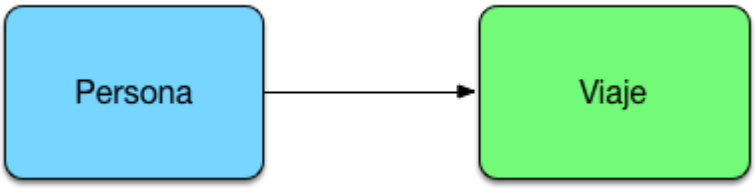
El uso de Java 8 FlatMap es algo que en muchas ocasiones cuesta entender . La programación funcional en Java 8 esta empezando y para la mayor parte de la gente es algo muy nuevo. Vamos a crear un ejemplo sencillo de flatMap, partiremos de dos clases relacionadas Personas y Viajes.



Una persona realiza varios viajes y contiene por lo tanto una lista de viajes:

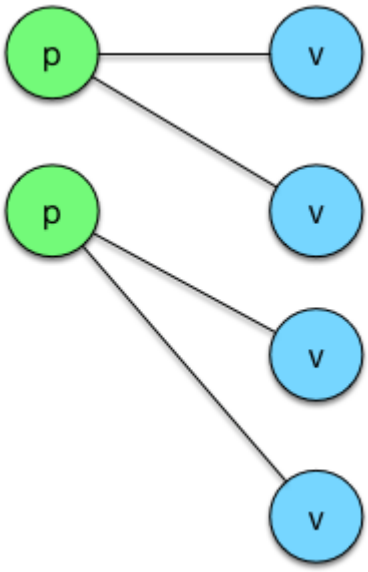
```
import java.util.List;

public class Persona {
    private String nombre;
    private List<Viaje> lista = new ArrayList<Viaje>();

    public Persona(String nombre) {
        super();
        this.nombre = nombre;
    }

    public String getNombre() {
        return nombre;
    }

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

    public void addViaje(Viaje v) {
```

```
        lista.add(v);
    }

    public List<Viaje> getLista() {
        return lista;
    }
}
```

A continuación se muestra el concepto de Viaje que únicamente contiene el pais:

```
package com.arquitecturajava;

public class Viaje {
  private String pais;

  public Viaje(String pais) {
    super();
    this.pais = pais;
  }

  public String getPais() {
    return pais;
  }

  public void setPais(String pais) {
    this.pais = pais;
  }
}
```

La estructura es anidada.

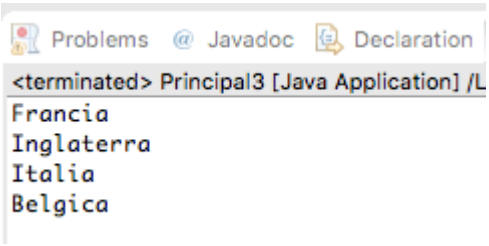


Es momento de crear y recorrer la estructura en código para obtener todos los paises a los cuales se ha viajado :

```
package com.arquitecturajava;

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

public class Principal {
    public static void main(String[] args) {
        Persona p = new Persona("pedro");
        Viaje v = new Viaje("Francia");
        Viaje v2 = new Viaje("Inglaterra");
        p.addViaje(v);
        p.addViaje(v2);

        Persona p1 = new Persona("gema");
        Viaje v3 = new Viaje("Italia");
        Viaje v4 = new Viaje("Belgica");
```

```
        p1.addViaje(v3);
        p1.addViaje(v4);

        List<Persona> lista = new ArrayList<Persona>();
        lista.add(p);
        lista.add(p1);

        for (Persona persona : lista) {
            for (Viaje viaje : persona.getLista()) {
                System.out.println(viaje.getPais());
            }
        }
    }
}
```

La consola nos mostrará:



Listas y bucles anidados

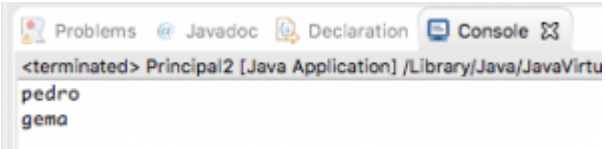
El problema es que en ningún caso hemos usado programación funcional para recorrer la lista, simplemente dos bucles anidados. No es una mala solución pero es mejorable con programación funcional.

Usando Java 8 Map

El primer paso va a ser usar la función map para que a través de programación funcional nos imprima los nombres de las personas.

```
lista.stream()
    .map(persona -> persona.getNombre())
    .forEach(new Consumer<String>() {
        @Override
        public void accept(String s) {
            System.out.println(s);
        }
    });
```

Hemos convertido la lista en un Stream y a través de map() hemos convertido el Stream de Personas en un Stream de “Strings” que imprime los nombres.



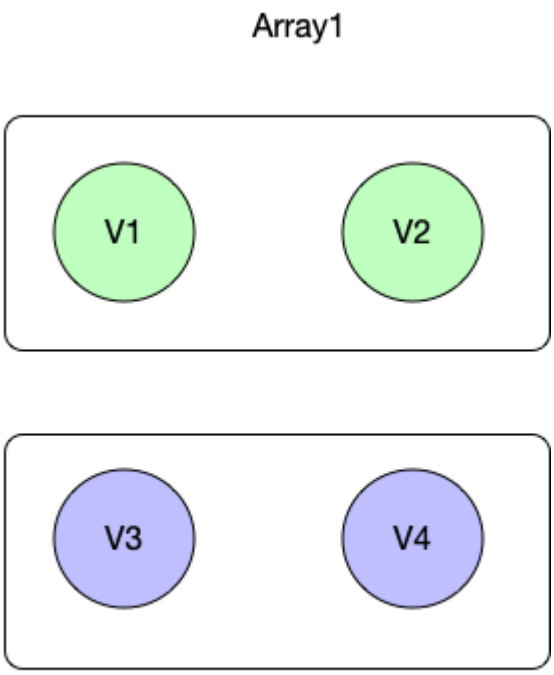
Sería mucho más óptimo usar varios Java reference method y realizarlo así :

```
lista.stream()
    .map(Persona::getNombre)
    .forEach(System.out::println);
```

Lamentablemente no es suficiente ya que lo que queríamos ya que necesitamos imprimir el nombre de los países. Así pues tenemos que trabajar un poquito más:

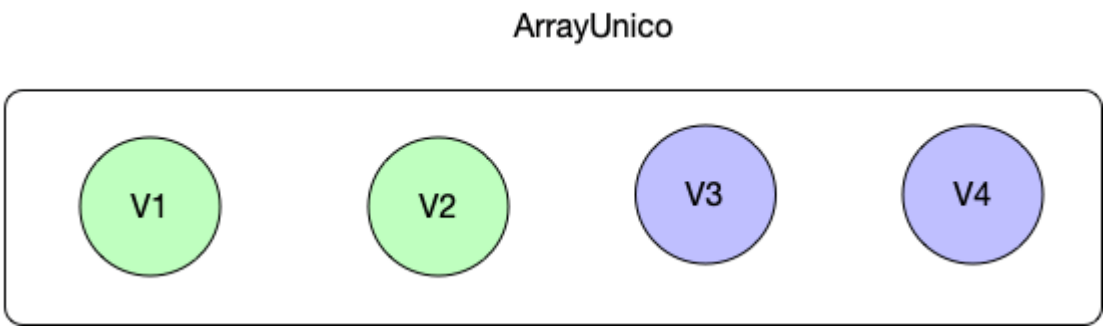
Usando Java 8 FlatMap

Para conseguir la lista de países debemos operar de otra manera y usar la función flatMap. Para ello combinamos map y flatMap. . Con map obtenemos la lista de viajes de cada Persona. Una vez que tenemos esto claro lo que abremos seleccionado en el Stream son multiples arrays de datos cada uno con una lista de Paises.



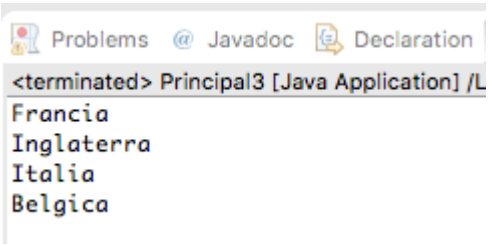

```
lista.stream()
    .map(persona -> persona.getLista())
    .flatMap(viajes -> viajes.stream())
    .forEach(v->System.out.println(v.getPais());
```

Lo que hacemos ahora es usar Java 8 FlatMap y obligar a que esos dos arrays se conviertan en uno solo ya que la operación de flatMap aplana el array y consigue que lo operemos como un único array con 4 elementos.

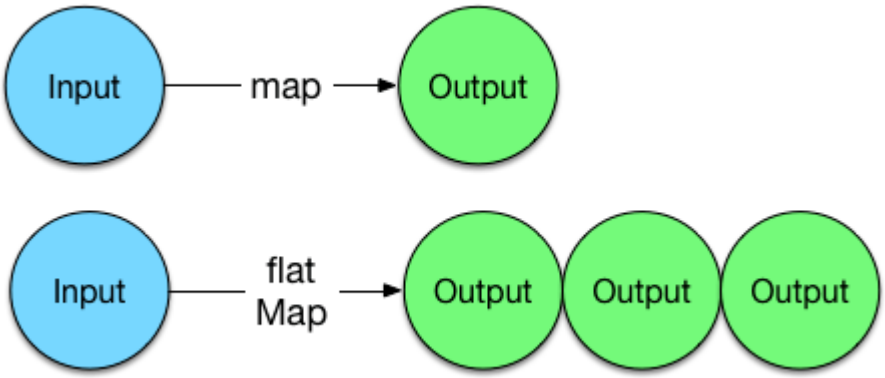


Si vemos el resultado las cosas quedan mucho más claras tenemos la lista que necesitamos

sin usar bucles anidados:



Ahora bien ,¿Cómo funciona exactamente flatMap? . FlatMap es una función que recibe una entrada y devuelve varias salidas para esa entrada . Esa es la diferencia con Map que tiene una entrada y devuelve una única salida.



Java 8 FlatMap nos ha ayudado a eliminar todo tipo de bucle del programa.

Otros artículos relacionados:

- ¿Que es un JavaScript Observable?
- Rx flatMap (mergeMap) y simplificación de observables
- ¿Que es un Java Predicate?
- Introducción a Spring Data y JPA