

Optionals

By TecheStop

A dark blue diagonal gradient bar that starts from the bottom left corner and extends towards the top right corner, covering the lower half of the slide.

What is an Optional class?

An optional object is a container object which may or may not contain a non-null value.

Why Optional class is needed?

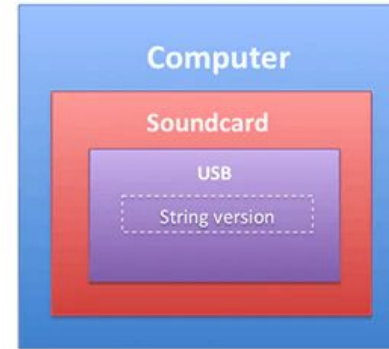


Image Source: [Oracle Article](#)

String version =
`computer.getSoundcard().getUSB().getVersion();`

Why Optional class is needed?

```
String version = "UNKNOWN";  
if(computer != null){  
    Soundcard soundcard =  
        computer.getSoundcard();  
    if(soundcard != null){  
        USB usb = soundcard.getUSB();  
        if(usb != null){  
            version = usb.getVersion();  
        }  
    }  
}
```

Optional Object creation

Optional object can be created in following ways.

1. Using static `empty()` method of `Optional` class.

```
Optional<String> optionalEmptyObj = Optional.empty();
```

2. Using static `of()` method of `Optional` class. The value passed to `of()` method should be non Null

```
Optional<String> optionalNonEmptyObj = Optional.of("Test");
```

3. Using static `ofNullable()` method

```
Optional<String> optionalNullableObj =  
Optional.ofNullable(null);
```

Value checking for Optional object

We can use `isPresent()` function to check if the optional object's value is non null.

For Instance:

```
Optional<String> optionalNonEmptyObj =  
Optional.of("Test");
```

```
System.out.println(optionalNonEmptyObj.isPresent());
```

ifPresent method

ifPresent() method is used to execute a functionality based on whether the value of an optional object is present or not.

For instance: Before Java 8, we used to check the null value of a variable through if condition and then execute some statement executing that variable. Something like

```
if(name != null){  
    System.out.println("Name: "+name);  
}
```

ifPresent method (Continued)

From Java 8, using the Optional type , we can write the same code in much simpler way as shown below.

```
Optional<String> optionalNonEmptyObj =  
Optional.of("Prateek");  
  
optionalNonEmptyObj.ifPresent((name) ->  
System.out.println("Name: "+name));
```


ifPresentOrElse method (Java 11)

The method is to execute a consumer function in case value is present else execute a runnable function as a default action.

For Instance:

```
Optional<String> optionalEmptyObj = Optional.empty();
```

```
optionalEmptyObj.ifPresentOrElse((s) ->  
System.out.println(s),() -> System.out.println("Value  
unknown"));
```

orElse and orElseGet method

Methods are used to return a string or call a supplier function in case value is not present in the optional object.

```
Optional<String> optionalNonEmptyObj =  
Optional.of("Prateek");
```

```
optionalNonEmptyObj.orElse("Hello");
```

```
optionalNonEmptyObj.orElseGet()->"Hello");
```

Get method

Method to get the value of the optional object.

```
Optional<String> optionalNonEmptyObj =  
Optional.of("Prateek");
```

```
System.out.println(optionalNonEmptyObj.get());
```

If we tries to get the value of a null object, then the call to the get method will throw `NoSuchElementException`.

```
Optional<String> optionalNullableObj =  
Optional.ofNullable(null);
```

```
optionalNullableObj.get();
```

Map method

The method is used to perform some action on the wrapped value of optional object and returns a new optional object.

```
String text = "My name is Prateek";
```

```
Optional<String> optionalText = Optional.of(text);
```

```
optionalText.map((s) -> s.length()).ifPresent((l) ->  
System.out.println("Length of the text: "+l));
```

Filter method

The method is used to apply a test condition to the optional object and return the object as it is if the condition is satisfied.

```
String text = "My name is Prateek";
```

```
Optional<String> optionalText = Optional.of(text);
```

```
System.out.println(optionalText.filter((s) ->  
s.contains("Prateek")).get());
```

```
System.out.println(optionalText.filter((s) ->  
s.contains("Test")).orElse("No string present"));
```

Equals method

Two optional objects are equal when

- It is also an Optional and;
- Both instances have no value present or;
- The present values are "equal to" each other via equals().

hashCode method

Returns hash code of the optional object if value is present or 0 if not.

Stream method

Method is used to convert the Optional object into a Stream of the contained value.

Thank You