**Java-8 – Supplier Interface**

* **Supplier Functional Interface:**

interface Supplier<R>{

R get();

}

R -> Return type.

Supplier won’t take any input, but it return something.

Supplier and Consumer are opposite interfaces.

Supplier interface doesn’t contain any default methods.

* **Program to get system date using supplier:**

import java.util.function.Supplier;

import java.util.Date;

class SupplierDemo{

public static void main(String[] args){

Supplier<Date> s = () -> new Date();

System.out.println(s.get());

}

}

* **Program to get random name using supplier:**

import java.util.function.Supplier;

class SupplierDemo{

public static void main(String[] args){

Supplier<String> s = () -> {

String[] s1 = {“Sunny”, “Bunny”, “Chinny”, “Pinny”};

int x = (int) Math.random()\*4);

return s1[x];

};

System.out.println(s.get());

System.out.println(s.get());

System.out.println(s.get());

}

}

* **Program to get random OTP using supplier:**

import java.util.function.Supplier;

class GetOTPDemo{

public static void main(String[] args){

Supplier<String> otps = () -> {

String otp = “”;

for(int i=0;i<=6;1++){

Otp = otp + (int) (Math.random()\*10);

}

return otp;

};

System.out.println(otps.get());

}

}

* **Program to get random password using Supplier:**

class PasswordGenerator{

public static void main(String[] args){

Supplier<String> s = () -> {

Supplier<Integer> d = () -> (int)(Math.random()\*10);

String symbols = “ABCDEFGHIJKLMNOPQRSTUVWXYZ#$@”;

Supplier<Character> c = () -> symbols.chartAt((int)(Math.random()\*29);

String pwd = “”;

for(int i=1;i<=8;i++){

if(i%2 == 0){

pwd = pwd + d.get();

} else{

pwd = pwd + c.get();

}

}

return pwd;

};

System.out.println(s.get());

}

}

* **Comparison table Predicate, Function, Consumer & Supplier:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property** | **Predicate** | **Function** | **Consumer** | **Supplier** |
| Purpose | To take some input and perform some conditional check. | 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 based on our requirement. |
| Interface declaration | interface Predicate<T> {  } | interface Function<T,R>{  } | interface Consumer<T> {  } | interface Supplier<R> {  } |
| Single Abstract Method (SAM) | public boolean test(T t); | public R apply(T t); | public void accept(T t); | public R get(); |
| Default methods | and(), or(), negate() | andThen(), compose() | andThen() | NA |
| Static methods | isEqual() | identity(); | NA | NA |

* **This document contains the content of 53 to 57.**