(colaAux.eliminar());
    }

    cola.vaciar(colaAux);
    cola.vaciar(colaAux2);
}

```

## EJERCICIO N° 14

- Mover al inicio todos los clientes mayores a 60 años.
- Es decir si el cliente es mayor a 60 deberá de moverlo al inicio de la cola. ○ El método recibe 2 parámetros
  - La Cola de Clientes
  - El valor(int) de la edad.

```
ColaClientes cola_client = new ColaClientes();
```

```
Cliente c1 = new Cliente( nombre: "Andres", apellidos: "Mendoza", edad: 19, pais: "Bolivia", genero: "M", tipo: "Gold");  
Cliente c2 = new Cliente( nombre: "Jose", apellidos: "Condori", edad: 71, pais: "Chile", genero: "M", tipo: "Gold");  
Cliente c3 = new Cliente( nombre: "Saul", apellidos: "Alanoca", edad: 21, pais: "Argentina", genero: "M", tipo: "Silver");  
Cliente c4 = new Cliente( nombre: "Luis", apellidos: "Choque", edad: 22, pais: "Bolivia", genero: "M", tipo: "Silver");  
Cliente c5 = new Cliente( nombre: "Marcos", apellidos: "Quispe", edad: 60, pais: "Chile", genero: "M", tipo: "Gold");
```

```
cola_client.Adicionar(c1);  
cola_client.Adicionar(c2);  
cola_client.Adicionar(c3);  
cola_client.Adicionar(c4);  
cola_client.Adicionar(c5);
```

```
ColaClientes cola_client2 = new ColaClientes();
```

```
Cliente c6 = new Cliente( nombre: "Sergio", apellidos: "Mendoza", edad: 19, pais: "Bolivia", genero: "M", tipo: "Gold");  
Cliente c7 = new Cliente( nombre: "Edson", apellidos: "Condori", edad: 71, pais: "Chile", genero: "M", tipo: "Gold");  
Cliente c8 = new Cliente( nombre: "Jhonatan", apellidos: "Alanoca", edad: 21, pais: "Argentina", genero: "M", tipo: "Silver");  
Cliente c9 = new Cliente( nombre: "Mijail", apellidos: "Choque", edad: 22, pais: "Bolivia", genero: "M", tipo: "Silver");  
Cliente c10 = new Cliente( nombre: "Victor", apellidos: "Quispe", edad: 60, pais: "Chile", genero: "M", tipo: "Gold");
```

```
cola_client2.Adicionar(c6);  
cola_client2.Adicionar(c7);  
cola_client2.Adicionar(c8);  
cola_client2.Adicionar(c9);  
cola_client2.Adicionar(c10);
```

## EJERCICIO N° 15

Por razones de promociones de vuelo, es necesario cambiar de vuelo a ciertos clientes.

- Crear 2 colas con 5 clientes.
  - Todos los clientes cuyo nombre sea Saul deberán ser agregados a la cola B al inicio.

```
Project ▾
defensa_H4 C:\JAVA_COURSE\defensa_H4
├── .idea
├── out
├── src
├── ColaClientes
│   ├── Cliente
│   └── ColaClientes
├── Main
├── Main
├── defensa_H4.iml
├── External Libraries
└── Scratches and Consoles

Main ▾
↑ Mostrando datos del cliente:
↓ Nombres: Mijail
Apellidos: Choque
Edad: 22
Pais: Bolivia
Genero: M
Tipo: Silver

Mostrando datos del cliente:
Nombres: Victor
Apellidos: Quispe
Edad: 60
Pais: Chile
Genero: M
Tipo: Gold

Mostrando datos del cliente:
Nombres: Saul
Apellidos: Alanoca
Edad: 21
Pais: Argentina
Genero: M
Tipo: Silver

Process finished with exit code 0
```

```
Main.java
19
20
21
22
23
24
25
26
27
28
29
30
31

ColaClientes cola_client2 = new ColaClientes();

Cliente c6 = new Cliente( nombre: "Sergio", apellidos: "Mendoza", edad: 19, pais: "Bolivia", genero: "M", tipo: "Gold");
Cliente c7 = new Cliente( nombre: "Edson", apellidos: "Condori", edad: 71, pais: "Chile", genero: "M", tipo: "Gold");
Cliente c8 = new Cliente( nombre: "Jhonatan", apellidos: "Alanoca", edad: 21, pais: "Argentina", genero: "M", tipo: "Silver");
Cliente c9 = new Cliente( nombre: "Mijail", apellidos: "Choque", edad: 22, pais: "Bolivia", genero: "M", tipo: "Silver");
Cliente c10 = new Cliente( nombre: "Victor", apellidos: "Quispe", edad: 60, pais: "Chile", genero: "M", tipo: "Gold");

cola_client2.Adicionar(c6);
```

## EJERCICIO N° 15

Por razones de promociones de vuelo, es necesario cambiar de vuelo a ciertos clientes.

- Crear 2 colas con 5 clientes.
  - Todos los clientes cuyo nombre sea Saul deberán ser agregados a la cola B al inicio.



## EJERCICIO N° 15

//EJERCICIO 15, MOVER A LA 2DA COLA A TODO QUIEN SE LLAME SAUL

1 usage

```
public static void agregarSaulAlInicio(ColaClientes cola1, ColaClientes cola2) {  
    ColaClientes colaAux = new ColaClientes();  
  
    while (!cola1.esVacia()) {  
        Cliente cliente = cola1.eliminar();  
  
        if (cliente.getNombre().equals("Saul")) {  
            cola2.Adicionar(cliente);  
        } else {  
            colaAux.Adicionar(cliente);  
        }  
    }  
  
    cola1.vaciar(colaAux);  
}
```

Por razones de promociones de vuelo, es necesario cambiar de vuelo a ciertos clientes.

- Crear 2 colas con 5 clientes.
  - Todos los clientes cuyo nombre sea Saul deberán ser agregados a la cola B al inicio.

[https://github.com/SergioAndresMendozaAlvarado/ESTRUCUTRA-DE-DATOS/blob/main/Hito%204/LAB1/Practica\\_Final\\_H4/CODIGO/main.java](https://github.com/SergioAndresMendozaAlvarado/ESTRUCUTRA-DE-DATOS/blob/main/Hito%204/LAB1/Practica_Final_H4/CODIGO/main.java)

The background is a blue gradient with decorative white circuit-like lines in the corners. The word "GRACIAS" is centered in the upper left area.

GRACIAS