Optional class ifPresent() 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 class introduction](http://data-structure-learning.blogspot.com/2015/07/optional-class-introduction.html) and Optional class [of(), ofNullable() and empty()](http://data-structure-learning.blogspot.com/2015/07/optional-class-of-ofnullable-empty.html) method.

In this post we will discuss ifPresent() method.

**public** **void** ifPresent(Consumer<? **super** T> consumer) {

**if** (value != **null**)

consumer.accept(value);

}

If the value is present then invoke the specified consumer with the value otherwise do nothing. This method will throw NullPointerException if value is present and consumer is null.

Now Consumer interface is functional interface hence we can apply Lambda Operator.

Let us take example that we took previously. We will write a method that will pick an element that starts with some prefix String.

**public** **static** **void** find1(List<String> days, String prefix) {

String found = **null**;

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

**if** (day.startsWith(prefix)) {

found = day;

**break**;

}

}

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

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

**} else {**

**System.*out*.println("No Matches");**

**}**

}

The if-else condition for null check is clutter in method. We can use ifPresent(..) for our work there.

ifPresent(..) will be used to print the value.

With Lambda Operator we can write like this

found.ifPresent(day -> System.***out***.println(day));

Let us write entire method again with declarative style.

**public** **static** **void** find(**final** List<String> days, **final** String prefix){

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

.filter(day->day.startsWith(prefix))

.findFirst();

**found.ifPresent(day -> System.*out*.println(day));**

}

So we can remove if condition using ifPresent(..) method.