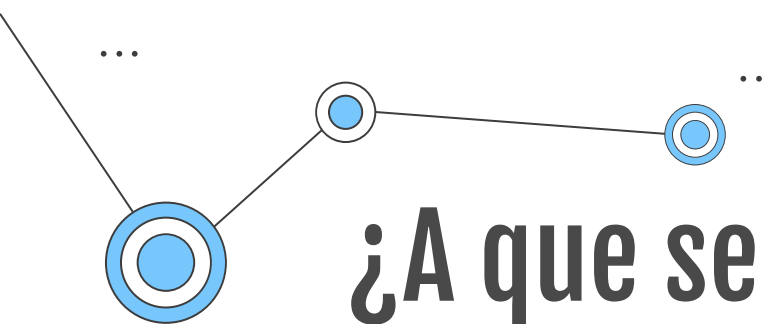
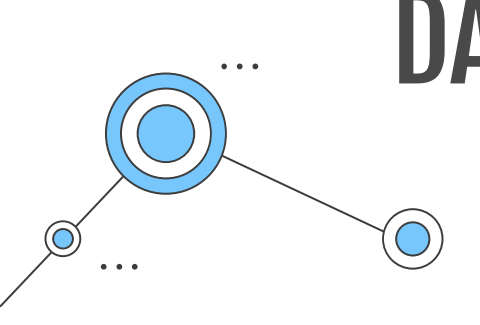




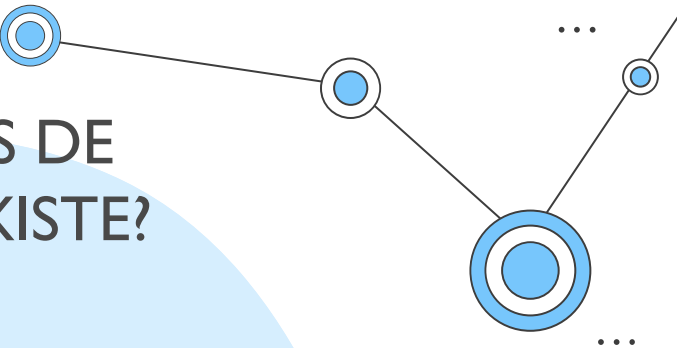
ESTRUCTURA DE DATOS II



¿A que se refiere cuando se habla de ESTRUCTURA DE DATOS?



Dentro de la informática, las construcciones de datos son esas que nos permiten, como desarrolladores, acomodar la información de forma eficiente, y en conclusión diseñar la solución adecuada para un definido problema.



¿Cuáles son los TIPOS DE ESTRUCTURA QUE EXISTE?

Existen dos tipos de datos

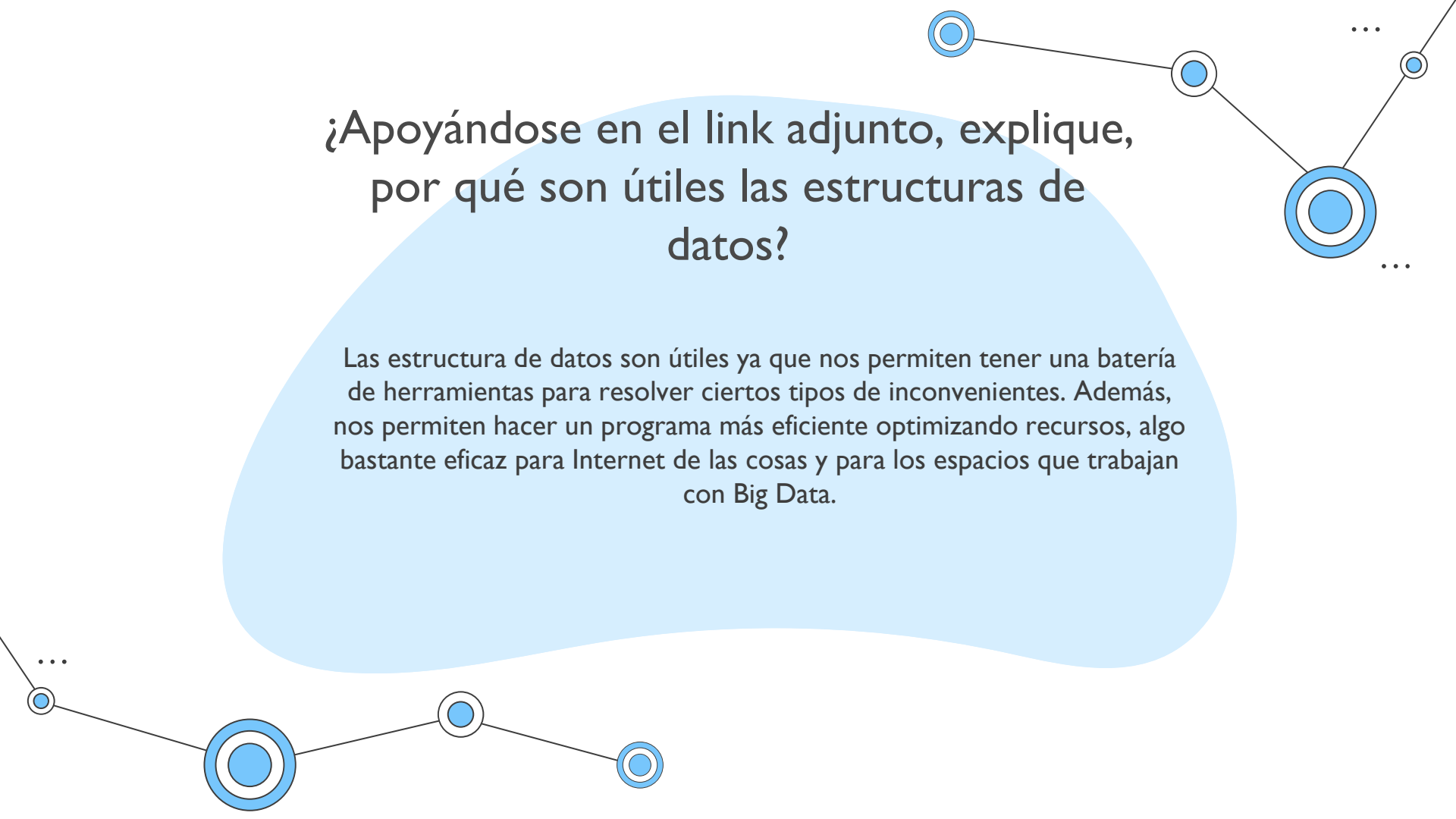
Estructuras de datos lineales.- Son aquellas en las que los elementos ocupan lugares sucesivos en la estructura y cada uno de ellos tiene un único sucesor y un único predecesor, es decir, sus elementos están ubicados uno al lado del otro relacionados en forma lineal.

Hay tres tipos de estructuras de datos lineales:

- Listas enlazadas
- Pilas
- Colas

Estructura de datos no lineales .- También llamadas multienlazadas, son aquellas en las que cada elemento puede estar enlazado a cualquier otro componente. Es decir, cada elemento puede tener varios sucesores o varios predecesores. Existen dos tipos:

- Árboles
 - Grafos
- 

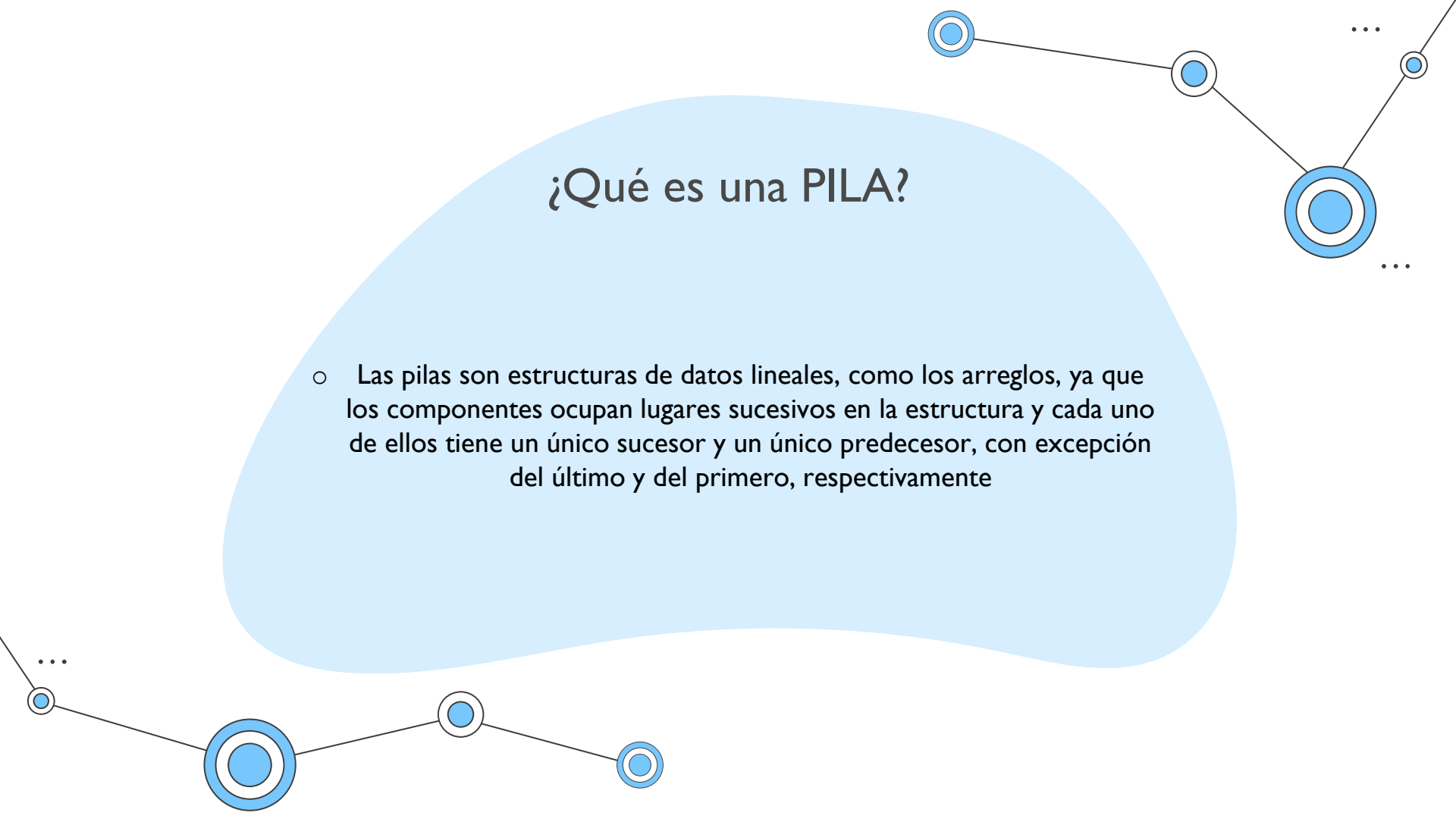



¿Apoyándose en el link adjunto, explique,
por qué son útiles las estructuras de
datos?

Las estructura de datos son útiles ya que nos permiten tener una batería de herramientas para resolver ciertos tipos de inconvenientes. Además, nos permiten hacer un programa más eficiente optimizando recursos, algo bastante eficaz para Internet de las cosas y para los espacios que trabajan con Big Data.

¿Qué es una PILA?

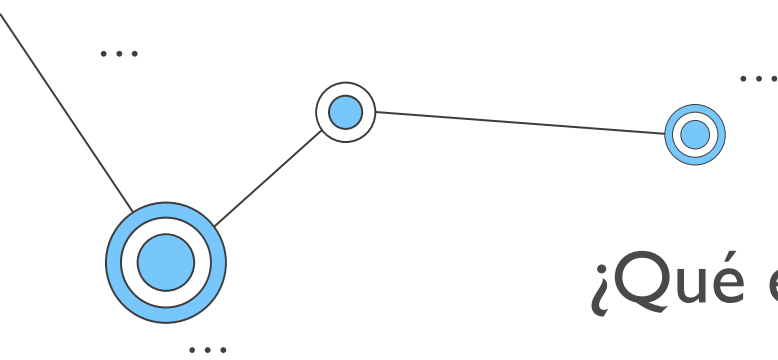
- Las pilas son estructuras de datos lineales, como los arreglos, ya que los componentes ocupan lugares sucesivos en la estructura y cada uno de ellos tiene un único sucesor y un único predecesor, con excepción del último y del primero, respectivamente





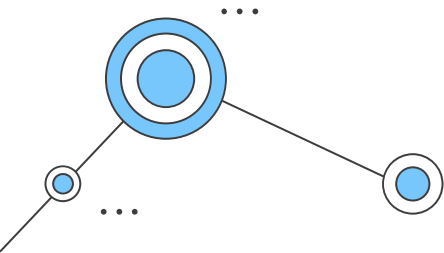
¿Qué es STACK en JAVA, una STACK será lo mismo que una PILA?

Un Stack es un objeto similar a una pila (**Pila en español**). Posibilita guardar objetos y después recuperarlos en el orden inverso en el que se insertaron, o sea, continuamente se recupera el último factor insertado. Para insertar un objeto a la pila se invoca el procedimiento push.



¿Qué es TOPE en una PILA?

La definición de pila especifica que un solo extremo de la pila se designa como tope. Tienen la posibilidad de posicionarse nuevos recursos en el tope de la pila o tienen la posibilidad de quitar recursos de él. La característica más relevante de la pila es que el último factor insertado en ella fue el primero en suprimirse.





¿Qué es MAX en una PILA?

Es una variable auxiliar a la cual se le nombra TOPE, esta variable se usa para indicar el ultimo factor que se inserto en la pila.



¿Qué es MAX en una PILA?

Es una variable auxiliar a la cual se le denomina TOPE, esta variable se utiliza para indicar el ultimo elemento que se inserto en la pila.

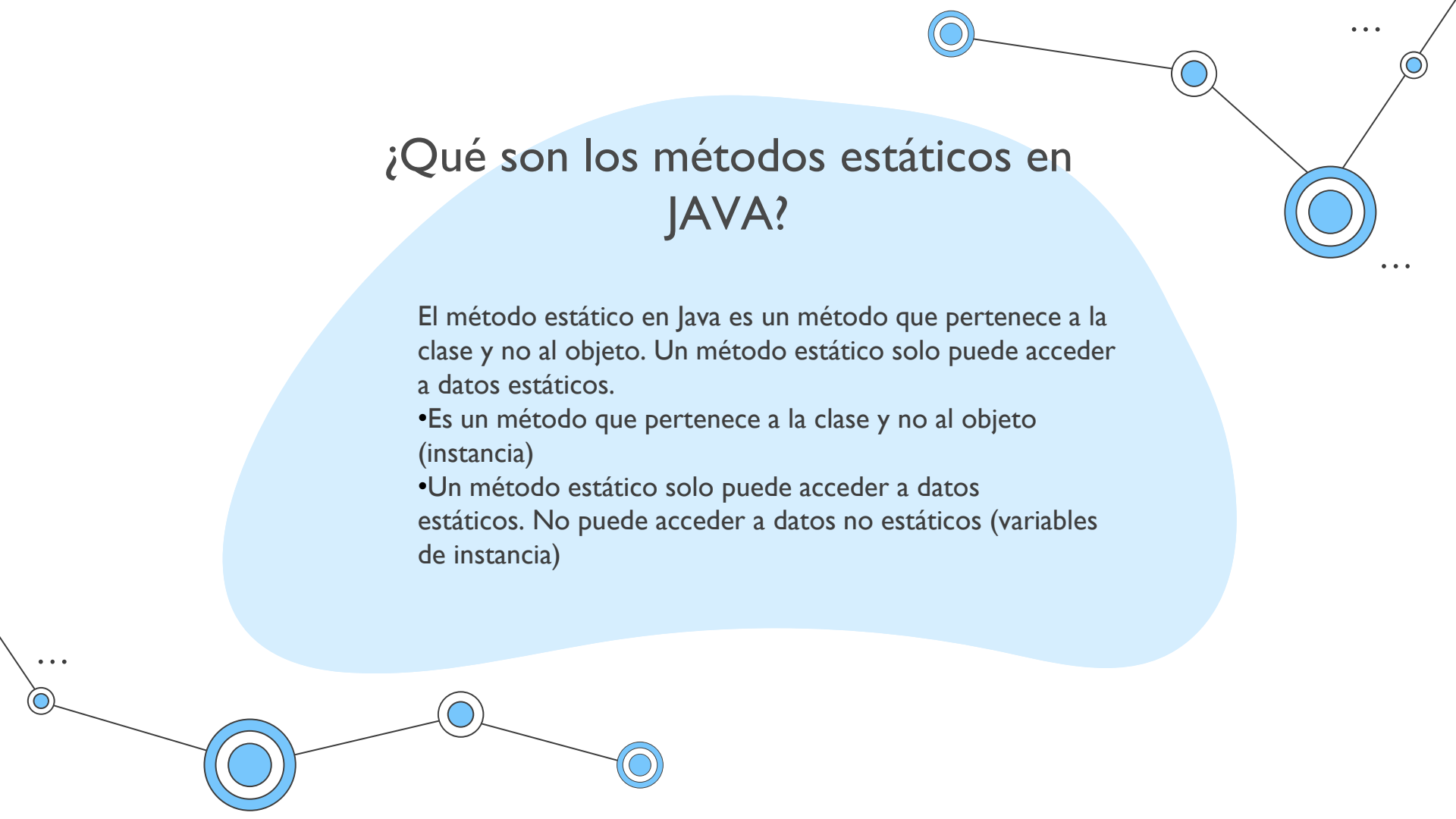
¿A que se refiere los métodos esVacia() y esLlena() en una PILA?

Regresa verdadero si la pila está
vacía, falso en caso opuesto

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

Regresa verdadero si la pila está
vacía, falso en caso opuesto

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



¿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)

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

Libros		
m	Libros(int)	
f	tope	int
f	libros	Libros[]
f	max	int
m	esllena()	boolean
m	nroElementos()	int
m	llenar()	void
m	vaciar(Libros)	void
m	adicionar(Libros)	void
m	eliminar()	Libros
m	mostrar()	void
m	esVacio()	boolean

Crear las clases necesarias para la PILA DE CLIENTES.

Crear la clase Cliente

Cliente	
Cliente (String, String, String, int, String)	
nombre	String
edad	int
apellido	String
genero	String
direccion	String
getApellido()	String
setDireccion (String)	void
getNombre()	String
setGenero (String)	void
getDireccion ()	String
setApellido (String)	void
setEdad (int)	void
getEdad()	int
getGenero()	String
setNombre (String)	void
mostrarDatos()	void

Crear la clase PilaCliente

PilaCliente	
PilaCliente (int)	
tope	int
pilita	Cliente []
max	int
vaciar (PilaCliente)	void
estaVacia()	boolean
estaLleno()	boolean
numeroElementos()	int
eliminar ()	Cliente
insertar (Cliente)	void
mostrar()	void

Crear la clase Main

Main	
Main ()	
reordenaPila (PilaCliente)	void
asignarDireccion (PilaCliente, String)	void
mayoresCiertaEdad (PilaCliente, int)	void
moverKesimo (PilaCliente, Cliente)	void
main (String[])	void

La clase MAIN con la creación de 5 clientes y agregados a la PILA.

```
Cliente cliente1 = new Cliente( nombre: "Alejandro", apellido: "Quevedo", direccion: "Zona 1", edad: 20, genero: "Masculino");  
Cliente cliente2 = new Cliente( nombre: "Maria", apellido: "Diaz", direccion: "Zona 2", edad: 28, genero: "Femenino");  
Cliente cliente3 = new Cliente( nombre: "Mariano", apellido: "Gonzales", direccion: "Zona 3", edad: 36, genero: "Masculino");  
Cliente cliente4 = new Cliente( nombre: "Rivers", apellido: "Vargas", direccion: "Zona 4", edad: 42, genero: "Femenino");  
Cliente cliente5 = new Cliente( nombre: "Roberto", apellido: "Ramos", direccion: "Zona 5", edad: 51, genero: "Masculino");
```

```
pila.mostrar();
```

Elementos de la pila

Datos del cliente:

Nombre: Roberto
Apellido: Ramos
Direccion: Zona 5
Edad: 51
Genero: Masculino

Datos del cliente:

Nombre: Rivers
Apellido: Vargas
Direccion: Zona 4
Edad: 42
Genero: Femenino

Datos del cliente:

Nombre: Mariano
Apellido: Gonzales
Direccion: Zona 3
Edad: 36
Genero: Masculino

Datos del cliente:

Nombre: Maria
Apellido: Diaz
Direccion: Zona 2
Edad: 28
Genero: Femenino

Datos del cliente:

Nombre: Alejandro
Apellido: Quevedo
Direccion: Zona 1
Edad: 20

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.

```
/*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. */  
  
1 usage  
public static void mayoresCiertaEdad(PilaCliente pila, int edadMayor){  
    int contador = 0;  
    Cliente elem = null;  
    if(pila.estaVacia()){  
        System.out.println("La pila esta vacia");  
    } else {  
        while(!pila.estaVacia()){  
            elem = pila.eliminar();  
            if(elem.getEdad() > edadMayor){  
                contador++;  
            }  
        }  
    }  
    System.out.println("Clientes Mayor A 20 " + edadMayor + " son: " + contador);  
}
```

```
mayoresCiertaEdad(pila, edadMayor: 20);
```

```
Clientes Mayor A 20 son: 4  
  
Process finished with exit code 0
```

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.

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

El método deberá llamarse `moverElemento(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 de `valorTope` */*

```
public static void kEsimoposicion(PilaCliente pila, Cliente valorTope){  
    PilaCliente aux= new PilaCliente( max: 10);  
    Cliente elem = null;  
    while(!pila.estaVacia()){  
        elem = pila.eliminar();  
        if(elem != valorTope){  
            aux.insertar(elem);  
        }  
    }  
    pila.vaciar(aux);  
    pila.insertar(valorTope);  
    pila.mostrar();  
}
```

`kEsimoposicion(pila,cliente2);`

Elementos de la pila

Datos del cliente:

Nombre: Mariano
Apellido: Gonzales
Direccion: Zona 3
Edad: 36
Genero: Masculino

Datos del cliente:

Nombre: Roberto
Apellido: Ramos
Direccion: Zona 5
Edad: 51
Genero: Masculino

Datos del cliente:

Nombre: Rivers
Apellido: Vargas
Direccion: Zona 4
Edad: 42
Genero: Femenino

Datos del cliente:

Nombre: Maria
Apellido: Diaz
Direccion: Zona 2
Edad: 28
Genero: Femenino

Datos del cliente:

Nombre: Alejandro
Apellido: Quevedo
Direccion: Zona 1
Edad: 20

14.Cambiar la dirección de algunos CLIENTES de la PILA

- El método deberá llamarse `asignarDireccion(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.

```
/*14.Cambiar la dirección de algunos CLIENTES de la PILA.
El método deberá llamarse asignarDireccion(Pila, nuevaDireccion)
El método debe ser creado en la clase MAIN como un método estático.
El método recibe 2 parámetros
-La Pila de Clientes
-La nueva dirección
Cambiar la dirección del cliente siempre y cuando el género sea FEMENINO.*/

public static void asignarDireccion(PilaCliente pila, String nuevaDireccion){
    Cliente elem = null;
    if(pila.estaVacia()){
        System.out.println("La pila esta vacia");
    } else {
        while(!pila.estaVacia()){
            elem = pila.eliminar();
            if(elem.getGenero().equals("Femenino")){
                elem.setDireccion(nuevaDireccion);
                elem.mostrarDatos();
            }
        }
    }
    pila.mostrar();
}
```

```
asignarDireccion(pila, nuevaDireccion: "LaPaz");
```

Datos del cliente:

Nombre: Rivers

Apellido: Vargas

Direccion: LaPaz

Edad: 42

Genero: Femenino

Datos del cliente:

Nombre: Maria

Apellido: Diaz

Direccion: LaPaz

Edad: 28

Genero: Femenino

La pila esta vacia

Process finished with exit code 0

I5.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.

```
/*I5.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  
El método debe reordenar la pila de tal manera que los clientes de género FEMENINO queden al final de la pila. */  
  
public static void reordenaPila(PilaCliente pila){  
    PilaCliente aux= new PilaCliente( max: 10);  
    Cliente valorExtraidoPila=null;  
    while(!pila.estaVacia()){  
        valorExtraidoPila = pila.eliminar();  
        if(valorExtraidoPila.getGenero().equals("Femenino")){  
            valorExtraidoPila.mostrarDatos();  
        }  
        else{  
            aux.insertar(valorExtraidoPila);  
        }  
    }  
    pila.vaciar(aux);  
    pila.mostrar();  
}
```

`reordenaPila(pila);`

Datos del cliente:
Nombre: Rivers
Apellido: Vargas
Direccion: Zona 4
Edad: 42
Genero: Femenino

Datos del cliente:
Nombre: Maria
Apellido: Diaz
Direccion: Zona 2
Edad: 28
Genero: Femenino
Elementos de la pila

Datos del cliente:
Nombre: Roberto
Apellido: Ramos
Direccion: Zona 5
Edad: 51
Genero: Masculino

Datos del cliente:
Nombre: Mariano
Apellido: Gonzales
Direccion: Zona 3
Edad: 36
Genero: Masculino

Datos del cliente:
Nombre: Alejandro
Apellido: Quevedo
Direccion: Zona 1
Edad: 20
Genero: Masculino