6장. 함수형 프로그래밍

함수형 프로그래밍의 개념은 프로그래밍 그 자체보다 앞서 등장했다. 기본이 되는 것은 람다(lambda)계산법이다.

정수를 제곱하기

- 25까지의 정수의 제곱을 출력하는 프로그램
  - 자바

```
public class Squint {
  public static void main(String args[]) {
    for (int i=0; i<25; i++)
      System.out.println(i*i);
  }
}
```

  - 클로저 (함수형 언어)

```
(println (take 25 (map (fn [x] (* x x)) (range))))
```

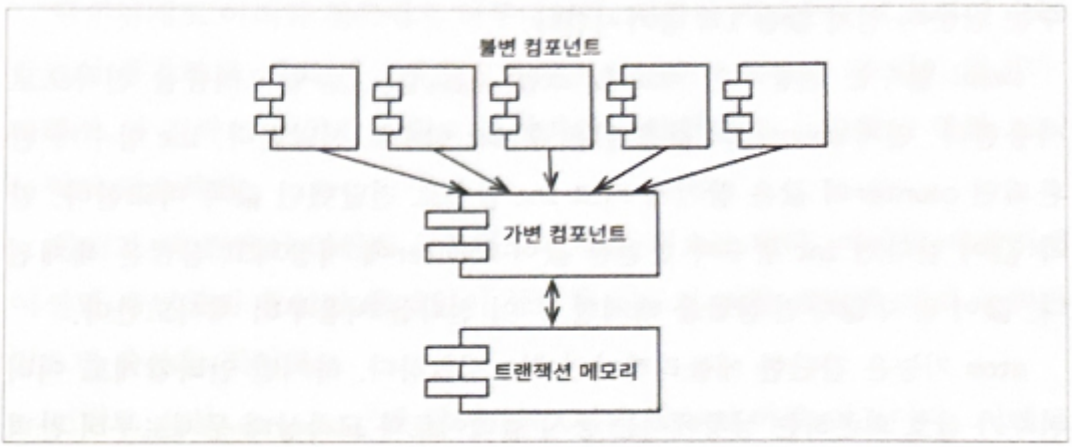
    - `(fn [x] (* x x))`는 익명함수이다.
    - range함수는 0부터 끝이 없는 정수 리스트를 반환한다.
    - 끝이 없는 정수를 map이 받고, 익명함수를 각각 적용한다.
    - 그중 take함수를 이용해 25개까지만 잘라서 새로운 리스트로 반환한다.
    - println함수로 결과를 출력한다.
  - 자바와 클로저의 차이
    - 자바는 가변 변수를 사용해서 실행 중에 상태가 변할 수 있다.
    - 클로저는 가변 변수가 전혀 없다. x와 같은 변수가 한번 초기화 되면 절대로 변하지 않는다.

불변성과 아키텍처

- 가변 변수는 경합(race)조건, 교착상태(deadlock), 동시 업데이트 문제를 야기시킬 수 있다.
- 동시성 문제에 대해 관심을 가져야하고, 불변성이 가능한지에 대해서 물어봐야한다. 실현 가능하지만 일종의 타협이 필요하다.

가변성의 분리

- 불변성과 관련해 가장 주요한 타협 중 하나는 애플리케이션, 또는 애플리케이션 내부의 서비스를 가변 컴포넌트와 불변 컴포넌트로 분리하는 일이다.
    - 불변 컴포넌트: 순수하게 함수형 방식으로만 작업이 처리되며 어떤 가변 변수도 사용되지 않는다. 순수 함수형 컴포넌트가 아닌 하나 이상의 다른 컴포넌트와 서로 통신한다.



- 상태 변경은 트랜잭션 메모리와 같은 실천법을 사용해 동시 업데이트와 경합 조건 문제로부터 가변 변수를 보호한다.
- 트랜잭션 메모리는 데이터베이스가 디스크의 레코드를 다루는 방식과 동일한 방식으로 메모리의 변수를 처리한다. 트랜잭션을 사용하거나 또는 재시도 기법을 통해 변수를 보호한다.
- 가능한 많은 처리를 불변 컴포넌트로 옮겨야하고, 가변 컴포넌트에서는 가능한 한 많은 코드를 빼내야한다.

이벤트 소싱

- 이벤트 소싱: 상태가 아닌 트랜잭션을 저장하는 전략. 상태가 피룡해지면 단순히 상태의 시작점부터 모든 트랜잭션을 처리한다.

- 애플리케이션은 CRUD가 아닌 CR로만 이루어진다. 변경과 삭제가 일어나지 않으므로 동시 업데이트 문제또한 일어나지 않는다.
- 저장공간과 처리능력이 충분하면 애플리케이션이 완전한 불변성을 갖도록 만들 수 있고, 따라서 완전한 함수형으로 만들 수 있다.

결론

- 구조적 프로그래밍은 제어흐름의 직접적인 전환에 부과되는 규율이다.
- 객체 지향 프로그래밍은 제어흐름의 간접적인 전환에 부과되는 규율이다.
- 함수형 프로그래밍은 변수 할당에 부과되는 규율이다.
- 소프트웨어는 급격히 발전하는 기술이 아니다.
- 소프트웨어는 순차, 분기, 반복, 참조로 구성된다.