



Java 8 New
Features in
Simple Way



Predefined Functional Interface

Predicate

Study Material



Functions

- Functions are exactly same as predicates except that functions can return any type of result but function should (can) return only one value and that value can be any type as per our requirement.
- To implement functions oracle people introduced Function interface in 1.8 version.
- Function interface present in *Java.util.function* package.
- Functional interface contains only one method i.e., `apply()`

```
1) interface function(T,R) {  
2)     public R apply(T t);  
3) }
```

Assignment: Write a function to find length of given input string.

Ex:

```
1) import Java.util.function.*;  
2) class Test {  
3)     public static void main(String[] args) {  
4)         Function<String, Integer> f = s -> s.length();  
5)         System.out.println(f.apply("Durga"));  
6)         System.out.println(f.apply("Soft"));  
7)     }  
8) }
```

Note: Function is a functional interface and hence it can refer lambda expression.

```
1) import Java.util.function.*;  
2) class Test {  
3)     public static void main(String[] args) {  
4)         Function<Integer, Integer> f = s -> s*s;  
5)         System.out.println(f.apply(5));  
6)         System.out.println(f.apply(10));  
7)     }  
8) }
```



Differences between predicate and function

Predicate	Function
To implement conditional checks We should go for predicate	To perform certain operation And to return some result we Should go for function.
Predicate can take one type Parameter which represents Input argument type. Predicate<T>	Function can take 2 type Parameters. First one represent Input argument type and Second one represent return Type. Function<T,R>
Predicate interface defines only one method called test()	Function interface defines only one Method called apply().
public boolean test(T t)	public R apply(T t)
Predicate can return only boolean value.	Function can return any type of value

Note: Predicate is a boolean valued function and(), or(), negate() are default methods present inside Predicate interface.



Program to remove spaces present in the given String by using Function:

```
1) import java.util.function.*;
2) class Test
3) {
4)     public static void main(String[] args)
5)     {
6)         String s="durga software solutions hyderabad";
7)         Function<String,String> f= s1->s1.replaceAll(" ","");
8)         System.out.println(f.apply(s));
9)     }
10) }
```

Output: durgasoftwaresolutionshyderabad

Program to find Number of spaces present in the given String by using Function:

```
1) import java.util.function.*;
2) class Test
3) {
4)     public static void main(String[] args)
5)     {
6)         String s="durga software solutions hyderabad ";
7)         Function<String,Integer> f= s1->s1.length() - s1.replaceAll(" ","").length();
8)         System.out.println(f.apply(s));
9)     }
10) }
```

Output: 3

Program to find Student Grade by using Function:

```
1) import java.util.function.*;
2) import java.util.*;
3) class Student
4) {
5)     String name;
6)     int marks;
7)     Student(String name,int marks)
8)     {
9)         this.name=name;
10)        this.marks=marks;
11)    }
12) }
```



Java 8 New Features in Simple Way



```
13) class Test
14) {
15)     public static void main(String[] args)
16)     {
17)         ArrayList<Student> l= new ArrayList<Student>();
18)         populate(l);
19)         Function<Student,String> f=s->{
20)             int marks=s.marks;
21)             if(marks>=80)
22)             {
23)                 return "A[Dictinction]";
24)             }
25)             else if(marks>=60)
26)             {
27)                 return "B[First Class]";
28)             }
29)             else if(marks>=50)
30)             {
31)                 return "C[Second Class]";
32)             }
33)             else if(marks>=35)
34)             {
35)                 return "D[Third Class]";
36)             }
37)             else
38)             {
39)                 return "E[Failed]";
40)             }
41)         };
42)         for(Student s : l)
43)         {
44)             System.out.println("Student Name:"+s.name);
45)             System.out.println("Student Marks:"+s.marks);
46)             System.out.println("Student Grade:"+f.apply(s));
47)             System.out.println();
48)         }
49)     }
50)     public static void populate(ArrayList<Student> l)
51)     {
52)         l.add(new Student("Sunny",100));
53)         l.add(new Student("Bunny",65));
54)         l.add(new Student("Chinny",55));
55)         l.add(new Student("Vinny",45));
56)         l.add(new Student("Pinny",25));
57)     }
58) }
```



Java 8 New Features in Simple Way



Output:

D:\durgaclass>java Test
Student Name:Sunny
Student Marks:100
Student Grade:A[Dictinction]

Student Name:Bunny
Student Marks:65
Student Grade:B[First Class]

Student Name:Chinny
Student Marks:55
Student Grade:C[Second Class]

Student Name:Vinny
Student Marks:45
Student Grade:D[Third Class]

Student Name:Pinny
Student Marks:25
Student Grade:E[Failed]

Program to find Students Information including Grade by using Function whose marks are >=60:

```
1) import java.util.function.*;  
2) import java.util.*;  
3) class Student  
4) {  
5)     String name;  
6)     int marks;  
7)     Student(String name,int marks)  
8)     {  
9)         this.name=name;  
10)        this.marks=marks;  
11)    }  
12) }  
13) class Test  
14) {  
15)     public static void main(String[] args)  
16)     {  
17)         ArrayList<Student> l= new ArrayList<Student>();  
18)         populate(l);  
19)         Function<Student,String> f=s->{  
20)             int marks=s.marks;  
21)             if(marks>=80)
```



Java 8 New Features in Simple Way



```
22)    {
23)        return "A[Dictinction]";
24)    }
25)    else if(marks>=60)
26)    {
27)        return "B[First Class]";
28)    }
29)    else if(marks>=50)
30)    {
31)        return "C[Second Class]";
32)    }
33)    else if(marks>=35)
34)    {
35)        return "D[Third Class]";
36)    }
37)    else
38)    {
39)        return "E[Failed]";
40)    }
41)    };
42)    Predicate<Student> p=s->s.marks>=60;
43)
44)    for(Student s : l)
45)    {
46)        if(p.test(s))
47)        {
48)            System.out.println("Student Name:"+s.name);
49)            System.out.println("Student Marks:"+s.marks);
50)            System.out.println("Student Grade:"+f.apply(s));
51)            System.out.println();
52)        }
53)    }
54) }
55) public static void populate(ArrayList<Student> l)
56) {
57)     l.add(new Student("Sunny",100));
58)     l.add(new Student("Bunny",65));
59)     l.add(new Student("Chinny",55));
60)     l.add(new Student("Vinny",45));
61)     l.add(new Student("Pinny",25));
62) }
63) }
```

Output:

D:\durgaclasses>java Test

Student Name:Sunny



Java 8 New Features in Simple Way



Student Marks:100

Student Grade:A[Dictinction]

Student Name:Bunny

Student Marks:65

Student Grade:B[First Class]

Program to find Total Monthly Salary of All Employees by using Function:

```
1) import java.util.*;
2) import java.util.function.*;
3) class Employee
4) {
5)     String name;
6)     double salary;
7)     Employee(String name,double salary)
8)     {
9)         this.name=name;
10)        this.salary=salary;
11)    }
12)    public String toString()
13)    {
14)        return name+":"+salary;
15)    }
16) }
17) class Test
18) {
19)     public static void main(String[] args)
20)     {
21)         ArrayList<Employee> l1= new ArrayList<Employee>();
22)         populate(l1);
23)         System.out.println(l1);
24)         Function<ArrayList<Employee>,Double> f= l1 ->{
25)             double total=0;
26)             for(Employee e: l1)
27)             {
28)                 total=total+e.salary;
29)             }
30)             return total;
31)         };
32)         System.out.println("The total salary of this month:"+f.apply(l1));
33)     }
34)
35)     public static void populate(ArrayList<Employee> l)
36)     {
37)         l.add(new Employee("Sunny",1000));
```




Java 8 New Features in Simple Way



```
38) l.add(new Employee("Bunny",2000));
39) l.add(new Employee("Chinny",3000));
40) l.add(new Employee("Pinny",4000));
41) l.add(new Employee("Vinny",5000));
42) }
43) }
```

Output:

D:\durgaclasses>java Test

[Sunny:1000.0, Bunny:2000.0, Chinny:3000.0, Pinny:4000.0, Vinny:5000.0]

The total salary of this month:15000.0

Program to perform Salary Increment for Employees by using Predicate & Function:

```
1) import java.util.*;
2) import java.util.function.*;
3) class Employee
4) {
5)     String name;
6)     double salary;
7)     Employee(String name,double salary)
8)     {
9)         this.name=name;
10)        this.salary=salary;
11)    }
12)    public String toString()
13)    {
14)        return name+":"+salary;
15)    }
16) }
17) class Test
18) {
19)     public static void main(String[] args)
20)     {
21)         ArrayList<Employee> l= new ArrayList<Employee>();
22)         populate(l);
23)
24)         System.out.println("Before Increment:");
25)         System.out.println(l);
26)
27)         Predicate<Employee> p=e->e.salary<3500;
28)         Function<Employee,Employee> f=e->{
29)             e.salary=e.salary+477;
30)             return e;
31)         };
32)
```



Java 8 New Features in Simple Way



```
33) System.out.println("After Increment:");
34) ArrayList<Employee> l2= new ArrayList<Employee>();
35) for(Employee e: l)
36) {
37)     if(p.test(e))
38)     {
39)         l2.add(f.apply(e));
40)     }
41) }
42) System.out.println(l);
43) System.out.println("Employees with incremented salary:");
44) System.out.println(l2);
45) }
46) public static void populate(ArrayList<Employee> l)
47) {
48)     l.add(new Employee("Sunny",1000));
49)     l.add(new Employee("Bunny",2000));
50)     l.add(new Employee("Chinny",3000));
51)     l.add(new Employee("Pinny",4000));
52)     l.add(new Employee("Vinny",5000));
53)     l.add(new Employee("Durga",10000));
54) }
55) }
```

Output:

Before Increment:

[Sunny:1000.0, Bunny:2000.0, Chinny:3000.0, Pinny:4000.0, Vinny:5000.0, Durga:10000.0]

After Increment:

[Sunny:1477.0, Bunny:2477.0, Chinny:3477.0, Pinny:4000.0, Vinny:5000.0, Durga:10000.0]

Employees with incremented salary:

[Sunny:1477.0, Bunny:2477.0, Chinny:3477.0]



Function Chaining:

We can combine multiple functions together to form more complex functions. For this Function interface defines the following 2 default methods:

f1.andThen(f2): First f1 will be applied and then for the result f2 will be applied.

f1.compose(f2): First f2 will be applied and then for the result f1 will be applied.

Demo Program-1 for Function Chaining:

```
1) import java.util.function.*;
2) class Test
3) {
4)     public static void main(String[] args)
5)     {
6)
7)         Function<String,String> f1=s->s.toUpperCase();
8)         Function<String,String> f2= s->s.substring(0,9);
9)
10)        System.out.println("The Result of f1:"+f1.apply("AishwaryaAbhi"));
11)        System.out.println("The Result of f2:"+f2.apply("AishwaryaAbhi"));
12)        System.out.println("The Result of f1.andThen(f2):"+f1.andThen(f2).apply("Aishwarya
    Abhi"));
13)        System.out.println("The Result of f1.compose(f2):"+f1.compose(f2).apply("Aishwarya
    Abhi"));
14)    }
15) }
```

Output:

The Result of f1:AISHWARYAABHI

The Result of f2:Aishwarya

The Result of f1.andThen(f2):AISHWARYA

The Result of f1.compose(f2):AISHWARYA

Demo program to Demonstrate the difference between andThen() and compose():

```
1) import java.util.function.*;
2) class Test
3) {
4)     public static void main(String[] args)
5)     {
6)         Function<Integer,Integer> f1= i->i+i;
7)         Function<Integer,Integer> f2= i->i*i*i;
8)         System.out.println(f1.andThen(f2).apply(2));
```



Java 8 New Features in Simple Way



```
9)      System.out.println(f1.compose(f2).apply(2));
10)    }
11) }
```

Output:

64

16

Demo Program for Function Chaining:

```
1) import java.util.function.*;
2) import java.util.*;
3) class Test
4) {
5)     public static void main(String[] args)
6)     {
7)
8)         Function<String,String> f1=s->s.toLowerCase();
9)         Function<String,String> f2= s->s.substring(0,5);
10)
11)         Scanner sc = new Scanner(System.in);
12)         System.out.println("Enter User Name:");
13)         String username=sc.next();
14)
15)         System.out.println("Enter Password:");
16)         String pwd=sc.next();
17)
18)         if(f1.andThen(f2).apply(username).equals("durga") && pwd.equals("java"))
19)         {
20)             System.out.println("Valid User");
21)         }
22)         else
23)         {
24)             System.out.println("Invalid User");
25)         }
26)     }
27) }
```

Output:

D:\durgaclasses>java Test

Enter User Name:

durga

Enter Password:

java

Valid User

D:\durgaclasses>java Test



Enter User Name:
durgasoftwaresolutions
Enter Password:
java
Valid User

D:\durgaclasses>java Test
Enter User Name:
DURGATECHNOLOGIES
Enter Password:
java
Valid User

D:\durgaclasses>java Test
Enter User Name:
javajava
Enter Password:
java
Invalid User

D:\durgaclasses>java Test
Enter User Name:
durga
Enter Password:
Java
Invalid User

Function interface Static Method : identity()

Function interface contains a static method.

`static <T> Function<T,T> identity()`

Returns a function that always returns its input argument.

Eg:

```
1) import java.util.function.*;  
2) class Test  
3) {  
4)     public static void main(String[] args)  
5)     {  
6)         Function<String,String> f1= Function.identity();  
7)         String s2= f1.apply("durga");  
8)         System.out.println(s2);  
9)     }  
10) }
```

Output: durga