BiFunction Interface

Before proceeding with reading this post I would highly recommend reading [Function Interface](http://data-structure-learning.blogspot.com/2015/07/higher-order-functions-using-function.html).

As we saw in Function interface, it takes one argument *T* and produces and returns result *R.* BiFunction interface represents a function that accepts two arguments *T* and *U* and produces and returns result *R.*

Let us start with understanding code base for BiFunction Interface.

|  |
| --- |
| **BiFunction Interface Declaration**  **public** **interface** BiFunction<T, U, R>  The BiFunction interface takes 3 arguments. *T* and *U* are arguments to be processed upon. The third argument *R* is a return type of the result of function. |
| **apply() method**  R apply(T t, U u);  Apply this function to given argument. T and U are function argument. It returns function result *R.* |
| **andThen() method**  **default** <V>  BiFunction<T, U, V> andThen(Function<? **super** R, ? **extends** V> after) {  Objects.*requireNonNull*(after);  **return** (T t, U u) -> after.apply(apply(t, u));  }  Returns a composed function that first applies this function to input and then applies the after function to result. |

Let us now write a simple BiConsumer implementation of adding two integers and returning the result.

BiFunction<Integer, Integer, Integer> adderFunction1 = (t, u) -> {

**return** t + u;

};

**int** result1 = adderFunction1.apply(10, 10);//Outputs 20

Now we will take example of andThen() method. andThen method takes Function as parameter.

BiFunction<String, String, String> stringPrint = (t, u) -> {

**return** t + u;

};

Function<String, String> adderFunction = t -> t + " Function Interface";

System.***out***.println(stringPrint.andThen(adderFunction).apply("Bi","Function"));

Above is the code for andThen() method. we chained the stringPrint function with adderFunction and passed arguments as ("Bi","Function") to stringPrint.

The above code prints

BiFunction Function Interface

That’s all on BiFunction interface.