Predicates

- A predicate is a function with a single argument and returns boolean value.
- To implement predicate functions in Java, Oracle people introduced Predicate interface in 1.8 version (i.e.,Predicate<T>).
- Predicate interface present in *Java.util.function* package.
- It's a functional interface and it contains only one method i.e., test()

Ex:

```
interface Predicate<T> {
    public boolean test(T t);
}
```

As predicate is a functional interface and hence it can refers lambda expression

Ex:1 Write a predicate to check whether the given integer is greater than 10 or not.

Ex:

```
public boolean test(Integer I) {
    if (I >10) {
        return true;
    }
    else {
        return false;
    }
}
```

```
(Integer I) → {
    if(I > 10)
        return true;
    else
        return false;
}
```

I → (I>10);

predicate<Integer> p = I →(I >10);
System.out.println (p.test(100)); true
System.out.println (p.test(7)); false

Program:

```
1)  import Java.util.function;
2)  class Test {
3)        public static void main(String[] args) {
4)            predicate<Integer> p = I → (i>10);
5)            System.out.println(p.test(100));
6)            System.out.println(p.test(7));
7)            System.out.println(p.test(true)); //CE
8)        }
9)  }
```

1 Write a predicate to check the length of given string is greater than 3 or not.
Predicate<String> p = s → (s.length() > 3);
System.out.println (p.test("rvkb")); true
System.out.println (p.test("rk")); false

#-2 write a predicate to check whether the given collection is empty or not.
Predicate<collection> p = c → c.isEmpty();

Predicate joining

It's possible to join predicates into a single predicate by using the following methods.
- and()
- or()
- negate()

these are exactly same as logical AND ,OR complement operators

Ex:

```
1)  import Java.util.function.*;
2)  class test {
3)        public static void main(string[] args) {
4)            int[] x = {0, 5, 10, 15, 20, 25, 30};
5)            predicate<integer> p1 = i->i>10;
6)            predicate<integer> p2=i -> i%2==0;
7)            System.out.println("The Numbers Greater Than 10:");
8)            m1(p1, x);
9)            System.out.println("The Even Numbers Are:");
10)           m1(p2, x);
11)           System.out.println("The Numbers Not Greater Than 10:");
```


```
12)             m1(p1.negate(), x);
13)             System.out.println("The  Numbers Greater Than 10 And Even Are:â€  );
14)             m1(p1.and(p2), x);
15)             System.out.println("The Numbers Greater Than 10 OR Even:â€  );
16)             m1(p1.or(p2), x);
17)       }
18)        public static void m1(predicate<integer>p, int[] x) {
19)              for(int x1:x) {
20)                   if(p.test(x1))
21)                        System.out.println(x1);
22)                   }
23)              }
24) }
```