Poco a poco las expresiones Lambda se comienzan a utilizar. Una de las novedades es el uso de iteradores forEach en Java 8 . Vamos a explicar como estos funcionan. Para ello vamos a partir de un ejemplo con un bucle forEach clásico que recorre una colección de Personas:

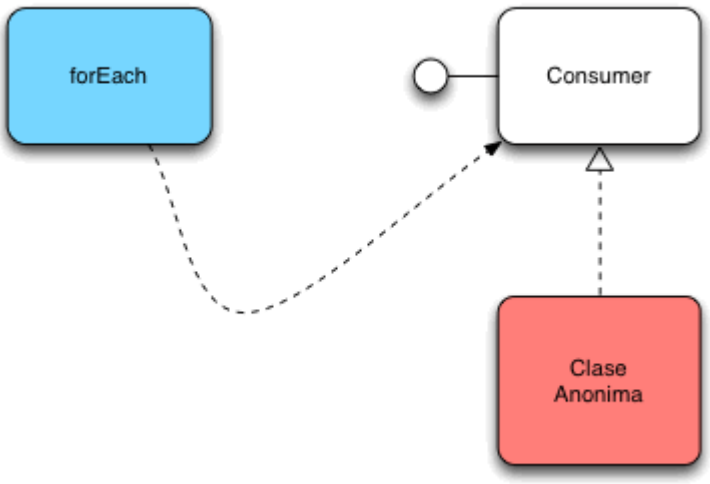
```
package com.arquitecturajava;

import java.util.ArrayList;

public class Principal4 {

  public static void main(String[] args) {

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

    for (Persona p: milista) {

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

  }

}
```

Esta operación la podemos realizar de similar forma utilizando el método forEach de Java 8 que las colecciones soportan a través del interface Iterable. Así pues el nuevo código sería :

```
package com.arquitecturajava;

import java.util.ArrayList;
```

```
import java.util.function.Consumer;

public class Principal3 {

  public static void main(String[] args) {

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

    milista.forEach(new Consumer < Persona > () {

      @Override
      public void accept(final Persona persona) {

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

      }

    });

  }

}
```

El método forEach soporta un nuevo tipo de Clase que se denomina “Consumer” la cual dispone de un método accept que realiza las operaciones que necesitemos sobre los objetos con los que estamos trabajando. Esta modificación parece de entrada un paso atrás ya que el código es más complejo al haber usado una clase anónima.



Sin embargo lo podemos simplificar sobre manera utilizando una expresión Lambda.

```
package com.arquitecturajava;

import java.util.ArrayList;
import java.util.function.Consumer;

public class Principal5 {

  public static void main(String[] args) {

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

    milista.forEach((final Persona persona) ->
System.out.println(persona.getNombre()));
  }

}
```

El compilador realiza las modificaciones necesarias para que todo funcione perfectamente y el código queda claramente simplificado.

CURSO Diseño Orientado Objeto

GRATIS

APUNTATE!!

Otros artículos relacionados:

- Java y Expresiones Lambda
- Java Iterator