**모던 자바 인 액션**

1. **기초**
2. **자바 8, 9, 10, 11 : 무슨 일이 일어나고 있는가?**

문제.

**“언어 생태계의 모든 언어는 변화해서 살아남거나 그대로 머물면서 사라지게 된다.”**

**– 64p.**

자바 8 이전의 자바는 멀티코어 프로세서를 온전히 활용하기 어려웠고(가변 공유 상태에서 발생하는 문제) 함수형 언어 미지원 등의 문제를 안고 있었다. 자바 8에서는 이를 해결하기 위해 람다와 스트림이 추가되었다.

함수형 언어 패러다임은 함수가 매개변수로 입력된 값에 언제나 동일한 결과를 반환하도록, 순수한 함수를 작성하는데 있다. 처음에는 자바와 C등 타 언어에도 함수가 있는데 이게 무슨 차이일까 싶었다. 알아본 결과 함수형 패러다임의 주안점은 “순수”에 있었다. 자바는 객체를 생성해 객체의 속성에 종속적이거나 영향을 미치는 메서드를 함수로 활용한다. 만약 메서드가 객체 내부의 속성을 이용하거나 객체 내부의 속성에 영향을 미친다면 이 함수(메서드)는 객체의 상태 변화에 관여하게 된다. 따라서 함수형 언어에서처럼 함수를 사용하기 위해서는 객체를 통하지 않고(객체를 생성해서 class.executeFunction()과 같이 사용하지 않고) 순수하게 일회용으로 사용할 수 있는 함수를 불러서 쓸 수 있어야 한다.

자바 8에 추가된 람다 표현식은 아래와 같은 표현으로 함수를 일회용으로 사용할 수 있게 만들어준다.

(s) -> System.out.println(s);

스트림은 메서드 체이닝과 비슷한 방식으로 함수를 이어서 활용한다. 스트림은 데이터를 함수형프로그래밍 패러다임으로 연속적으로 처리하기 위해 메서드 체이닝과 같은 방식을 활용한다. 메서드 체이닝은 객체를 생성, 활용, 변형하기 위해 함수를 잇는다는 점에서 스트림과 차이가 있다.

람다와 스트림 개념은 대규모 컴포넌트 기반 프로그래밍이나 멀티 프로세서를 위한 병렬성 처리를 유용하게 하기위해 활용된다. 자바로는 작은 프로젝트나 PS밖에 하지 않아 실제 용례에서 람다와 스트림이 어떻게 얼마만큼 혁신적인지 이해하는 데에는 한계가 있었다. 다독과 클론 코딩, 실제 적용으로 더 깊은 이해가 필요하다.