람다식이란?

: 자바에서는 함수적 프로그래밍(병렬처리와 이벤트 지향 프로그래밍에 적합)을 위해 자바 8부터 람다식을 지원했다. 람다식은 익명함수를 생성하기 위한 식으로 객체 지향 언어보다는 함수지향 언어에 가깝다. 자바 코드의 간결성, 컬렉션 요소들의 필터링 및 매핑을 위함이다.

람다식의 기본 문법

(타입 매개변수, …) -> {실행문; …}

* 예외 사항

1. 매개변수 타입은 런타임 시에 대입되는 값에 따라 자동으로 인식될 수 있어 적지 않아도 된다.
2. 매개변수가 하나라면, () 소괄호를 생략해도 된다.
3. 실행문이 하나라면, {} 중괄호도 생략할 수 있다.
4. 매개변수가 없다고 해도 빈 중괄호 ()를 반드시 입력해야 한다.
5. 실행문에 return문만 있을 경우, return문을 사용하지 않고 작성하는 것이 정석이다.

Ex) (x, y) -> x + y // 이는 (x, y) -> return x + y 와 동일한 기능을 한다.

타겟 타입과 함수적 인터페이스

: 자바는 메소드를 단독으로 선언할 수 없다. 그렇기에 항상 클래스의 구성 멤버로 선언하게 된다.

람다식은 인터페이스 변수에 대입되며, 이 말은 람다식은 인터페이스의 익명 구현 객체를 생성한다.

람다식은 대입될 인터페이스의 종류에 따라 작성 방법이 달라지기 때문에 람다식이 대입될 인터페이스를 람다식의 타겟 타입이라고 한다.

함수적 인터페이스(@FuncationalInterface)

: 모든 인터페이스를 람다식의 타겟 타입으로 사용할 수는 없다. 람다식이 하나의 메소드를 정의하기 때문에 두 개 이상의 추상 메소드가 선언된 인터페이스는 람다식을 이용해서 구현 객체를 생성할 수 없다.

하나의 추상 메소드만 선언된 인터페이스를 람다식의 타겟 타입이 될 수 있는데, 이를 함수적 인터페이스라고 한다.

함수적 인터페이스를 작성할 때, 두 개 이상의 추상 메소드가 선언되지 않도록 컴파일러가 체킹해주는 기능이 있는데, @FunctionalInterface 어노테이션이다. ( 두 개 이상 선언 시, 컴파일 오류 생성)

로컬 변수 사용

: 람다식에서 바깥 클래스의 필드나 메소드는 제한 없이 사용할 수 있으나, 메소드의 매개변수 또는 로컬 변수를 사용하면 이 두 변수는 final 특성을 가져야 한다.

매개 변수 또는 로컬 변수를 람다식에서 읽는 것은 허용되지만 람다식 내부 또는 외부에서 변경할 수 없다.

표준 API의 함수적 인터페이스

Consumer : 매개값 O, 리턴값 X

Supplier : 매개값 X, 리턴값 O

Function : 매개값 O, 리턴값 O, 주로 매개값을 리턴값으로 매핑(타입변환)

Operator : 매개값 O, 리턴값 O, 주로 매개값을 연산하고 결과를 리턴

Predicate : 매개값 O, 리턴값 O, 매개값을 조사해서 true/false를 리턴

이 후에는 consumer supplier 등의 람다식 추상 메소드들에 대한 설명 다수 내포 교과서 참조

andThen()과 compose() 디폴트 메소드

: 디폴트 및 정적 메소드는 추상 메소드가 아니기 때문에 함수적 인터페이스에 선언되어도 여전히 함수적 인터페이스의 성질을 잃지 않는다. 위의 디폴트 메소드들은 두 개의 함수적 인터페이스를 순차적으로 연결하고, 첫 번째 처리 결과를 두 번째 매개값으로 제공해서 최종 결과값을 얻을 때 사용한다.( 일종의 파이프라인 )

Consumer, Function, Operator 종류의 함수적 인터페이스는 andThen(), compose() 디폴트 메소드를 가지고 있다.

\*\* 함수적 인터페이스 성질이란?

: 하나의 추상 메소드를 가지고 있고, 람다식으로 익명 구현 객체를 생성할 수 있는 것

Ex) 형식

인터페이스AB = 인터페이스A.andThen()(인터페이스B);

최종결과 = 인터페이스AB.method();

처리과정

: 인터페이스AB.method 호출 -> 인터페이스B(람다식) 호출 및 처리 -> 결과값 매개변수 전달 -> 인터페이스A 매개변수를 활용한 최종결과 리턴

꽤나 복잡한 과정이기에 사용 전에 교재를 몇 번 더 살펴보자