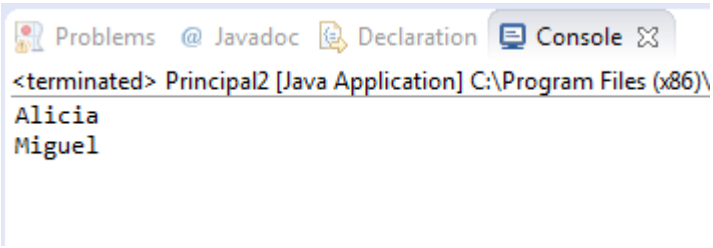
Poco a poco Java 8 se va dando a conocer como plataforma de desarrollo e incluye una serie de novedades. Entre las cuales hay que destacar las lambda expresions (Expresiones Lambda) que se echaban bastante ya que otros lenguajes como C# ya soportaban estructuras similares. El uso de Expresiones Lambda nos permitirá simplificar de forma muy clara algunos de los bloques de código que construíamos hasta ahora. Bloques que en muchos casos hacían uso de clases anónimas para solventar problemas que una expresión lambda expresa de una forma mucho mas directa. Vamos a ver un sencillo bloque de código que usa un Comparador para ordenar una lista de Personas.

```
package com.arquitecturajava;

public class Persona {

 private String nombre;

public String getNombre() {
 return nombre;
 }

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

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

```
package com.arquitecturajava;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

public class Principal {

 public static void main(String[] args) {

 ArrayList<Persona> milista= new ArrayList<Persona>();
 milista.add(new Persona("Miguel"));
 milista.add(new Persona("Alicia"));

 Collections.sort(milista,new Comparator<Persona>() {

 public int compare(Persona p1,Persona p2) {

 return p1.getNombre().compareTo(p2.getNombre());
 }

 });

 for (Persona p: milista) {

 System.out.println(p.getNombre());

 }
```

```
 }

}
```

Hemos usado un Comparator para ordenar a las Personas por su nombre.



Eso si el código es bastante engorroso. y realmente si nos ponemos a pensar lo único importante en todo el bloque del comparador es:

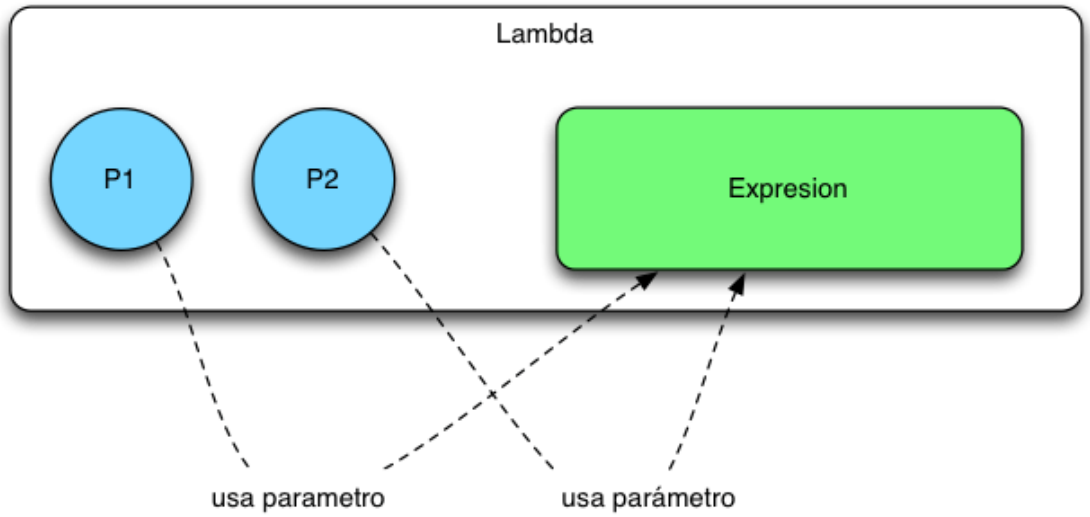
```
(Persona p1,Persona p2) {

 return p1.getNombre().compareTo(p2.getNombre());
 }
```

Esta parte del código es la que implemente la lógica de ordenación y como vemos es practicamente una función. Las expresiones lambda de Java 8 convierten a las funciones en elementos de primer nivel en el Lenguaje Java de tal forma que se puedan usar de una forma más sencilla y directa.

Lambda Expressions

Una expresión lambda se compone de dos elementos. En primer lugar de un conjunto de

parámetros y en segundo lugar de una expresión que opera con los parámetros indicados.



Vamos a hacer uso de una expresión lambda en el ejemplo anterior para simplificar el manejo de los interfaces :

```
package com.arquitecturajava;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

public class Principal2 {

public static void main(String[] args) {
```

```
ArrayList<Persona> milista= new ArrayList<Persona>();
milista.add(new Persona("Miguel"));
milista.add(new Persona("Alicia"));

Collections.sort(milista,
(Persona p1,Persona p2)-> p1.getNombre().compareTo(p2.getNombre()));

for (Persona p: milista) {

System.out.println(p.getNombre());

}
}

}
```

Como podemos ver la siguiente linea de código simplifica sobre manera el programa:

```
Collections.sort(milista,
(Persona p1,Persona p2)-> p1.getNombre().compareTo(p2.getNombre()));
```

La programación funcional y el uso de expresiones lambda nos ayudará a construir programas más claros y más flexibles

Otros artículos similares :

String y StringBuilder

Java Generics