Java 8 has introduced the concept of functional interface. Functional interface is nothing but a simple java interface containing only one method. In the old style functional interface can be used with inner class or anonymous class to implement. But in java 8, we have lambda expressions to use functional interface. The difference is that lambda expression is clean and small code in comparison to inner or anonymous class. java.util.function.Function is a functional interface introduced in java 8.

## Use @FunctionalInterface

Functional interface will have one method and can have any number of arguments. The interface will be annotated with *@FunctionalInterface.* By this annotation we ensure that the functional interface will not have more than one method. If we try to add more than one method, then there will be compile time error as *Unexpected @FunctionalInterface annotation*.  
Find the functional interface with no argument.

@FunctionalInterface

**public** **interface** Add {

**public** **int** addData(**int** n1, **int** n2);

}

@FunctionalInterface

**public** **interface** Multiply {

**public** **int** multiply(**int** num);

}

@FunctionalInterface

**public** **interface** Display {

**public** String show();

}

**public** **class** FunctionalInterfaceDemo {

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

// functional interface with no argument

Display display = () -> "Functional interface with no argument";

String s= display.show();

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

// functional interface with one argument

Multiply multiply = (**int** num) -> num\*10;

**int** res= multiply.multiply(5);

System.***out***.println(res);

// functional interface with two argument

Add add =(**int** a, **int** b) -> a+b;

**int** rs = add.addData(15, 20);

System.***out***.println(rs);

}

}

Functional Interfaces

In Java, a Marker interface is an interface with no methods or fields declaration. In simple words, marker interface is an empty interface. Similarly, a Functional Interface is an interface with just one abstract method declared in it.

ava.lang.Runnable is an example of a Functional Interface. There is only one method void run() declared in Runnable interface. Similarly ActionListener interface is also a Functional Interface. We use Anonymous inner classes to instantiate objects of functional interface. With Lambda expressions, this can be simplified.

Runnable r = () -> System.out.println("hello world");

new Thread(

() -> System.out.println("hello world")

).start();

[@FunctionalInterface](http://download.java.net/jdk8/docs/api/java/lang/FunctionalInterface.html)

@FunctionalInterface

public interface WorkerInterface {

public void doSomeWork();

}

public class WorkerInterfaceTest {

public static void execute(WorkerInterface worker) {

worker.doSomeWork();

}

public static void main(String [] args) {

//invoke doSomeWork using Annonymous class

execute(new WorkerInterface() {

@Override

public void doSomeWork() {

System.out.println("Worker invoked using Anonymous class");

}

});

//invoke doSomeWork using Lambda expression

execute( () -> System.out.println("Worker invoked using Lambda expression") );

}

}