Java의 Enum

Enum 왜 씀? Enum 쓸 바엔 뜨끈한 상수 든든하게 쓰고 말지

30기 정회원 김은솔

왜 써요?

Enum도 열거형 ‘상수’고 … 상수는 상수니까… 그냥 상수 여러 개 넣어둔 거 아닌가

상수란 무엇일까

- 변수의 반댓말
- 변하지 않고 고정된 값을 담는 변수

```
public static final String HELLO_WORLD = "Hello World!!";

public final int MAX_COUNT = 10;
```

상수란 무엇일까

- 상수값에 따라서 값에 해당하는 의미를 고정하고 싶을 때 사용한다.

```java
package org.opentutorials.javatutorials.constant2;

public class ConstantDemo {
    private final static int APPLE = 1;
    private final static int PEACH = 2;
    private final static int BANANA = 3;
    public static void main(String[] args) {
        int type = APPLE;
        switch(type){
            case APPLE:
                System.out.println(57+" kcal");
                break;
            case PEACH:
                System.out.println(34+" kcal");
                break;
            case BANANA:
                System.out.println(93+" kcal");
                break;
        }
    }
}
```

type 변수 안에 담긴 상수의 값에 따라 switch 문이 실행 된다.

상수는 switch 문에서 사용할 수 있다

상수란 무엇일까

```java
public class ConstantDemo {
    // fruit
    private final static int FRUIT_APPLE = 1;
    private final static int FRUIT_PEACH = 2;
    private final static int FRUIT_BANANA = 3;

    // company
    private final static int COMPANY_GOOGLE = 1;
    private final static int COMPANY_APPLE = 2;
    private final static int COMPANY_ORACLE = 3;

    public static void main(String[] args) {
        int type = FRUIT_APPLE;
        switch(type){
            case FRUIT_APPLE:
                System.out.println(57+" kcal");
                break;
            case FRUIT_PEACH:
                System.out.println(34+" kcal");
                break;
            case FRUIT_BANANA:
                System.out.println(93+" kcal");
                break;
        }
    }
}
```

다른 APPLE 상수가 필요할 때는 어떨까?
이름이 같은 상수는 안 되니까 접두사를 붙여 본다.
문제는 해결된 것처럼 보인다.

int type = COMPANY_APPLE;
로 코드를 바꾸면 어떨까?

정수 열거 패턴

- 타입 안정성을 보장할 수 없다. (타입을 판별하지 못해 runtime 시 타입 관련 문제가 발생)
- 표현력이 좋지 않다. (겹치는 이름인 경우에 접두어로 이름 충돌 방지하는 것)
- 상수값(final)은 컴파일을 하면 값이 파일에 새겨지므로, 상수 값이 바뀌면 컴파일을 다시 해야 한다.

```
public class Fruits {
  private static final int APPLE = 1;
  private static final int GRAPE = 2;
  private static final int ORANGE = 3;
}
```

열거 타입 (Enum)

- Java의 Enum은 완전한 기능을 갖춘 클래스
- compile time에 타입 안정성을 제공한다. (같은 enum에서 비교연산자 == 가능)
- 같은 이름의 상수가 공존할 수 있다. (namespace 문제 해결)

Enum 써보기

```java
public static int calculate(String grade) {
    if(grade.equals("A")) {
        return 10000;
    }
    else if(grade.equals("B")) {
        return 5000;
    }
    else if(grade.equals("C")) {
        return 1000;
    }
    else {
        return 0;
    }
}
```

- grade 값에 따라서 값을 나눠서 전달해야 하는 경우를 살펴보자.
- grade와 계산이 별도의 메소드로 진행되는 경우, 서로 관계가 있음을 코드로 표현할 수 없다.
- grade가 추가되는 경우 if가 무한히 추가되어야 한다…

```java
0개의 사용위치
public static void main(String[] args) {
    String gradeCode = getGrade();
    int money = MoneyCalculator.calculate(gradeCode);
}
```

Enum 써보기

```java
public enum Grade {
    0개의 사용위치
    A( money: 10000), B( money: 5000), C( money: 1000);

    2개 사용 위치
    private final int money;

    3개 사용 위치
    Grade(int money) {
        this.money = money;
    }

    1개 사용 위치
    public int getMoney() {
        return money;
    }
}
```

- 이름이 grade 정보
- 각 성적과 연관되는 money

상수와 연관된 데이터를 상수 자체에 넣고 싶다면?

```java
public enum Planet {
    MERCURY(3.302e+23, 2.439e6),
    VENUS(4.869e+24, 6.052e6),
    EARTH(5.975e+24, 6.378e6),
    MARS(6.419e+23, 3.393e6),
    JUPITER(1.899e+27, 7.149e7),
    SATURN(5.685e+26, 6.027e7),
    URANUS(8.683e+25, 0.556e7),
    NEPTUNE(1.024e+26, 2.477e7);

    private final double mass;              //질량(kg)
    private final double radius;            //반지름(meter)
    private final double surfaceGravity;    //표면중력(m / s^2)

    //중력상수(m^3 / kg s^2)
    private static final double G = 6.67300E-11;

    Planet(double mass, double radius) {
        this.mass = mass;
        this.radius = radius;
        surfaceGravity = G * mass / (radius * radius);
    }

    public double mass() { return mass; }
    public double radius() { return radius; }
    public double surfaceGravity() { return surfaceGravity; }

    public double surfaceWeight(double mass) {
        return mass * surfaceGravity; // F = ma
    }

}
```

- 각 행성마다 질량과 반지름이 존재
- 각 속성을 이용해 표면 중력 계산 가능
- 데이터와 함수를 한 곳에서 관리할 수 있다.
- 객체에게 질문하는 코드

```java
Planet earth = Planet.EARTH;
earth.mass();
earth.surfaceGravity();
```

상수별 메서드를 Lambda로 구현하기

```java
public enum Operation {
    1개 사용 위치
    PLUS, MINUS, TIMES, DIVIDE;

    0개의 사용위치
    public double apply(double x, double y) {
        switch (this) {
            case PLUS: return x + y;
            case MINUS: return x - y;
            case TIMES: return x * y;
            case DIVIDE: return x / y;
        }
        throw new AssertionError( detailMessage: "알 수 없는 연산: "+ this);
    }
}
```

- 사칙연산의 종류를 enum으로 선언하고 동작하는 연산까지 열거타입으로 한다면?
- switch 문을 이용해서 분기처리하는 방법이 있을 수 있다.
- throw는 도달할 일이 없지만 코드상으로는 도달 가능하므로 생략할 수 없다. (런타임 에러 발생)
- 새로운 상수 추가 시 case 문 추가

상수별 메서드를 Lambda로 구현하기

```java
public enum Operation {
    0개의 사용위치
    PLUS( symbol: "+", (a, b) -> a + b),
    0개의 사용위치
    MINUS( symbol: "-", (a, b) -> a - b),
    0개의 사용위치
    MULTIPLY( symbol: "*", (a, b) -> a * b),
    0개의 사용위치
    DIVIDE( symbol: "/", (a, b) -> a / b);

    2개 사용 위치
    private final String symbol;
    2개 사용 위치
    public final ToDoubleBiFunction<Double, Double> function;

    4개 사용 위치
    Operation(String symbol, ToDoubleBiFunction<Double, Double> function) {
        this.symbol = symbol;
        this.function = function;
    }

    0개의 사용위치
    public double apply(double a, double b) {
        return this.function.applyAsDouble(a, b);
    }


    @Override
    public String toString() {
        return symbol;
    }
}
```

- 메소드를 if로 찾지 않고 각각의 함수 만들어주기
- 객체에게 질문하기
- 열거 타입 상수끼리 코드 공유가 어렵다.
- 전략 패턴을 사용하여 개선할 수 있다.

```java
Operation operation = Operation.PLUS;
double result = operation.apply( a: 10,  b: 20);
```

Tip

```java
public static Operation findOperation(String symbol) {
    Operation resultOperation = Arrays.stream(Operation.values()) Strea
            .filter(operation -> operation.toString().equals(symbol))
            .findAny() Optional<Operation>
            .orElse( other: null);
    return resultOperation;
}
```

- stream을 사용하면 symbol로 해당 Operation을 찾아 리턴하는 메서드를 만들 수 있다.

```java
String symbol = "*";
double secondResult = Operation.findOperation(symbol)
        .apply( a: 10,  b: 20);
```

주의할 것

- 언제나 switch case 문이 나쁜 것이 아니다.
- 대부분의 경우 열거 타입의 성능이 정수 상수와 비교하여 크게 차이가 나지 않는다.
- 필요한 원소를 컴파일 타임에 다 알 수 있는 상수 집합의 경우 열거 타입을 사용하는 것이 좋다.

참고한 자료들

https://techblog.woowahan.com/2527/

https://opentutorials.org/module/516/6091