Optional class flatMap() method

In previous posts we saw how to [avoid Null Pointer Exception](http://data-structure-learning.blogspot.com/2015/07/avoid-null-pointer-using-optional-class.html) using Optional<T> class, [Optional<T> class introduction](http://data-structure-learning.blogspot.com/2015/07/optional-class-introduction.html), Optional<T> class [of(), ofNullable() and empty()](http://data-structure-learning.blogspot.com/2015/07/optional-class-of-ofnullable-empty.html) method, [Optional<T> class ifPresent() method](http://data-structure-learning.blogspot.com/2015/07/optional-class-ifpresent-method.html), [Optional<T> class filter()](http://data-structure-learning.blogspot.com/2015/07/optional-class-filter-method.html) method, Optional<T> class [isPresent()](http://data-structure-learning.blogspot.com/2015/07/optional-class-ispresent-and-get-method.html) and [get()](http://data-structure-learning.blogspot.com/2015/07/optional-class-ispresent-and-get-method.html) method, Optional<T> class [orElse()](http://data-structure-learning.blogspot.com/2015/07/optional-class-orelse-and-orelseget.html) and [orElseGet()](http://data-structure-learning.blogspot.com/2015/07/optional-class-orelse-and-orelseget.html) method and Optional<T> class [map()](http://data-structure-learning.blogspot.com/2015/07/optional-class-map-method.html) method.

I will first copy paste the Javadoc for this method then I will explain it.

**If a value is present, apply the provided Optional-bearing mapping function to it, return that result, otherwise return an empty Optional. This method is similar to map(Function), but the provided mapper is one whose result is already an Optional, and if invoked, flatMap does not wrap it with an additional Optional.**

We discussed [Function<T, R> interface](http://data-structure-learning.blogspot.com/2015/07/higher-order-functions-using-function.html). T is the argument and R is the result produced. Applies function to T which is argument of Function.

flatMap() is similar to map() method the difference is the return type of lambda expression passed to method. In flatMap() the return type of lambda expression is always an instance of Optional. But for map method return type of lambda expression can be anything but the value is wrapped within Optional instance before it is returned from map() method.

**public**<U> Optional<U> flatMap(Function**<? super T,** **Optional<U>**> mapper) {

Objects.*requireNonNull*(mapper);

**if** (!isPresent())

**return** *empty*();

**else** {

**return** Objects.*requireNonNull*(mapper.apply(value));

}

}

Look at the method parameter. The result is Optional<U>.

Example for flatMap()

List<String> words = **new** ArrayList<String>(Arrays.*asList*("Mobilize", "Diversity", "Nation", "Mobilizers", "Mobilization"));

Optional<String> found = words.stream()

.filter(word -> word.startsWith("Mob"))

.findFirst();

**final** Optional<String> upperCase = found.flatMap(word -> Optional.*of*(word.toUpperCase()));

upperCase.ifPresent(word -> System.***out***.println(word));