¿Qué es la programacion Imperativa? . Todos estamos habituados a programar y cuando lo hacemos habitualmente usamos un enfoque imperativo. ¿Qué quiere decir esto? , pues simplemente que le decimos al lenguaje de programación como tiene que realizar cada uno de los pasos .Le decimos que variables usar, que bucles y sentencias etc. Es decir construimos un algoritmo muy concreto para solventar un problema. Vamos a ver un ejemplo con una lista de Personas:

```
package com.arquitecturajava;

public class Persona {

    private String nombre;
    private int edad;

    // Constructor
    public Persona(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }

    // Getters y Setters
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getEdad() {
        return edad;
```

```
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }
}
```

Programacion Imperativa

Vamos a crear un programa que nos imprima la media de edad de las personas mayores de 18 años . Para ello vamos a usar un enfoque de programacion imperativa.

```
package com.arquitecturajava;

import java.util.ArrayList;
import java.util.List;

public class Principal {

    public static void main(String[] args) {

        Persona p1 = new Persona("pepe", 20);
        Persona p2 = new Persona("juan", 12);
        Persona p3 = new Persona("angela", 30);
        Persona p4 = new Persona("gema", 40);
        Persona p5 = new Persona("david", 15);

        List<Persona> lista = new ArrayList<>();

        lista.add(p1);
        lista.add(p2);
        lista.add(p3);
```

```
        lista.add(p4);
        lista.add(p5);

        int totalEdad = 0;
        int totalPersonas = 0;

        for (Persona p : lista) {
            if (p.getEdad() >= 18) {
                totalEdad += p.getEdad();
                totalPersonas++;
            }
        }

        if (totalPersonas > 0) {
            System.out.println(totalEdad / totalPersonas);
        } else {
            System.out.println("No hay personas mayores de edad.");
        }
    }
}
```

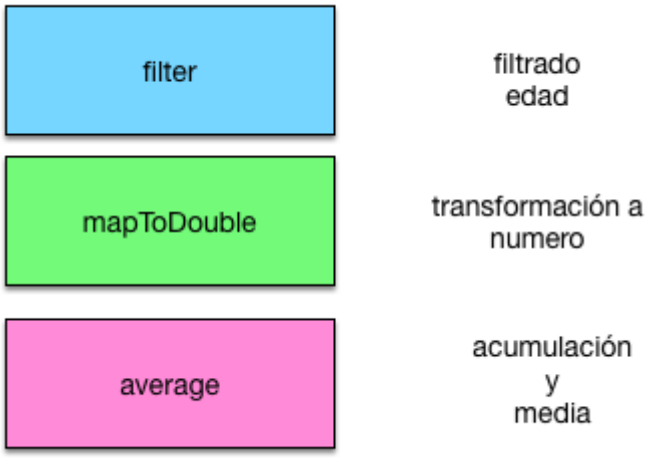
En este caso hemos declarado dos variables que hacen de acumuladores y utilizado un bucle for y una sentencia if. El programa imprime un resultado.

Sin embargo hoy en día esta opción ya no es la mejor opción . Vamos a realizar la misma operativa con programación declarativa.

```
package com.arquitecturajava;

import java.util.ArrayList;
import java.util.List;
import java.util.OptionalDouble;

public class Principal2 {

    public static void main(String[] args) {

        Persona p1 = new Persona("pepe", 20);
        Persona p2 = new Persona("juan", 12);
        Persona p3 = new Persona("angela", 30);
        Persona p4 = new Persona("gema", 40);
        Persona p5 = new Persona("david", 15);

        List<Persona> lista = new ArrayList<>();

        lista.add(p1);
        lista.add(p2);
        lista.add(p3);
        lista.add(p4);
        lista.add(p5);

        OptionalDouble resultado = lista.stream()
                .filter(persona -> persona.getEdad() >= 18)
                .mapToInt(Persona::getEdad)
                .average();

        if (resultado.isPresent()) {
            System.out.println(resultado.getAsDouble());
```

```
        } else {
            System.out.println("No hay personas mayores de edad.");
        }
    }
}
```

En este caso hemos solicitado al programa que realice una serie de tareas apoyándonos en programación funcional “el enfoque ha sido declarativo”.



Hemos declarado de forma abstracta las tareas a realizar y el lenguaje se ha encargado de los detalles. En este caso hemos utilizado expresiones Lambda. El resultado es idéntico.

```
<terminated> Principal2 (5) [Java Application] /Library/Java/Java
30.0
```

La programación declarativa nos permite crear programas más compactos , reutilizables y thread safe , poco a poco nos iremos acostumbrando a ella en el mundo Java.

Otros artículos relacionados :

- Javascript const vs var vs let
- JavaScript var y sus curiosidades
- Java var keyword y su uso con JDK10
- El concepto de Java 8 reference method