

Defensa Procesual hito 3

MANEJO DE PILAS(Cadenas, Objetos, Números)

MANEJO DE CONCEPTOS

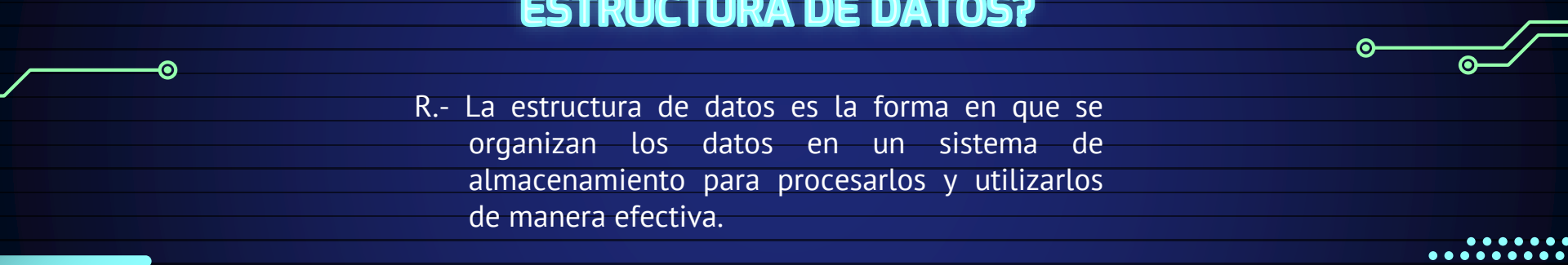
En este apartado se responden las preguntas dadas en la defensa en base a lo avanzado en clases





1. ¿A qué se refiere cuando se habla de ESTRUCTURA DE DATOS?

R.- La estructura de datos es la forma en que se organizan los datos en un sistema de almacenamiento para procesarlos y utilizarlos de manera efectiva.



2.- ¿Cuáles son los TIPOS DE ESTRUCTURA QUE EXISTE?

R.- Los tipos de estructuras de datos son: lineales, jerárquicas e indexadas.

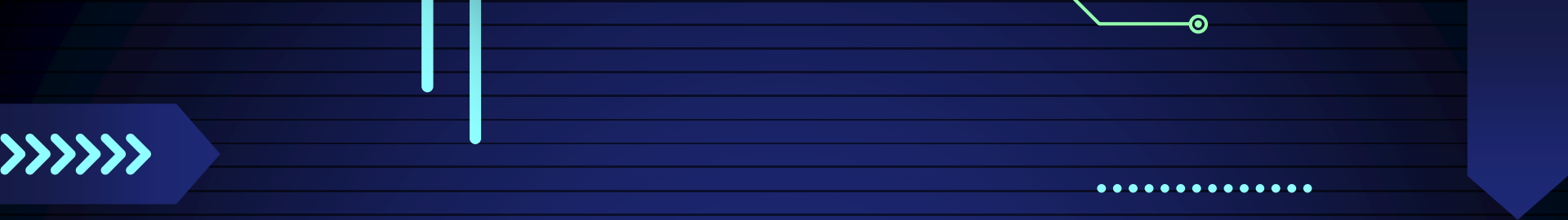
3. ¿Apoyándose en el link adjunto, explique, por qué son útiles las estructuras de datos?

o Tutorial WEB

R.- Las estructuras de datos son útiles porque optimizan el almacenamiento y acceso a los datos, mejoran el rendimiento del software y permiten manipulaciones eficientes de los mismos.

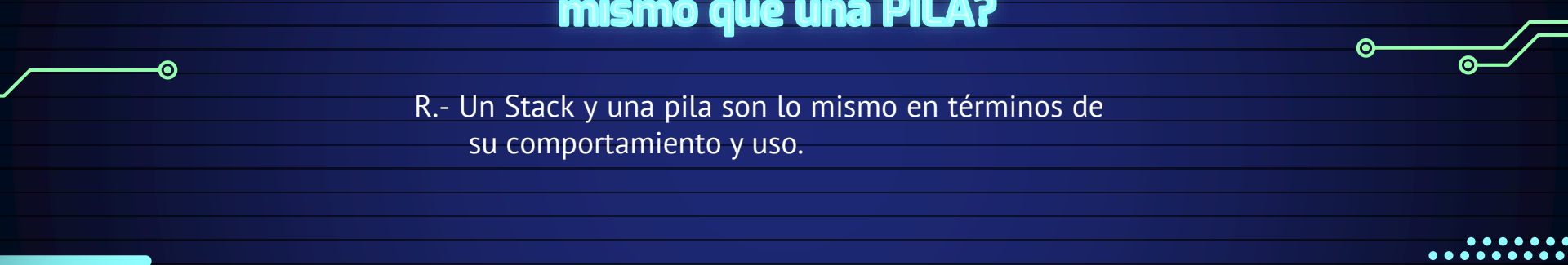
4. ¿Qué es una PILA?

R.- Una pila es una estructura de datos lineal donde los elementos se agregan y eliminan en la parte superior.



5. ¿Qué es STACK en JAVA, una STACK será lo mismo que una PILA?

R.- Un Stack y una pila son lo mismo en términos de su comportamiento y uso.



6. ¿Qué es TOPE en una PILA?

R.- Es el numero de elementos que tiene una pila

7. ¿Qué es MAX en una PILA?

R.- Es el numero máximo de elementos que soporta una pila, o en otras palabras es el tamaño de la pila.

8. ¿A qué se refiere los métodos esVacia() y esLlena() en una PILA?

R.- Los métodos esVacia() y esLlena() se refieren a la capacidad de la pila de contener elementos. El método esVacia() devuelve verdadero si la pila está vacía y falso si contiene elementos. El método esLlena() devuelve verdadero si la pila está llena y falso si aún tiene capacidad para más elementos



9. ¿Qué son los métodos estáticos en JAVA?

R.- Los métodos estáticos en Java son métodos que se pueden llamar sin crear una instancia de la clase. Los métodos estáticos se definen utilizando la palabra clave "static" y se pueden llamar utilizando el nombre de la clase en lugar de una instancia de la misma.

10. ¿A través de un gráfico, muestre los métodos mínimos que debería de tener una PILA?

m	eliminar ()	Cliente
m	esLlena ()	boolean
m	vaciar (PilaCliente)	void
m	nroElem ()	int
m	esVacio ()	boolean
m	mostrar ()	void
m	adicionar (Cliente)	void

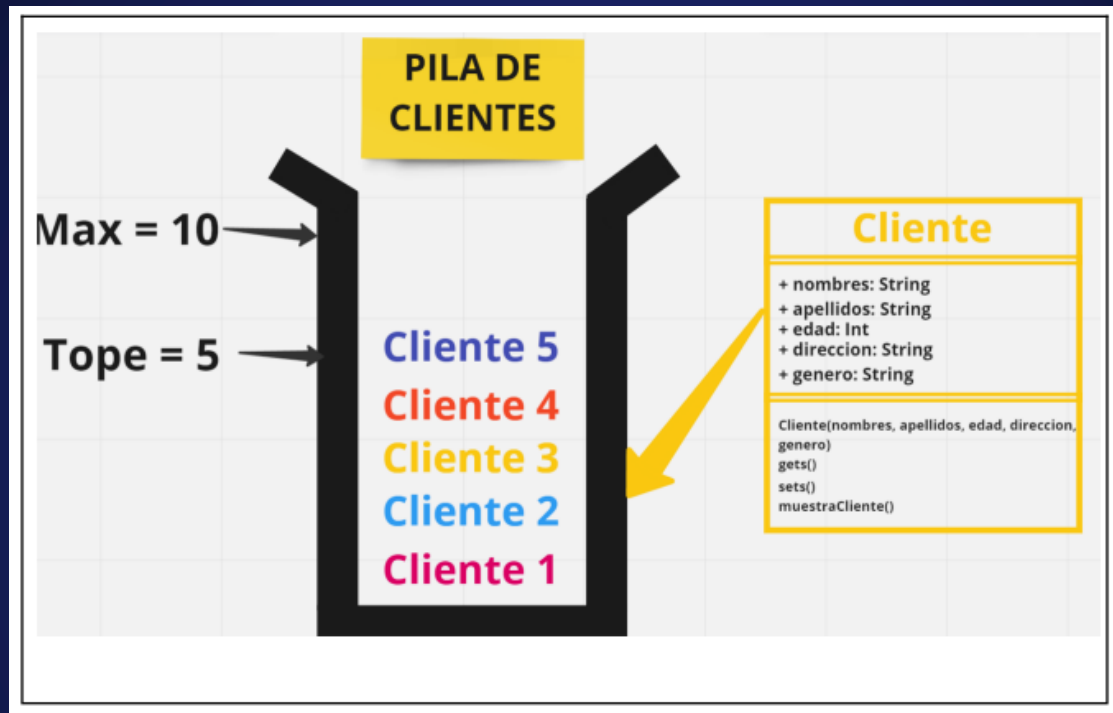
PARTE PRACTICA

En este apartado se resolverán los ejercicios propuestos en la defensa



11. Crear las clases necesarias para la PILA DE CLIENTES.

- Crear la clase Cliente
- Crear la clase PilaCliente
- Crear la clase Main.
- Crear un paquete de nombre PilaDeClientes (todas las clases deberán de estar dentro de este paquete)
- Adjuntar los siguientes. ■ La clase MAIN con la creación de 5 clientes y agregados a la PILA. ■ Una imagen de la salida de la consola en donde se muestran todos los ítems de la pila



Modelo a Realizar





Cliente		
m	Cliente(String, String, int, String, String)	
f	nombres	String
f	apellidos	String
f	edad	int
f	genero	String
f	direccion	String
m	getApellidos()	String
m	getEdad()	int
m	setApellidos(String)	void
m	setNombres(String)	void
m	setEdad(int)	void
m	getGenero()	String
m	setDireccion(String)	void
m	muestraCliente()	void
m	setGenero(String)	void
m	getNombres()	String
m	getDireccion()	String

PilaCliente		
m	PilaCliente ()	
f	max	int
f	cliente	Cliente []
f	tope	int
m	eliminar ()	Cliente
m	esLlena ()	boolean
m	vaciar(PilaCliente)	void
m	nroElem ()	int
m	esVacio ()	boolean
m	mostrar ()	void
m	adicionar (Cliente)	void

Diagrama de Clases





```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
    Cliente c1 = new  
    Cliente("Nombre1", "Apellido1", 18, "Direccion1", "Masculino");  
    Cliente c2 = new  
    Cliente("Nombre2", "Apellido2", 18, "Direccion2", "Femenino");  
    Cliente c3 = new  
    Cliente("Nombre3", "Apellido3", 18, "Direccion3", "Masculino");  
    Cliente c4 = new  
    Cliente("Nombre4", "Apellido4", 20, "Direccion4", "Femenino");  
    Cliente c5 = new  
    Cliente("Nombre5", "Apellido5", 21, "Direccion5", "Femenino");  
    PilaCliente pc1 = new PilaCliente();  
  
    pc1.adicionar(c1);  
    pc1.adicionar(c2);  
    pc1.adicionar(c3);  
    pc1.adicionar(c4);  
    pc1.adicionar(c5);  
  
    pc1.mostrar();  
}
```

MAIN



Mostrando la PILA DE CLIENTES

Cliente

Nombre: Nombre5

Apellido: Apellido5

Edad: 21

Direccion: Direccion5

Genero: Femenino

Cliente

Nombre: Nombre4

Apellido: Apellido4

Edad: 20

Direccion: Direccion4

Genero: Femenino

Cliente

Nombre: Nombre3

Apellido: Apellido3

Edad: 18

Direccion: Direccion3

Genero: Masculino

Cliente

Nombre: Nombre2

Apellido: Apellido2

Edad: 18

Direccion: Direccion2

Genero: Femenino

Cliente

Nombre: Nombre1

Apellido: Apellido1

Edad: 18

Direccion: Direccion1

Genero: Masculino

CONSOLA

12. Determinar cuántos CLIENTES son mayores de 20 años.

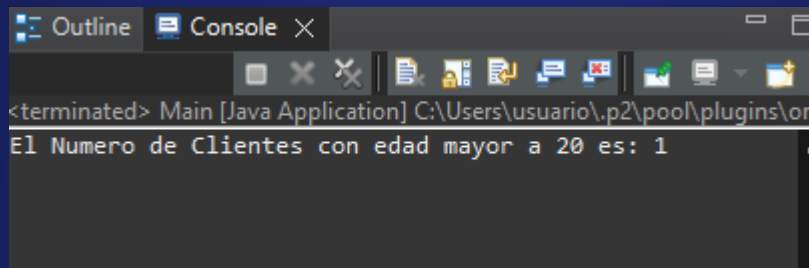
- El método deberá llamarse mayoresCiertaEdad(Pila, edadMayor)
- El método debe ser creado en la clase MAIN como un método estático.
- El método recibe 2 parámetros
 - La Pila de Clientes
 - El valor de la edad.
- Adjuntar los siguientes
 - El código del método que resuelve el problema.
 - Una imagen de la salida de la consola.



```
private static void mayoresCiertaEdad(PilaCliente pila, int edadMayor) {  
    // TODO Auto-generated method stub  
    PilaCliente aux = new PilaCliente();  
    Cliente item = null;  
    int cont = 0;  
    while(!pila.esVacio()) {  
        item = pila.eliminar();  
        if(item.getEdad() > edadMayor) {  
            cont++;  
        }  
        aux.adicionar(item);  
    }  
    pila.vaciar(aux);  
    System.out.println("El Numero de Clientes con edad mayor a " + edadMayor + " es: " + cont);  
}
```

METODO QUE RESUELVE EL PROBLEMA





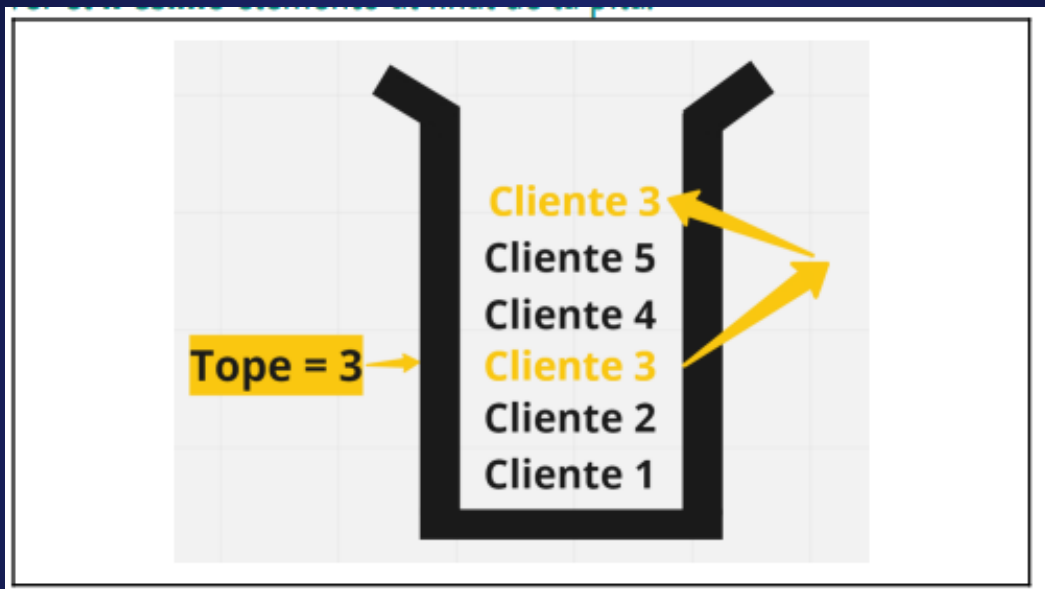
```
<terminated> Main [Java Application] C:\Users\usuario\.p2\pool\plugins\or  
El Numero de Clientes con edad mayor a 20 es: 1
```

CONSOLA



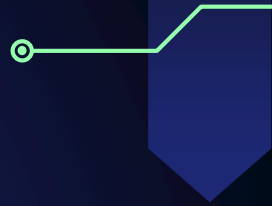
13. Mover el k-ésimo elemento al final de la pila

- El método deberá llamarse kEsimoPosicion(Pila, valorTope)
- El método debe ser creado en la clase MAIN como un método estático.
- El método recibe 2 parámetros
 - La Pila de Clientes
 - El valor(int) de la posición que moverá al final de la pila.
- Adjuntar los siguientes
 - El código del método que resuelve el problema.
 - Una imagen de la salida de la consola



Modelo a Realizar





```
private static void kEsimoPosicion(PilaCliente pila, int valorTope) {  
    // TODO Auto-generated method stub  
    PilaCliente aux = new PilaCliente();  
    Cliente item = null;  
    Cliente itemAux = null;  
    aux.vaciar(pila);  
    while(!aux.esVacio()) {  
        item = aux.eliminar();  
        if(aux.nroElem() == valorTope - 1) {  
            itemAux = item;  
        }  
        else {  
            pila.adicionar(item);  
        }  
    }  
    pila.adicionar(itemAux);  
}
```



METODO QUE RESUELVE EL PROBLEMA





Mostrando la PILA DE CLIENTES

Cliente
Nombre: Nombre3
Apellido: Apellido3
Edad: 18
Direccion: Direccion3
Genero: Masculino

Cliente
Nombre: Nombre5
Apellido: Apellido5
Edad: 21
Direccion: Direccion5
Genero: Femenino

Cliente
Nombre: Nombre4
Apellido: Apellido4
Edad: 20
Direccion: Direccion4
Genero: Femenino

Cliente
Nombre: Nombre2
Apellido: Apellido2
Edad: 18
Direccion: Direccion2
Genero: Femenino

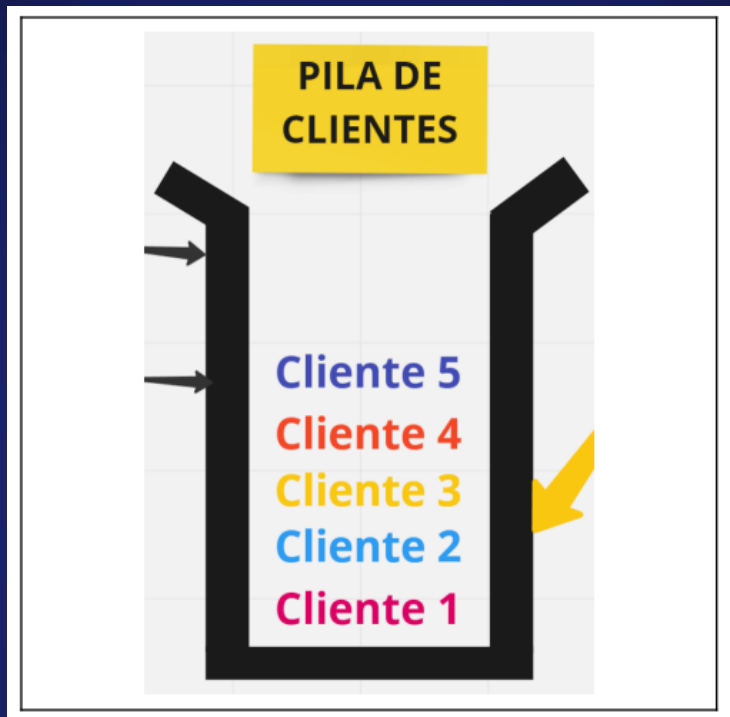
Cliente
Nombre: Nombre1
Apellido: Apellido1
Edad: 18
Direccion: Direccion1
Genero: Masculino

CONSOLA



14. Cambiar la dirección de algunos CLIENTES de la PILA.

- El método deberá llamarse asignaDireccion(Pila, nuevaDireccion)
- El método debe ser creado en la clase MAIN como un método estático.
- El método recibe 2 parámetros
 - La Pila de Clientes
 - El valor(String) de la nueva dirección.
- Cambiar la dirección del cliente siempre y cuando el género sea FEMENINO.
- Adjuntar los siguientes
 - El código del método que resuelve el problema
 - Una imagen de la salida de la consola



Modelo a Realizar





```
private static void asignaDireccion(PilaCliente_pila, String nuevaDireccion)
{
    // TODO Auto-generated method stub
    PilaCliente aux = new PilaCliente();
    Cliente item = null;
    aux.vaciar(pila);
    while(!aux.esVacio()) {
        item = aux.eliminar();
        if(item.getGenero().equals("Femenino")) {
            item.setDireccion(nuevaDireccion);
            pila.adicionar(item);
        }
        else {
            pila.adicionar(item);
        }
    }
}
```



METODO QUE RESUELVE EL PROBLEMA





Mostrando la PILA DE CLIENTES

Cliente
Nombre: Nombre5
Apellido: Apellido5
Edad: 21
Direccion: New Direccion
Genero: Femenino

Cliente
Nombre: Nombre4
Apellido: Apellido4
Edad: 20
Direccion: New Direccion
Genero: Femenino

Cliente
Nombre: Nombre3
Apellido: Apellido3
Edad: 18
Direccion: Direccion3
Genero: Masculino

Cliente
Nombre: Nombre2
Apellido: Apellido2
Edad: 18
Direccion: New Direccion
Genero: Femenino

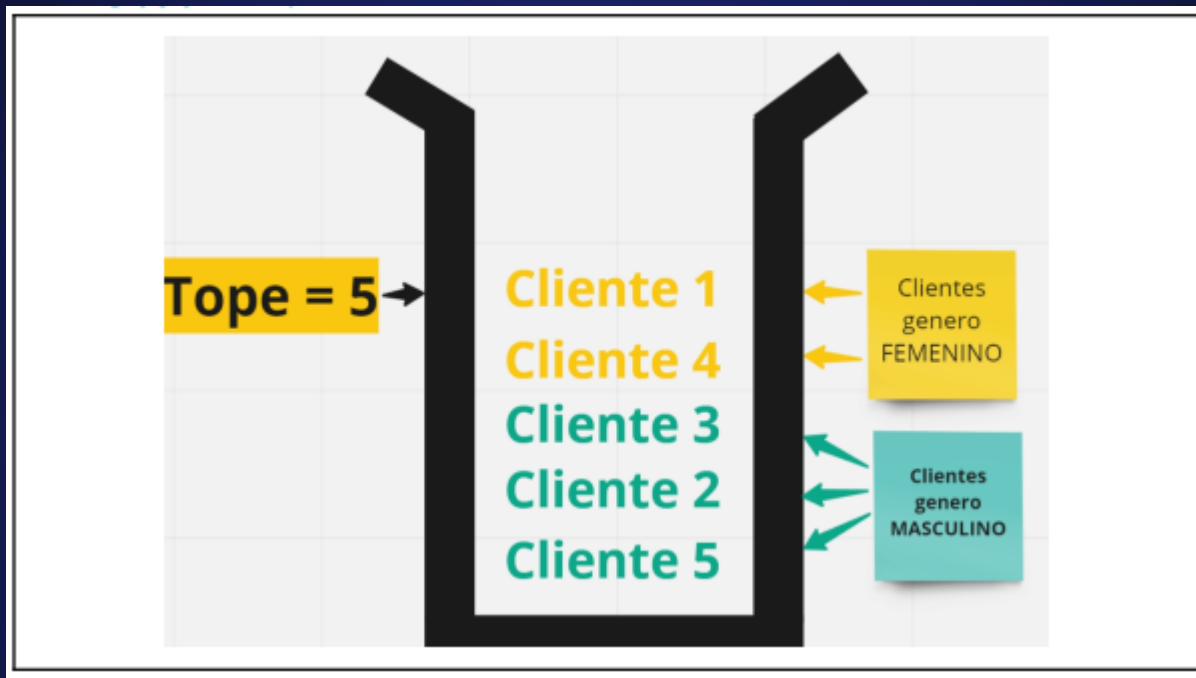
Cliente
Nombre: Nombre1
Apellido: Apellido1
Edad: 18
Direccion: Direccion1
Genero: Masculino

CONSOLA



15. Mover ÍTEMS de la PILA.

- El método deberá llamarse reordenaPila(Pila)
- El método debe ser creado en la clase MAIN como un método estático.
- El método recibe 1 parámetro
 - La Pila de Clientes
- Mover a la base todos los clientes del género masculino y los del género femenino moverlos al final.
- Adjuntar los siguientes
 - El código del método que resuelve el problema.
 - Una imagen de la salida de la consola



Modelo a Realizar



```
private static void reordenaPila(PilaCliente_pila) {  
    // TODO Auto-generated method stub  
    PilaCliente aux = new PilaCliente();  
    PilaCliente pilamasc = new PilaCliente();  
    PilaCliente pilafem = new PilaCliente();  
    aux.vaciar(pila);  
    pilamasc = pilaGenMasc(aux);  
    pilafem = pilaGenFem(aux);  
    pila.vaciar(pilamasc);  
    pila.vaciar(pilafem);  
}
```



METODO QUE RESUELVE EL PROBLEMA





```
private static PilaCliente pilaGenFem(PilaCliente pila) {  
    // TODO Auto-generated method stub  
    PilaCliente aux = new PilaCliente();  
    aux.vaciar(pila);  
    Cliente item = null;  
    PilaCliente pilaFem = new PilaCliente();  
    while(!aux.esVacio()) {  
        item = aux.eliminar();  
        if(item.getGenero().equals("Femenino")) {  
            pilaFem.adicionar(item);  
        }  
        pila.adicionar(item);  
    }  
    return pilaFem;  
}
```



METODO QUE RESUELVE EL PROBLEMA





```
private static PilaCliente pilaGenMasc(PilaCliente pila)
{
    // TODO Auto-generated method stub
    PilaCliente aux = new PilaCliente();
    aux.vaciar(pila);
    Cliente item = null;
    PilaCliente pilaMasc = new PilaCliente();
    while(!aux.esVacio()) {
        item = aux.eliminar();
        if(item.getGenero().equals("Masculino")) {
            pilaMasc.adicionar(item);
        }
        pila.adicionar(item);
    }
    return pilaMasc;
}
```



METODO QUE RESUELVE EL PROBLEMA





Mostrando la PILA DE CLIENTES

Cliente
Nombre: Nombre5
Apellido: Apellido5
Edad: 21
Direccion: Direccion5
Genero: Femenino

Cliente
Nombre: Nombre4
Apellido: Apellido4
Edad: 20
Direccion: Direccion4
Genero: Femenino

Cliente
Nombre: Nombre2
Apellido: Apellido2
Edad: 18
Direccion: Direccion2
Genero: Femenino

Cliente
Nombre: Nombre3
Apellido: Apellido3
Edad: 18
Direccion: Direccion3
Genero: Masculino

Cliente
Nombre: Nombre1
Apellido: Apellido1
Edad: 18
Direccion: Direccion1
Genero: Masculino

CONSOLA





Thanks!

Do you have any questions?

mijailChoque35@gmail.com
+591 76591425



Credits: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, infographics & images by **Freepik**