람다식(Lambda Expression)

람다식이 나오기 전과 후로 프로그램 구현 방식을 구분하자면 이전에 구현하던 방식을 '명령형 스타일', 람다식으로 구현하는 방식을 '함수형 스타일'이라고 한다.

명령형 스타일은 프로그램 구현 시 모든 작업내용을 자바 코드로 자세하게 작성하는 방식이다. 그래서 구현해야 할 코드양이 많다. 하지만 함수형 스타일은 개발자가 핵심내용만 구현하고 나머지는 자바 언어에서 자동으로 처리하는 방식이어서 코드가 간결하다.

람다식(Lambda Expression)은 메서드를 하나의 식(expression)으로 표현한 것

람다의 특징

람다식이란 '식별자없이 실행가능한 함수' **즉 익명함수이다.**

익명함수란 말그대로 함수의 이름이 없는 함수이다.

함수인데 함수를 따로 만들지 않고 간단한 익명함수를 쓰는 방식이다.

함수와 메소드 차이

- 근본적으로는 동일하지만 함수는 일반적용어, 메소드는 객체지향개념 용어이다.
- 함수는 클래스에 독립적 메소드는 클래스에 종속적이다.

람다식의 장단점

장점

1. 코드를 간결하게 만들 수 있다.
2. 식에 개발자의 의도가 명확히 드러나므로 가독성이 향상된다.
3. 함수를 만드는 과정없이 한번에 처리할 수 있기에 코딩하는 시간이 줄어든다.

단점

1. 람다를 사용하면서 만드는 무명함수는 재사용이 불가능하다.
2. 디버깅이 다소 까다롭다.
3. 불필요하게 너무 사용하게 되면 오히려 가독성을 떨어 뜨릴 수 있다.

람다식 사용법

(매개변수, ...) -> { 실행문 ... }

(매개변수, ...)는 오른쪽 중괄호 { } 블록을 실행하기 위해 필요한 값을 제공하는 역할을 한다. 매개 변수의 이름은 개발자가 자유롭게 지정할 수 있으며 인자타입도 명시하지 않아도 된다. **'->'** 기호는 매개 변수를 이용해서 중괄호 { } 바디를 실행한다는 뜻으로 해석하면 된다.

람다식 사용법

매개변수가 하나인 경우, 괄호() 생략가능(타입이 없을 때만)

(a) -> a * a;	a -> a * a; (O)
(int a) -> a * a;	int a -> a * a; (X)

매개변수의 타입이 추론 가능하면 생략가능

(int x, int y) -> x > y ? x : y;	(x, y) -> x > y ? x : y;

java.util.function 패키지의 표준 API

함수적 인터페이스를 사용하면 코드를 더 간결하고 가독성 있게 작성할 수 있으며, 다양한 함수형 인터페이스가 자바 표준 라이브러리에서 제공된다. 이러한 함수적 인터페이스들은 java.util.function 패키지 내에 정의되어 있다.

종류	매개값	리턴값	메소드 형태	활용
Consumer	O	X	accept	
Supplier	X	O	get...	
Function	O	O	apply...	매개값을 매핑(=타입변환)해서 리턴
Operator	O	O	apply...	매개값을 연산해서 결과 리턴
Predicate	O	O	test	매개값이 조건에 맞는지 확인해서 boolean 리턴