IntConsumer Functional Interface

IntConsumer Interface represents an operation that accepts int valued argument and returns no result. Previously we had discussed [Consumer](http://data-structure-learning.blogspot.com/2015/07/java-lambda-consumer-functional.html) interface. I would recommend you to read that as IntConsumer is primitive int data type specialization of [Consumer](http://data-structure-learning.blogspot.com/2015/07/java-lambda-consumer-functional.html) interface.

There are two different methods in IntConsumer interface. Let us understand both of them one by one.

**accept() method**

**void** accept(**int** value);

accept() method performs this operation on give argument. This method does not return any value. It accepts a parameter of type int.

Let us take simple example to understand this method.

IntConsumer intConsumer = (a) -> System.***out***.println(a);

intConsumer.accept(10); //Prints 10

**andThen()** method

**default** IntConsumer andThen(IntConsumer after) {

Objects.*requireNonNull*(after);

**return** (**int** t) -> { accept(t); after.accept(t); };

}

andThen() method returns a composed version of this and after IntConsumer that will perform respective operation in sequence. This method will throw NullPointerException if after is null.

Let us write two different IntConsumer. One for adding and another for multiplication.

IntConsumer adder = (a) -> System.***out***.println(a + a);

IntConsumer multiplier = (a) -> System.***out***.println(a \* a);

adder.andThen(multiplier).accept(20);//First adder is executed then multiplier.

Output

40

400

Read about important java.util.function package’s interface [here](http://data-structure-learning.blogspot.com/p/functional-programming-in-java.html). [Consumer](http://data-structure-learning.blogspot.com/2015/07/java-lambda-consumer-functional.html), [Function](http://data-structure-learning.blogspot.com/2015/07/java-lambda-function-functional.html), [Supplier](http://data-structure-learning.blogspot.com/2015/07/java-lambda-supplier-functional.html), [BinaryOperator](http://data-structure-learning.blogspot.com/2015/07/java-lambda-binaryoperator-functional.html) & [Predicate](http://data-structure-learning.blogspot.com/2015/07/java-lambda-predicate-functional.html) Functional Interfaces. I have also written on [High Order functions](http://data-structure-learning.blogspot.com/2015/07/higher-order-functions-using-function.html) using Function functional interface.