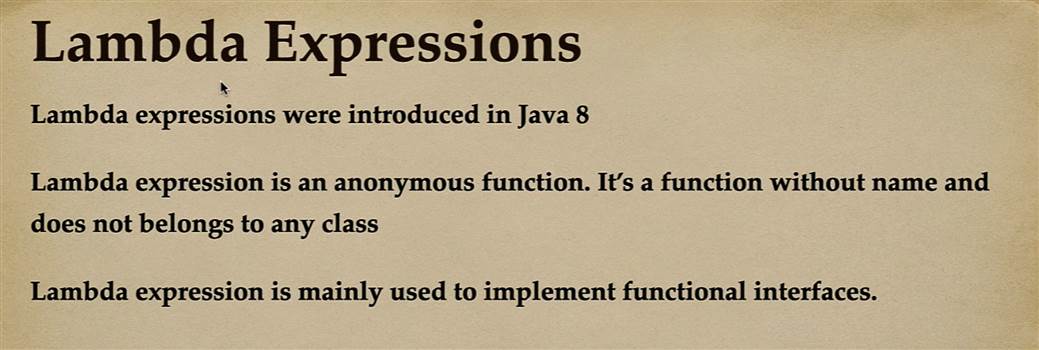
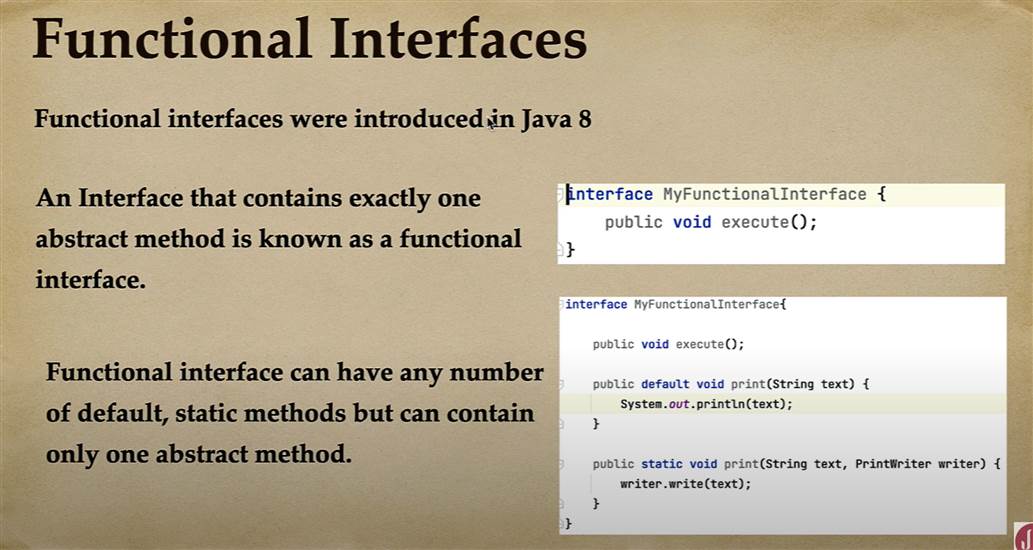
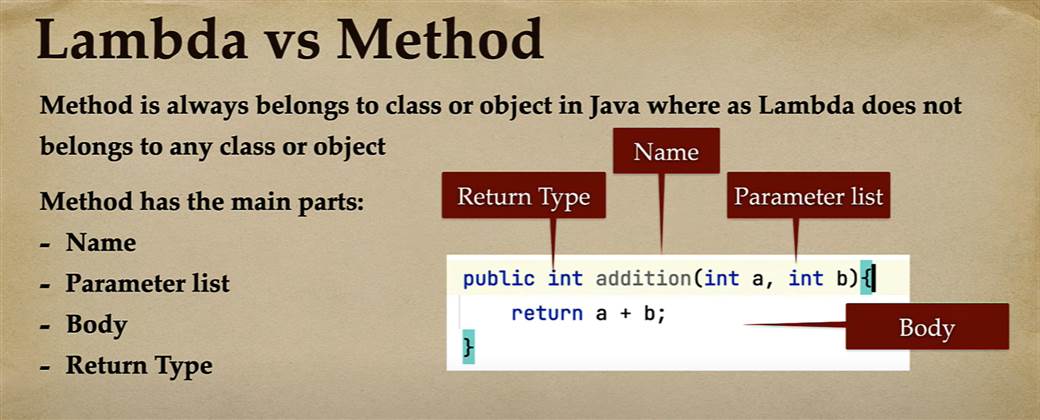
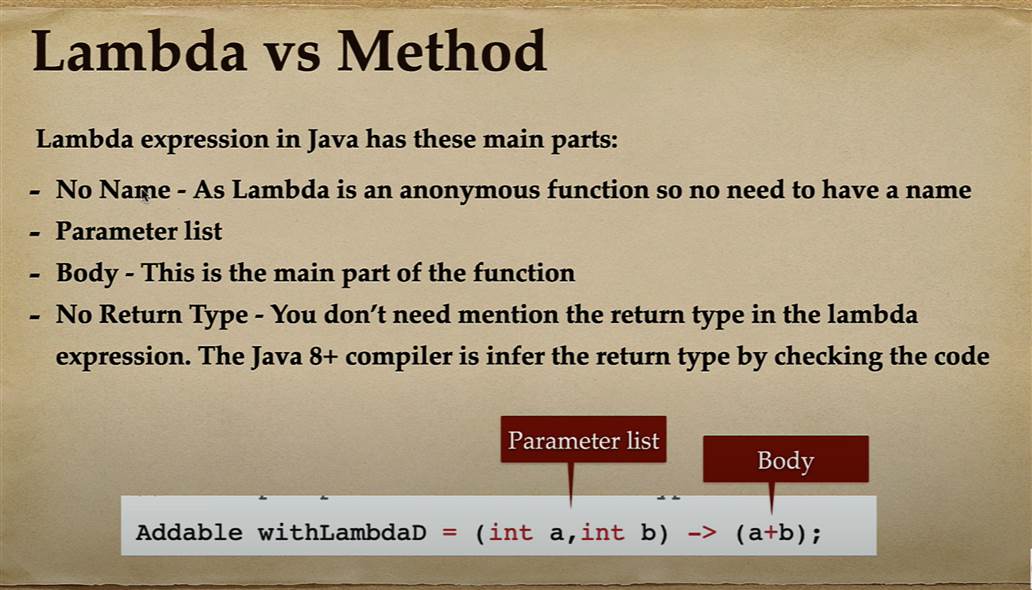
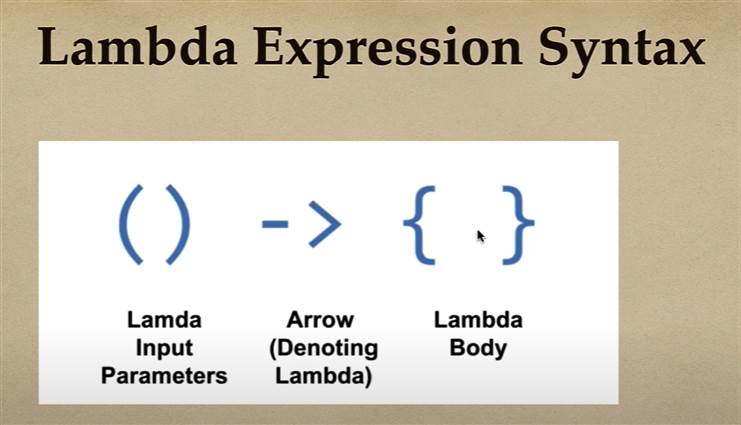
Chapter one : lambda expression

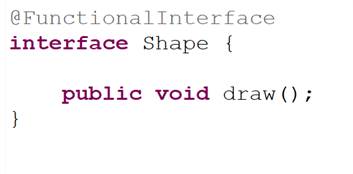


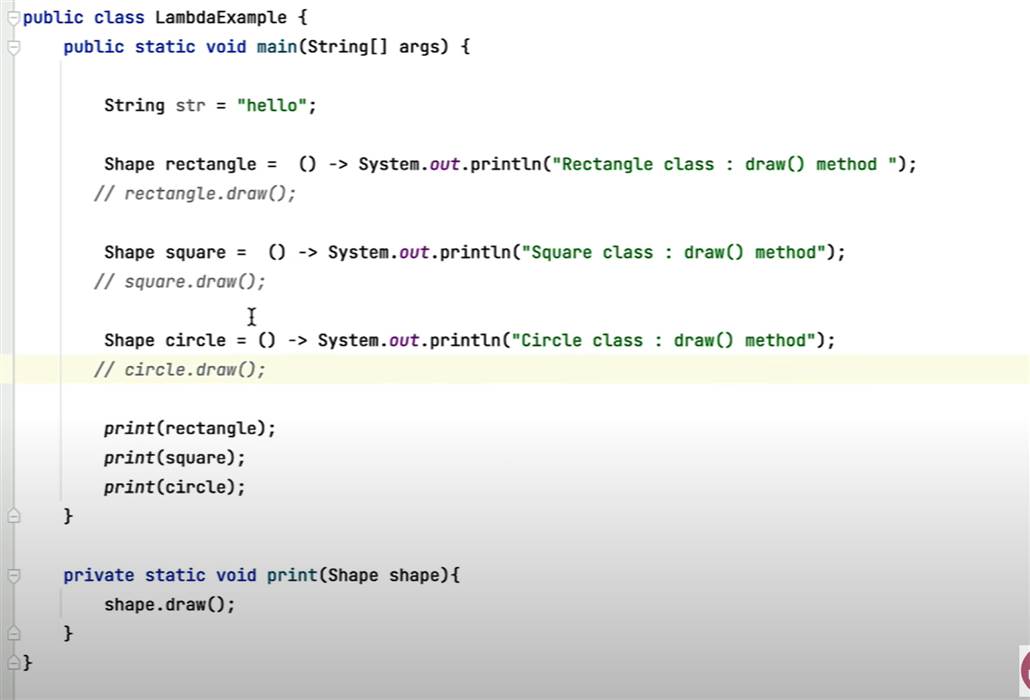




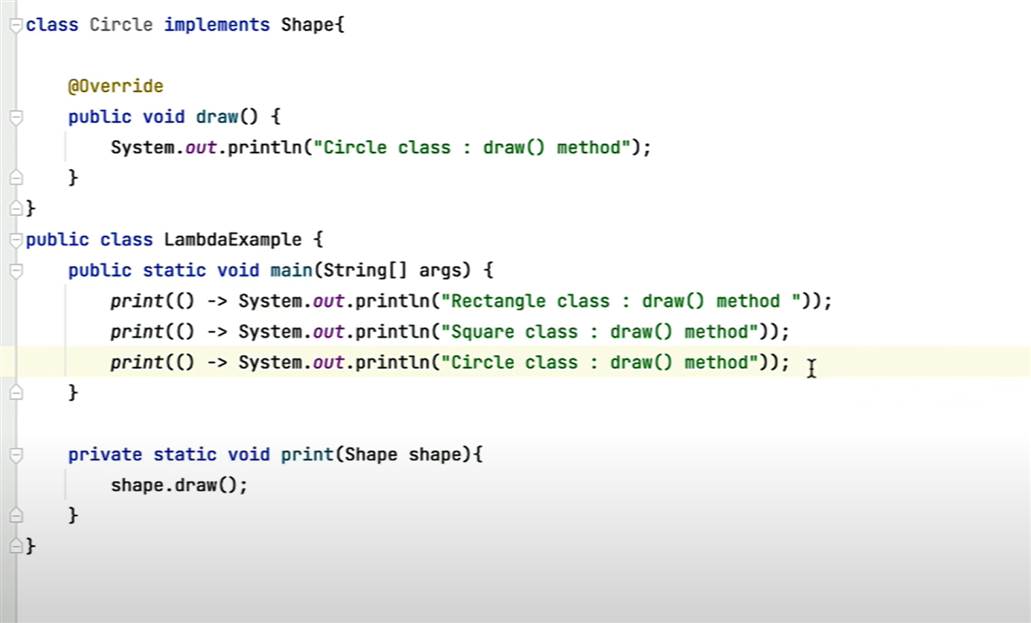


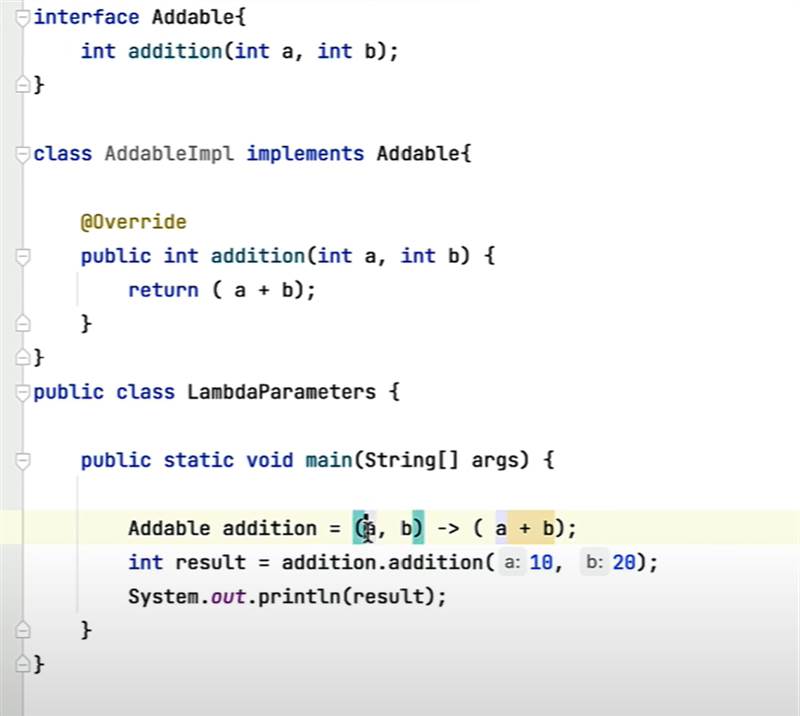




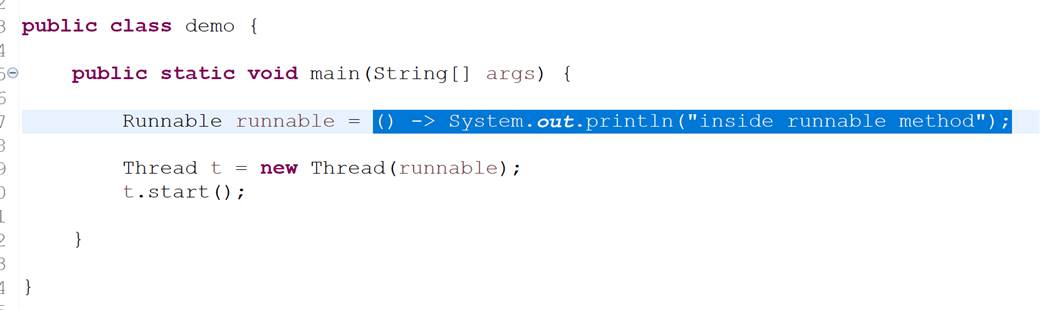


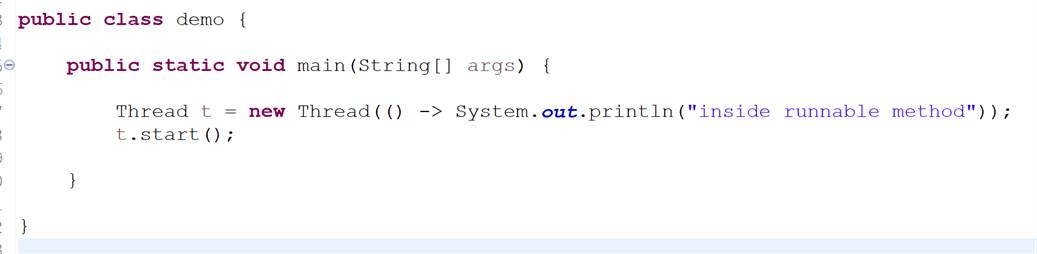
Or we can pass lambda expresssion directly into print method.





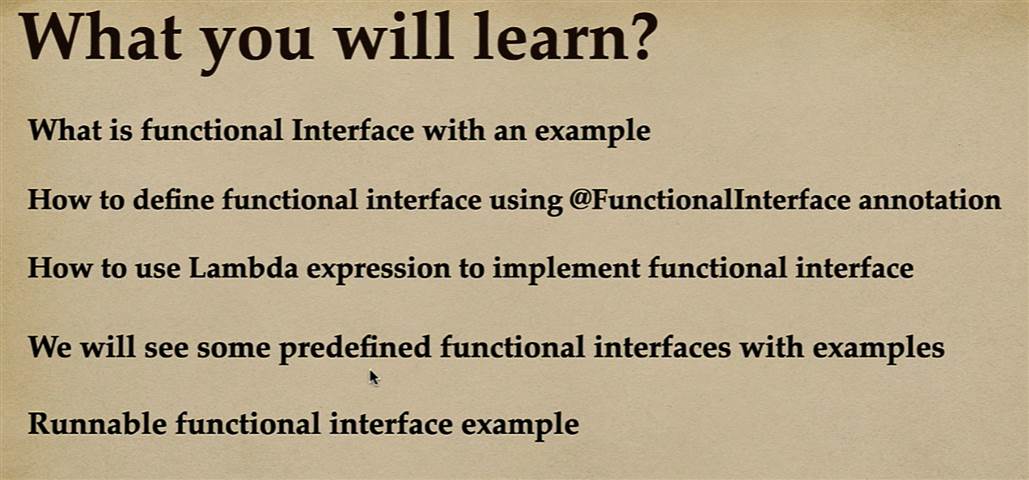
Implement Runnable using lambda expression;

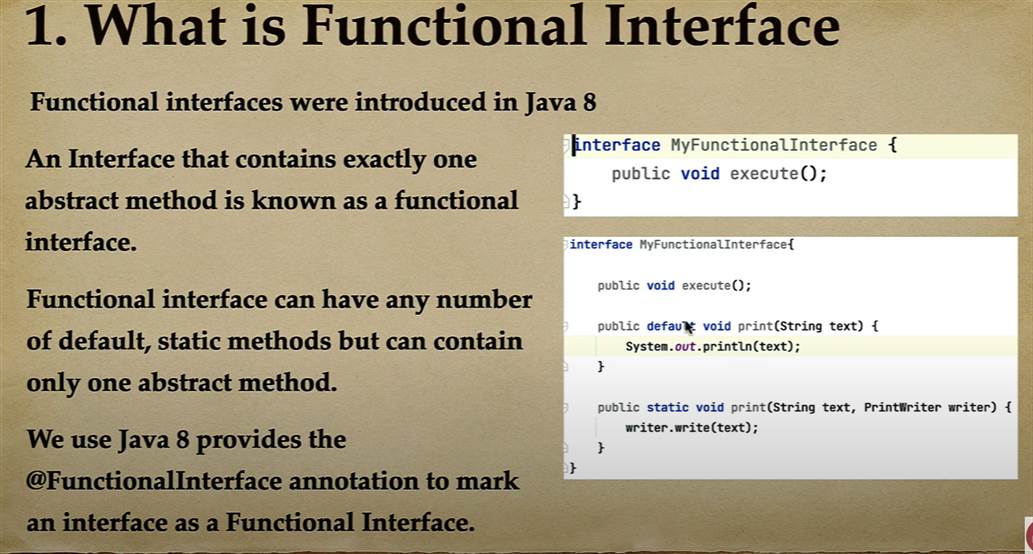


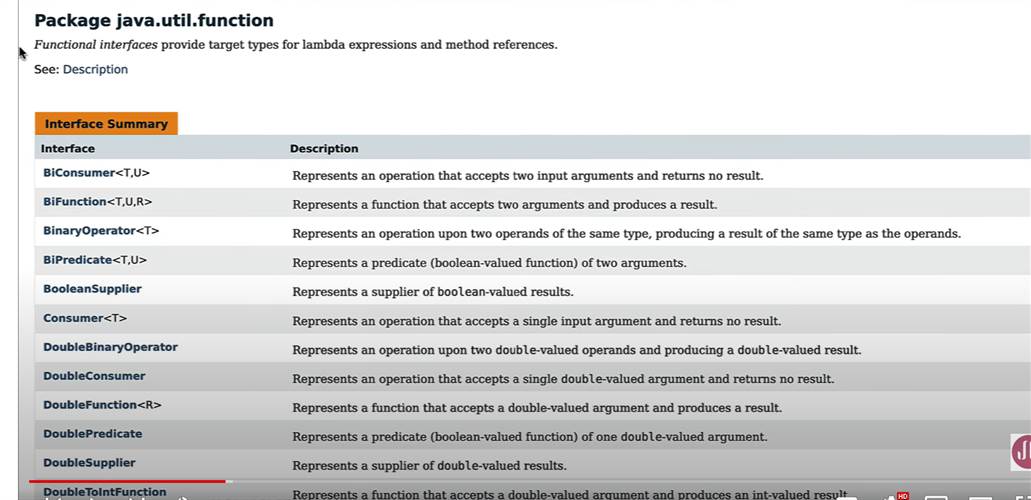


This is how we can pass lambda expression as a parameter to thread constructor.

Chapter two : functional interface





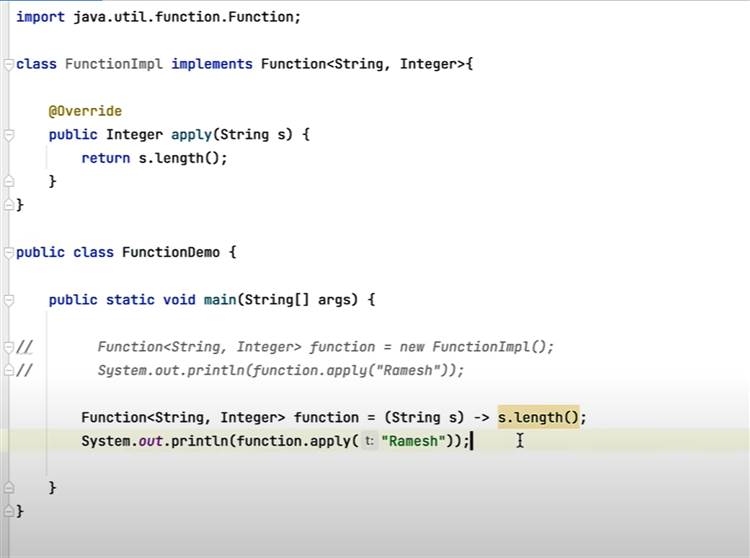


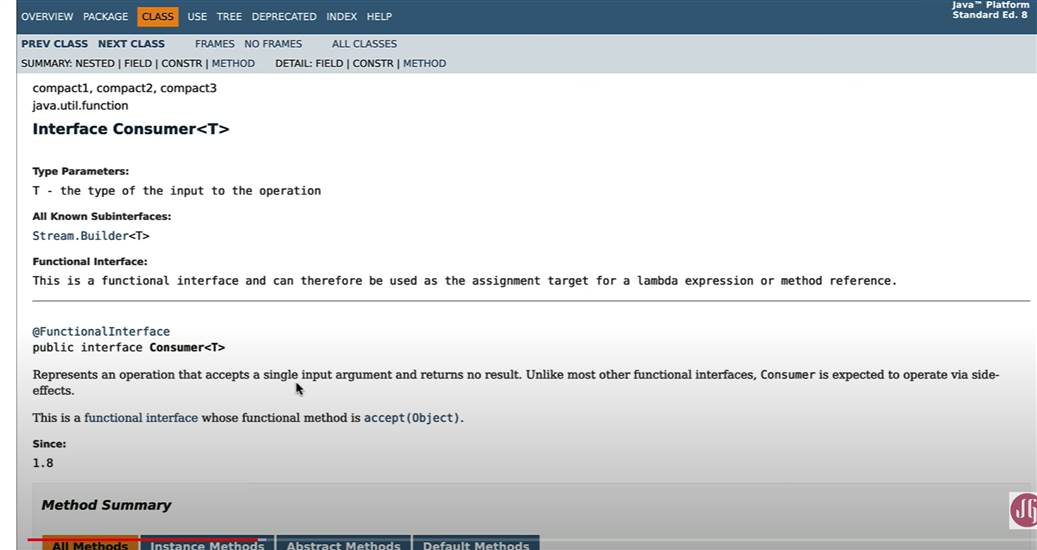
Take a input and return as output

Public interface Function<T,R>{

R apply(T t);

}





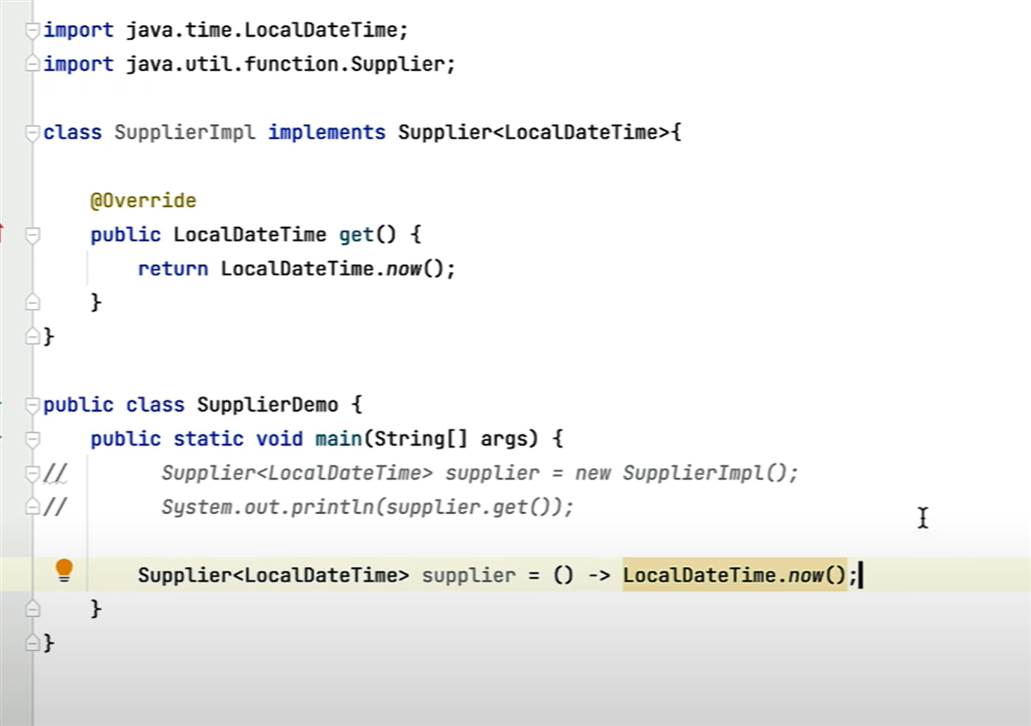


**Supplier :** basically itsupply some result to the client.

Interface Supplier<T>{

Get();

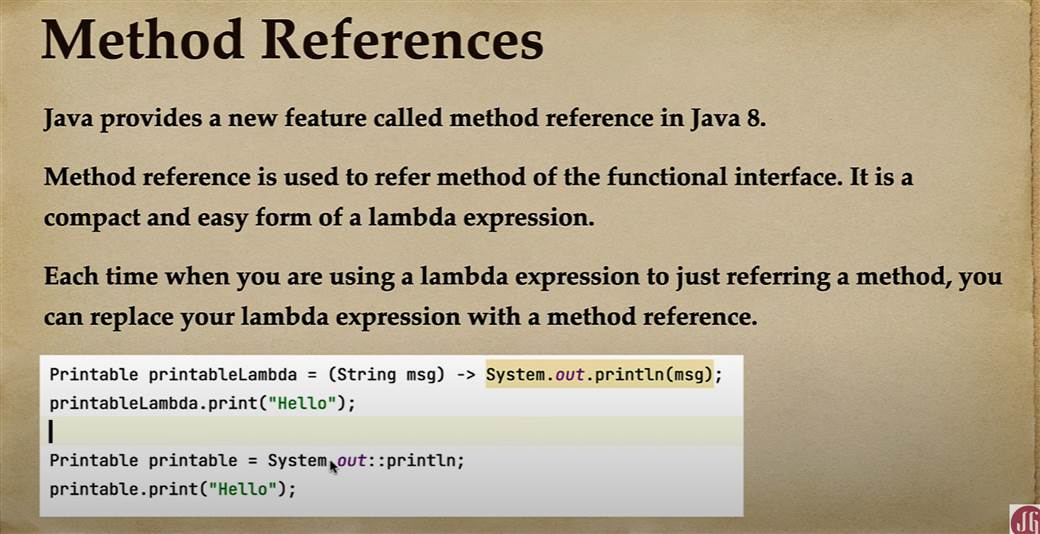
};

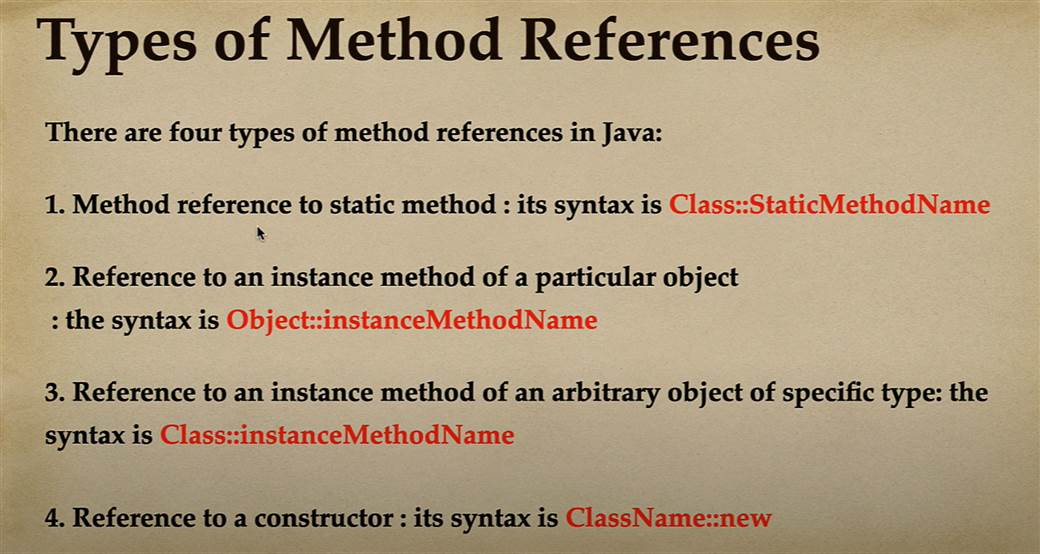




**Chapter 3 :) Method reference**

**We** can replace lambda expression with method reference



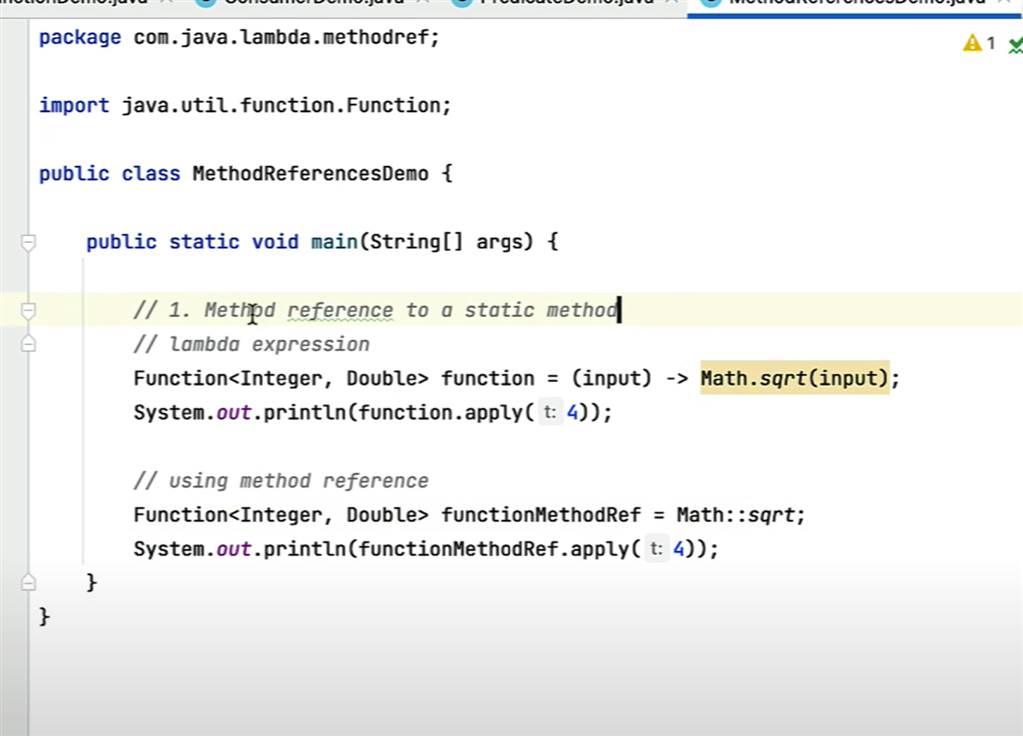


**Note :**

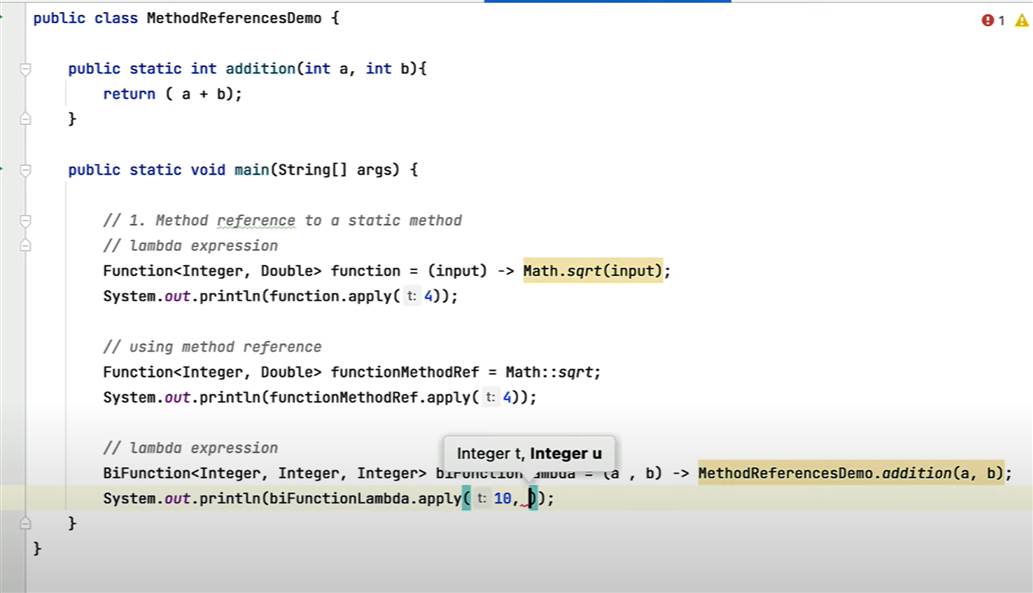
First will write a lambda expression then will convert into method reference.

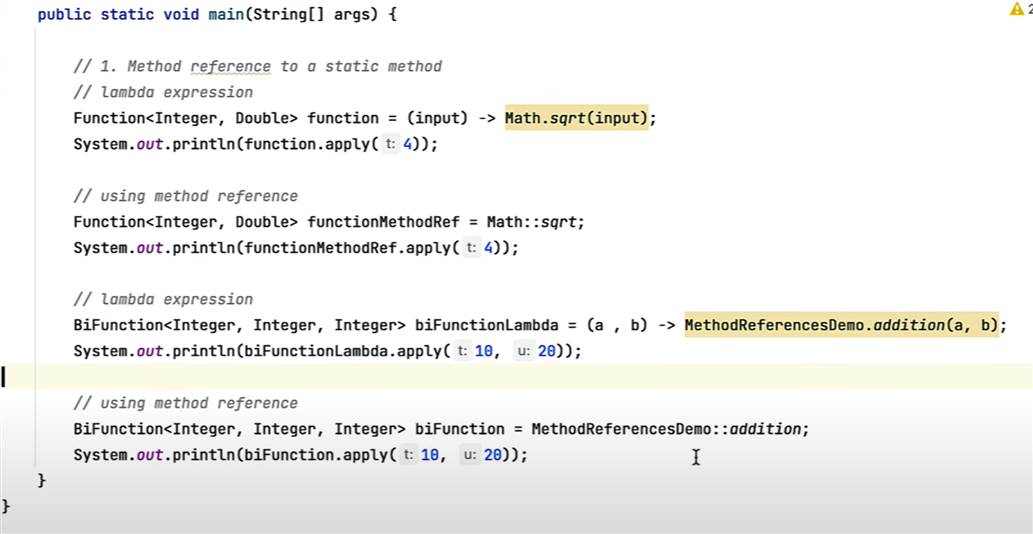
**Class:static Method name**

**Example 1**

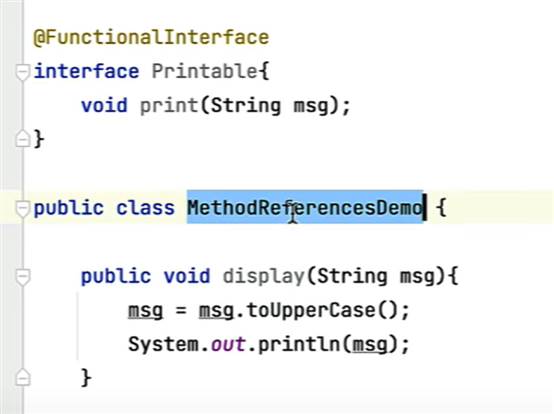


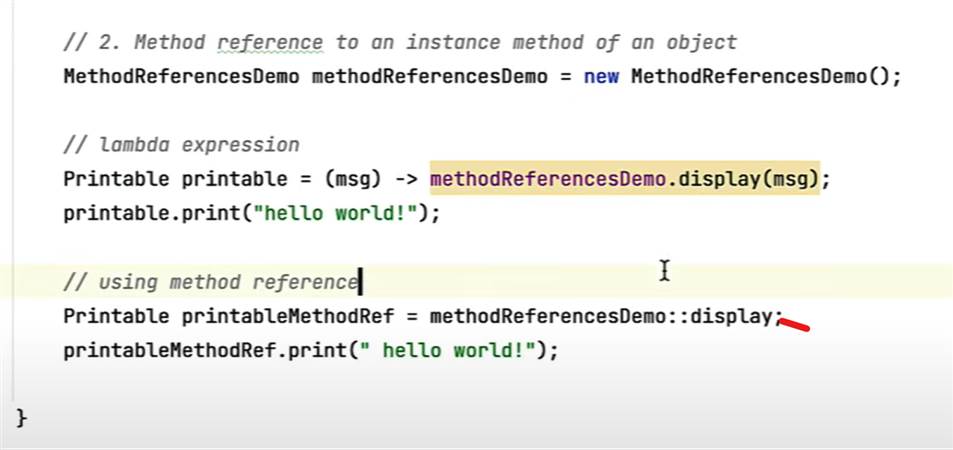
Example 2



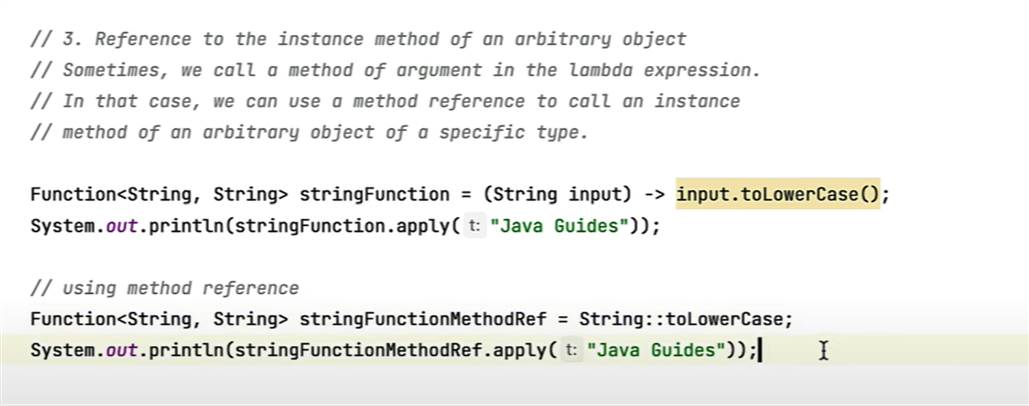


2: object :: instanceMethodName

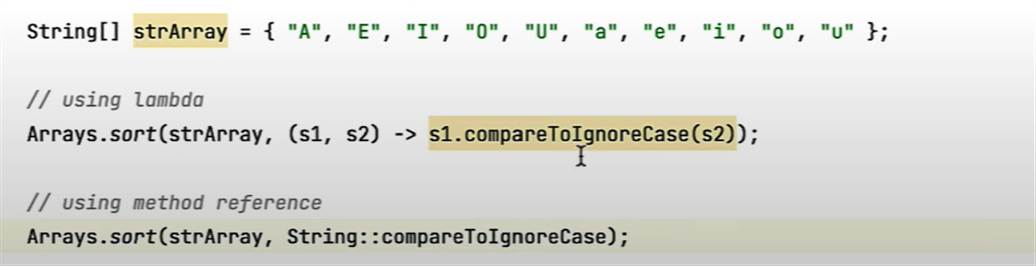




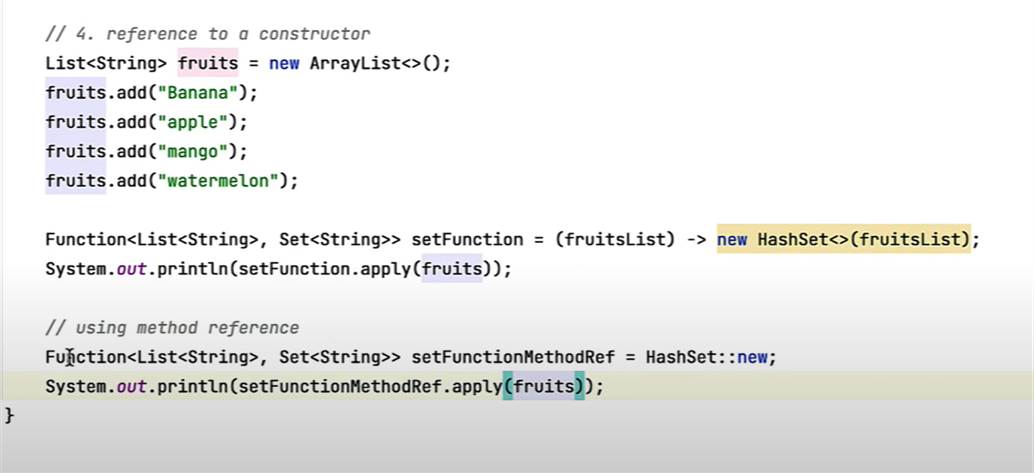
Type 3 : method reference



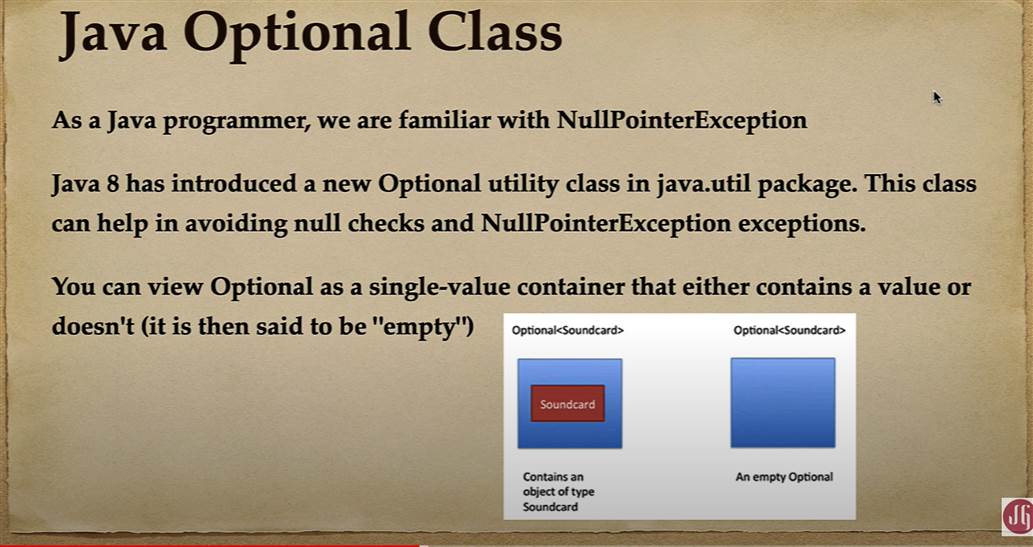
One more example

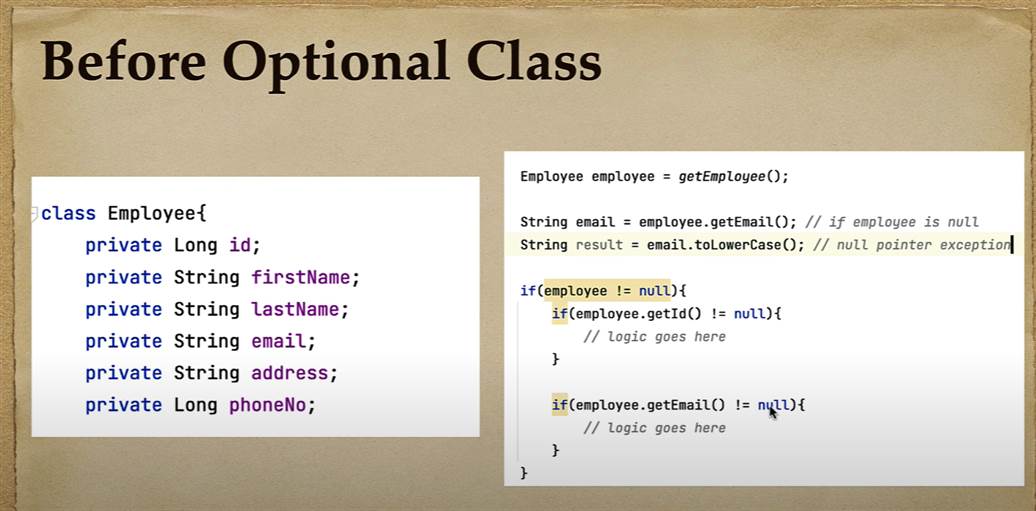


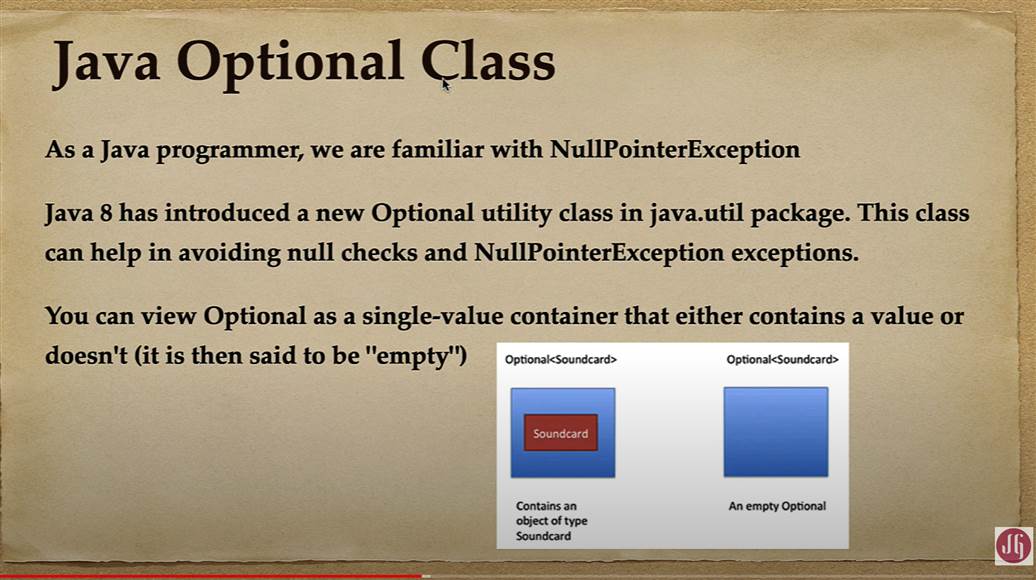
**Method reference using new**

****

**Chapter 4 :- optional class**

****

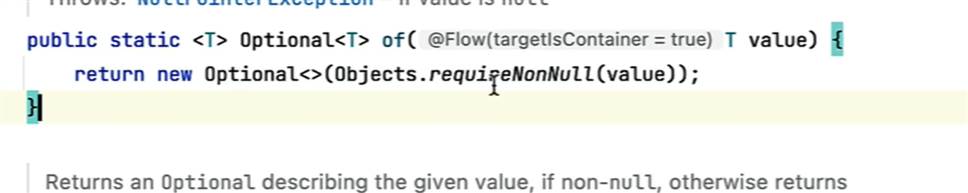
****

****

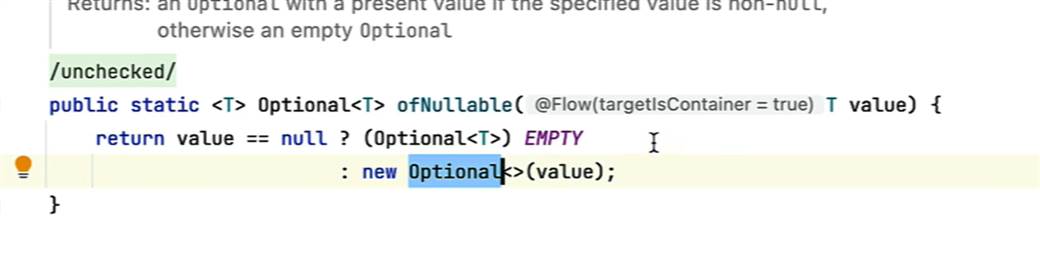
**We can create**  object of optional class by below static method

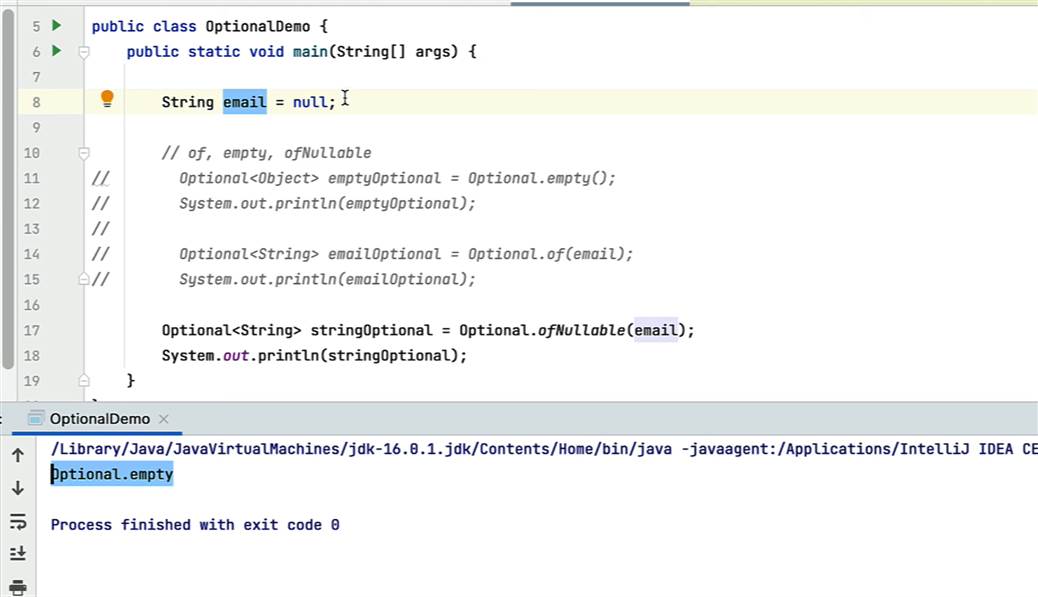
**Empty**

, **of(value),** :- internally it will do null check and if null is there it will throw null pointer exception

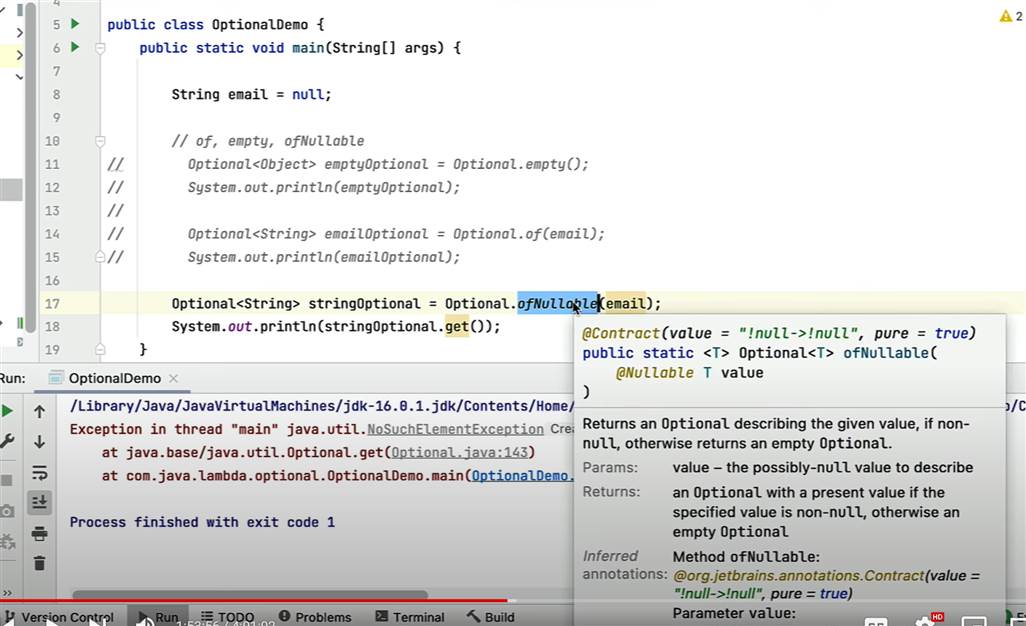


**ofNullable(value)** :---- if value is null then it will return empty object of optional not null . this means it will not throw null pointer exception . it will handle null pointer exception by returning empty object of optional class





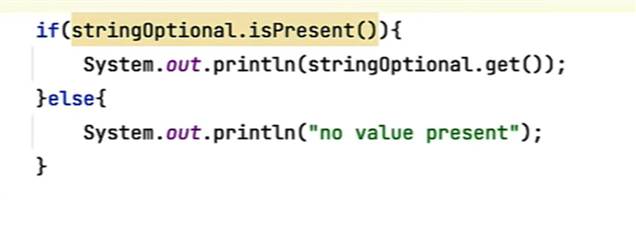
optionalObject.get() ----- will return value



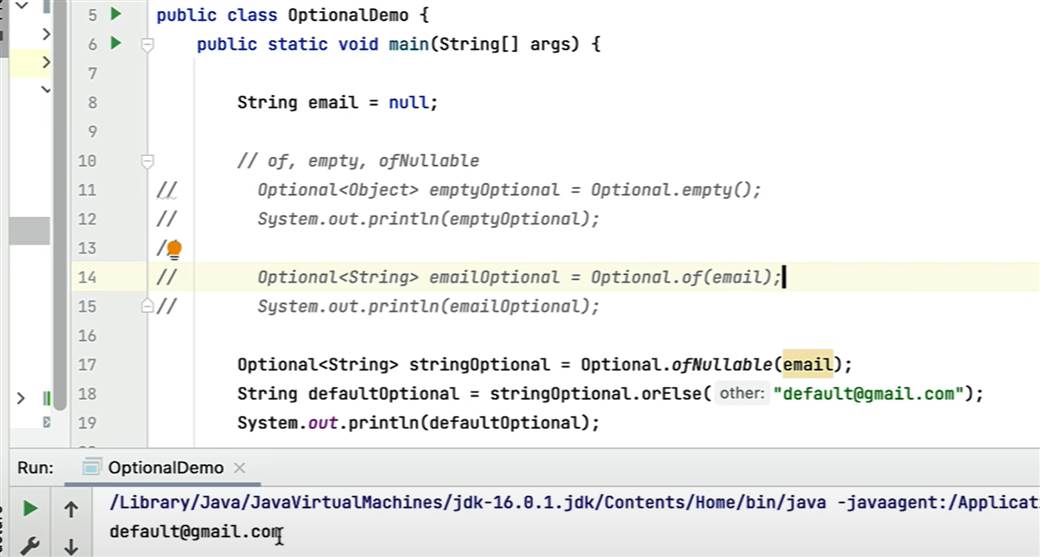
**Will get NoSuchElementException**

**Note :**  in real time project we don’t call get method directly , we need to check optional class contain value or not

**isPresent : : return boolean value**

****

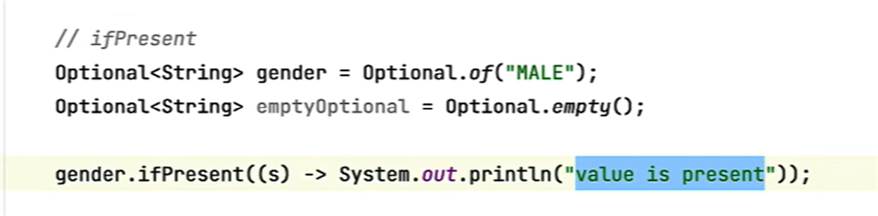
**Sometimes we have requirement i**f the value is not present in optional class then we have to return default values.



**orElse and orElseGet both** are similar only diff, first one will take string while second one will take supllier type

**orElseThrow :**  if value is not present in optional class then we have to throw the exception

**// ifPresent()**

****

**If value is present in optional then only it will execute otherwise it will not execute.**

**A computer screen shot of a program code

Description automatically generated**

**Optional class contain filter method**

**A computer code with text

Description automatically generated with medium confidence**

**Note**

we can use map value in optional to transform from type to another

chapter 5 default and static method

if we will provide default method no need to override by default it will be available to all implement class.

**Advantages :**  preserve backward compatibility is there.

Static : good for utility method like null check

Interface.utility method

Chapter 6 stream API



A screenshot of a computer

Description automatically generated

This contain interfaces, classes and enum

A screen shot of a computer

Description automatically generated

Create stream object

A computer screen shot of a program

Description automatically generated

**Filter stream**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

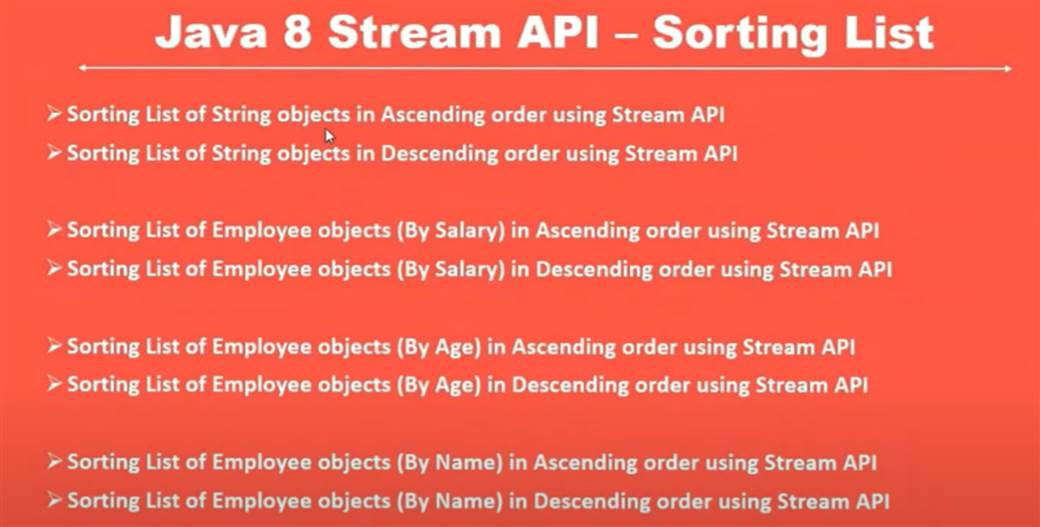
Traditional way filtering product

Using java 8

A screenshot of a computer

Description automatically generated

**Sorting using stream api**

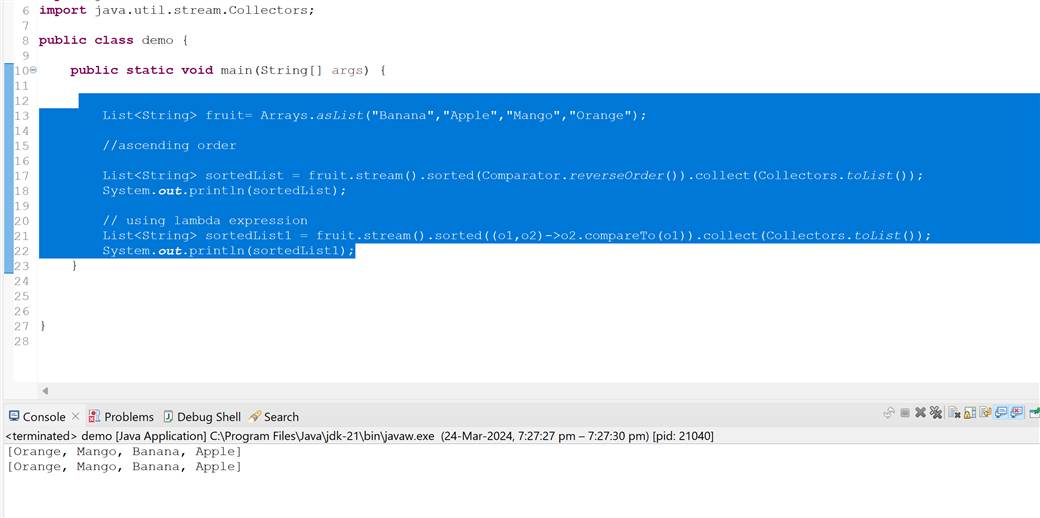
****

**Three different**  way to sort list in ascending order

A screenshot of a computer program

Description automatically generated

Sort list of string in decending order



**Sort employee using salary by annonymous implementation**

**A white background with text

Description automatically generated**

**Sort employee using lambda expression for id**

**A screenshot of a computer

Description automatically generated**

**Descending order sort**

**A screen shot of a computer

Description automatically generated**

**Sort using Comparator.compairing**

**A white background with text

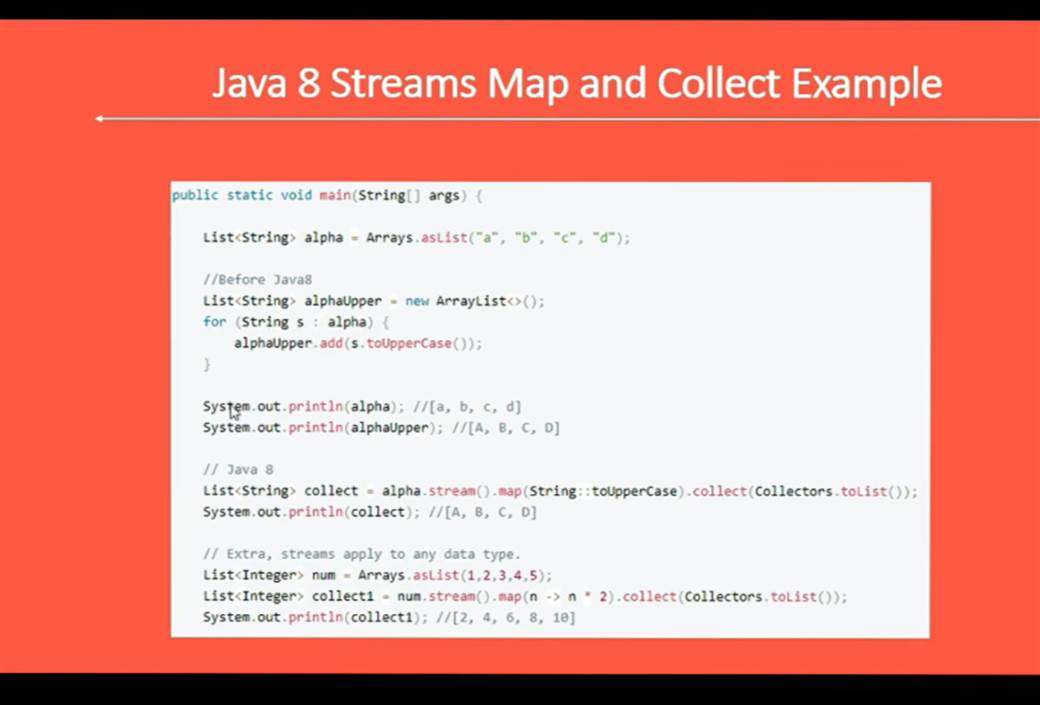
Description automatically generated**

**Descending order**

**A screenshot of a computer

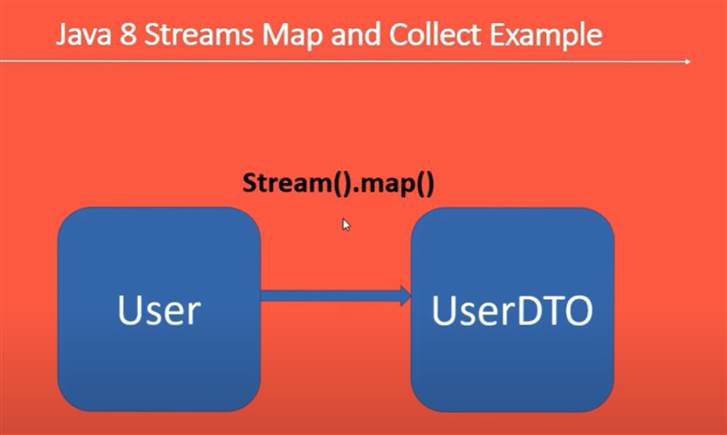
Description automatically generated**

**Java 8 stream amp and collect example**

****

**Use case**

**Fetch user object**  using JPA and map to user dto object



**One object**  map to another object

A screen shot of a computer

Description automatically generated

findFirst , findAny

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

List<Integer> list = Arrays.*asList*(1, 2, 3, 4);

Optional<Integer> first = list.stream().findFirst();

**if** (first.isPresent()) {

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

}**else** {

System.***out***.println("not present");

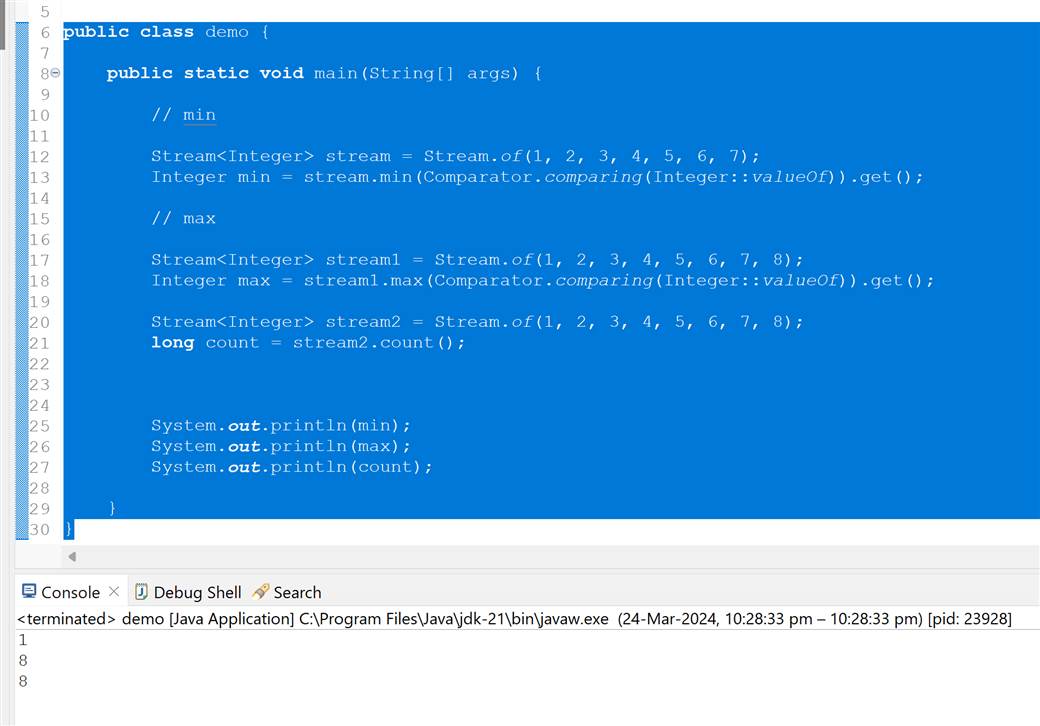
}

}

**Count () : use to** count number of element in string

Max() -> use to find max element in string

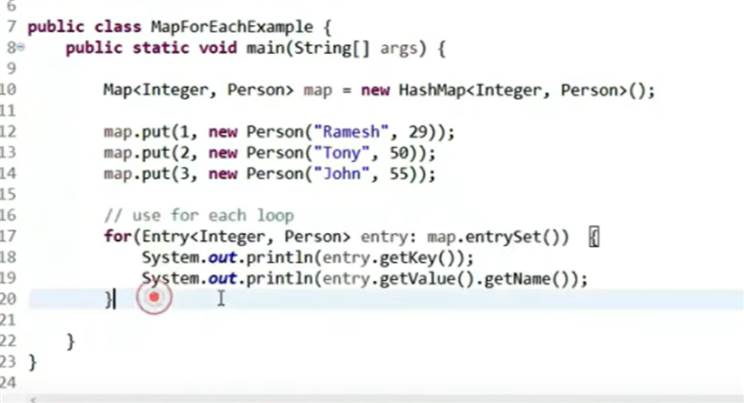
Min():- use to find min element in a string



**Chapter 7 :----- java 8 forEach() method**

**Iterate over map**

**Earlier**

****

**A screenshot of a computer screen

Description automatically generated**