CS2030 (2310) Exercise #4: Maybe

Tags & Categories

Tags:

Categories:

Related Tutorials

Task Content

Maybe

Topic Coverage

- Generics
- Bounded and unbounded wildcards
- Static methods
- Functional interfaces

Problem Description

In this exercise, we are going to extend our generic `Maybe` context first introduced during the lecture with several other higher order functions, much like Java's `Optional` class.

Task

You are given the `Maybe` class with the following methods defined:

- `static <T> Maybe<T> of(U thing)`
- `static <T> Maybe<T> empty()`
- `public String toString()`

You are to implement the additional methods:

- `equals`
- `filter`
- `ifPresent` and `ifPresentOrElse`
- `or`, `orElse` and `orElseGet`
- `map` and `flatMap`

The `Maybe.java` class is provided to you. You may refer to the Java API on the `Optional` class for the implementation details.

Level 1

Write the overriding `equals` method that takes in `other` of type `Object`. The other object is considered equal if:

- it is also a `Maybe` and;
- both instances contain no value or;
- the contained values are "equal to" each other via their respective `equals` method.

```
jshell> Maybe<Integer> mi = Maybe.<Integer>of(1)
mi ==> Maybe[1]
```

```
jshell> Maybe<String> ms = Maybe.<String>of("1")
ms ==> Maybe[1]

jshell> Maybe<Integer> ei = Maybe.<Integer>empty()
ei ==> Maybe.empty

jshell> Maybe<String> es = Maybe.<String>empty()
es ==> Maybe.empty

jshell> mi.equals(mi)
$.. ==> true

jshell> mi.equals(ms)
$.. ==> false

jshell> mi.equals(1)
$.. ==> false

jshell> mi.equals(ei)
$.. ==> false

jshell> mi.equals(es)
$.. ==> false

jshell> ei.equals(es)
$.. ==> true
```

Level 2

Write the `filter` method which takes in an appropriately type-parameterized `Predicate` such that if the value encapsulated in the `Maybe` exists and the value matches the predicate, the `Maybe` is returned, otherwise `Maybe.empty` is returned.

```
jshell> Maybe<Integer> mi = Maybe.<Integer>of(1)
mi ==> Maybe[1]

jshell> mi.filter(x -> x % 2 == 0)
$.. ==> Maybe.empty

jshell> mi.filter(x -> x % 2 == 1)
$.. ==> Maybe[1]

jshell> mi.filter(x -> x % 2 == 0).filter(x -> x > 0)
$.. ==> Maybe.empty

jshell> mi.filter(x -> x % 2 == 1).filter(x -> x > 0)
$.. ==> Maybe[1]

jshell> Predicate<Object> pred = x -> x.hashCode() == 1
pred ==> $Lambda$21/0x00000008000ac840@675d3402

jshell> mi.filter(pred)
$.. ==> Maybe[1]
```

Level 3

Write the methods `ifPresent` and `ifPresentOrElse` that takes in an appropriately type-parameterized `Consumer` as an action. If there exists a value in the `Maybe`, apply the action on this value; otherwise do nothing. For the case of `ifPresentOrElse`, the method takes in an additional parameter `Runnable` whereby the runnable will be invoked when there is no value. Note that both methods have a `void` return type, and that `java.lang.Runnable` is a non-generic functional interface.

```
jshell> Maybe<Integer> mi = Maybe.<Integer>of(1)
mi ==> Maybe[1]

jshell> mi.filter(x -> x % 2 == 0).ifPresent(x -> System.out.println(x))

jshell> mi.filter(x -> x % 2 == 1).ifPresent(x -> System.out.println(x))
1

jshell> mi.filter(x -> x % 2 == 0).
   ...> ifPresentOrElse(x -> System.out.println(x),
   ...> () -> System.out.println("No value"))
No value

jshell> mi.filter(x -> x % 2 == 1).
   ...> ifPresentOrElse(x -> System.out.println(x),
   ...> () -> System.out.println("No value"))
1

jshell> Consumer<Object> consumer = x -> System.out.println(x.hashCode())
consumer ==> $Lambda$25/0x00000008000ad840@1b604f19

jshell> Runnable action = () -> {
   ...>    for (int i = 3; i >= 0; i--) {
   ...>       System.out.print(i + " ");
   ...>    }
   ...>    System.out.println("!");
   ...> }
action ==> $Lambda$26/0x00000008000adc40@457e2f02

jshell> Maybe<String> ms = Maybe.<String>of("one")
ms ==> Maybe[one]

jshell> ms.filter(x -> x.equals("ONE")).
   ...> ifPresentOrElse(consumer, action)
3 2 1 0 !

jshell> ms.filter(x -> x.equalsIgnoreCase("ONE")).
   ...> ifPresentOrElse(consumer, action)
110182
```

Level 4

Write the method `orElse` that takes in an alternative value of type `T`. If the value exists in `Maybe`, then return that value; otherwise return the alternative value.

A similar method is `orElseGet` that takes in an appropriate type-parameterized `Supplier` that produces the alternative value instead. In this case, if a value does not exist in `Maybe`, the value produced by the supplier is returned.

Yet another variant is the `or` method which takes in a `Supplier` of a `Maybe<T>` as the alternative. If a value does not exist in `Maybe`, the alternative `Maybe` produced by the supplier is returned.

```
jshell> Maybe<Integer> mi = Maybe.<Integer>of(1)
mi ==> Maybe[1]

jshell> mi.filter(x -> x % 2 == 0).orElse(2)
$.. ==> 2

jshell> mi.filter(x -> x % 2 == 1).orElse(2)
$.. ==> 1

jshell> mi.filter(x -> x % 2 == 0).orElseGet(() -> 2)
$.. ==> 2

jshell> mi.filter(x -> x % 2 == 1).orElseGet(() -> 2)
$.. ==> 1

jshell> Maybe<Object> mo = Maybe.<Object>of(1)
mo ==> Maybe[1]

jshell> Supplier<Integer> supp = () -> 2
supp ==> $Lambda$21/0x00000008000ac840@51565ec2

jshell> mo.filter(x -> x.hashCode() == 0).orElseGet(supp)
$.. ==> 2

jshell> mo.filter(x -> x.hashCode() == 1).orElseGet(supp)
$.. ==> 1

jshell> Supplier<Maybe<Integer>> suppMaybe = () -> Maybe.<Integer>of(2)
suppMaybe ==> $Lambda$24/0x00000008000adc40@5c7fa833

jshell> mo.filter(x -> x.hashCode() == 0).or(suppMaybe)
$.. ==> Maybe[2]
```

```
jshell> mo.filter(x -> x.hashCode() == 1).or(suppMaybe)
$.. ==> Maybe[1]
```

Level 5

Finally, write the methods `map` and `flatMap` that takes an appropriately type-parameterized `Function` where the resultant value of the function is a `Maybe`.

```
jshell> Maybe<String> ms = Maybe.<String>of("123")
ms ==> Maybe[123]

jshell> Function<String,Integer> f = x -> x.length()
f ==> $Lambda$15/0x00000008000aa840@7dc7cbad

jshell> ms.map(f)
$.. ==> Maybe[3]

jshell> Function<String,Maybe<Integer>> g = x -> Maybe.<Integer>of(x.length())
g ==> $Lambda$16/0x00000008000aac40@1753acfe

jshell> ms.map(g)
$.. ==> Maybe[Maybe[3]]

jshell> ms.flatMap(g)
$.. ==> Maybe[3]

jshell> Maybe<Object> mo = ms.flatMap(g)
mo ==> Maybe[3]
```

Hint: Refrain from casting to the `Maybe` type and using `SuppressWarnings` to quell the unchecked type warning. You should just create a `Maybe<R>` and return it, just like `map`.

Submission (Course)

Select course: CS2030 (2023/2024 Sem 1) - Programming Methodology II

Your Files:

BROWSE

SUBMIT (only .java, .c, .cpp, .h, .jsh, and .py extensions allowed)

To submit multiple files, click on the Browse button, then select one or more files. The selected file(s) will be added to the upload queue. You can repeat this step to add more files. Check that you have all the files needed for your submission. Then click on the Submit button to upload your submission.

© Copyright 2009-2023 National University of Singapore. All Rights Reserved.
Terms of Use | Privacy | Non-discrimination

MySoC | Computing Facilities | Search | Campus Map
School of Computing, National University of Singapore