Effective Java Study

Woowacourse_study 4th

Item 38 by 알파

타입 안전 열거 패턴..?

JDK 1.5 이전에서 공식적으로 enum을 지원하지 않았기 때문에 생겨난 열거 패턴

타입 안전 열거 패턴..?

```
public class Country {

    private String name;

    private Shape(String name) {
        this.name = name;
    }

    public static final Country ROK = new Shape("republic of korea");
    public static final Shape USA = new Shape("united states");
    public static final Shape JPA = new Shape("japan");
}
```

타입 안전 열거 패턴..?

타입 안전 열거 패턴은 확장이 가능하지만, enum의 경우는 확장이 불가능하다

그런데 인터페이스를 곁들인..

그러나, enum은 interface를 implements하여 확장성을 갖는 방법이 있다

그런데 인터페이스를 곁들인..

```
public interface Operation {
    double apply(double x, double y);
}
```

그런데 인터페이스를 곁들인..

여기서 곱하기와 나누기를

추가하려면..?

```java
public enum BasicOperation implements Operation {
    PLUS("+") {
        @Override
        public double apply(double x, double y) {
            return x + y;
        }
    },
    MINUS("-") {
        @Override
        public double apply(double x, double y) {
            return x - y;
        }
    };

    private final String symbol;

    BasicOperation(String symbol) {
        this.symbol = symbol;
    }
}
```

그런데 인터페이스를 곁들인..

여기서 곱하기와 나누기를

추가하려면..!

```java
public enum ExtendedOperation implements Operation {
    EXP("*") {
        @Override
        public double apply(double x, double y) {
            return x * y;
        }
    },
    REMAINDER("/") {
        @Override
        public double apply(double x, double y) {
            return x / y;
        }
    };

    private final String symbol;

    ExtendedOperation(String symbol) {
        this.symbol = symbol;
    }
}
```

Exp, Modulo 연산도 간단하게 Operation interface를 implement하면 된다!

interface를 구현하는 여러 enum들에 접근할 수 있다!

```
public static void main(String[] args) {
    double x = 10;
    double y = 2;
    test(ExtendedOperation.class, x, y);
}

public static <T extends Enum & Operation> void test(
  Class opEnumType, double x, double y) {
    for (Operation op : opEnumType.getEnumConstants()) {
        System.out.printf("%f %s %f = %f%n", x, op, y, op.apply(x, y));
    }
}
```

interface를 구현하는 여러 enum들에 접근할 수 있다!

```
public static void main(String[] args) {
    double x = 10;
    double y = 2;
    test(Arrays.asList(ExtendedOperation.values()), x, y);
}

public static void test(Collection<? extends Operation> opSet, double x, double y) {
    for (Operation op : opSet) {
        System.out.printf("%f %s %f = %f%n", x, op, y, op.apply(x, y));
    }
}
```

단점은..?

같은 interface를 implements하는

구현체 enum 간의 상속은 불가

해결책은..?

공통 부분이 많지 않다면

같은 interface 내에서 디폴트 메소드로 구현!

해결책은..?

공통 부분이 많다면

별도의 도우미 클래스나 정적 도우미 메소드로 분리

References

Joshua Bloch, 『Effective Java 3/E』, 이복연 역 (서울 : 인사이트, 2018), pp. 232 - 236

E.O.D

Item 38 by 알파