Professional Skills
Agile Fundamentals
Jira
Git
Databases Introduction
Java Beginner
Maven
Testing (Foundation)
Java Intermediate
● Optionals
○ JDBC CRUD
○ Exceptions
○ SOLID Principles
○ Single Responsibility
○ Open/Closed
○ Liskov Substituiton
○ Interface Segregation
○ Dependency Inversion
○ Best Practice
○ Design Patterns
○ Creational Design Patterns
○ Structural Design Patterns
○ Behavioural Design Patterns
○ Collection & Map
○ HashSets
○ HashMaps
○ Enums
○ Logging
○ Generics
○ Lambda Expressions
○ Streams
○ Complexity
○ Input and Output
○ Local Type Inference
HTML
CSS

Optionals

Contents

Overview

Optional is a wrapper class in Java that can be used to contain not null objects.
This class has a variety of utility methods to help us with handling values as "available" or "not available" as opposed to checking null values.
Optional is a value based class, so avoid using == for equality checks.
We should also avoid passing optionals as parameters as we would then have to run a null check within the method, thus defeating the purpose of using the Optional wrapper class.

Tutorial

The following code snippets show you how you can wrap a value with the Optional wrapper class, and shows you what sort of methods the class provides.
Optionals are very useful when it comes to interacting with Databases because we don't know whether they will contain values or not.

Import

To import the optionals class use the following.

```
import java.util.Optional;
```

ofNullable

We can assign a value to an optional with the ofNullable method.
If we were to use Optional.of(item) it would throw an exception if it is given a null value.

```
String item;
if(Math.random() >= 0.5) {
    item = "Hello";
} else {
    item = null;
}
Optional.ofNullable(item);
```

Any reference to item in the following code snippets is referencing the item variable above.

isPresent

The isPresent method returns a boolean telling us whether a record was found or not.

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

orElse

The orElse method will return the item if it is not null or return whatever we specify if it is.

```
System.out.println(item.orElse("Item not found"));
```

map

We can perform maps, filters and other methods that take lambda expressions to perform some kind of functionality on the optional.

```
item = item.map(theItem ->  "goodbye");
item.ifPresent(theItem -> System.out.println(theItem + "found"));
```

orElseThrow

If an item is null then we can throw our own custom exceptions.

```
System.out.println(item.orElseThrow(() -> new RuntimeException("Item Not
Found")));
```

get

To retrieve the item within the optional we can use the get method, this will throw an error if the optional contains a null value.

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

Exercises

There are no exercises for this module.