

## **Predefined Functional Interface**

# **Supplier**

# Supplier (I)

Sometimes our requirement is we have to get some value based on some operation like

supply Student object  
Supply Random Name  
Supply Random OTP  
Supply Random Password  
etc

For this type of requirements we should go for Supplier.  
Supplier can be used to supply items (objects).

Supplier won't take any input and it will always supply objects.  
Supplier Functional Interface contains only one method get().

```
1) interface Supplier<R>
2) {
3)     public R get();
4) }
```

Supplier Functional interface does not contain any default and static methods.

## Demo Program-1 For Supplier to generate Random Name:

```
1) import java.util.function.Supplier;
2) class Test
3) {
4)     public static void main(String[] args)
5)     {
6)         Supplier<String> s =()->{
7)             String[] s1={"Sunny", "Bunny", "Chinny", "Pinny"};
8)             int x =(int)(Math.random()*4);
9)             return s1[x];
10)        };
11)        System.out.println(s.get());
12)        System.out.println(s.get());
13)        System.out.println(s.get());
14)    }
15) }
```

## Java 8 New Features in Simple Way

---

### Output:

```
D:\durgaclasses>java Test
Chinny
Bunny
Pinny
```

```
D:\durgaclasses>java Test
Bunny
Pinny
Bunny
```

### Demo Program-2 For Supplier to supply System Date:

```
1) import java.util.function.*;
2) import java.util.*;
3) class Test
4) {
5)     public static void main(String[] args)
6)     {
7)         Supplier<Date> s=()->new Date();
8)         System.out.println(s.get());
9)         System.out.println(s.get());
10)        System.out.println(s.get());
11)    }
12) }
```

```
D:\durgaclasses>java Test
Sun Aug 05 21:34:06 IST 2018
Sun Aug 05 21:34:06 IST 2018
Sun Aug 05 21:34:06 IST 2018
```

### Demo Program-3 For Supplier to supply 6-digit Random OTP:

```
1) import java.util.function.*;
2) import java.util.*;
3) class Test
4) {
5)     public static void main(String[] args)
6)     {
7)         Supplier<String> otps=()->{
8)             String otp="";
9)             for(int i =1;i<=6;i++)
10)            {
11)                otp=otp+(int)(Math.random()*10);
12)            }
13)            return otp;
14)        }
15) }
```

## Java 8 New Features in Simple Way

```
14)    };
15)    System.out.println(otps.get());
16)    System.out.println(otps.get());
17)    System.out.println(otps.get());
18)    System.out.println(otps.get());
19)    }
20) }
```

### Output:

492499  
055284  
531389  
925530

### Demo Program-4 For Supplier to supply Random Passwords:

#### Rules:

1. length should be 8 characters
2. 2,4,6,8 places only digits
3. 1,3,5,7 only Capital Uppercase characters,@,#,\$

```
1) import java.util.function.*;
2) import java.util.*;
3) class Test
4) {
5)     public static void main(String[] args)
6)     {
7)         Supplier<String> s={()->
8)         {
9)             String symbols="ABCDEFGHJKLMNOPQRSTUVWXYZ#$@";
10)            Supplier<Integer> d={()->(int)(Math.random()*10);
11)            Supplier<Character> c={()->symbols.charAt((int)(Math.random()*29));
12)            String pwd="";
13)            for(int i =1;i<=8;i++)
14)            {
15)                if(i%2==0)
16)                {
17)                    pwd=pwd+d.get();
18)                }
19)                else
20)                {
21)                    pwd=pwd+c.get();
22)                }
23)            }
24)            return pwd;
25)        };
26)        System.out.println(s.get());
```

## Java 8 New Features in Simple Way

```

27) System.out.println(s.get());
28) System.out.println(s.get());
29) System.out.println(s.get());
30) System.out.println(s.get());
31) }
32) }

```

### Output:

G2W3Y8W5  
W7M8\$0L3  
T5T5N2F3  
O9N2L0V2  
A4I1\$1P6

### Comparison Table of Predicate, Function, Consumer and Supplier

Property	Predicate	Function	Consumer	Supplier
1) Purpose	To take some Input and perform some conditional checks	To take some Input and perform required Operation and return the result	To consume some Input and perform required Operation. It won't return anything.	To supply some Value base on our Requirement.
2) Interface Declaration	<pre>interface Predicate &lt;T&gt; { ..... }</pre>	<pre>interface Function &lt;T, R&gt; { ..... }</pre>	<pre>interface Consumer &lt;T&gt; { ..... }</pre>	<pre>interface Supplier &lt;R&gt; { ..... }</pre>
3) Single Abstract Method (SAM)	public boolean test (T t);	public R apply (T t);	public void accept (T t);	public R get();
4) Default Methods	and(), or(), negate()	andThen(), compose()	andThen()	-
5) Static Method	isEqual()	identify()	-	-