Lecture 52 Supplier Introduction:

**public** **interface** Supplier<T> {

/\*\*

\* Gets a result.

\*

\* **@return** a result

\*/

T get();

}

Lecture 53: Program to get System Date using System date:

**package** com.durgaSoft.section7.lecture52;

**import** java.util.Date;

**import** java.util.function.Supplier;

**public** **class** Example1 {

**public** **static** **void** main(String[] args) {

Supplier<Date> getSystemDateAndTime = () -> **new** Date();

System.***out***.println(getSystemDateAndTime.get());

}

}

Lecture 54: Program to generate Random number:

**package** com.durgaSoft.section7.lecture52;

**import** java.util.Random;

**import** java.util.function.Supplier;

**import** javax.xml.stream.events.Namespace;

**public** **class** Example2 {

**public** **static** **void** main(String[] args) {

Random random = **new** Random();

String[] names = { "Sunny", "Bunny", "Chinny", "Vinny" };

Supplier<String> getRandomName = () -> {

**return** names[random.nextInt(4)];

};

System.***out***.println(getRandomName.get());

}

}

Lecture 55:

Write a function to generate 6 digit OTP.

**package** com.durgaSoft.section7.lecture52;

**import** java.util.Random;

**import** java.util.function.Supplier;

**public** **class** Example3 {

**public** **static** **void** main(String[] args) {

Random random = **new** Random();

Supplier<Integer> generteOTP = () -> random.nextInt(1000000);

System.***out***.println(generteOTP.get());

}

}

Lecture 56:

Logic to generate Random Password:

* Length should be 8 Characters only
* 2,4,6,8 places only digits are allowed
* 1,3,5,7 place only upper case Alphabets or special characters.
* Special characters allowed are @,#,$.

**package** com.durgaSoft.section7.lecture52;

**import** java.util.Random;

**import** java.util.function.Function;

**import** java.util.function.Supplier;

/\*

\* Logic to generate Random Password:

\*

\* 1) Length should be 8 Characters only

\* 2) 2,4,6,8 places only digits are allowed

\* 3) 1,3,5,7 place only upper case Alphabets or special characters.

\* Special characters allowed are @,#,$.

\*/

**public** **class** Example4 {

**public** **static** **void** main(String[] args) {

Random random = **new** Random();

String[] symbols = {"@","#","$"};

Supplier<Integer> getDigits = () -> random.nextInt(10);

Supplier<Character> getCharacter = () -> (**char**)(random.nextInt(26)+65);

Supplier<String> getSymbol =() -> symbols[random.nextInt(symbols.length)];

Supplier<String> generatePassword = () ->{

String password ="";

**for**(**int** i=0;i<8;i++) {

**if**(i%2==0) {

password =password+getDigits.get();

}**else** {

**if**(random.nextBoolean()) {

password = password+getCharacter.get();

}**else** {

password = password+getSymbol.get();

}

}

}

**return** password;

};

System.***out***.println("Generated Password is "+generatePassword.get());

}

}

Lecture 57: Comparison Table Predicate, Function, Consumer and Supplier.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SI | 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 consumer some input and perform required operation.   It won’t return anything. | To Supply some value based on our requirement. |
|  |  |  |  |  |  |
| 2 | Interface Declaration | Interface Predicate<T>{  … } | interface Function<T,R>{ … } | Interface Consumer<T>{ .. } | Interface Supplier<R>{  …  } |
|  |  |  |  |  |  |
| 3 | Single Abstract method | public boolean test(T t) | public R apply(T t) | public void accept(T t) | public R get(T t) |
|  |  |  |  |  |  |
| 4 | Default methods | and(), or(), negate() | andThen(), and compose() | andThen() | ---- |
|  |  |  |  |  |  |
| 5 | Static method | isEqual() | Identity | --- | --- |