Predicate

Functional Interface

Java 8

Shiva Prasad Gurram

- This interface was introduced in Java 8.
- This Functional Interface is used for conditional check.
- It represents a boolean-valued-function of one argument.
- This is mainly used to filter data from a Java Stream

- Since this is a functional interface and can therefore be used as the assignment target for a lambda expression or method reference.

- We can use this whenever we want to check something and return true or false based on condition.

- **It has only one abstract method and few default and static methods.**

✅ **boolean test(T t);**
✅ **default Predicate<T> and(Predicate<? super T> other);**
✅ **default Predicate<T> or(Predicate<? super T> other);**
✅ **static <T> Predicate<T> isEquals(Object targetRef);**
✅ **default Predicate<T> negate();**

Real time use case

- Let's assume we have list of orders and we need to apply the discount of 10% on orders whose purchase value is greater than 1000.

- Assume our Order class has below data members

  orderId
  price
  quantity
  purchaseValue

Lets implement the requirement

Here we are going to use "filter(Predicate<? super T> predicate)" which is an intermediate operation and it returns the stream of elements that matches the given condition.

Coding:

```
List<Order> orders = Arrays.asList(
        new Order(1001, 500, 3, 3 * 500),
        new Order(1002, 200, 4, 4 * 200),
        new Order(1003, 350, 3, 3 * 350),
        new Order(1004, 600, 7, 7 * 600));
```

contd..

```
List<Double> discountEligibleOrders = orders
    .stream()
    .filter(order -> order.getPurchaseValue() > 1000)
    .peek(order -> System.out.println("orderId:" +order.getOrderId() +"
purchaseValue:"+ order.getPurchaseValue()))
    .map(order -> order.getPurchaseValue() - (order.getPurchaseValue()/100) *
10)
    .collect(Collectors.toList());

System.out.println(discountEligibleOrders);
```

filter method taking Predicate to determine eligible orders

contd..

Output of code

orderId:1001 purchaseValue:1500.0
orderId:1003 purchaseValue:1050.0
orderId:1004 purchaseValue:4200.0
[1350.0, 945.0, 3780.0]

Being a Java developer we often use this predicate FI in so many places. This slideshow demonstrates basic behaviour of Predicate FI.

Other way

You can write code like below to print the discount eligible orders.

```
Predicate<Order> pFI = (order) -> item.getPurchaseValue() > 1000;
orders.stream().filter(pFI).forEach(order -> System.out.println(order));
```

Output:

```
Order [orderId=1001, price=500, quantity=3, purchaseValue=1500.0]
Order [orderId=1003, price=350, quantity=3, purchaseValue=1050.0]
Order [orderId=1004, price=600, quantity=7, purchaseValue=4200.0]
```

Thank you!