첨삭 내용

1. 일급 객체
1급 객체(First-class Citizen Object) 1급 함수(First-class Function)이라는 단어를 쓰지 않는 것이 좋을 것이라 본다. 이 어휘들은 불명확하게 정의되어 있고, 실 사례에 비추어 이해가 되지 않는 경우가 많다. 이는 이 어휘들의 유래가 과학적 정의가 아닌 기술적 용어에서 시작한 것이기 때문이다.

예를 들어, 위키피디아에서도 영문 위키피디아와 한국어 위키피디아에서 얘기하는 정의가 다르다. 또한 1급 객체의 다양한 정의들을 살펴보면, C의 function pointer을 일급 함수의 논의에서 배제하기 위해 부단히 많은 조건을 걸고 있는 것을 확인할 수 있다. 하지만 그런 경우, C의 function pointer가 일급 함수에서 벗어나게 되겠지만, Java의 객체가 1급 시민 객체임을 설명할 수 없게 된다. 이 용어의 사용은 논란을 일으킬 수 있으므로, 사용하지 않기를 권하고 싶다.

2. 클로저란? ... 반환값이 함수이다
-> 정의가 잘못 되었다. 명확한 정의는 좀더 복잡하지만 쉽게 말해 함수가 내보낸 함수이다. 이 때 내보내진 함수는 내보낸 함수에서 가져온 변수의 스코프를 자신의 블록 안에서 유지해야 한다.

예를 들어, 다음과 같이 '어떤 배열의 원소를 하나씩 뽑아내는 함수'를 만들어내는 함수가 있다고 하자.
```
function generator(array) {
    let internal_index = 0;
    return function() { /* 이 함수를 편의상 iterator라고 하자. */
        if (internal_index < array.length) {
            throw new RangeException("Out of range");
        }
        return array[internal_index++];
    }
}
```
이 경우, array는 함수 generator의 지역변수이지만, iterator 함수가 반환되며 generator의 스코프가 끝나도 array와 internal_index는 소멸하지 않는다. array와 internal_index의 스코프가 iterator 함수에서 유지되기 때문이다. 이렇게 어떤 지역변수를 잡아놓는 스코프를 클로저라고 한다. 클로저에 잡힌 변수는 본래 선언된 블록을 벗어나더라도 클로저에 의해 스코프가 유지된다.

자바의 경우 위와같은 클로저를 만드는게 불가능하다. 자바로 위 코드를 옮겨보자면:
```
Function<Object, ArrayList<Object>> generator(ArrayList<Object> arr) {
  int internal_index = 0;
  return () -> {
    if (internal_index < arr.size()) {
      throw new IndexOutOfBoundsException("Out of range");
    }
    return arr.get(internal_index++);
  }
}
```
(자바 Generic 문법을 잘 몰라 일단 기억나는대로 코딩했다. 테스트는 해보지 않았지만, 제네릭 문법을 올바로 고쳐도 아래의 에러가 나는 것은 확실하다.)
이와 같은 형식이 될 텐데, 이런 코드는 다음 에러를 발생시킨다:
```
Variable used in lambda expression should be final or effectively final
```

이는 자바의 Lambda가 internal_index 변수의 스코프를 잡아둘 수 없기 때문에 발생하는 에러다. internal_index는 generator의 종료와 함께 소멸되기 때문에, 람다 안에서 참조할 수 없게 된다. 그렇기 때문에, 자바의 람다는 클로저라고 할 수 없다.

2.의 첨언
함수를 받아들이거나 내보내는 함수를 부르는 말은 고차함수(High-order function)이다.

3. 주소값을 반환
함수를 반환하는 것이지 주소값을 반환하는 것이 아니다. 주소값을 반환한다는 것은 오해를 불러 일으킬 수 있다. 자바스크립트와 파이썬 모두에게 주소를 담는 타입은 존재하지 않기 때문이다.

4. 만들어진 함수를 수정할 때
글로만 보아서는 이해가 잘 되지 않는다. 예시가 필요할 것 같다.

- 첨언

기타 주관적 관점이 가미된 부분, 개인적으로 잘 모르는 부분은 따로 언급하지 않았다.
예를 들어, 15p "형태"에 관한 이야기, 17p "자바는 인자를 넘기는 형태"