02

JAVA 8

- Java 8 Features
- Lambda Expressions
- Interfaces Changes
- Functional Interfaces
- Stream API
- Optional Class
- Method References and Constructor References
- Date And Time API Changes
- StringJoiner

What are the java8 Features?

- Lambda expression
- Default method
- static method
- Functional Interfaces
- Method Reference
- Constructor Reference
- Stream API
- Foreach
- Optional class
- Collectors class

What is Lambda expression?

- A Lambda is a function
  - no name
  - no modifier
  - no return types

Why to use lambda expression?

- To write functional programming in java
- To write more readable, maintainable and concise code

What are new changes in Interface or why required default method in interface?

- Till the java7 definition of method not allowed
- If a added new method in interfaces child must be override that method this problem identify by java and in the java8 version allow method with default keyword
- Why default keyword because by default interfaces method are public to declare default modifier java8 introduce default keyword explicitly
- We can write multiple default methods in interface

Static Method Interface

1. Java interface static method is part of interface, we can't use it for implementation class objects.
2. Java interface static methods are good for providing utility methods, for example null check, collection sorting etc.
3. Java interface static method helps us in providing security by not allowing implementation classes to override them.
4. We can't define interface static method for Object class methods, we will get compiler error as "This static method cannot hide the instance method from Object". This is because it's not allowed in java, since Object is the base class for all the classes, and we can't have one class level static method and another instance method with same signature.

Is functional interfaces available in java7 if yes give

- **Runnable –>** This interface only contains the run() method.
- **Comparable –>** This interface only contains the compareTo() method.
- **ActionListener –>** This interface only contains the actionPerformed() method.
- **Callable –>** This interface only contains the call() method.
- **Comparator->** it contains compare method

What is Functional Interfaces?

- If an interface contains only one single abstract method, then it is called as functional interface.
- Functional interface is used to invoke lambda expression
- Functional interface can contain default and static method
- To represent our interface as functional interface we use @FunctionalInterface to make sure only one SAM

  Below are the most used built in functional interfaces
  All the functional interfaces available in java.util.function package
    - Function
    - Supplier
    - Consumer
    - Predicate

What is Function Interface?

- A function is a type of functional interface in Java it receives only a single argument and returns a value after the required processing.
- It has 4 methods

Method	Description
default <V> Function<T,V> andThen(Function<? super R,? extends V> after)	It returns a composed function that first applies this function to its input, and then applies the after function to the result. If evaluation of either function throws an exception, it is relayed to the caller of the composed function.
static <T> Function<T,T> identity()	It returns a function that always returns its input argument.
R apply(T t)	It applies this function to the given argument.
default <V> Function<V,R> compose(Function<? super V,? extends T> before)	It Returns a composed function that first applies the before function to its input, and then applies this function to the result. If evaluation of either function throws an exception, it is relayed to the caller of the composed function.

What is Consumer Functional Interface or explain scenario

Method	Description
void accept(T t)	It performs this operation on the given argument.
default Consumer<T> andThen(Consumer<? super T> after)	It returns a composed Consumer that performs, in sequence, this operation followed by the after operation. If performing either operation throws an exception, it is relayed to the caller of the composed operation. If performing this operation throws an exception, the after operation will not be performed.

What is Supplier Interface?

A supplier is any method which takes no arguments and returns a value. Its job is to supply an instance of an expected class.

It represents a function which does not take in any argument but produces a value of

`T get()`: This abstract method does not accept any argument but instead returns newly generated values

What is Predicate?

It is a functional interface which represents a predicate (boolean-valued function) of one argument.

Java Predicate Interface Methods

Methods	Description
boolean test(T t)	It evaluates this predicate on the given argument.
default Predicate<T> and(Predicate<? super T> other)	It returns a composed predicate that represents a short-circuiting logical AND of this predicate and another. When evaluating the composed predicate, if this predicate is false, then the other predicate is not evaluated.
default Predicate<T> negate()	It returns a predicate that represents the logical negation of this predicate.
default Predicate<T> or(Predicate<? super T> other)	It returns a composed predicate that represents a short-circuiting logical OR of this predicate and another. When evaluating the composed predicate, if this predicate is true, then the other predicate is not evaluated.
static <T> Predicate<T> isEqual(Object targetRef)	It returns a predicate that tests if two arguments are equal according to Objects.equals(Object, Object).

What Method Reference?

Method reference is used to refer method of functional interface.

It is compact and easy form of lambda expression.

Each time when you are using lambda expression to just referring a method, you can replace your lambda expression with method reference.

Types of Method References

There are following types of method references in java:

1. Reference to a static method.

   ContainingClass::staticMethodName

```
public static void main(String[] args) {
    // Referring static method
    Sayable sayable = MethodReference::saySomething;
    // Calling interface method
    sayable.say();
}
```

2. Reference to an instance method.

   We can refer non-static method using object.

   Syntax

```
containingObject::instanceMethodName
```

3. Reference to a constructor.

You can refer a constructor by using the new keyword.

Syntax

```
ClassName::new
```

What Is Stream in Java

- A stream is not a data structure instead it takes input from the Collections, Arrays or I/O channels.
- Streams don't change the original data structure; they only provide the result as per the pipelined methods.
- Streams don't change the original data structure; they only provide the result as per the pipelined methods.
- Each intermediate operation is lazily executed and returns a stream as a result, hence various intermediate operations can be pipelined. Terminal operations mark the end of the stream and return the result.

What is are the operations in stream api

1) Intermediate Operation -Java 8 Stream intermediate operations return another Stream which allows you to call multiple operations in a form of a query. Stream intermediate operations do not get executed until a terminal operation is invoked. All Intermediate operations are lazy, so they're not executed until a result of a processing is needed.
    i. filter()
    ii. map()
    iii. flatMap()
    iv. distinct()
    v. sorted()
    vi. peek()
    vii. limit()
    viii. skip()
2) Terminal Operation- Java-8 Stream terminal operations produces a non-stream, result such as primitive value, a collection or no value at all.

3) anyMatch()
4) allMatch()
5) noneMatch()
6) collect()
7) count()
8) findAny()
9) findFirst()
10) forEach()
11) min()
12) max()
13) reduce()
14) toArray()