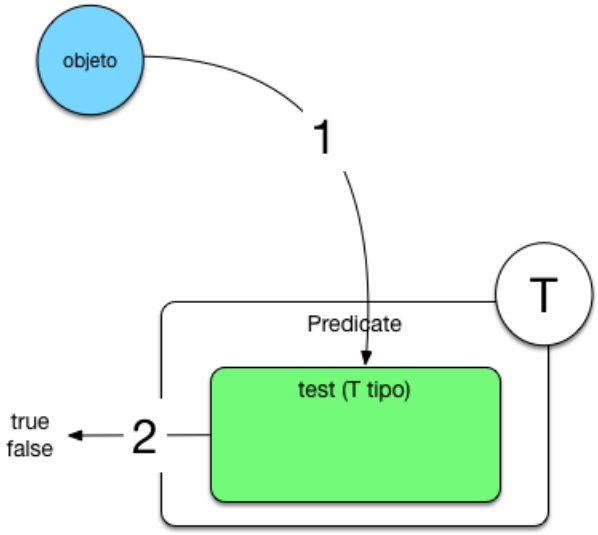
Utilizando Java 8 Predicate

25 mayo, 2016 Por Cecilio Álvarez Caules 6 comentarios

Crear un Java 8 Predicate , es una de las operaciones que más realizaremos cuando trabajemos **con expresiones Lambda y Streams en Java 8**. **¿Qué es un Predicado?** , un Predicado es un **interface funcional** que define una condición que un objeto determinado debe cumplir . **¿Por ejemplo es una Persona mayor de 20 años?**.



El Predicado dispone de **un único método denominado test que recibe el objeto** y comprueba si cumple la condición. Vamos a construir en ejemplo apoyándonos en la clase Persona.

```
1  package com.arquitecturajava;
2
3  public class Persona {
4
5  private String nombre;
6  private String apellidos;
7  private int edad;
8  public String getNombre() {
9  return nombre;
10 }
11 public void setNombre(String nombre) {
12 this.nombre = nombre;
13 }
14 public Persona(String nombre, String apellidos, int edad) {
15 super();
16 this.nombre = nombre;
17 this.apellidos = apellidos;
18 this.edad = edad;
19 }
20 public String getApellidos() {
21 return apellidos;
22 }
23 public void setApellidos(String apellidos) {
```

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

}
```

Vamos a **construir una lista de Personas** para más adelante aplicar un Predicado sobre ella.

```
public class Principal {

public static void main(String[] args) {

List<Persona> lista = new ArrayList<>();
Persona p1 = new Persona("pepe", "perez", 20);
Persona p2 = new Persona("angel", "sanchez", 30);
Persona p3 = new Persona("pepe", "blanco", 40);
lista.add(p1);
lista.add(p2);
lista.add(p3);


}

}
```

Finalmente convertiremos la lista **en un Stream de datos y la recorreremos**:

```
lista.stream().forEach(new Consumer<Persona>() {

@Override
public void accept(Persona p) {

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

}

});
```

El resultado será

pepe
angel
pepe

Java 8 Predicate

Hemos recorrido la lista , vamos a **crear un Predicado** que se encargue de definir una condición que permita filtrar la lista.

```
Predicate<Persona> predicadoNombre = new Predicate<Persona>() {

@Override
 public boolean test(Persona p) {

 return p.getNombre().equals("pepe");
 }

```

```
};
```

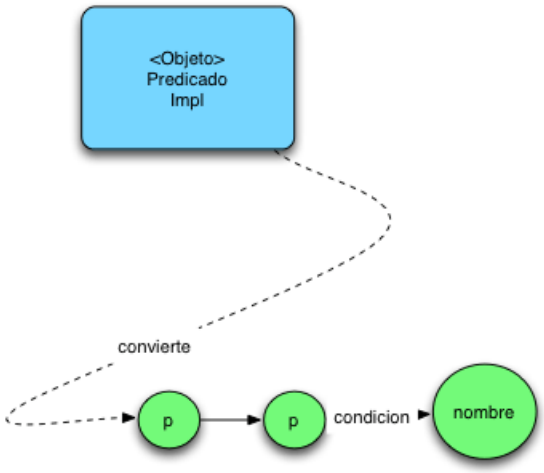
En este caso acabamos de crear un Predicado que solo cumplirán las personas **que se llamen "pepe".** Usamos la función **filter a nivel de Streams para pasar este objeto Predicado** y **obligar a que se filtre en base a la condición que el Predicado define.**

```
lista.stream().filter(predicadoNombre).forEach(p -> System.out.println(p.getApellidos()
```

El resultado en pantalla será:

```
***************
perez
blanco
***************
```

El Stream se ha apoyado en el Predicado para realizar el filtrado y quedarnos con las Personas que se llaman "pepe" . Recordemos que los interfaces funcionales se pueden ver como expresiones Lambda y su gestión es más sencilla.



Es momento de convertir nuestro **Java 8 Predicate a una expresión lambda para ganar claridad:**

```
lista.stream()
.filter(p -> p.getNombre().equals("pepe"))
.forEach(p -> System.out.println(p.getApellidos()));
```

El resultado será idéntico:

```
***************
perez
blanco
***************
```

Los Predicados de Java 8 son inprescindibles

Otros artículos relacionados: Java Lambda Java Streams