Enum (열거형)

2016년 12월 3일 토요일 오전 12:15

Enumeration Type (열거형)

- 한정된 값 만을 갖는 데이터 타입
- 열거형은 열거 상수 (Enumeration Constant) 중에 하나의 상수를 저장하는 데이터 타입
  - 요일 : 월, 화, 수, 목, 금, 토, 일
  - 계절 : 봄, 여름, 가을, 겨울

열거형 선언

1. 열거형 이름으로 .java 파일 생성

   열거형이름.java

2. .java 파일에 열거형을 선언한다.

   public enum 열거형이름 {

   }

3. 열거형에 열거 상수를 선언한다.
   (열거 상수는 모두 대문자가 관례, 열거 상수가 여러개의 단어이면 단어 사이에 _ 로 연결)

   Ex)
   ```
   public enum Week {
       MONDAY,
       TUESDAY,
       WEDNESDAY,
       THURSDAY,
       FRIDAY,
       SATURDAY,
       SUNDAY
   }
   ```

열거형 변수 선언

1. 열거형도 데이터 타입이므로 변수를 선언하고 사용해야 한다.

   열거형 변수이름;
   ```
   Week today;
   Week reservationDay;
   ```

2. 열거 상수를 저장하기

   열거형 변수이름 = 열거타입.열거상수;
   ```
   Week today = Week.SUNDAY;
   ```

열거 상수 (Enumeration Constant)

- 열거형 안에 열거 상수가 7개 있으면 총 7개의 열거객체로 선언이 되고,
  열거 상수가 각각의 열거객체를 참조한다.

Week 객체의 경우 MONDAY ~ FRIDAY 까지
각각에 해당하는 7개의 Week 객체 생성
↓
열거 상수는 Method 영역에 생성
↓
열거 상수 MONDAY ~ FRIDAY는 Heap 영역에 생성된 Week 객체 참조
↓
열거형 변수를 선언하고 열거 상수를 저장하면 결과적으로 같은 Week 객체를 참조한다.
↓
그러므로 today == Week.SUNDAY 는 true

열거형 예제

```
// Week.java
public enum Week { // Enumeration Type
     // Enumeration Constant
     MONDAY,
     TUESDAY,
     WEDNESDAY,
     THURSDAY,
     FRIDAY,
     SATURDAY,
     SUNDAY
}

// EnumerationExample.java
import java.util.Calendar;

public class EnumerationType {
     public static void main(String[] args){
          Week today = null; // 열거형 변수 선언 (Stack Area)

          // 컴퓨터의 시간 제공 Class
          Calendar calendar = Calendar.getInstance();
          int day = calendar.get(Calendar.DAY_OF_WEEK); // 현재 요일 정수 값으로 반환

          switch(day){
               case 1:
                    today = Week.SUNDAY; break;
               case 2:
                    today = Week.MONDAY; break;
               case 3:
                    today = Week.TUESDAY; break;
               case 4:
                    today = Week.WEDNESDAY; break;
               case 5:
                    today = Week.THURSDAY; break;
               case 6:
                    today = Week.FRIDAY; break;
               case 7:
```

```
                today = Week.SATURDAY; break;
        }

        System.out.println("오늘요일: " + today.toString());

        // Stack 영역의 today 변수와 Method 영역의 열거상수가 같은 객체를 참조하면
        if(today == Week.WEDNESDAY){
            System.out.println("Break Day");
        } else {
            System.out.println("Java Day");
        }
    }
}
```

열거 객체의 메소드

Week today = Week.SUNDAY

- **name()**
  - **String name = today.name();**
  - **열거객체가 가지고 있는 문자열 반환 (문자열은 상수이름과 동일)**
- **ordinal()**
  - **int ordinal name = today.ordinal();**
  - **전체 열거객체 중 몇 번째 열거 객체인지 순번 반환 (0 번부터 시작)**
- **compareTo()**
  - **Week day1 = Week.MONDAY;**
    Week day2 = Week.WEDNESDAY;
    int result1 = day1.compareTo(day2);
    int result1 = day2.compareTo(day1);
  - **인자로 주어진 열거 객체를 기준으로 순서가 앞서는지 뒤서는지 비교**

    int result1 = day1.compareTo(day2); // day1가 day2보다 2 앞서므로 -2 반환

    int result1 = day2.compareTo(day1); // day2가 day1 보다 2 뒤서므로 2 반환
- **valueOf()**
  - **Week weekDay = Week.valueOf("SATURDAY");**
  - **인자로 주어진 문자열과 동일한 문자열을 가지는 열거객체 반환**
- **values()**
  - **Week[] days = Week.values();**
    for(Week day : days){
        System.out.println(day.toString());
    }
  - **열거형의 모든 열거 객체들을 배열형태로 반환**
