Predicate Interface Part 2: Using and operation in Predicates.

In previous post we saw how to filter the collection using the Predicate. In this post we will see how to combine different predicates to get results.

In basic introduction post of Predicate we saw different default methods in Predicate Interface. In this post we will explore and(..) method.

We will take example of Person class that was discussed in previous post. For convenience, I will copy the code here.

**package** com.example.javase8.filtercollections;

**public** **class** Person {

**private** String name;

**private** **int** age;

**public** Person(String name, **int** age) {

**this**.name = name;

**this**.age = age;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

@Override

**public** String toString() {

**return** "[name=" + name + ", age=" + age + "]";

}

}

Now let us insert few Person objects with their name and age into the List.

List<Person> list = **new** ArrayList<>();

list.add(**new** Person("Ned", 10));

list.add(**new** Person("Ryan", 20));

list.add(**new** Person("James", 30));

list.add(**new** Person("Walder", 35));

list.add(**new** Person("Roose", 37));

list.add(**new** Person("Roy", 40));

list.add(**new** Person("Joshua", 50));

Let us define two predicates

Predicate<Person> predicate1 = (p) -> p.getAge() > 20;

Predicate<Person> predicate2 = (p) -> p.getAge() < 40;

predicate1 – This predicate is used to get all the objects or elements from List whose age is greater than 20. Executing predicate1 alone yields following output.

Output

[name=James, age=30]

[name=Walder, age=35]

[name=Roose, age=37]

[name=Roy, age=41]

[name=Joshua, age=50]

predicate2 - This predicate is used to get all the objects or elements from List whose age is less than 40. Executing predicate2 alone yields following output.

Output

[name=Ned, age=10]

[name=Ryan, age=20]

[name=James, age=30]

[name=Walder, age=35]

[name=Roose, age=37]

Now let say if both the predicates are joined then we will get those Person object’s whose age is greater than 20 but less than 40. i.e. 30, 35 and 37.

Ok so let us join both the above predicates.

Predicate<Person> norPred1NorPred2 = **predicate1.and(predicate2);**

So we in all we have 3 predicates.

Predicate<Person> **predicate1** = **(p) -> p.getAge() > 20;**

Predicate<Person> **predicate2** = **(p) -> p.getAge() < 40;**

Predicate<Person> **norPred1NorPred2** = **predicate1.and(predicate2);**

Now let us execute the third predicate.

list.forEach(p -> {

**if** (norPred1NorPred2.test(p)) {

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

}

});

Now, what we can do is we can replace third predicate **norPred1NorPred2.**

Predicate<Person> norPred1NorPred2 = (p) -> p.getAge() > 20

&& p.getAge() < 40;

This will work same as above. I combined two predicates just to give an example that we can join 2 different predicates and get third predicate.

Let us now add this predicate to stream and filter it out.

list.stream()

.filter(norPred1NorPred2)

.forEach(p -> System.***out***.println(p));