<https://www.journaldev.com/2389/java-8-features-with-examples>

<https://www.javatpoint.com/java-8-features>

<https://dzone.com/articles/java-lambda-expressions-basics>

# How to Fix The Target Type of This Expression Must be a Functional Interface Error

This post talks about how to resolve "**the target type of this expression must be a functional interface**" error while trying to write a lambda expression.

Let's first get some background on what are **functional interfaces** and **lambda expressions** in Java; that will help you to get an idea why this error is coming.

**Functional Interface in Java**

A functional interface is an interface with **only one abstract method**. A functional interface is also known as SAM type where SAM stands for (Single Abstract Method).

* Refer this post to know more about functional interfaces - [Functional interfaces & lambda expression](http://netjs.blogspot.com/2015/06/functional-interfaces-and-lambda-expression-in-java-8.html)

**Lambda expression in Java**

Lambda expression is an instance of the functional interface and **provides implementation of the abstract method defined by the functional interface**. Thus the functional interface specifies its **target type**.

* Refer this post to know more about lambda expression - [Overview of lambda expressions](http://netjs.blogspot.com/2015/06/lambda-expression-in-java-8-overview.html)

As you can see Functional interface is an interface with only one abstract method and lambda expression provides implementation of the abstract method defined by the functional interface.

So *this error may come when you have a functional interface which has more than one abstract methods*. Let's see it with an **example**. Here I am trying to write a lambda block which counts the number of words in a string.

interface IMyFunc {

int func(String n);

int func1(String n1, String n2);

}

public class FuntionalIntError {

public static void main(String[] args) {

String inStr = "Lambdas are a new addition to Java";

// lamda block assigned to a functional interface reference

IMyFunc myFunc = (str) -> {

int c = 0;

char ch[]= new char[str.length()];

for(int i = 0; i < str.length(); i++){

ch[i] = str.charAt(i);

if(((i > 0) && (ch[i] != ' ') && (ch[i-1] == ' ')) ||

((ch[0] != ' ') && (i == 0)))

c++;

}

return c;

};

// calling the func method of that functional interface

System.out.println("The word count is " + myFunc.func(inStr));

}

}

This code will give me the same error "**The target type of this expression must be a functional interface**" because of the fact that functional interface **IMyFunc** has **two abstract methods** so lambda expression won't know which of these two methods it is implementing and what is its target type.

**How to resolve this error**

By now you may have guessed that resolving this error "*the target type of this expression must be a functional interface*" means functional interface should have only one abstract method. So we need to delete the second method.

interface IMyFunc {

int func(String n);

//int func1(String n1, String n2);

}

In order to *make sure that the functional interface you write has one and only one abstract method*, you can use **@FunctionalInterface** annotation with your functional interface.

@FunctionalInterface

interface IMyFunc {

int func(String n);

}

* Refer this post to know more about functional interface annotaton - [Functional interface annotation in Java 8](http://netjs.blogspot.com/2015/06/functional-interface-annotation-java-8.html)

With this annotation @FunctionalInterface in place, any attempt to add another abstract method to this functional interface will result in compiler error.

That's all for this topic **How to Fix The Target Type of This Expression Must be a Functional Interface Error**. If you have any doubt or any suggestions to make please drop a comment. Thanks!