Optional class map() 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 and 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.

Let’s say that we have a value in Optional<T> object and we want to apply the function on it. This is the purpose of map() method. So if we need to apply transformation on value only that value is not null hence avoiding NullPointerException.

So let’s say we from the List of strings we want to change case to upper case for the element whose prefix is “Wed” and length > 5. So we write something like this.

**for** (String day : days) {

**if** (day != **null**) {

**if** (day.startsWith("Wed") && day.length() > 5) {

String upper = day.toUpperCase();

System.***out***.println(upper);

}

}

}

map() method

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

Objects.*requireNonNull*(mapper);

**if** (!isPresent())

**return** *empty*();

**else** {

**return** Optional.*ofNullable*(mapper.apply(value));

}

}

map() applies to given function on value inside Optional only if Optional is present. Otherwise empty() is returned.

List<String> days = **new** ArrayList<String>(Arrays.*asList*("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"));

**final** Optional<String> found=days.stream()

.filter(day -> day.startsWith("Wed"))

.findFirst();

found.filter(day -> day.length() > 5)

.map(String::toUpperCase)

.ifPresent(System.***out***::print); //WEDNESDAY

map() applies to given function on value inside Optional only if Optional is present. Otherwise empty() is returned.

That’s all on map() method. In next post e we will see flatMap() method.