

# ***PROCESUAL HITO 3***

**Josias Jonathan Leon Luis**

**Estructura De Datos**



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

En el ámbito de la informática, las estructuras de datos son aquellas que nos permiten, como desarrolladores, organizar la información de manera eficiente, y en definitiva diseñar la solución correcta para un determinado problema.



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

- Arrays.
- Listas enlazadas.
- Pilas.
- Colas.
- Árboles binarios





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

Las estructuras de datos son una forma de organizar los datos en la computadora, de tal manera que nos permita realizar unas operaciones con ellas de forma **muy eficiente**.

Es decir, igual que un array introducimos un dato y eso es prácticamente inmediato, no siempre lo es, según qué estructuras de datos y qué operaciones.

**Depende que algoritmo queramos ejecutar**, habrá veces que sea mejor utilizar una estructura de datos u otra estructura que nos permita más velocidad.

Por este motivo es interesante conocer algo más que simplemente los arrays o los hashmaps que casi todo el mundo conoce.



## 4. ¿Qué es una PILA?

Una **pila** (***stack*** en inglés) es una lista ordenada que permite almacenar y recuperar datos, siendo el modo de acceso a sus elementos de tipo LIFO (del inglés *Last In, First Out*, «último en entrar, primero en salir»).





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

La clase Stack es una clase de las llamadas de tipo LIFO (Last In - First Out, o último en entrar - primero en salir).

Stack es lo mismo que Pila.



## 6. ¿Qué es TOPE en una PILA?

Una colección de datos a los cuales se les puede acceder mediante un extremo, que se conoce generalmente como tope.

## 7. ¿Qué es MAX en una PILA?

Es la máxima cantidad de elementos que puede tener almacenada una pila.





## 8. ¿A que se refiere los métodos esVacia() y esLLena() en una PILA?

El método esVacia() es cuando la pila no contiene elementos almacenados, de forma que para realizar cambios, primero habrá que almacenar información en la pila.

El método esLLena() es cuando el tope de la pila es igual al max, impidiendo poder seguir llenando la pila.





## 9. ¿Qué son los métodos estáticos en JAVA?

Un método estático es un método que tiene **sentido invocarla sin crear previamente ningún objeto.**



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

```
public class PilaCliente {  
  
    4 usages  
    private int max;  
    10 usages  
    private int tope;  
    3 usages  
    private Cliente[] Clientes;  
  
    7 usages  
    public PilaCliente(int max) {  
        this.tope = 0;  
        this.max = max;  
        this.Clientes = new Cliente[this.max + 1];  
    }  
  
    public boolean esVacio () {  
        if (tope == 0) {  
            return true;  
        } else {  
            return false;  
        }  
    }  
  
    1 usage  
    public boolean esLleno () {  
        if (tope == max) {  
            return true;  
        } else {  
            return false;  
        }  
    }  
  
    public int nroElem () {  
        return this.tope;  
    }  
  
    public void adicionar (Cliente nuevoCliente) {
```

```
        public void adicionar (Cliente nuevoCliente) {  
            if (this.esLleno() == false) {  
                this.tope = this.tope + 1;  
                this.Clientes[this.tope] = nuevoCliente;  
            } else {  
                System.out.println("La pila de numeros está llena");  
            }  
        }  
  
        public Cliente eliminar () {  
            Cliente elementoEliminado = null;  
  
            if (!this.esVacio()) {  
                elementoEliminado = (this.Clientes[this.tope]);  
                this.tope = this.tope - 1;  
            } else {  
                System.out.println("La pila de libros está vacia");  
            }  
            return elementoEliminado;  
        }  
  
        public void llenar () {  
        }  
  
        public void mostrar () {  
            Cliente elem = null;  
            if (esVacio())  
                System.out.println("Pila Vacía");  
            else {  
                System.out.println("\nDatos de la Pila de clientes");  
                PilaCliente aux = new PilaCliente(this.max);  
                while (!esVacio()) {  
                    elem = this.eliminar();  
                    aux.adicionar (elem);  
                    elem.mostrarCliente();  
                }  
                vaciar(aux);  
            }  
        }  
    }  
}
```

```
    }  
  
    public void mostrar () {  
        Cliente elem = null;  
        if (esVacio())  
            System.out.println("Pila Vacía");  
        else {  
            System.out.println("\nDatos de la Pila de clientes");  
            PilaCliente aux = new PilaCliente(this.max);  
            while (!esVacio()) {  
                elem = this.eliminar();  
                aux.adicionar (elem);  
                elem.mostrarCliente();  
            }  
            vaciar(aux);  
        }  
    }  
  
    public void vaciar (PilaCliente pila) {  
        while (!pila.esVacio())  
            adicionar(pila.eliminar());  
    }  
}
```



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

```
public class Cliente {  
    3 usages  
    private String Nombres;  
    3 usages  
    private String Apellidos;  
    3 usages  
    private int Edad;  
    3 usages  
    private String Direccion;  
    3 usages  
    private String Genero;  
  
    5 usages  
    public Cliente(String Nombres, String Apellidos, int Edad, String Direccion, String Genero) {  
        this.Nombres = Nombres;  
        this.Apellidos = Apellidos;  
        this.Edad = Edad;  
        this.Direccion = Direccion;  
        this.Genero = Genero;  
    }  
  
    1 usage  
    public String getNombres() {  
        return Nombres;  
    }  
  
    public void setNombres(String nombres) {  
        Nombres = nombres;  
    }  
  
    1 usage  
    public String getApellidos() {  
        return Apellidos;  
    }  
  
    public void setApellidos(String apellidos) {  
        Apellidos = apellidos;  
    }  
}
```

```
public int getEdad() {  
    return Edad;  
}  
  
public void setEdad(int edad) {  
    Edad = edad;  
}  
  
1 usage  
public String getDireccion() {  
    return Direccion;  
}  
  
1 usage  
public void setDireccion(String direccion) {  
    Direccion = direccion;  
}  
  
3 usages  
public String getGenero() {  
    return Genero;  
}  
  
public void setGenero(String genero) {  
    Genero = genero;  
}  
  
1 usage  
public void mostrarCliente() {  
    System.out.println("\nMostrando datos del jugador");  
    System.out.println("Nombre: " + this.getNombres());  
    System.out.println("Apellidos: " + this.getApellidos());  
    System.out.println("Edad: " + this.getEdad());  
    System.out.println("Direccion: " + this.getDireccion());  
    System.out.println("Genero: " + this.getGenero());  
  
    System.out.println("\n");  
}
```



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

```
1 package Hito3.PilaDeClientes;
2
3 public class Main {
4     public static void main(String [] args) {
5         Cliente cli1 = new Cliente( Nombres: "Adolf", Apellidos: "Hitler", Edad: 56, Direccion: "Alemania", Genero: "Masculino");
6         Cliente cli2 = new Cliente( Nombres: "Josef", Apellidos: "Stalin", Edad: 74, Direccion: "Rusia", Genero: "Masculino");
7         Cliente cli3 = new Cliente( Nombres: "Alejandra", Apellidos: "Maine", Edad: 19, Direccion: "Estados Unidos", Genero: "Femenino");
8         Cliente cli4 = new Cliente( Nombres: "Emma", Apellidos: "Stone", Edad: 33, Direccion: "Estados Unidos", Genero: "Femenino");
9         Cliente cli5 = new Cliente( Nombres: "Natalia", Apellidos: "Poklonskaya", Edad: 42, Direccion: "Rusia", Genero: "Femenino");
10
11         PilaCliente pila = new PilaCliente( max: 100);
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
21 }
22
23 }
```

Main  
Edad: 74  
Direccion: Rusia  
Genero: Masculino

Mostrando datos del jugador  
Nombre: Adolf  
Apellidos: Hitler  
Edad: 56  
Direccion: Alemania  
Genero: Masculino





# 12.Determinar cuántos CLIENTES son mayores de 20 años.

```
13      pila.adicionar(cli1);
14      pila.adicionar(cli2);
15      pila.adicionar(cli3);
16      pila.adicionar(cli4);
17      pila.adicionar(cli5);
18      //pila.mostrar();
19      mayoresCiertasEdad(pila, edadMayor: 40);
20
21  }
22  //Determinar cuantos clientes son mayores a 20 años
23  1 usage
24  @
25  public static void mayoresCiertasEdad(PilaCliente pila, int edadMayor) {
26      PilaCliente aux = new PilaCliente( max: 10);
27      int MayoresXEdad = 0;
28      Cliente Valorextraido = null;
29      while (pila.esVacio() == false) {
30          Valorextraido = pila.eliminar();
31          if (Valorextraido.getEdad() > edadMayor) {
32              MayoresXEdad = MayoresXEdad + 1;
33          }
34          aux.adicionar(Valorextraido);
35      }
36      System.out.println("\nLa cantidad de clientes con mas de " + edadMayor + " son: " + MayoresXEdad);
37  }
```

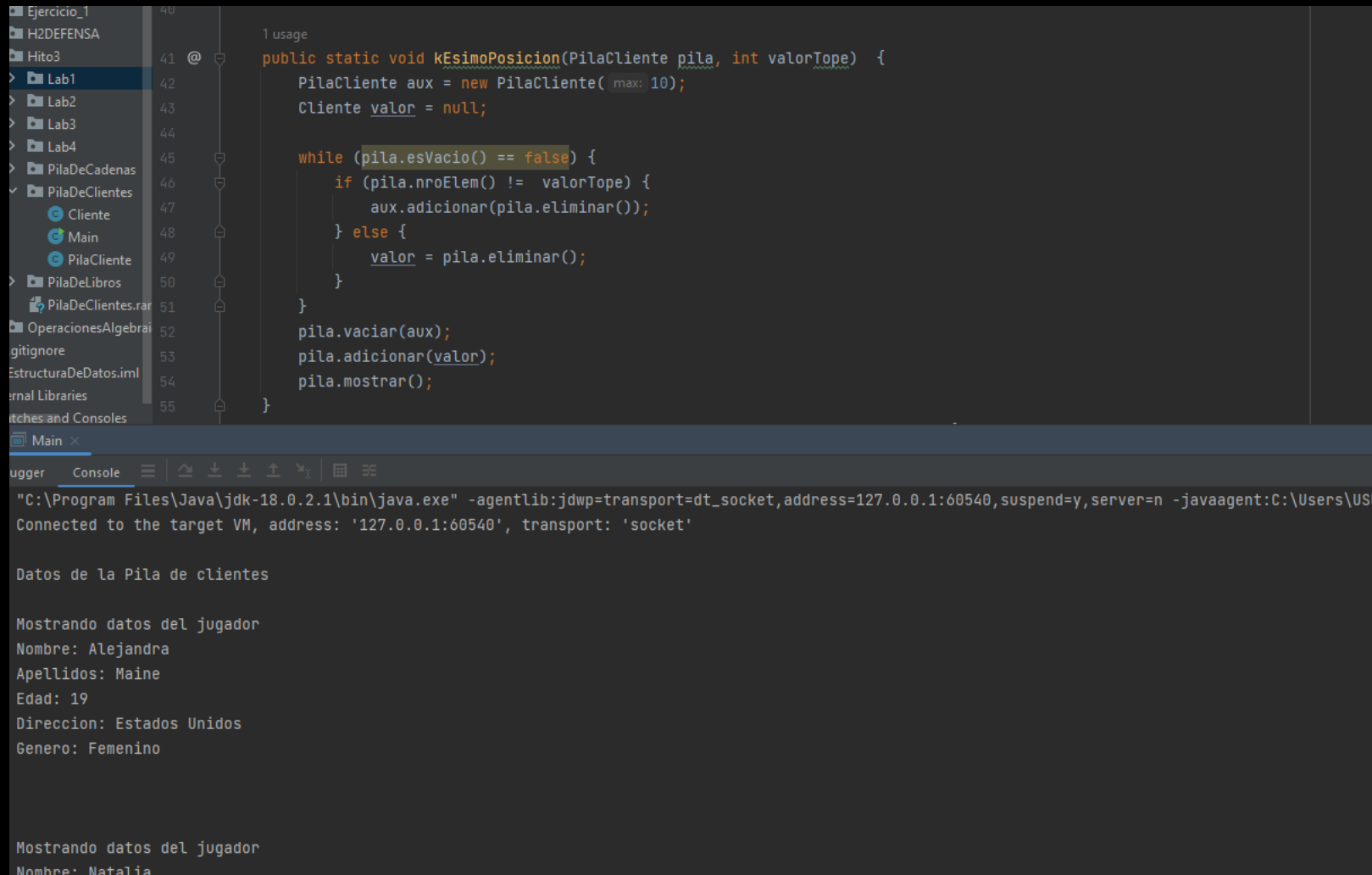
Main x  
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.2.2\lib\idea\_rt.jar=53541:C:\Program

La cantidad de clientes con mas de 40 son: 3

Process finished with exit code 0



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



The screenshot shows an IDE with a project named 'Ejercicio\_1'. The file explorer on the left shows a package structure with 'PilaDeClientes' containing 'Cliente', 'Main', and 'PilaCliente'. The main editor displays the implementation of the 'kEesimoPosicion' method in 'PilaCliente.java'. The method moves the k-th element of the stack to the bottom. The console output shows the state of the stack after two operations.

```
1 usage
41 @
42 public static void kEesimoPosicion(PilaCliente pila, int valorTope) {
43     PilaCliente aux = new PilaCliente( max: 10);
44     Cliente valor = null;
45
46     while (pila.esVacio() == false) {
47         if (pila.nroElem() != valorTope) {
48             aux.adicionar(pila.eliminar());
49         } else {
50             valor = pila.eliminar();
51         }
52     }
53     pila.vaciar(aux);
54     pila.adicionar(valor);
55     pila.mostrar();
56 }
```

Console Output:

```
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" -agentlib:jdwp=transport=dt_socket,address=127.0.0.1:60540,suspend=y,server=n -javaagent:C:\Users\USI
Connected to the target VM, address: '127.0.0.1:60540', transport: 'socket'

Datos de la Pila de clientes

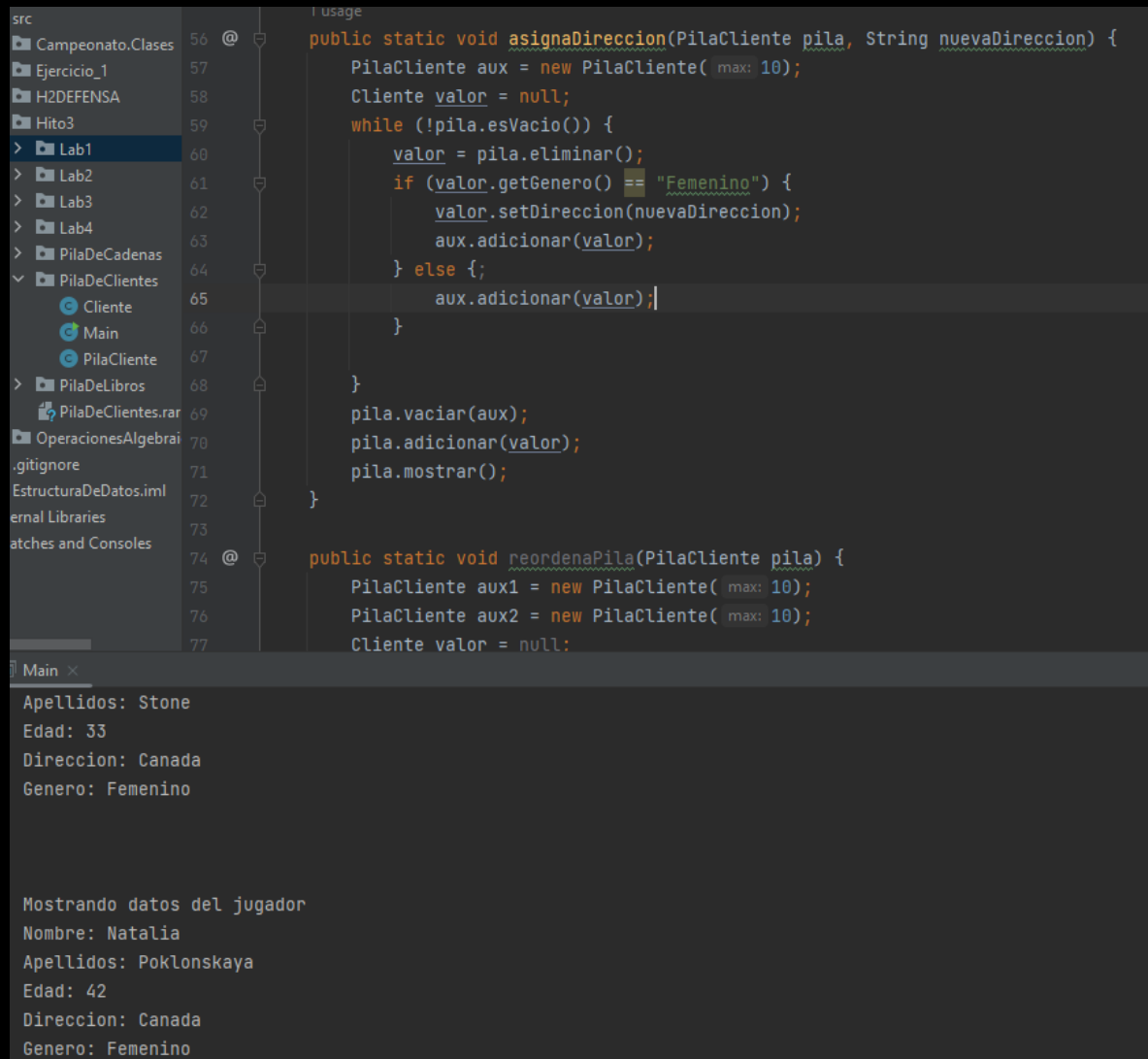
Mostrando datos del jugador
Nombre: Alejandra
Apellidos: Maine
Edad: 19
Direccion: Estados Unidos
Genero: Femenino

Mostrando datos del jugador
Nombre: Natalia
```





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



The screenshot shows an IDE with a project named 'Campeonato.Clases'. The file explorer on the left lists several files, including 'Lab1', 'Lab2', 'Lab3', 'Lab4', 'PilaDeCadenas', 'PilaDeClientes', 'PilaDeLibros', and 'OperacionesAlgebrai'. The main editor displays the code for 'PilaDeClientes'. The code includes a method 'asignaDireccion' that iterates through a stack of clients and updates the direction of female clients to 'Canada'. The output window at the bottom shows the results of the program execution.

```
src
├── Campeonato.Clases
├── Ejercicio_1
├── H2DEFENSA
├── Hito3
├── Lab1
├── Lab2
├── Lab3
├── Lab4
├── PilaDeCadenas
├── PilaDeClientes
│   ├── Cliente
│   ├── Main
│   └── PilaCliente
├── PilaDeLibros
├── PilaDeClientes.rar
├── OperacionesAlgebrai
├── .gitignore
├── EstructuraDeDatos.iml
├── External Libraries
├── Attachments and Consoles
└── Main

1 usage
56 @
57 public static void asignaDireccion(PilaCliente pila, String nuevaDireccion) {
58     PilaCliente aux = new PilaCliente( max: 10);
59     Cliente valor = null;
60     while (!pila.esVacio()) {
61         valor = pila.eliminar();
62         if (valor.getGenero() == "Femenino") {
63             valor.setDireccion(nuevaDireccion);
64             aux.adicionar(valor);
65         } else {
66             aux.adicionar(valor);
67         }
68     }
69     pila.vaciar(aux);
70     pila.adicionar(valor);
71     pila.mostrar();
72 }
73
74 @
75 public static void reordenaPila(PilaCliente pila) {
76     PilaCliente aux1 = new PilaCliente( max: 10);
77     PilaCliente aux2 = new PilaCliente( max: 10);
78     Cliente valor = null;
```

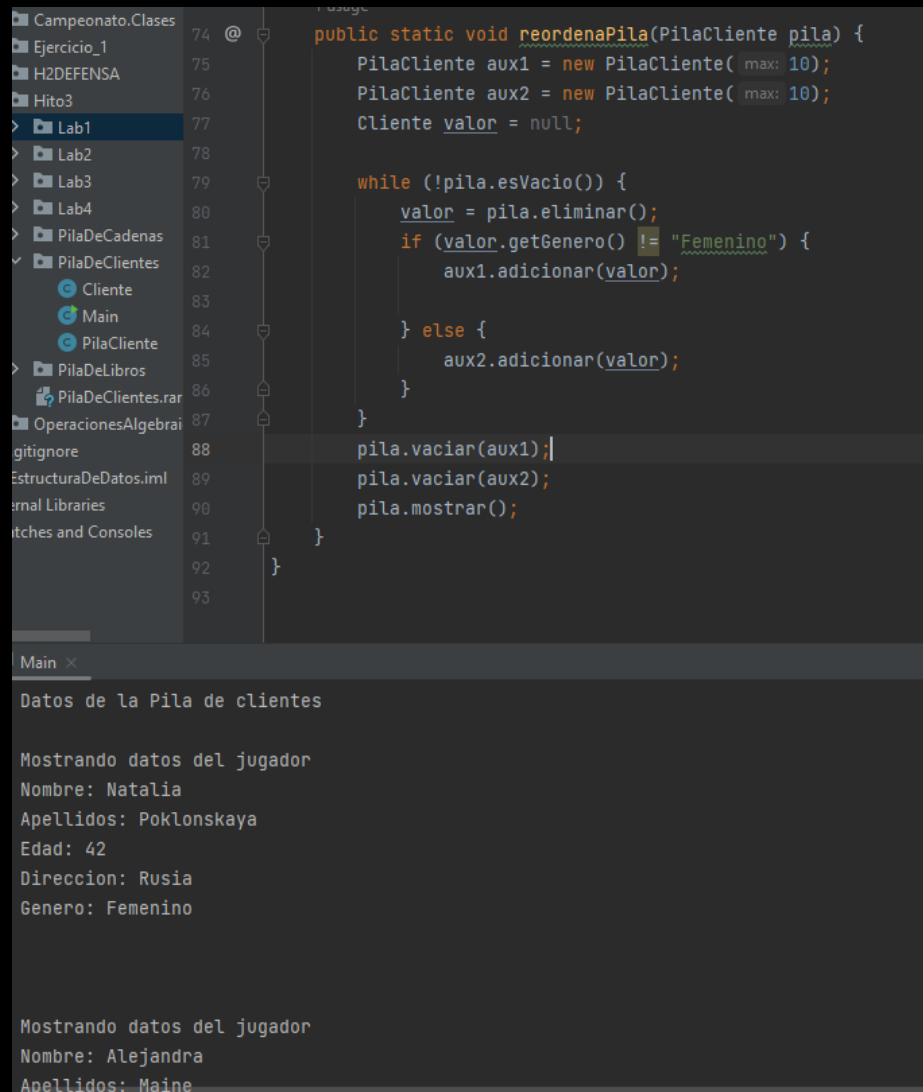
Apellidos: Stone  
Edad: 33  
Direccion: Canada  
Genero: Femenino

Mostrando datos del jugador  
Nombre: Natalia  
Apellidos: Poklonskaya  
Edad: 42  
Direccion: Canada  
Genero: Femenino

La nueva dirección usada para los clientes de género femenino para este ejemplo fue Canadá



# 15.Mover ÍTEMS de la PILA.



```
74 @
75 public static void reordenaPila(PilaCliente pila) {
76     PilaCliente aux1 = new PilaCliente( max: 10);
77     PilaCliente aux2 = new PilaCliente( max: 10);
78     Cliente valor = null;
79
80     while (!pila.esVacio()) {
81         valor = pila.eliminar();
82         if (valor.getGenero() != "Femenino") {
83             aux1.adicionar(valor);
84         } else {
85             aux2.adicionar(valor);
86         }
87     }
88     pila.vaciar(aux1);
89     pila.vaciar(aux2);
90     pila.mostrar();
91 }
92
93 }
```

Datos de la Pila de clientes

Mostrando datos del jugador  
Nombre: Natalia  
Apellidos: Poklonskaya  
Edad: 42  
Direccion: Rusia  
Genero: Femenino

Mostrando datos del jugador  
Nombre: Alejandra  
Apellidos: Maine

Mover a la base todos los clientes del género masculino y los del género femenino moverlos al final.

