Optional 자료

- http://www.daleseo.com/java8-optional-before/
- http://www.daleseo.com/java8-optional-after/
- http://www.daleseo.com/java8-optional-effective/

실습 환경

- jshell 이용 (인터프리터)

1. `jshell -v`

   혹은

2. jshell 수행 후
   `/set feedback verbose`

3. Optional import
   `import java.util.Optional`

Empty (undefined Value ⊥)

- 타입과 값 : T x 형태의 타입과 수식
  - 이론 적으로 타입은 원소들의 집합으로 해석함
  - int x 는 x의 값(value)이 int 타입의 원소임을 의미함.
- 자바의 Wrapper 타입
  - 자바만의 특징 : Integer는 int의 wrapper 타입으로서, Integer 객체는 메모리 공간을 할당 받음을 표현하는 타입.
  - 일반적인 수 10 은 int 타입이며, new Integer(10) 의 객체가 Integer 타입의 원소이다
- empty (Undefined Value ⊥)
  - 2/0, 3/0, 5/0 등의 산술식의 값이 정의되지 않는다 (undefined) 라고 함.
  - 모든 undefined 값은 모두 동일하게 처리되며, 이를 기호 ⊥ 로 표현함.
  - Java에서는 Optional.empty() 로서 생성됨

Optional<T> 와 T 의 차이점

- T가 Integer일 때
  - Integer : 임의의 한 정수
  - Optional<Integer> : 임의의 한 정수뿐만이라 empty 도 함께 포함될 수 있는 타입
- Optional<T>
  - T 타입의 값을 Optional로 wrapping 하는 타입 (wrapper 클래스), 혹은 container.
  - T는 Optional 타입의 인수(parameter) 임. 즉, Optional 타입은 T 타입을 인수로 취하는 함수임.
- T 대신 Optional<T>를 사용해야 하는 경우
  - 예) (x / y) 에서 y가 0 일 경우, exception 발생. 프로그램 측에서는 아무런 대응을 할 수 없음. (x / y) 의 타입을 Float라고 할 때, Float가 이 연산에 대한 값을 표현 못함.
  - 타입을 Optional<Float> 형태로 표현하면, y가 0 일 경우, 그 값을 empty 로 취급하면서 프로그래밍을 진행할 수 있음

- Optional 타입 객체의 두 부류
  - empty 이 거나
  - empty 가 아닌 객체 (실제 값이 있는 객체)

- Empty는 Singletone Instance
  - empty 객체는 모든 타입의 원소가 될 수 있다.
  - 즉, empty는 Optional<Integer>, Optional<Boolean>, … 등의 타입을 가질 수 있다.

- 주의 : Optional.empty() 는 null 과 다르다.
  - Optional.empty() // Optional 내에 존재하는 empty 값
  - null // (문제가 있는) 일반 객체
  - null을 ofNullable() 메소드로 Optional에 넣을 때, 이것은 empty로 매핑 됨.
  - Optional 내에서는 empty에 대한 연산만 할 뿐, null에 대한 연산을 할 수 없음
  - Optional<String> s = null 의 표현은 존재하지만, 실제로는 무의미 함.

Class Optional<T>

static <T> Optional<T>	empty()
static <T> Optional<T>	of (T value)
static <T> Optional<T>	ofNullable (T value)
T	get()
void	ifPresent (Consumer<T> action)
boolean	isPresent()
Optional<T>	or□(Supplier<Optional<T>> supplier)
T	orElse□(T other)
<U> Optional<U>	map□(Function<T, U> mapper)
<U> Optional<U>	flatMap□(Function<T, Optional<U>> mapper)

Optional 컨테이너 안의 값 넣기

- Optional.empty()
  - `Optional.empty() : empty (undefined 값) 을 의미함.`
  - `Optional<Boolean> b = Optional.empty()`
  - `Optional<Integer> n = Optional.empty()`

- `Optional.of(value)`
  - 어떤 null이 아닌 값을 `Optional<T>` 타입의 객체로 변환시킴 (lifting)
  - 만약 value가 null이면 NPE 에러 발생

- `Optional.ofNullable(value)`
  - null일 경우 empty 로 변환시킴
  - null 이 아닌 객체를 `Optional<T>` 컨테이너로 집어 넣음 (lifting)

Optional.empty() vs null

```
jshell> var o1 = Optional.of(null)
  Exception java.lang.NullPointerException

jshell> Optional<String> str = null                    // 사용하지 말아야 할 표현

jshell> var o1 = Optional.ofNullable(str)              // OK, null이 empty로 변환됨
o1 ==> Optional.empty

jshell> o1.isPresent()                                 // empty 가 아니면 true, empty이면 false
$21 ==> false

jshell> var o3 = Optional.empty()
o3 ==> Optional.empty
```

Optional에 empty를 넣는 방법은 Optional.empty(), 혹은 Optional.ofNullable(null) 두 가지 방법이 있다.
Null은 일반 객체에서, empty는 Optional 타입 내에서 적용됨
Optional 타입의 값을 null 로 return 하는 메소드는 없다

ifPresent(Consumer<? super T> consumer)

- isPresent() 와 혼동하지 말 것

- Optional.ofNullable(null).ifPresent(x -> System.out.println(x))
  // do nothing

- jshell> Optional.ofNullable("abc").ifPresent(x -> System.out.println(x))
  "abc" 화면 출력

- Optional<T> 타입의 값 만들기

  - Optional.empty()

    예) Optional<Boolean> b = Optional.empty()
        Optional<Integer> i = Optional.empty()

  - Optional.of(value) : null 이 아닌 T 타입의 값일 때

  예) Optional<Boolean> b2 = Otional.of(true)
      Optional<Integer> n = Optional.of(123)

  - Optional.ofNullable(value) : null이거나 null 이 아닌 두 가지 모두 사용가능

  예) Optional<Integer> n3 = Optional.ofNullable(null)
      Optional<Integer> n4 = Optional.ofNullable(567)

Optional 컨테이너 안의 값 추출

- get() : empty 아닌 값 꺼내 오기 : Optional<T> => T

```
n4.get() => 567
```

- orElse(T value) : empty 일 때, 주어지는 value로 return

```
Optional<Boolean> b = Optional.empty()
b.orElse(false)   => false
n4.orElse (new Integer(100)) => 567
n3.orElse (new Integer(100)) => 100        // empty 경우
```

- isPresent() : Optional<T> 객체의 값이 empty 인지 여부 판단

```
n3.isPresent() => false   (n3 는 empty)
n4.isPresent() => true    (n4 는 empty 아님)
```

map

- Haskell의 map 타입
  - map :: (a -> b) -> (m a -> m b) (여기서 m은 Optional 등의 parameterized 타입)
  - 개념적으로 Java 8에서 m 은 Stream 이나 Optional 등이 될 수 있음
- Optional 에서의 map : obj.map(functionalMethod)
  - obj가 Optional 타입의 객체이면, map을 적용한 결과 또한 Optional 타입이다.
  - functionalMethod는 람다 식이나 Functional Interface 의 메소드

- 예)

```
Optional<Boolean> t = Optional.of(true)
t.map(x -> !x)   => Optional[false]  :  Optional<Boolean> 타입의 false
Optional<Boolean> f = t.map(x -> !x)
f   => Optional[false]
```

map 은 structure-preserving 함수

- obj.map(f) 타입
  - obj 의 타입 : Optional<S>
  - f 의 타입 : S -> T 형태의 함수
  - obj.map(f) 결과의 타입 : Optional<T>
  - obj 가 empty 인 경우, obj.map(f) 결과의 값 또한 empty

- 예)
  - jshell> Optional.of(2).map(f = x -> x == 1 ? 10 : null) // Optiona.empty
  - jshell> Optional.of(1).map(f = x -> x == 1 ? 10 : null) // Optional[10]
  - jshell> Optional.empty().map(x -> x) // Optional.empty

```
Optional.ofNullable(order)
            .map(Order::getMember)
            .map(Member::getAddress)
            .map(Address::getCity)
            .orElse("Seoul");
```

각 연산 과정 도중 한 번이라도 empty 값이면, 전체 결과 값은 empty 이다.

filter

- map 과 유사한 구조
- obj.filter(predicateMethod) :
  - predicateMethod는 결과 값이 true/false
  - predicateMethod를 적용한 결과 값이 true일 때만 obj가 선택되고, false일 때는 empty
- 예)
  - `Optional<Boolean> t = Optional.ofNullable(true)  // Optional[true]`
  - `Optional<Boolean> f = t.map(x -> !x)  // Optional[false]`
  - `t.filter(x -> x)  // Optional[true]`
  - `t.filter(x -> !x)  // Optional.empty`
  - `f.filter(x -> !x)  // Optional[false]`
  - `Optional<Integer> n = Optional.of(555)`
  - `n.filter(x -> x > 100)  // Optional[555]`
  - `n.filter(x -> x < 100)  // Optional.empty`

```
jshell -v (혹은 /set feedback verbose)

var n1 = Optional.ofNullable(new Integer(2))
var n2 = Optional.ofNullable(null)
var n3 = Optional.empty()
Optional<Integer> n4 = Optional.ofNullable(new Integer(2))

n1.get()                                    // 2
n4.orElse(new Integer(100))                 // 2
n2.orElse(new Integer(100))                 // 100

n1.map(x -> x * x)                          // Optional[4]
n1.map(x -> x < 10)                         // Optional[false]
n2.map(x -> x)
n1.map(x -> null)                           // Optional.empty()
n2.map(x -> x + 2)                          // error 타입 문제


Optional<Boolean> b = Optional.empty()
b.map(x -> !x)                              //empty

n1.filter(x -> x < 4)                       // Optional[2]
n1.filter(x -> x > 4)                       // Optional.empty
n2.filter(x -> x == null)                   // Optional.empty
n2.filter(x -> x != null)                   // Optional.empty
```