EJ

이펙티브 자바 CP.1

작성 일시	@2022년 12월 30일 오후 8:23
최종 편집 일시	@2022년 12월 30일 오후 10:44
유형	이펙티브 자바
작성자	
참석자	

1️⃣ 객체 생성과 파괴

▼ 1. 생성자 대신 정적 팩토리 메소드를 고려해라

public ClassName(), static getWhat()

클래스는 public 생성자 대신 정적 팩토리 메소드를 제공할 수 있다.

정적 팩토리 메소드의 장점

1. 이름을 가질 수 있다.

   명확화 할 수 있다.

2. 호출할 때마다 인스턴스를 새로 생성하지 않아도 된다.

   lib 같이 자주 쓰는 파일에 활용 가능해보인다.

3. 반환 타입을 하위 타입 객체로 반환할 수 있는 능력이 있다.

4. 입력 매개변수에 따라 매번 다른 클래스의 객체를 반환할 수 있다.

   3, 4 항목 안에서

   이건 단점이 아닌가 생각 했지만, Map 을 예로 생각해보면 편하다. TreeMap, LinkedMap, HashMap 등을 입력 변수에 따라 변경할 수 있다는건, 유연한 개발이 가능 할 것으로 보인다.

5. 정적 팩토리 메소드는 작성하는 시점에는 반환할 객체의 클래스가 존재하지 않아도 된다.

```
// Order 라는 interface 안에 구현체가 없어도 반환 가능하다.
public static List<Order> getOrders() {
  return new ArrayList<>();
}

interface Order {
}
```

6. 서비스 제공자 프레임워크를 만드는 배경이된다. (ex. JDBC)

정적 팩토리 메소드의 단점

1. 상속을 하려면 public 이나 protected 생성자가 필요하니, 정적 팩터리 메소드만 제공하면 하위 클래스를 만들 수 없다.

   어쩌면, 이 제약은 상속(3-4)보다 컴포지션을 사용한다면, 오히려 장점일 수 있다.

2. 정적 패터리 메소드는 프로그래머가 찾기 어렵다.

   흔히 사용하는 명명 방식

```
// from 매개수를 하나 받아서 해당 타입의 인스턴스를 반환하는 형변환 메소드
Data d = Date.from(instant);

// of 여러 매개변수를 받아 적합한 타입의 인스턴스를 반환하는 집계 메소드
Set<Rank> faceCards = EnumSet.of(JACK, QUEEN, KING);

// instance 혹은 getInstance 매개변수로 명시한 인스턴스를 반환하지만, 같은 인스턴스임을 보장하지 않음
StackWalker luke = StackWalker.getInstance(option);

// create 혹은 newInstance, 위와 같지만, 매번 새로운 인스턴스를 반환함을 보장함
Object newArray = Array.newInstance(classObject, arrayLen);

// getType getInstance 와 같으나, 생성할 클래스가 아닌 다른 클래스에 팩터리 메소드를 정의할 때 사용한다.
FileStore fs = Files.getFileStore(path);

// newType newInstance 와 같으나, 위와 같음
BufferedReader br = Files.newBufferedReader(paht);

// type 위 getType, newType의 간결한 버전
List<Complaint> litany = Collections.list(legacyLitany);
```

▼ 2. 생성자에 매개 변수가 많다면 빌더를 고려해라

▼ 3. prviate 생정자나 열거 타입으로 싱글턴임을 보증해라

▼ 4. 인스턴스화를 막으려거든 private 생성자를 사용해라

▼ 5. 자원을 직접 명시하지 말고 의존 객체를 주입해라

▼ 6. 불 필요한 객체 생성을 피해라

▼ 7. 다 쓴 객체 참조를 해제해라

▼ 8. finalizer, cleaner 사용을 피해라

▼ 9. try-finally 보다는 try-with-resources를 사용해라