프로그래밍 공부의 활용!

람다(Lambda)

람다식 (Lambda Expression)

메서드를 하나의 '식(expression)'으로 간략하면서도 명확하게 표현한것

```
(매개 변수, ...) -> { 실행문 }
```

람다식 문법(Lambda expression syntax)

1번 타입 생략 가능

```
(int a, int b) -> a + b;

(a,b) -> a + b;
```

람다식 문법(Lambda expression syntax)

2번 매개 변수가 하나인 경우 괄호 생략 가능

```
a -> { a * a; }
```

람다식 문법(Lambda expression syntax)

3번 람다식의 바디 부분에 하나의 표현식만 오는 경우 중괄호 생략 가능

```
a -> a * a
```

람다식 문법(Lambda expression syntax)

4번 람다식의 바디에 'return' 문이 있는 경우 중괄호를 생략 할 수 없음

```
OK
(a, b) -> { return a > b ? a : b }

Error
(a, b) -> return a > b ? a : b
```

람다식 문법(Lambda expression syntax)

5번 중괄호 생략 하고 싶으면 표현식으로 변경하여 사용함

```
OK
(a, b) -> a > b ? a : b
```

람다식 사용

1. 인터페이스 생성

2. 인터페이스에 람다식으로 구현할 메서드를 선언

함수형 인터페이스(Functional Interface)

람다식을 저장할 수 있는 변수는 '함수형 인터페이스(functional interface)'타입이여야 함

```
인터페이스명 변수 = 람다식;
```

함수형 인터페이스(@FunctionalInterface) 정의

함수형 인터페이를 정의하고 '@FunctionalInterface' 이노테이션을 붙여주면 자바 컴파일러가 함수형 인터페이스의 정의를 검증 해줌

```
@FunctinalInterface
interface 인터페이스명 {
    public 메소드명;
}
```

익명 클래스(Anonymous class)

이름이 없는 클래스를 의미 하며 객체 사용시에 클래스의 선언과 객체 생성이 동시에 이루어지며, 일회성으로 딱 하나의 객체만 필요할 경우 사용.

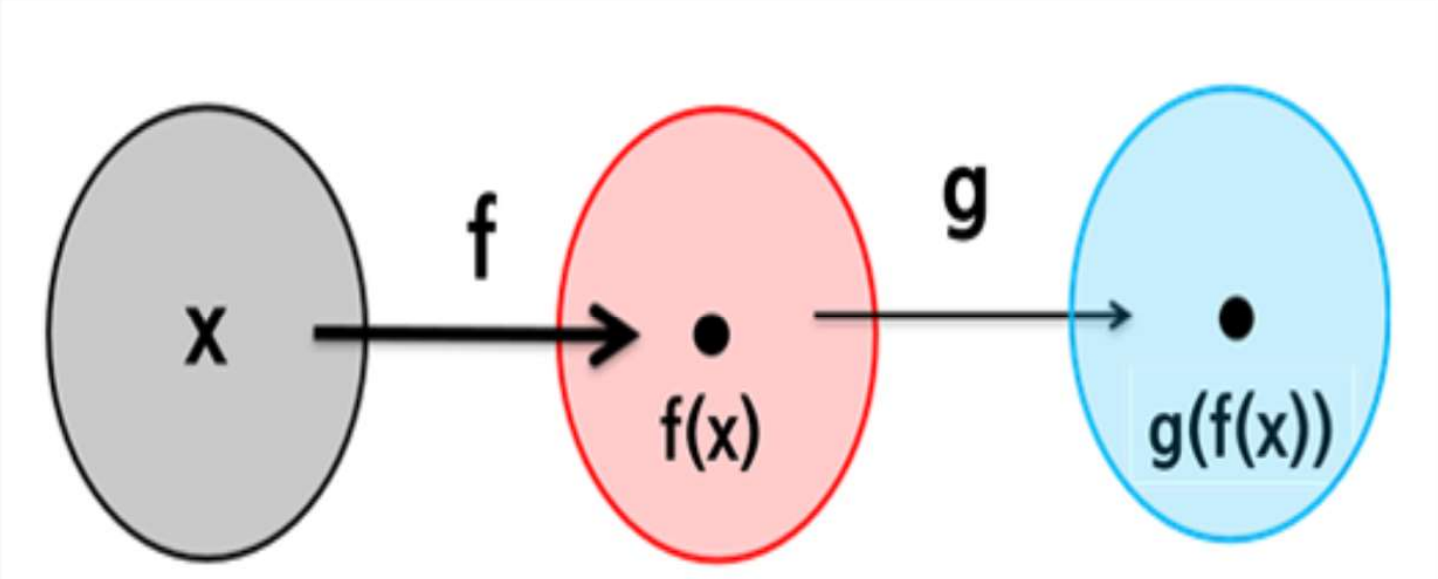
```
부모클래스명 변수명 = new 부모클래스명 { ... 내용 구현 ... };

인터페이스명 변수명 = new 인터페이스명 { ... 내용 구현 ... };
```

java.util.function 패키지

기본적인 함수형 인터페이스

함수형 인터페이스	메서드
java.lang.Runnable	void run();
Supplier<T>	T get();
Consumer<T>	void accept(T t);
Function<T, R>	R apply (T t);
Predicate<T>	boolean test(T t);

Function의 합성




See you
Again!
다음 이 시간에 만나요!