Functional Interface with Arguments

In previous post we saw how to write a functional interface and use Lambda Operator. In this post we will write a Functional Interface with Arguments.

Below is the Functional Interface with 2 int arguments.

**package** com.example.javase8.interfaces;

@FunctionalInterface

**public** **interface** InterfaceWithArgs {

**public** **int** calculate(**int** value1, **int** value2);

}

Now we will write a code to call this method and perform operation on them.

**package** com.example.javase8;

**import** com.example.javase8.interfaces.InterfaceWithArgs;

**public** **class** UseInterfaceWithArgs {

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

InterfaceWithArgs obj1 = (v1, v2) -> {

**int** result = v1 \* v2;

**return** result;

};

**int** result1 = obj1.calculate(3, 4);

InterfaceWithArgs obj2 = (v1, v2) -> {

**int** result = v1 + v2;

**return** result;

};

**int** result2 = obj2.calculate(3, 4);

System.***out***.println(result1+" "+result2);

}

}

We are using one Interface and performing two different operations multiplication and addition.

The two parameters are in brackets. Also did you notice carefully there are no data type for the arguments.

Reason for that is there is only one interface in Functional Interface which is abstract. So compiler can infer the types itself. But, if you want to write the types explicitly then you are allowed to do that. But there is one more catch. If the method has multiple parameters then you need to write types for all of them else none of them.