Lambda

Lambda식이란?

메서드를 하나의 식으로 표현한 것이며 익명 함수라고도 한다.

Lambda식 작성

메서드를 람다식으로 변환하는 과정은 다음과 같다.

1. 메서드의 이름과 반환 타입을 제거하고 ->를 블록{} 앞에 추가한다.
2. 반환 값이 있는 경우, 식이나 값만 적고 return문 생략 가능(끝에 ';'안 붙임)
3. 매개변수의 타입이 추론 가능하면 생략 가능

주의사항은 다음과 같다.

- 매개변수가 하나인 경우, 괄호() 생략 가능하다.(타입이 없을 때만)
- 블록 안의 문장이 하나뿐일 때, 괄호{} 생략 가능하다.(끝에 ';'안 붙임)
- 단, 하나뿐인 문장이 return문이면 괄호{} 생략 불가능하다.

정확히는 람다식은 익명 함수가 아니라 익명 객체이다.
그러므로 참조변수로 람다식을 참조할 수 있는데, 참조변수의 타입은 함수형 인터페이스 타입으로 하며
함수형 인터페이스의 메서드와 람다식의 매개변수 개수와 반환타입이 일치해야 하고 형변환이 필요하다.

함수형 인터페이스

함수형 인터페이스는 단 하나의 추상 메서드만 선언된 인터페이스이다.
함수형 인터페이스에 @FunctionalInterface 어노테이션을 붙여야 컴파일러가 올바르게 정의하였는지 확인해준다.

함수형 인터페이스 타입의 매개변수와 반환타입

메서드의 매개변수가 함수형 인터페이스라면, 이 메서드를 호출할 때 람다식을 참조하는 참조변수를 매개변수로 지정해야한다는 뜻이다.
메서드의 반환타입이 함수형 인터페이스라면, 이 함수형 인터페이스의 추상 메서드와 동등한 람다식을 가리키는 참조변수를 반환하거나 람다식을 직접 반환할 수 있다.

java.util.finction 패키지

자주 사용되는 다양한 함수형 인터페이스를 제공한다.
가능하면 이 패키지의 인터페이스를 활용하는 것이 재사용성이나 유지보수에 좋다.

자주 쓰이는 가장 기본적인 함수형 인터페이스는 다음과 같다.

함수형 인터페이스	메서드	설명
java.lang.Runnable	void run()	매개변수도 없고 반환값도 없다.
Supplier	T get()	매개변수는 없고, 반환값만 있다.

함수형 인터페이스	메서드	설명
Consumer	void accept(T t)	Suplier와 반대로 매개변수만 있고, 반환값이 없다.
Function<T, R>	R apply(T t)	일반적인 함수로, 하나의 매개변수를 받아서 결과를 반환한다.
Predicate	boolean test(T t)	조건식을 표현하는데 사용되며, 매개변수는 하나이다.

매개변수가 두 개인 함수형 인터페이스는 다음과 같다.

함수형 인터페이스	메서드	설명
BiConsumer<T, U>	void accept(T t, U u)	두 개의 매개변수만 있고, 반환값이 없다.
BiPredicate<T, U>	boolean test(T t, U u)	조건식을 표현하는데 사용되며, 매개변수는 두 개이다.
BiFunction<T, U, R>	R apply(T t, U u)	두 개의 매개변수를 받아서 하나의 결과를 반환한다.

매개변수의 타입과 반환타입이 일치하는 함수형 인터페이스는 다음과 같다.

함수형 인터페이스	메서드	설명
UnaryOperator	T apply(T t)	Function의 자손으로 매개변수와 결과의 타입이 같다.
BinaryOperator	T apply(T t, T t)	BiFunction의 자손으로 매개변수와 결과의 타입이 같다.

기본형을 사용하는 함수형 인터페이스는 다음과 같다.

함수형 인터페이스	메서드	설명
AToBFunction	B applyAsB(A a)	입력이 A타입이고 출력이 B타입이다.
ToBFunction	B applyAsB(T value)	입력은 지네릭스 타입이고 출력이 B타입이다.
AFunction	R apply(A a)	입력이 A타입이고 출력은 지네릭스 타입이다.
ObjAConsumer	void accpet(T t, A a)	입력이 T, A타입이고 출력은 없다.

Function의 합성

f.andthen(g)는 함수 f를 먼저 적용하고, 그 다음에 함수 g를 적용한다.
f.compose(g)는 함수 g를 먼저 적용하고, 그 다음에 함수 f를 적용한다.

Predicate의 결합

여러 Predicate를 and(), or(), negate()로 연결해서 하나의 새로운 Predicate로 결합할 수 있다.
isEqual()는 등가 비교를 할 때 사용한다.

메서드 참조

하나의 메서드만 호출하는 람다식을 더 간결하게 표현할 수 있는 방법이다.

메서드 참조의 방법은 다음과 같다.

종류	람다식	메서드 참조
static메서드 참조	x -> ClassName.method(x)	ClassName::method
인스턴스메서드 참조	(obj, x) -> obj.method(x)	ClassName::method
특정 객체 인스턴스메서드 참조	(x) -> obj.method(x)	obj::method
생성자 메서드 참조	() -> new MyClass()	MyClass:new
배열 메서드 참조	x -> new int[x]	int[]::new