프로그래밍언어론 기말고사 준비 (2022 9 주~14 주)

주별 강의 내용 (강의자료 파일)

1. Week 9: Function application, β-reduction, η- conversion, Method Reference
   - LambdaCalculus.pdf (8, 9, 10, 11, 20, 21, 22 쪽)
   - Lambda.java, LambdaEx.java (jshell 로 실습)
   - lambda.scala
   - LambdaCalculus.pdf (23 쪽)
   - funInterEx.pdf (메소드 레퍼런스 표현)
   - QsortStream2.java
   - Student.java

2. Week 10: Java 의 Optional
   - Optional.pdf
   - NpeOrder.java

3. Week 11: Haskell 3, 4 장, twice 함수 정의하기
   - function types
   - curried function
   - Polymorphic (generic) function
   - Overloaded function
   - 재귀함수
   - Twice 함수를 Haskell, Java, Scala 로 정의하기
     (TwiceGen.java, twice.hs, twice.scala)

4. Week 12: Haskell 의 타입추론 및 Abstract Data Type 정의하기
   - poly 폴더 / PolyType.pdf typeInf.pdf
   - Haskell 8 장
   - Java, Scala, Kotlin 코딩: exp 폴더
     shape.hs shape.scala
     ShapeDemo.java ExpEval.java
     exp.kt
     expEval.scala expEvalCase.scala enum-expr.scala

5. Week 13: Lazy Evaluation
   - Haskell 의 infinite list 와 take 함수

- Java Stream 의 limit 함수
- Scala 의 call-by-name 함수 정의, LazyList
- lazy 폴더 / take.hs Take.java Lazy.java callByname.scala fib.hs LazyFib.scala

6. Week 14: Monad (Haskell 의 >>=, Java 의 flatMap)
   - 강의자료 : practMon.pdf sigpl15-Monad.pdf
   - StreamFlatMap.java
   - StringFlatMap.png stringFlatMap.hs StringFlatMap.java
   - Optional vs Maybe: AddOpt.java

예상 문제

1. 다음 람다 식의 β-reduction 적용 과정을 표현하라.
   ((λx. λy. y + x) 5) 7

2. Java 을 이용하여 위의 람다 식 정의 및 β-reduction 적용 과정을 표현하라 (람다 식의 이름은 plus 로 하고, 이에 적절한 타입 표현 및 apply 상황을 표현하라.)

3. Java 코드 var list = List.of(2,3,4) 가 주어졌을 때, stream, map, forEach 를 적용하여 다음을 계산하라 (jshell 로 확인 해 볼 것)
   a. list 의 각 원소를 제곱하여 list2 를 계산하라 (즉, list2 ==> [4, 9, 16])
   b. list 의 각 원소를 제곱한 결과를 표준 출력 (println) 하는 상황을 두 가지로 코딩하라 (람다 식 및 method reference 적용)

4. 파일 funInterEx.pdf 의 (5.3 Method Reference) 코딩 예 문제 풀어 보기

5. QsortStream2.java 이해하기
   a. stream1 과 stream2 는 동일한 데이터를 반복 적용하여 정의한다. 처음 얻어진 stream1 을 그대로 stream2 에 사용할 수는 없는가? 이에 대한 이유를 설명하라.
   b. Stream 의 이러한 특성을 Haskell 의 List 와 비교하라 (유사성, 차이점을 설명하라. 문제 12. d 와 동일함)
   c. 타입 `Supplier<IntStream>` 은 무엇을 의미하는가?
   d. 코드 `IntStream upto50 = supplier.get().filter(n -> n <= 50);` 에서 `get()` 메소드를 적용하는 이유는 무엇인가?

6. Student.java
   a. 다음 코드는 어떤 기능을 하는가?
   ```
   Map<String, List<Stud>> studByName =
   studs.stream().collect(Collectors.groupingBy(Stud::getName))
   ```
   b. 다음 코드에서 sort 를 수행한 결과는 어디에 저장되는가?
   Compator.comparing 메소드의 파라메터에 어떤 내용이 표현되어야 하는가?
   ```
   Collections.sort(studs, Comparator.comparing(Stud::getScore));
   ```

7. Optional.pdf
    a. 세 타입 int, Integer, Optional<Integer> 는 각각의 값에 있어서 어떤 차이를 보이는가?
    b. Optional.empty() 의 타입은 Optional<T> 로 표현된다. 이것은 무엇을 의미하는가?
    c. Optional.of(value) 와 Optional.ofNullable(value) 는 어떤 다른 점이 있는가?
    d. 다음의 코드를 가정한다.

```
class A { static Boolean even(Integer n) { return n % 2 == 0;} }
Optional<Integer> n = Optional.of(3);
Predicate<Integer> evenLambda = n -> n % 2 == 0;
Optional<Boolean> m;
```

n 이 짝수이면 true, 홀수이면 false 의 값을 m 에 할당하는 코드를 다음과 같은 다양한 방법으로 코딩하라.

(1) even 메소드 이용 :

(2) evenLambda 이용 :

(3) map 의 mapper 함수를 를 람다 식으로 표현 :

(4) flatMape과 람다 식을 이용:

(5) orElse 가 활용되기 좋은 경우에 대해서 설명하라. 또한, get 과 orElse 의 차이점을 설명하라.

8. Haskell
    a. 타입 체킹 (type checking)과 타입 추론(type inference)에 대해서 비교 설명하라.
    b. Java 에서 타입 추론이 적용되는 예를 들어라.
    c. 다형성(Polymorphism) 은 Polymorphic 함수와 overloaded 함수로서 구현된다. 다형성의 핵심 개념 및 이 개념이 Java 에서 적용되는 상황을 간단히 설명하라.

    d. 다음 Haskell의 first, p, g, h 함수의 추론 결과는 무엇인가?

```
first x y z = x
p x = ['a', x, 'z']
g f (x, y) = f x y
h = g . first
```

9. (Haskell 과 Java 코드) 다음 Haskell 을 고려하여 물음에 답하라.

```
twice1 f x = f (f x)
twice2 f x = (f . f) x            -- function composition
twice3 f = \x -> (f . f) x
twice4 f = f . f                  -- from twice3, eta conversion
twice5 = \f -> \x -> f (f x)
twice6 = \f -> f . f              -- from twice4, point-free
```

a. twice1 과 twice2 의 타입을 써라.

b. twice1 ~ twice6 은 파라메터를 2 개, 1 개 0 개의 갖는 형태로 정의되어 있으며, 함수의 합성, 람다 식, eta conversion 등의 원리에 따라 다양하게 정의될 수 있음을 보여 준다. 이 함수에 각각에 대한 Java 함수를 정의하라.

c. `twice3 (\a -> a+2) 6` 의 reduction 과정을 보여라. 또한 이에 해당하는 Java 의 twice3 함수 호출을 코딩하라.

a.

b. Java 의 twice 함수

```
class TwiceGen <T,U,V> {
  T twice1 (Function<T,T> f, T x) {return f.apply(f.apply(x));}
  T twice2 (Function<T,T> f, T x) {return (f.andThen(f)).apply(x);}
  Function<T,T> twice3(Function<T,T> f) {return x -> (f.andThen (f)).apply(x);}
  Function<T,T> twice4(Function<T,T> f) { return f.andThen(f) ; }
  Function<Function<T,T>, Function<T,T>> twice5
      = (Function<T,T> f) -> (T x) -> f.apply(f.apply(x));
 Function<Function<T,T>, Function<T,T>> twice6
    = (Function<T,T> f) -> f.andThen(f);
}
```

c. Haskell 의 reduction

```
twice3 (\a -> a+2) 6
= (\x -> ((\a -> a+2) . (\a -> a+2)) x) 6
= ((\a -> a+2) . (\a -> a+2)) 6
= (\a -> a+2)((\a -> a+2) 6)
= (\a -> a+2) (6+2) = (\a -> a+2) 8
= 8+2 = 10
```

10. Algebraic Data Type: shape.hs 와 ShapeDemo.java

a. shape.hs 에 해당하는 ShapeDemo.java 를 코딩해 보기

b. Java 의 객체 생성은 Haskell 에서 어떻게 표현되는가?

c. Haskell 의 pattern-matching 은 어떤 기능을 하며, 이것은 Java 에서 어떻게 표현되는가?
d. Haskell 의 Algebraic Data Type 은 Sum (or, |) 과 Product 타입으로 구성된다. Product 란 Constructor 다음에 일정하게 주어지는 파라메터들을 의미함. Algebraic Data Type 은 Java 에서 어떻게 정의되는가?
e. Haskell 에서 rect1 은 생성자 Rect 를 사용하여 생성되었지만, react2 는 일반 함수를 실행한 결과이다. 이 상황이 Java 에서 어떻게 코딩되는가?

11. Algebraic Data Type: ExpEval.java
    a. 수식 1 * (2 + 3) + 4 에 대한 Abstract Syntax Tree 를 그리고, 이를 Haskell 로 코딩하라.
    b. Haskell 코딩을 Java 로 표현하기

12. Lazy Evaluation
    a. Haskell 로 map 함수를 정의하라.
    b. Haskell 로 take 함수를 정의하라.
    c. 다음 Java 코드에 해당하는 Haskell 코드를 써라.

```
IntStream.iterate(5, i -> i+1)
  .limit(3)
  .boxed()
  .collect(Collectors.toList());
```

    d. Java 의 Stream 과 Haskell List 의 공통점과 차이점을 써라.

    e. Java, Scala, Haskell 언어는 각각 어떤 형태의 파라메터 패싱 방법을 적용하고 있는가? 또한 lazy evaluation 을 표현하기 위해 어떤 방법을 적용하고 있는가?

13. flatMap 함수
    a. flatMap 함수의 타입을 써라

    b. 다음 Haskell 코드에 해당되는 Java 코드를 써라.

```
Prelude> [1,2,3] >>= \x -> [x, x*x]
[1,1,2,4,3,9]

List<Integer> list = List.of(1,2,3);
list.stream().flatMap(x -> _______)
.collect(Collectors.toList());
```

c. 위의 코드는 map 함수로 구현할 수 없다. map 은 flatMap 에 비해서 어떤 기능이 떨어지는가?

d. 다음 map 으로 정의된 코드를 flatMap 으로 정의하라.

```
list.stream().map(x -> x*x).collect(Collectors.toList());
```

```
Optional of(3).map(x -> x*x)
```

e. 다음 filter 로 정의된 코드를 flatMap 으로 정의하라.

```
list.stream().filter(x -> (x>=2)).collect(Collectors.toList());
```

f. 다음 코드 실행 결과를 써라.

```
Optional.of(Optional.of(2)).flatMap(x -> x)
```