HowToDoInJava

Functional Interfaces in Java

Last Updated: February 26, 2022 By: Lokesh Gupta Java Streams Functional Interface, Java Stream Basics

Introduced in Java 8, **a functional interface is simply an interface that has exactly one abstract method**. Learn more about functional interfaces in this tutorial.

Table Of Contents

1. What is a Functional Interface?

1.1. Only one `abstract` method is allowed

Functional interfaces are new additions in Java 8. **As a rule, a functional interface can contain exactly one abstract method.** These functional interfaces are also called **Single Abstract Method interfaces (SAM Interfaces)**.

Apart from one abstract method, a **functional interface can also have the following methods that do not count** for defining it as a functional interface.

- Default methods
- Static methods
- Public methods inherited from the *Object* class

1.2. Implemented by Lambda Expressions

In Java, *lambda expressions* can be used to represent an instance of a functional interface. For example, Comparator interface is a functional interface.

```
@FunctionalInterface
public interface Comparator<T> {
    int compare(T o1, T o2);
    boolean equals(Object obj);

    //and multiple default methods...
}
```

Comparator interface has only two abstract methods `compare()` and `equals()`. But *equals()* has been inherited from the *Object* class, so it is not counted. Other than these two methods, all other methods are *default methods*. So *Comparator* is qualified to be declared as a functional interface.

Java program to implement *Comparator* using a lambda expression.

```
//Compare by Id
Comparator<Employee> compareById = Comparator.comparing(e -> e.getId());

Comparator<Employee> compareByFirstName = Comparator.comparing(e -> e.getFirstName());
```

2. @FunctionalInterface Annotation

Java 8 introduced the annotation `@FunctionalInterface` to mark an interface as a functional interface. The primary use of this annotation is **for compiler-level errors when the interface violates the contracts of precisely one abstract method**.

> Note that **using the annotation *@FunctionalInterface* is optional**.

If the interface has one abstract method and does not have *@FunctionalInterface* annotation, the interface is still a functional interface, and it can be the target type for lambda expressions.

The presence of the annotation protects us from inadvertently changing a functional interface into a non-functional interface, as the compiler will catch it.

Let's build our first functional interface. Note that methods in an interface are, by default, *abstract*.

```
@FunctionalInterface
public interface MyFirstFunctionalInterface
{
    public void firstWork();
}
```

Let's try to add another abstract method:

```
@FunctionalInterface
public interface MyFirstFunctionalInterface
{
    public void firstWork();
    public void doSomeMoreWork();   //error
}
```

The above code will result in a compiler error:

```
Unexpected @FunctionalInterface annotation
@FunctionalInterface ^ MyFirstFunctionalInterface is not a functional interface
multiple non-overriding abstract methods found in interface MyFirstFunctionalInterface
```

Read More : Generic Functional Interfaces

3. Functional Interfaces in JDK

The following is a list of Java's most commonly used functional interfaces.

- Runnable: contains only the *run()* method.
- Comparable: contains only the *compareTo()* method.
- ActionListener: contains only the *actionPerformed()* method.
- *Callable*: contains only the *call()* method.
- *Predicate*: a boolean-valued function that takes an argument and returns true or false.
- *BiPredicate*: a predicate with two arguments.
- *Consumer*: an operation that takes an argument, operates on it, and returns no result.
- *BiConsumer*: a consumer with two arguments.
- *Supplier*: a supplier that returns a value.
- *Function<T, R>*: takes an argument of type T and returns a result of type R.
- *BiFunction<T, U, R>*: takes two arguments of types T and U and returns a result of type R.

4. Demo

Let's see a quick example of creating and using functional interfaces in Java.

We are using a functional interface *Function* to create the formula for mathematical squares.

```
Function<Integer, Integer> square = x -> x * x;
```

The Function interface has one abstract method `apply()` that we have implemented above. we can execute the above method as follows:

```
System.out.println( square.apply(5) );  //Prints 25
```

5. Conclusion

In this tutorial, we learned to create and manage functional interfaces in Java. We learned that a *functional interface* has only one *abstract* method and they can be implemented by the lambda expressions.

We also saw the JDK provided existing functional interfaces, and finally how to create an use a functional interface.

Happy Learning !!

Sourcecode on Github

Was this post helpful?

Let us know if you liked the post. That's the only way we can improve.

Yes

No

Recommended Reading:

1. Generic Functional Interfaces in Java
2. Sealed Classes and Interfaces
3. Java Streams API
4. Creating Streams in Java
5. Primitive Type Streams in Java
6. Java Stream sorted()
7. Java Stream max()
8. Java Stream limit()

9. Java Stream findFirst() vs findAny() API With Example

0. Sorting a Stream by Multiple Fields in Java

Join 7000+ Awesome Developers

Get the latest updates from industry, awesome resources, blog updates and much more.

Email Address

Subscribe

** We do not spam !!*

15 thoughts on “Functional Interfaces in Java”

jyothi

March 3, 2020 at 6:48 pm

thank you…somehow it helped me to gather info on functional interface

Reply

Irfan

July 12, 2019 at 12:21 am

Thanks for the details
# It will really help readers if we can add more scenarios when this particular feature can/should be used
# Which current features it can replace or act as an alternative to
# What would be the BIG-O or any performance metric we can back up with

Reply

Ravi Kant Verma

November 14, 2018 at 2:10 am

Very Nice Post

Reply

Raja M S

September 25, 2018 at 4:15 pm

Do a Functional Interface can extends another Functional or Non Functional Interface?
Any behavioral change?

Reply

Abhishek Kumar

February 4, 2020 at 6:54 pm

No A Functional Interface can not extends another Funtional Interface as the child funtional interface will be having two abstract method which will give exception.

Reply

Ullas

September 7, 2018 at 1:29 am

Excellent post. Kudos to the creator!

Reply

Priya

September 7, 2017 at 11:29 am

This site has very useful content very easy to understand.

Reply

Harsh

December 7, 2016 at 12:12 pm

Fantastic Post regarding Functional interface

Reply

Himansu

July 22, 2016 at 1:06 am

Hi Lokesh,

A few things to possibly add are below are list of FunctionalInterface in jdk1.8 libraries

https://stackoverflow.com/questions/27743315/a-summary-of-the-parameters-and-return-type-of-functional-interfaces-in-the-pack/28162720#28162720

Reply

Lalita Kamde

January 22, 2015 at 1:38 pm

Thanks…it clears the confusion of which are functional interface .

Reply

Binh Thanh Nguyen

January 9, 2015 at 3:53 am

Thanks, nice post

Reply

Manohar

June 17, 2014 at 7:11 am

I have doubt . What is the main purpose of implementing functional interfaces in java8 ?

Thanks to you lokesh, now I know about functional interfaces but dont know when to use it ?

Reply

Lokesh Gupta

June 17, 2014 at 8:35 am

"The type in which lambda expressions are converted, are always of functional interface type". You would like to read more in this section: https://howtodoinjava.com/java8/lambda-expressions/#functional_interface

Reply

Tony

April 6, 2014 at 5:48 pm

Nice one.This is absolutely clear to understand.
But only one doubt in which scenario we will use Functional Interface/SAM Interface ?

Thanks & Regards
Tony

Reply

Lokesh Gupta

April 6, 2014 at 6:08 pm

Whenever you are planning to write a method where you want to pass a lambda expression as argument.

Reply

Leave a Comment

Name *

Email *

Website

☐ Add me to your newsletter and keep me updated whenever you publish new blog posts

Post Comment

Search ...

Two fifths of all older people in the UK say television is their main form of company.
Want to help them get out of the house?
DONATE HERE
SOTHEBY MEWS

HowToDoInJava

A blog about Java and related technologies, the best practices, algorithms, and interview questions.

Meta Links

- About Me
- Contact Us
- Privacy policy
- Advertise
- Guest Posts

Blogs

REST API Tutorial

Copyright © 2022 · Hosted on Cloudways · Sitemap