



Internacionalízate

CARRERA
INGENIERIA DE SISTEMAS

DEFENSA HITO 3

ESTUDIANTE:
MICHAEL RIGOBERTO ALCON VILLCA

EL ALTO - BOLIVIA

MANEJO DE CONCEPTOS

1. ¿A QUÉ SE REFIERE CUANDO SE HABLA DE ESTRUCTURA DE DATOS?

LAS ESTRUCTURAS DE DATOS EN PROGRAMACIÓN SON UN MODO DE REPRESENTAR INFORMACIÓN EN UNA COMPUTADORA, AUNQUE Además, CUENTAN CON UN COMPORTAMIENTO INTERNO. ¿QUÉ SIGNIFICA? QUE SE RIGE POR DETERMINADAS REGLAS/RESTRICCIONES QUE HAN SIDO DADAS POR LA FORMA EN QUE ESTÁ CONSTRUIDA INTERNAMENTE.

2. ¿CUÁLES SON LOS TIPOS DE ESTRUCTURA QUE EXISTE?

Primero, debemos diferenciar entre estructura de dato estática y estructura de dato dinámica.

- Las estructuras de datos estáticas son aquellas en las que el tamaño ocupado en memoria se define antes de que el programa se ejecute y no puede modificarse dicho tamaño durante la ejecución del programa,

- estructura de datos dinámica es aquella en la que el tamaño ocupado en memoria puede modificarse durante la ejecución del programa.

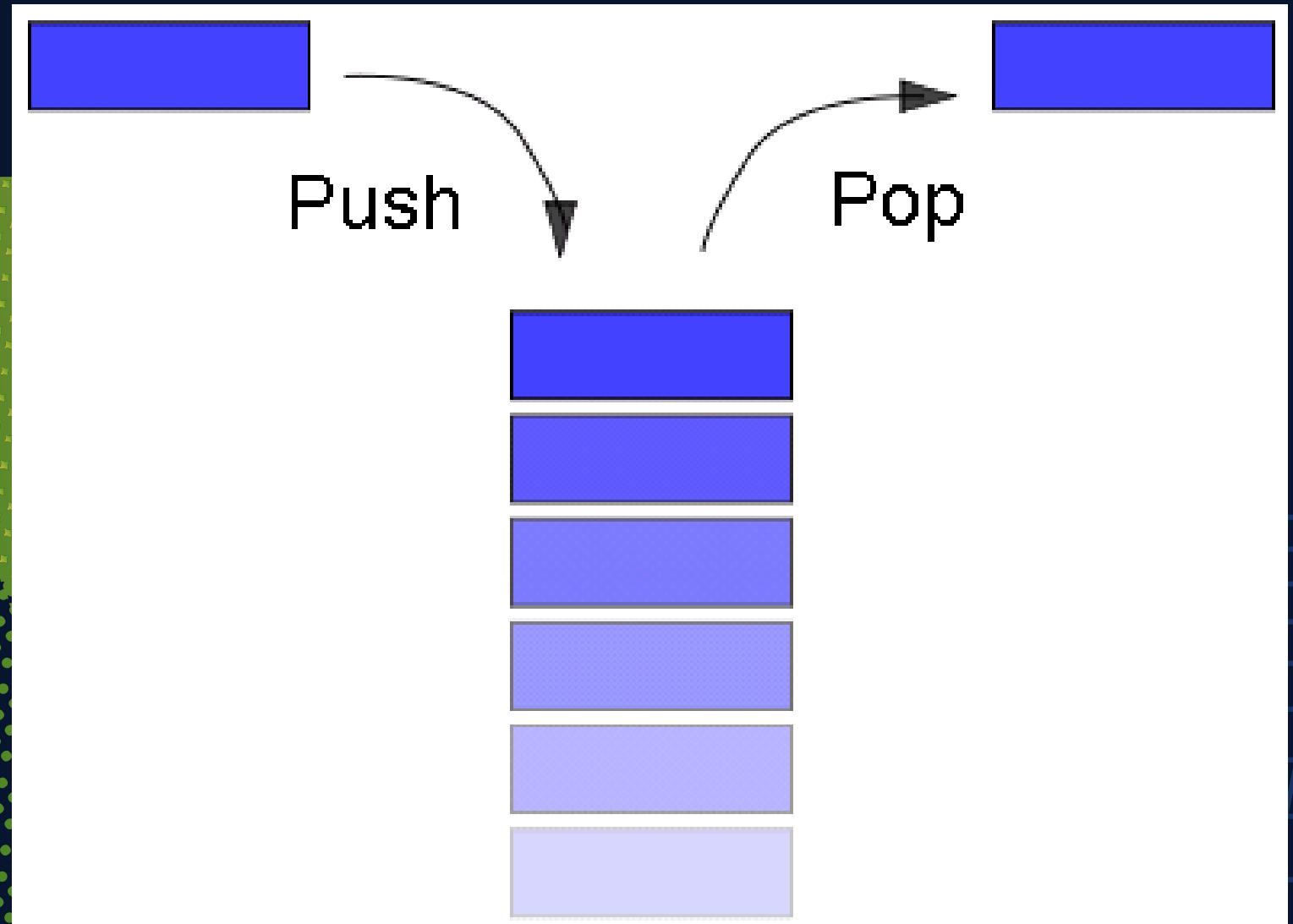
Cada tipo de estructura dependerá del tipo de aplicación que se requiera. Una típica dentro de las estructuras de datos estáticas son los arrays:



3. ¿APOYÁNDOSE EN EL LINK ADJUNTO, EXPLIQUE, POR QUÉ SON ÚTILES LAS ESTRUCTURAS DE DATOS?



4. ¿QUÉ ES UNA PILA?



UNA PILA ES UNA LISTA ORDINAL O ESTRUCTURA DE DATOS EN LA QUE EL MODO DE ACCESO A SUS ELEMENTOS ES DE TIPO LIFO QUE PERMITE ALMACENAR Y RECUPERAR DATOS. ESTA ESTRUCTURA SE APLICA EN MULTITUD DE OCASIONES EN EL ÁREA DE INFORMÁTICA DEBIDO A SU SIMPLICIDAD Y ORDENACIÓN IMPLÍCITA DE LA PROPIA ESTRUCTURA.

PARA EL MANEJO DE LOS DATOS SE CUENTA CON DOS OPERACIONES BÁSICAS: APIALAR (PUSH), QUE COLOCA UN OBJETO EN LA PILA, Y SU OPERACIÓN INVERSA, RETIRAR (O DESAPILAR, POP), QUE RETIRA EL ÚLTIMO ELEMENTO APIALADO.

EN CADA MOMENTO SÓLO SE TIENE ACCESO A LA PARTE SUPERIOR DE LA PILA, ES DECIR, AL ÚLTIMO OBJETO APIALADO . LA OPERACIÓN RETIRAR PERMITE LA OBTENCIÓN DE ESTE ELEMENTO, QUE ES RETIRADO DE LA PILA PERMITIENDO EL ACCESO AL SIGUIENTE, QUE PASA A SER EL NUEVO TOS.

POR ANALOGÍA CON OBJETOS COTIDIANOS, UNA OPERACIÓN APIALAR EQUIVALDRÍA A COLOCAR UN PLATO SOBRE UNA PILA DE PLATOS, Y UNA OPERACIÓN RETIRAR A RETIRARLO.

¿QUÉ ES STACK EN JAVA?

La clase Stack es una clase de las llamadas de tipo LIFO (Last In - First Out, o último en entrar - primero en salir). Esta clase hereda de la clase que ya hemos estudiado anteriormente en el curso Vector y con 5 operaciones permite tratar un vector a modo de pila o stack.

Las operaciones básicas son push (que introduce un elemento en la pila), pop (que saca un elemento de la pila), peek (consulta el primer elemento de la cima de la pila), empty (que comprueba si la pila está vacía) y search (que busca un determinado elemento dentro de la pila y devuelve su posición dentro de ella).

Esta clase es muy sencilla y al crear un objeto de tipo Stack con el constructor básico evidentemente no contendrá ningún elemento.

Un conjunto mucho más completo y consistente para operaciones de stack LIFO es proporcionado en la interface Deque y sus implementaciones, pero nosotros de momento vamos a limitarnos al estudio de la clase Stack.

```
/* Ejemplo Interface List, clase Stack aprenderaprogramar.com */
import java.util.Stack;
public class Programa {
    public static void main(String arg[]) {
        String cadenano = "(Cadena no equilibrada en paréntesis((000)))";
        String cadenasi = "(Cadena equilibrada en parentesis())";
        System.out.println("Verificación equilibrado en paréntesis para cadenano:");
        System.out.println(verificaParentesis(cadenano));
        System.out.println("Verificación equilibrado en paréntesis para cadenasi:");
        System.out.println(verificaParentesis(cadenasi));
    }

    public static boolean verificaParentesis(String cadena) {
        Stack<String> pila = new Stack<String>();      int i = 0;
        while (i<cadena.length()) { // Recorremos la expresión carácter a carácter
            if(cadena.charAt(i)=='(') {pila.push("(");} // Si el paréntesis es de apertura apilamos siempre

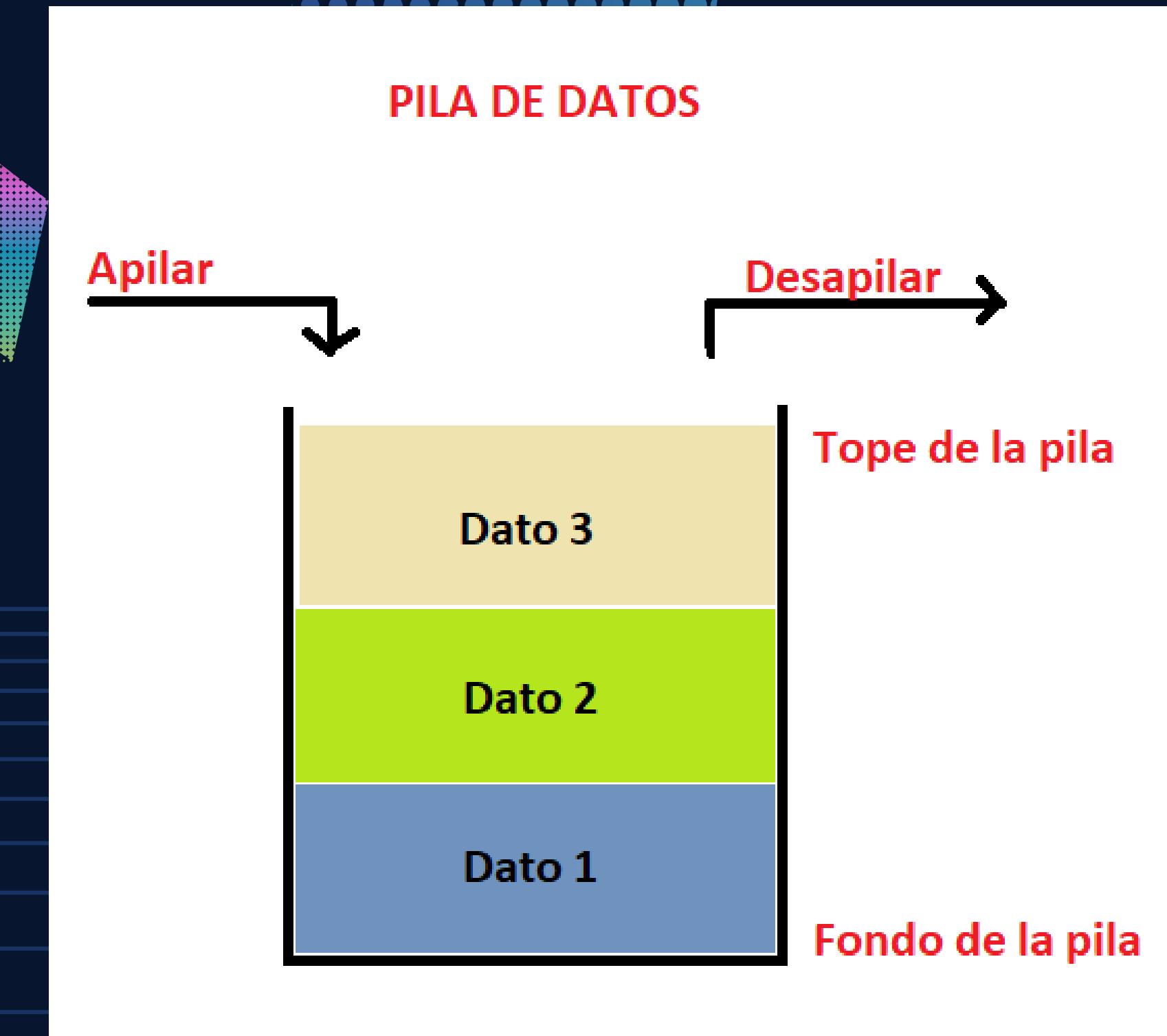
            else if (cadena.charAt(i)==')') { // Si el paréntesis es de cierre actuamos según el caso
                if (!pila.empty()){ pila.pop();} // Si la pila no está vacía desapilamos
                else { pila.push(")"); break;} // La pila no puede empezar con un cierre, apilamos y salimos
            }
            i++;
        }
        if(pila.empty()){ return true;} else { return false;}
    }
}
```

6.

¿QUÉ ES TOPE EN UNA PILA?

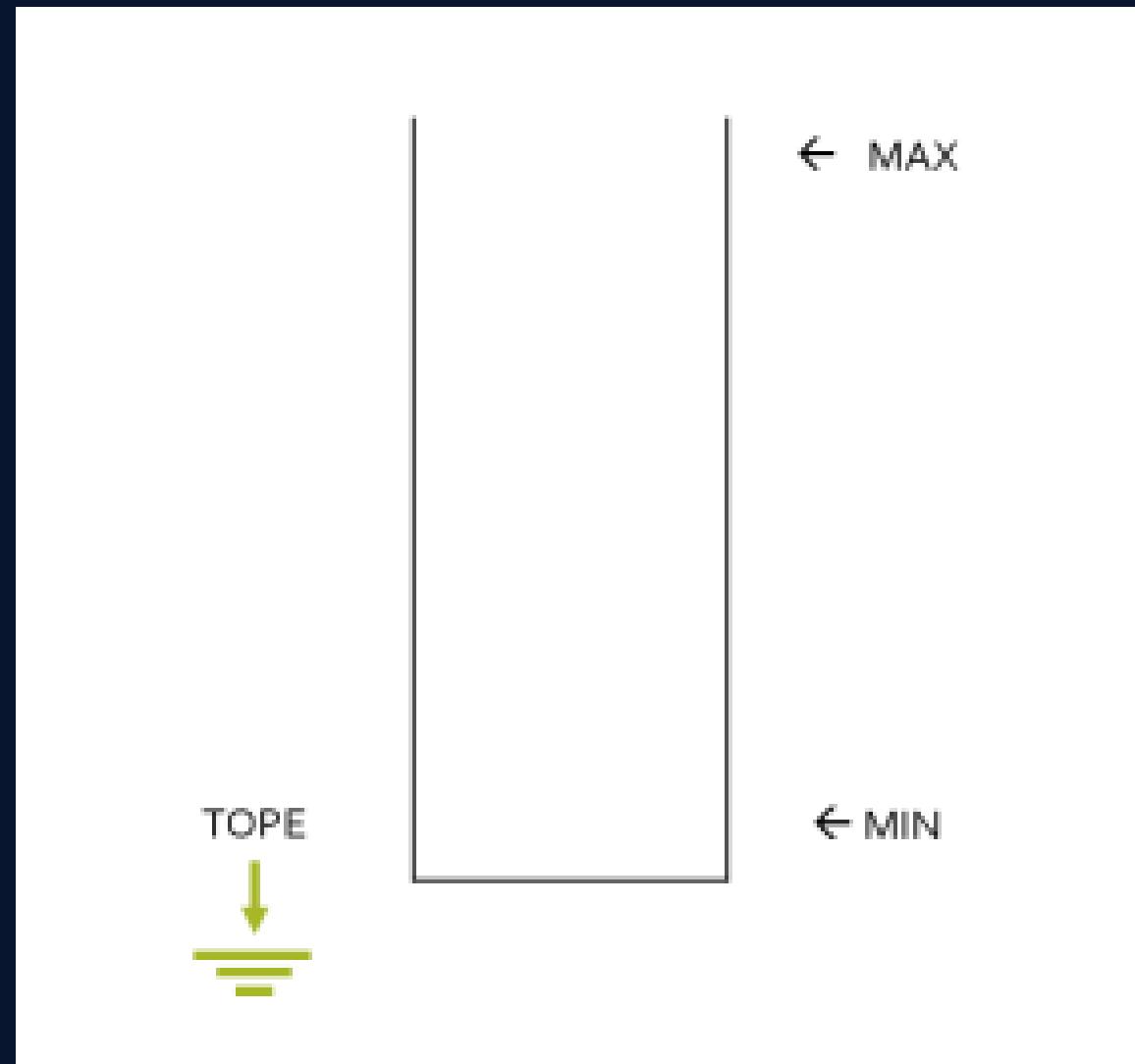
Una pila es un conjunto ordenado de elementos en el cual se pueden agregar y eliminar elementos de un extremo, el cual es llamado el tope de la pila. La pila es un objeto dinámico en constante cambio. La definición de pila especifica que un solo extremo de la pila se designa como tope.

Pueden colocarse nuevos elementos en el tope de la pila o se pueden quitar elementos de él. La característica más importante de la pila es que el último elemento insertado en ella es el primero en suprimirse. Por esta razón, una pila se denomina como una estructura LIFO (Last In First Out) en la cual, el último elemento insertado, será el primero en ser eliminado.



¿QUÉ ES MAX EN UNA PILA?

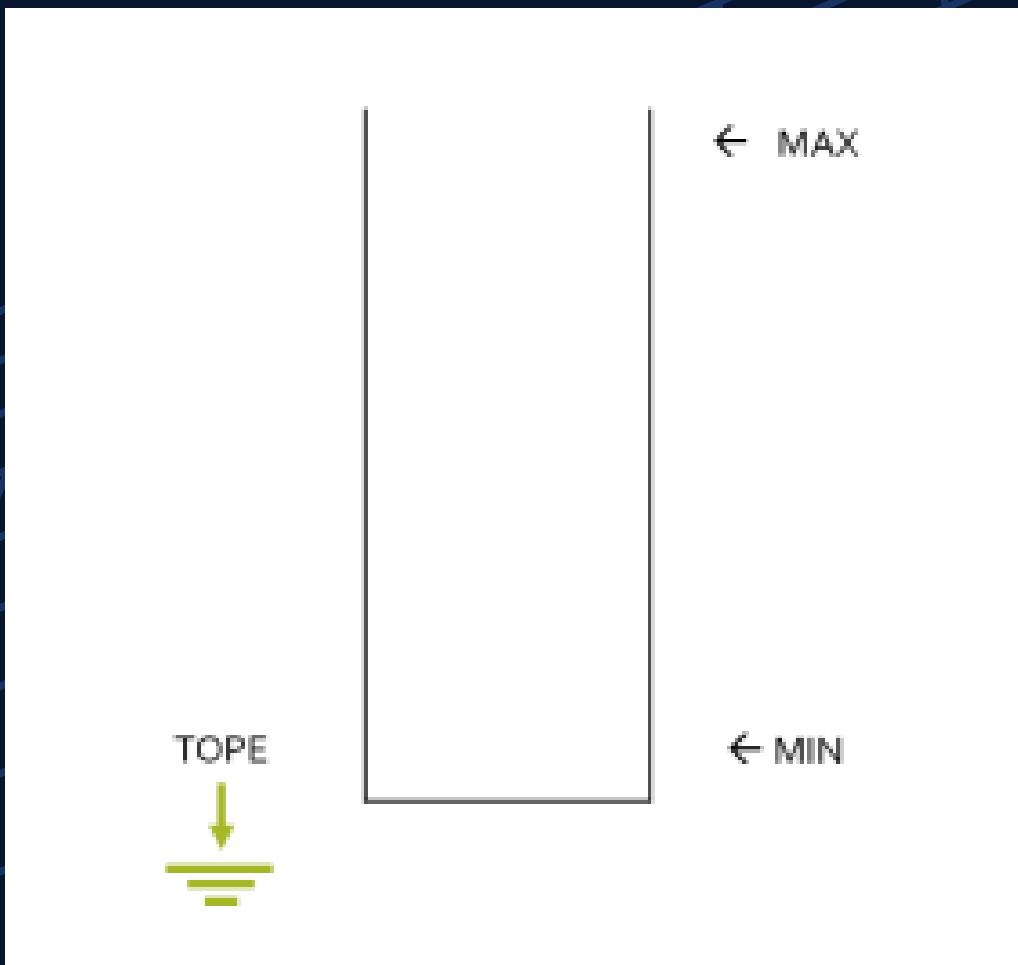
Ademas de un tope la pila tiene un maximo, que representa el tamaño de la pila.



8. ¿A QUÉ SE REFIERE LOS MÉTODOS ESVACIA() Y ESLLENA() EN UNA PILA?

PILA VACÍA

UNA PILA VACÍA NO CONTIENE ELEMENTO ALGUNO DENTRO DE LA ESTRUCTURA Y EL TOPE DE LA MISMA APUNTA A NULO.

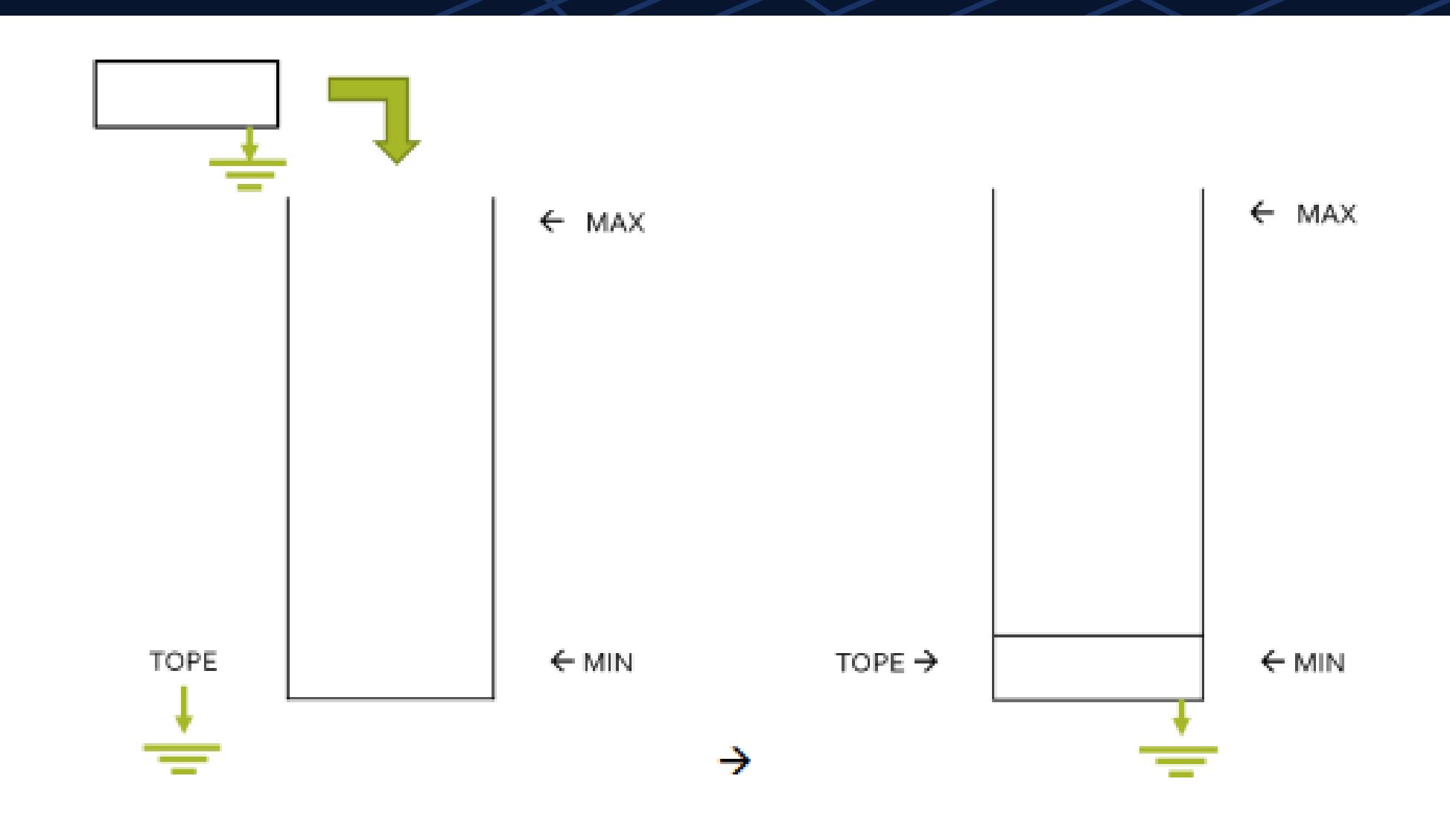


EN UNA PILA VACÍA NO ES POSIBLE REALIZAR POP, DEBIDO A QUE LA ESTRUCTURA NO CONTIENE INFORMACIÓN.



8. ¿A QUÉ SE REFIERE LOS MÉTODOS ESVACIA() Y ESENTRA() EN UNA PILA?

CUANDO LA PILA ESTÁ VACÍA SÍ SE PUEDE REALIZAR PUSH, EN TAL CASO, EL NODO QUE ENTRA A LA ESTRUCTURA SERÍA EL ÚNICO ELEMENTO DE LA PILA Y EL TOPE APUNTARÍA A ÉL

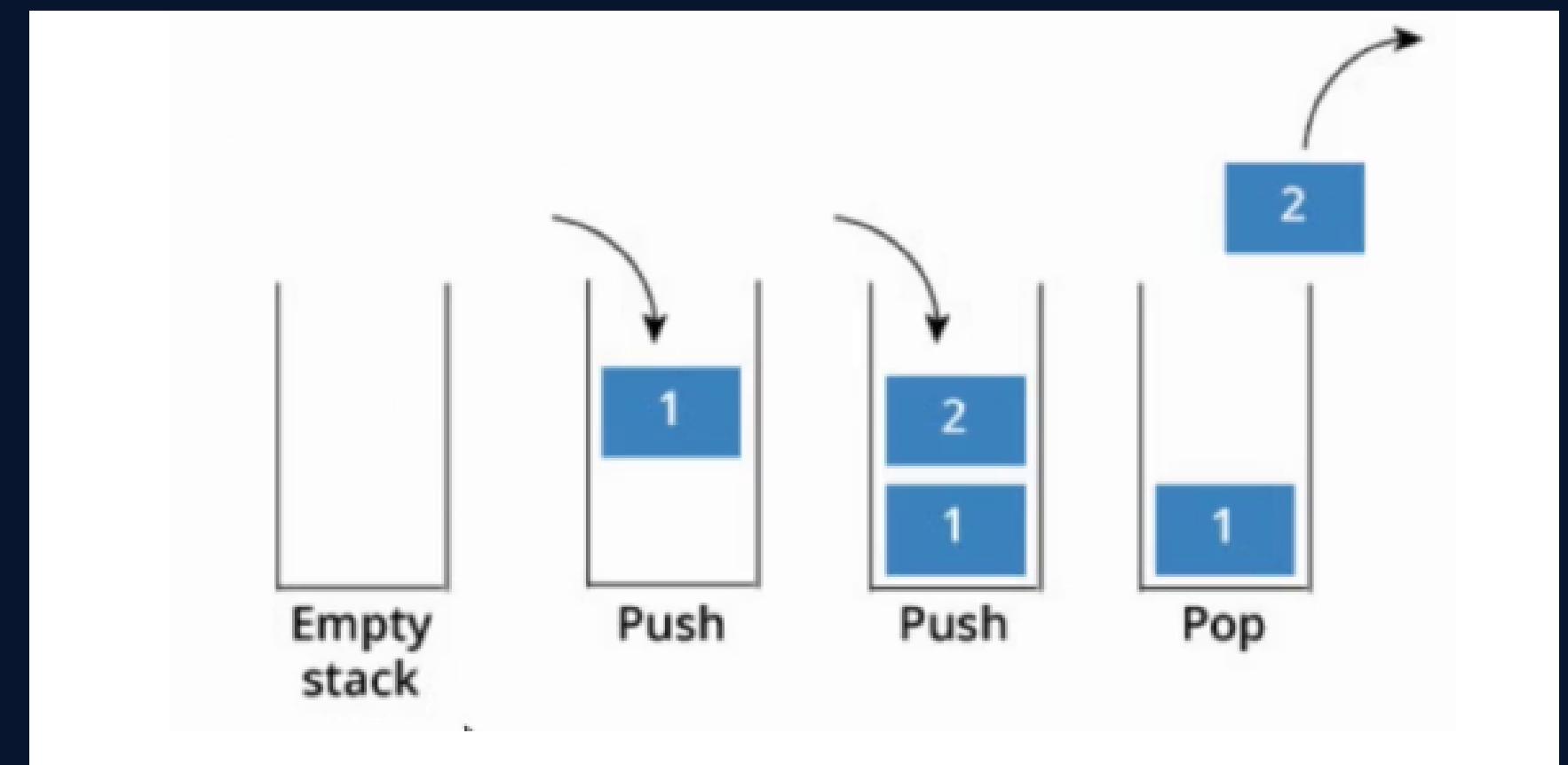
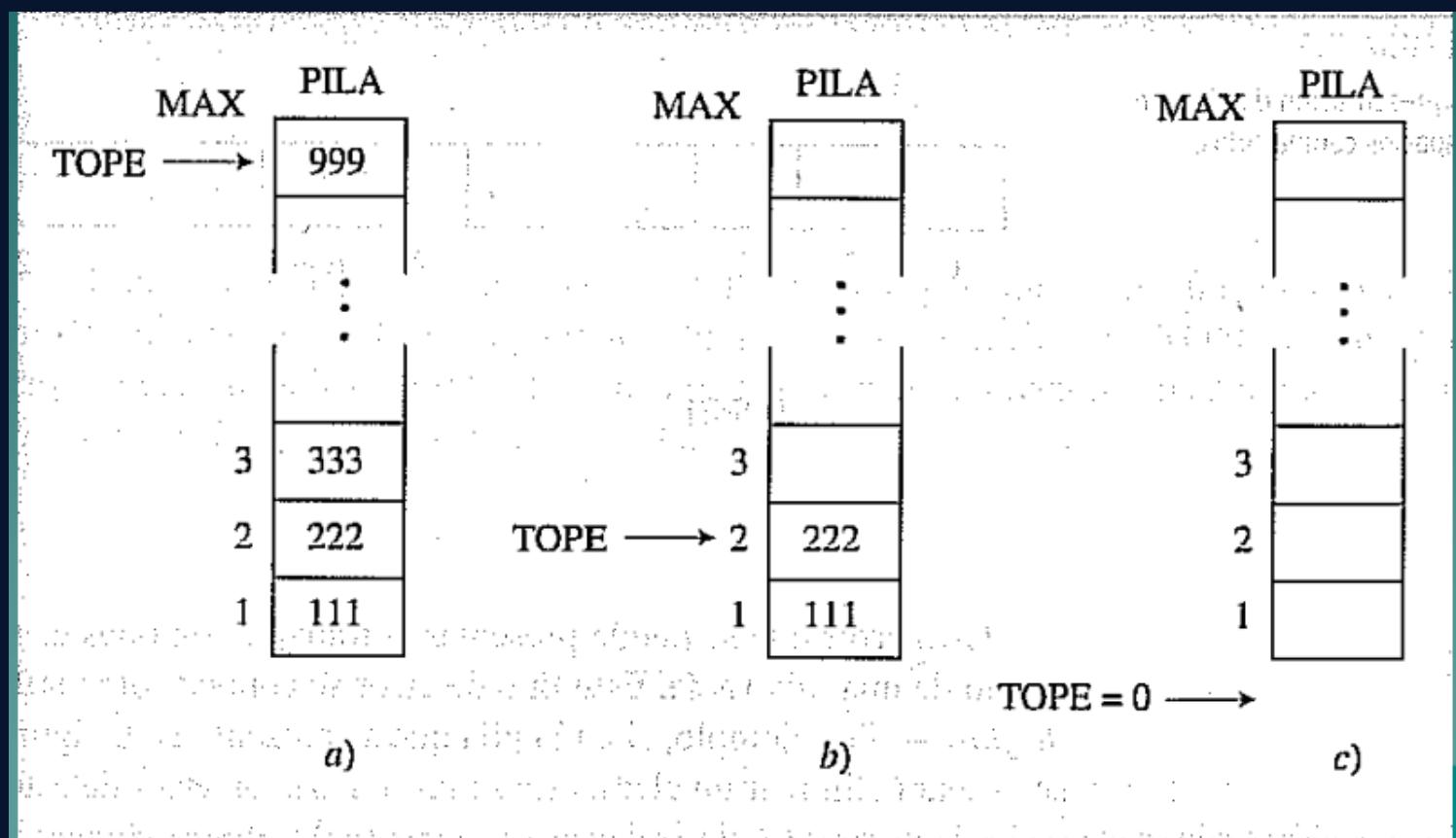


9. ¿QUÉ SON LOS MÉTODOS ESTÁTICOS EN JAVA?

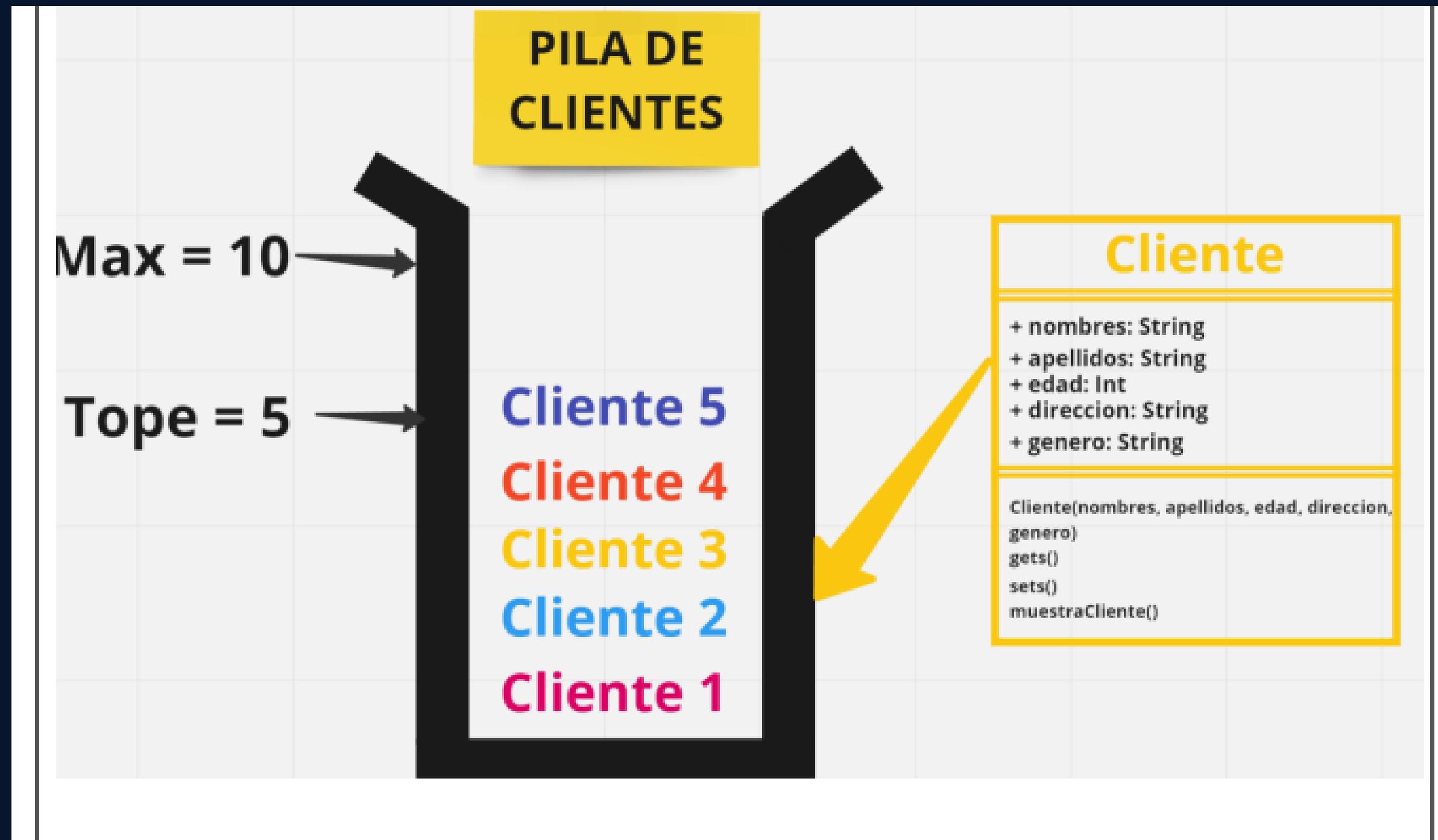
El método estático en Java es un método que pertenece a la clase y no al objeto. Un método estático solo puede acceder a datos estáticos.

- **Es un método que pertenece a la clase y no al objeto (instancia).**
- **Un método estático solo puede acceder a datos estáticos. No puede acceder a datos no estáticos (variables de instancia).**
- **Un método estático puede llamar solo a otros métodos estáticos y no puede invocar un método no estático a partir de él.**
- **Un método estático se puede acceder directamente por el nombre de la clase y no necesita ningún objeto.**
- **Un método estático no puede hacer referencia a "este" o "super" palabras clave de todos modos.**

10. A TRAVÉS DE UN GRÁFICO, MUESTRE LOS MÉTODOS MÍNIMOS QUE DEBERÍA DE TENER UNA PILA?



II. 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

- ✓ Tarea_hito3 C:\Users\MICHAEL\IdeaProjects\Tarea_hito3
 - > .idea
 - > out
 - ✓ src
 - calculadora
 - ✓ defensahito2
 - Departamento
 - Main
 - Main1
 - Pais
 - Provincia
 - ✓ PilasDeClientes
 - Cliente
 - Main
 - PilaCliente
 - > polimorfismo
 - ✓ .gitignore
 - ✓ Tarea_hito3.iml
- > External Libraries
- ✓ Scratches and Consoles

```
1 package PilasDeClientes;
2
3 public class Cliente {
4     3 usages
5     private String Nombres;
6     3 usages
7     private String Apellidos;
8     3 usages
9     private int Edad;
10    3 usages
11    private String direccion;
12    3 usages
13    private String Genero;
14
15    5 usages
16    public Cliente(String nombres, String apellidos, int edad, String direccion, String genero) {
17        Nombres = nombres;
18        Apellidos = apellidos;
19        Edad = edad;
20        this.direccion = direccion;
21        Genero = genero;
22    }
23
24    1 usage
25    public String getNombres() {
26        return Nombres;
27    }
28
29    public void setNombres(String nombres) {
30        Nombres = nombres;
31    }
32
33    1 usage
34    public String getApellidos() {
35        return Apellidos;
36    }
37}
```

clase Cliente

```
 35     public int getEdad() {
 36         return Edad;
 37     }
 38
 39     public void setEdad(int edad) {
 40         Edad = edad;
 41     }
 42
 43     public String getDireccion() {
 44         return direccion;
 45     }
 46
 47     public void setDireccion(String direccion) {
 48         this.direccion = direccion;
 49     }
 50
 51     public String getGenero() {
 52         return Genero;
 53     }
 54
 55     public void setGenero(String genero) {
 56         Genero = genero;
 57     }
 58
 59     public void mostrarCliente() {
 60         System.out.println("mostrar datos del cliente =");
 61         System.out.println("Nombre: " + this.getNombres());
 62         System.out.println("Apellidos: " + this.getApellidos());
 63         System.out.println("Edad: " + this.getEdad());
 64         System.out.println("Direccion: " + this.getDireccion());
 65         System.out.println("Genero: " + this.getGenero());
 66         System.out.println("\n");
 67     }
 68 }
```

src
calculadora
defensahito2
Departamento
Main
Main1
Pais
Provincia
PilasDeClientes
Cliente
Main
PilaCliente
polimorfismo
.gitignore
Tarea_hito3.iml
External Libraries
Scratches and Consoles

Project + - X Tarea_hito3 C:\Users\MICHAEL\IdeaProjects\1 Clientejava X PilaCliente.java X Main.java X

A 8 X 28

```
package PilasDeClientes;

public class PilaCliente {

    private int max;
    private int tope;
    private Cliente[] Cliente;

    public PilaCliente(int max) {
        this.tope = 0;
        this.max = max;
        this.Cliente = new Cliente[this.max + 1];
    }

    public boolean esVacio () {
        if (tope == 0) {
            return true;
        } else {
            return false;
        }
    }

    public boolean esLleno () {
        if (tope == max) {
            return true;
        } else {
            return false;
        }
    }

    public int nroElem () {
        return this.tope;
    }

    public void push(Cliente cliente) {
        Cliente[tope] = cliente;
        tope++;
    }

    public Cliente pop() {
        Cliente cliente = Cliente[tope];
        tope--;
        return cliente;
    }

    public Cliente peek() {
        return Cliente[tope];
    }
}
```

clase- PilaCliente

clase- PilaCliente

```
Project: Tarea_hito3 C:\Users\MICHAEL\IdeaProjects\Tarea_hito3
  .idea
  out
  src
    calculadora
    defensahito2
    PillasDeClientes
      Cliente
      Main
      PilaCliente
    polimorfismo
    .gitignore
    Tarea_hito3.iml
External Libraries
Scratches and Consoles
```

```
35     public void adicionar (Cliente nuevoCliente) {
36         if (this.esLleno() == false) {
37             this.tope = this.tope + 1;
38             this.Cliente[this.tope] = nuevoCliente;
39         } else {
40             System.out.println("La pila de cliente está llena");
41         }
42     }
43     2 usages
44     public Cliente eliminar () {
45         Cliente elementoEliminado = null;
46
47         if (!this.esVacio()) {
48             elementoEliminado = (this.Cliente[this.tope]);
49             this.tope = this.tope - 1;
50         } else {
51             System.out.println("La pila de cliente está vacia");
52         }
53         return elementoEliminado;
54     }
55     public void llenar () {
56
57     }
58     1 usage
59     public void mostrar () {
60         Cliente elem = null;
61         if (esVacio())
62             System.out.println("Pila Vacia");
63         else {
64             System.out.println("\nDatos de la Pila de cliente");
65             System.out.println("\n");
66             PilaCliente aux = new PilaCliente(this.max);
67             while (!esVacio()) {
68                 elem = this.eliminar();
69                 aux.adicionar (elem);
70                 elem.mostrarCliente();
71             }
72             vaciar(aux);
73         }
74     }
75 }
```

clase-main

```
1 package PilasDeClientes;
2
3 public class Main {
4     public static void main(String [] args){
5         Cliente cli1 = new Cliente( nombres: "carlos", apellidos: "alarcon", edad: 25, direccion: "villavictoria/av.2#215", genero: "masculino");
6         Cliente cli2 = new Cliente( nombres: "juan", apellidos: "valcasar", edad: 18, direccion: "santiago/av.1#358", genero: "masculino");
7         Cliente cli3 = new Cliente( nombres: "nataly", apellidos: "calle", edad: 50, direccion: "mariscalsantacruz/av.3#78", genero: "femenino");
8         Cliente cli4 = new Cliente( nombres: "richard", apellidos: "villa", edad: 32, direccion: "villamercedesa/av.5#2415", genero: "masculino");
9         Cliente cli5 = new Cliente( nombres: "marcos", apellidos: "escobar", edad: 38, direccion: "pagador/calle.16#1883", genero: "masculino");
10
11         PilaCliente pila = new PilaCliente( max: 10);
12         pila.adicionar(cli1);
13         pila.adicionar(cli2);
14         pila.adicionar(cli3);
15         pila.adicionar(cli4);
16         pila.adicionar(cli5);
17         pila.mostrar();
18     }
19 }
20 }
```

Run: Main X



Datos de la Pila de cliente

```
mostrar datos del cliente =
Nombre: marcos
Apellidos: escobar
Edad: 38
Direccion: pagador/calle.16#1883
Genero: masculino
```

```
mostrar datos del cliente =
Nombre: richard
Apellidos: villa
Edad: 32
Direccion: villamercedesa/av.5#2415
Genero: masculino
```

```
mostrar datos del cliente =
Nombre: nataly
Apellidos: calle
Edad: 50
Direccion: mariscalsantacruz/av.3#78
Genero: femenino
```

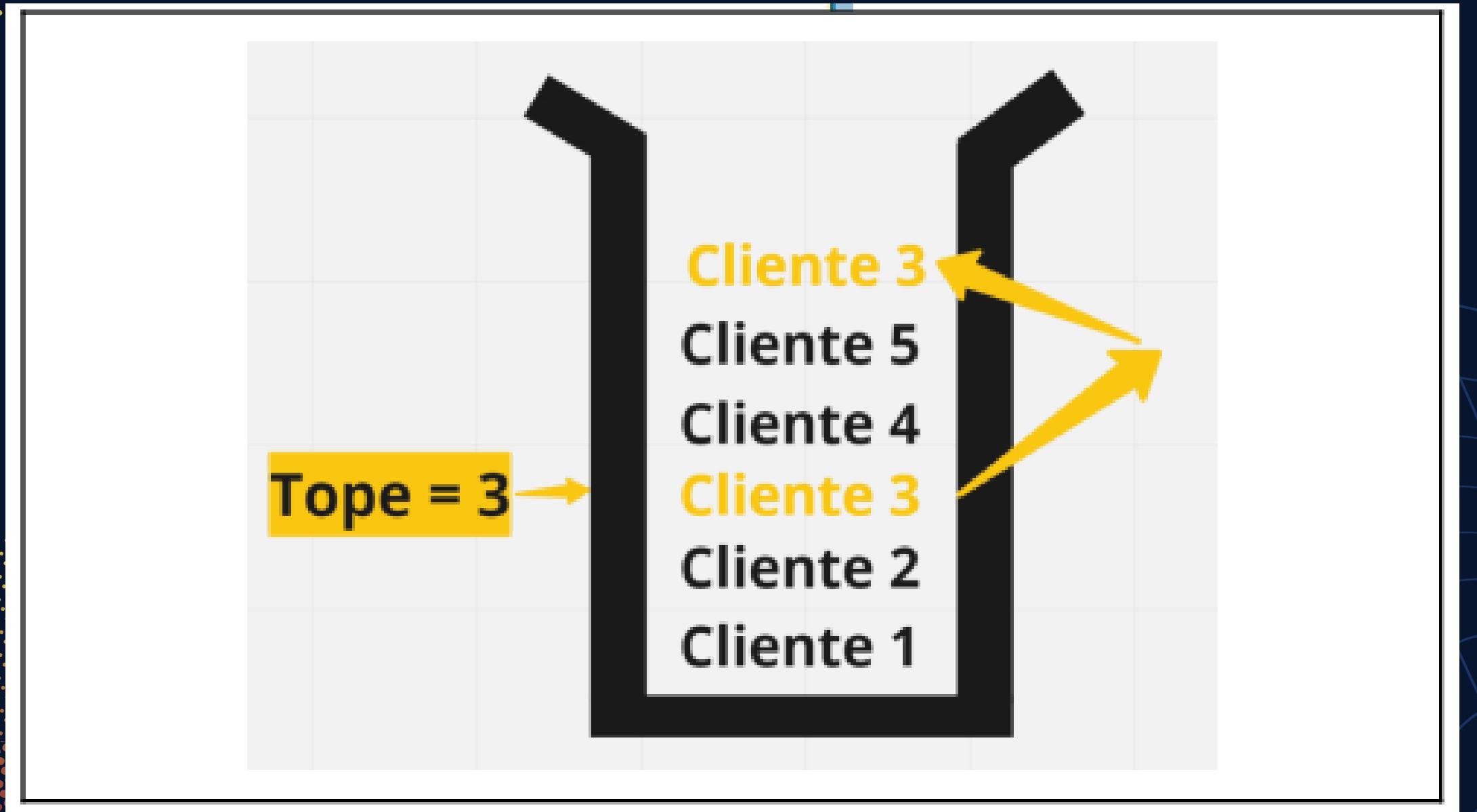
```
mostrar datos del cliente =
Nombre: juan
Apellidos: valcasar
Edad: 18
Direccion: santiago/av.1#358
Genero: masculino
```

```
mostrar datos del cliente =
Nombre: carlos
```

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.

13. MOVER EL H-ÉSIMO ELEMENTO AL FINAL DE LA PILA.



```
//Mover el k-ésimo elemento al final de la pila.  
1 usage  
public static void kEsimoPosicion(PilaCliente pila, int valorTope) {  
    PilaCliente aux = new PilaCliente( max: 10);  
    Cliente valor = null;  
    if (valorTope < pila.nroElem()) {  
        while (pila.esVacio() == false) {  
            if (pila.nroElem() != valorTope) {  
                aux.adicionar(pila.eliminar());  
            } else {  
                valor = pila.eliminar();  
            }  
        }  
    }  
}
```

```
//Mover el k-ésimo elemento al final de la pila.  
1 usage  
public static void kEsimoPosicion(PilaCliente pila, int valorTope) {  
    PilaCliente aux = new PilaCliente( max: 10);  
    Cliente valor = null;  
    if (valorTope < pila.nroElem()) {  
        while (pila.esVacio() == false) {  
            if (pila.nroElem() != valorTope) {  
                aux.adicionar(pila.eliminar());  
            } else {  
                valor = pila.eliminar();  
            }  
        }  
        pila.vaciar(aux);  
        pila.adicionar(valor);  
        pila.mostrar();  
    } else {  
        System.out.println("No se puede mover porque el valor de Tope es mayor");  
    }  
}  
}  
61 }  
62 }
```

Run: Main ×

▶ ↑ Datos de la Pila de cliente
🔧 ↓
➡ mostrar datos del cliente =
Nombre: juan
Apellidos: valcasar
Edad: 18
Direccion: santiago/av.1#358
Genero: masculino

➡ mostrar datos del cliente =
Nombre: marcos
Apellidos: escobar
Edad: 38
Direccion: pagador/calle.16#1883
Genero: masculino