



Java 8 New  
Features in  
Simple Way



---

**Predefined Functional Interface**

# **Predicate**

# **Study Material**



# 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 refer 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
```



## Java 8 New Features in Simple Way



### 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€œ");  
14)        m1(p1.and(p2), x);  
15)        System.out.println("The Numbers Greater Than 10 OR Even:â€œ");  
16)        m1(p1.or(p2), x);  
17)    }
```



## Java 8 New Features in Simple Way



---

```
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) }
```



## Program to display names starts with 'K' by using Predicate:

```
1) import java.util.function.Predicate;
2) class Test
3) {
4)     public static void main(String[] args)
5)     {
6)         String[] names={"Sunny","Kajal","Mallika","Katrina","Kareena"};
7)         Predicate<String> startsWithK=s->s.charAt(0)=='K';
8)         System.out.println("The Names starts with K are:");
9)         for(String s: names)
10)        {
11)            if(startsWithK.test(s))
12)            {
13)                System.out.println(s);
14)            }
15)        }
16)    }
17) }
```

### Output:

The Names starts with K are:  
Kajal  
Katrina  
Kareena

## Predicate Example to remove null values and empty strings from the given list:

```
1) import java.util.ArrayList;
2) import java.util.function.Predicate;
3) class Test
4) {
5)     public static void main(String[] args)
6)     {
7)         String[] names={"Durga","",null,"Ravi","", "Shiva",null};
8)         Predicate<String> p=s-> s != null && s.length()!=0;
9)         ArrayList<String> list=new ArrayList<String>();
10)        for(String s : names)
11)        {
12)            if(p.test(s))
13)            {
14)                list.add(s);
15)            }
16)        }
17)        System.out.println("The List of valid Names:");
```



## Java 8 New Features in Simple Way



```
18)    System.out.println(list);
19)    }
20) }
```

### Output:

The List of valid Names:

[Durga, Ravi, Shiva]

### Program for User Authentication by using Predicate:

```
1) import java.util.function.Predicate;
2) import java.util.Scanner;
3) class User
4) {
5)     String username;
6)     String pwd;
7)     User(String username,String pwd)
8)     {
9)         this.username=username;
10)        this.pwd=pwd;
11)    }
12) }
13) class Test
14) {
15)     public static void main(String[] args)
16)     {
17)         Predicate<User> p = u->u.username.equals("durga")&& u.pwd.equals("java");
18)         Scanner sc= new Scanner(System.in);
19)         System.out.println("Enter User Name:");
20)         String username=sc.next();
21)         System.out.println("Enter Password:");
22)         String pwd=sc.next();
23)         User user=new User(username,pwd);
24)         if(p.test(user))
25)         {
26)             System.out.println("Valid user and can avail all services");
27)         }
28)         else
29)         {
30)             System.out.println("invalid user you cannot avail services");
31)         }
32)     }
33) }
```

D:\durgaclasses>java Test

Enter User Name:

durga



## Java 8 New Features in Simple Way



Enter Password:

java

Valid user and can avail all services

D:\durgaclasses>java Test

Enter User Name:

ravi

Enter Password:

java

invalid user you cannot avail services

### Program to check whether SoftwareEngineer is allowed into pub or not by using Predicate?

```
1) import java.util.function.Predicate;
2) class SoftwareEngineer
3) {
4)     String name;
5)     int age;
6)     boolean isHavingGf;
7)     SoftwareEngineer(String name,int age,boolean isHavingGf)
8)     {
9)         this.name=name;
10)        this.age=age;
11)        this.isHavingGf=isHavingGf;
12)    }
13)    public String toString()
14)    {
15)        return name;
16)    }
17) }
18) class Test
19) {
20)     public static void main(String[] args)
21)     {
22)         SoftwareEngineer[] list={ new SoftwareEngineer("Durga",60,false),
23)                                     new SoftwareEngineer("Sunil",25,true),
24)                                     new SoftwareEngineer("Sayan",26,true),
25)                                     new SoftwareEngineer("Subbu",28,false),
26)                                     new SoftwareEngineer("Ravi",17,true)
27)     };
28)     Predicate<SoftwareEngineer> allowed= se -> se.age>= 18 && se.isHavingGf;
29)     System.out.println("The Allowed Members into Pub are:");
30)     for(SoftwareEngineer se : list)
31)     {
32)         if(allowed.test(se))
33)         {
```



## Java 8 New Features in Simple Way



```
34)      System.out.println(se);
35)    }
36)  }
37) }
38) }
```

### Output:

The allowed members into Pub are:

Sunil

Sayan

### Employee Management Application:

```
1) import java.util.function.Predicate;
2) import java.util.ArrayList;
3) class Employee
4) {
5)     String name;
6)     String designation;
7)     double salary;
8)     String city;
9)     Employee(String name,String designation,double salary,String city)
10)    {
11)        this.name=name;
12)        this.designation=designation;
13)        this.salary=salary;
14)        this.city=city;
15)    }
16)    public String toString()
17)    {
18)        String s=String.format("[%s,%s,%.2f,%s]",name,designation,salary,city);
19)        return s;
20)    }
21)    public boolean equals(Object obj)
22)    {
23)        Employee e=(Employee)obj;
24)        if(name.equals(e.name)&&designation.equals(e.designation)&&salary==e.salary && c
ity==e.city)
25)        {
26)            return true;
27)        }
28)        else
29)        {
30)            return false;
31)        }
32)    }
33) }
```





## Java 8 New Features in Simple Way



```
34) class Test
35) {
36)     public static void main(String[] args)
37)     {
38)         ArrayList<Employee> list= new ArrayList<Employee>();
39)         populate(list);
40)
41)         Predicate<Employee> p1=emp->emp.designation.equals("Manager");
42)         System.out.println("Managers Information:");
43)         display(p1,list);
44)
45)         Predicate<Employee> p2=emp->emp.city.equals("Bangalore");
46)         System.out.println("Bangalore Employees Information:");
47)         display(p2,list);
48)
49)         Predicate<Employee> p3=emp->emp.salary<20000;
50)         System.out.println("Employees whose slaray <20000 To Give Increment:");
51)         display(p3,list);
52)
53)         System.out.println("All Managers from Bangalore city for Pink Slip:");
54)         display(p1.and(p2),list);
55)
56)         System.out.println("Employees Information who are either Managers or salary <2000
57)         0");
58)         display(p1.or(p3),list);
59)
60)         System.out.println("All Employees Information who are not managers:");
61)         display(p1.negate(),list);
62)
63)         Predicate<Employee> isCEO=Predicate.isEqual(new Employee("Durga","CEO",30000,"
64)         Hyderabad"));
65)         Employee e1=new Employee("Durga","CEO",30000,"Hyderabad");
66)         Employee e2=new Employee("Sunny","Manager",20000,"Hyderabad");
67)         System.out.println(isCEO.test(e1));//true
68)         System.out.println(isCEO.test(e2));//false
69)     }
70)     public static void populate(ArrayList<Employee> list)
71)     {
72)         list.add(new Employee("Durga","CEO",30000,"Hyderabad"));
73)         list.add(new Employee("Sunny","Manager",20000,"Hyderabad"));
74)         list.add(new Employee("Mallika","Manager",20000,"Bangalore"));
75)         list.add(new Employee("Kareena","Lead",15000,"Hyderabad"));
76)         list.add(new Employee("Katrina","Lead",15000,"Bangalore"));
77)         list.add(new Employee("Anushka","Developer",10000,"Hyderabad"));
78)         list.add(new Employee("Kanushka","Developer",10000,"Hyderabad"));
79)         list.add(new Employee("Sowmya","Developer",10000,"Bangalore"));
```



## Java 8 New Features in Simple Way



```
80) list.add(new Employee("Ramya","Developer",10000,"Bangalore"));
81) }
82) public static void display(Predicate<Employee> p,ArrayList<Employee> list)
83) {
84)     for (Employee e: list )
85)     {
86)         if(p.test(e))
87)         {
88)             System.out.println(e);
89)         }
90)     }
91)     System.out.println("*****");
92) }
93) }
```

### Output:

Managers Information:

[Sunny,Manager,20000.00,Hyderabad]

[Mallika,Manager,20000.00,Bangalore]

\*\*\*\*\*

Bangalore Employees Information:

[Mallika,Manager,20000.00,Bangalore]

[Katrina,Lead,15000.00,Bangalore]

[Sowmya,Developer,10000.00,Bangalore]

[Ramya,Developer,10000.00,Bangalore]

\*\*\*\*\*

Employees whose salary <20000 To Give Increment:

[Kareena,Lead,15000.00,Hyderabad]

[Katrina,Lead,15000.00,Bangalore]

[Anushka,Developer,10000.00,Hyderabad]

[Kanushka,Developer,10000.00,Hyderabad]

[Sowmya,Developer,10000.00,Bangalore]

[Ramya,Developer,10000.00,Bangalore]

\*\*\*\*\*

All Managers from Bangalore city for Pink Slip:

[Mallika,Manager,20000.00,Bangalore]

\*\*\*\*\*

Employees Information who are either Managers or salary <20000

[Sunny,Manager,20000.00,Hyderabad]

[Mallika,Manager,20000.00,Bangalore]

[Kareena,Lead,15000.00,Hyderabad]

[Katrina,Lead,15000.00,Bangalore]

[Anushka,Developer,10000.00,Hyderabad]

[Kanushka,Developer,10000.00,Hyderabad]

[Sowmya,Developer,10000.00,Bangalore]

[Ramya,Developer,10000.00,Bangalore]

\*\*\*\*\*



## Java 8 New Features in Simple Way



---

All Employees Information who are not managers:

[Durga,CEO,30000.00,Hyderabad]

[Kareena,Lead,15000.00,Hyderabad]

[Katrina,Lead,15000.00,Bangalore]

[Anushka,Developer,10000.00,Hyderabad]

[Kanushka,Developer,10000.00,Hyderabad]

[Sowmya,Developer,10000.00,Bangalore]

[Ramya,Developer,10000.00,Bangalore]

\*\*\*\*\*

true

false



# Predicate Practice Bits

**Q1. Which of the following abstract method present in Predicate interface?**

- A. test()
- B. apply()
- C. get()
- D. accept()

Answer: A

**Explanation:** Predicate functional interface contains only one abstract method: test()

**Q2. Which of the following is the static method present in Predicate interface?**

- A. test()
- B. and()
- C. or()
- D. isEqual()

Answer: D

**Explanation:** Predicate functional interface contains only one static method: isEqual()

**Q3. Which of the following default methods present in Predicate interface?**

- A. and()
- B. or()
- C. negate()
- D. All the above

Answer: D

**Explanation:** Predicate Functional interface contains the following 3 default methods:  
and(),or(),not()



---

**Q4. Which of the following is Predicate interface declaration?**

A.

```
1) interface Predicate<T>
2) {
3)     public boolean test(T t);
4) }
```

B.

```
1) interface Predicate<T>
2) {
3)     public boolean apply(T t);
4) }
```

C.

```
1) interface Predicate<T,R>
2) {
3)     public R test(T t);
4) }
```

D.

```
1) interface Predicate<T,R>
2) {
3)     public R apply(T t);
4) }
```

Answer: A

**Explanation:**

```
1) interface Predicate<T>
2) {
3)     public boolean test(T t);
4) }
```

Predicate interface can take only one Type parameter which represents only input type. We are not required to specify return type because return type is always boolean type.



---

**Q5. Which of the following is valid Predicate to check whether the given Integer is divisible by 10 or not?**

- A. `Predicate<Integer> p = i -> i%10 == 10;`
- B. `Predicate<Integer, Boolean> p = i -> i%10 == 0;`
- C. `Predicate<Boolean, Integer> p = i -> i%10 == 0;`
- D. None of the above

Answer: A

**Explanation:**

```
1) interface Predicate<T>
2) {
3)     public boolean test(T t);
4) }
```

Predicate interface can take only one Type parameter which represents only input type. We are not required to specify return type because return type is always boolean type.

**Q6. Which of the following is valid regarding Predicate functional interface?**

- A. Predicate Functional interface present in `java.util.function` package
- B. It is introduced in java 1.8 version
- C. We can use Predicate to implement conditional checks
- D. It is possible to join 2 predicates into a single predicate also.
- E. All the above

Answer: E

**Q7. Which of the following is valid Predicate to check whether the given user is admin or not?**

- A. `Predicate<User> p = user -> user.getRole().equals("Admin");`
- B. `Predicate<Boolean> p = user -> user.getRole().equals("Admin");`
- C. `Predicate<User> p = (user, s = "admin") -> user.getRole().equals(s);`
- D. None of the above

Answer: A



### Explanation:

```
1) interface Predicate<T>
2) {
3)     public boolean test(T t);
4) }
```

Predicate interface can take only one Type parameter which represents only input type. We are not required to specify return type because return type is always boolean type.

### Q8. Consider the following Predicates

`Predicate<Integer> p1=i->i%2==0;`

`Predicate<Integer> p1=i->i>10;`

### Which of the following are invalid ?

- A. `p1.and(p2)`
- B. `p1.or(p2)`
- C. `p1.negate(p2)`
- D. `p1.negate()`

Answer: C

Explanation: `negate()` method won't take any argument