03-7. enum 클래스

Object Oriented Programming

Enum 클래스

열거형 클래스 선언 및 이용

- 상수 여러 개를 열거형 타입에 선언하고 이 열거형 타입으로 선언되는 변수는 선언된 상수값중 하나를 지정하게 하는 기법.
- enum 이라는 예약어로 만들어지는 클래스

- 열거 상수는 기본으로 name과 original 프로퍼티를 제공합니다.
- name은 열거 상수의 문자열이며, original은 열거한 순서를 나타내는 인덱스 번호입니다.

```
enum class Direction {
    NORTH, SOUTH, WEST, EAST
}
fun main(args: Array<String>) {
    val direction: Direction = Direction.NORTH
    println("${direction.name} ... ${direction.ordinal}")
    val directions: Array<Direction> = Direction.values()
    directions.forEach { t -> println(t.name) }
    val direction1=Direction.valueOf("WEST")
    println("${direction1.name} .. ${direction1.ordinal}")
}
```

Enum 클래스

개발자 임의 데이터 삽입

- 열거형 클래스를 선언할 때 주 생성자의 매개변수에 프로퍼티를 선언해야 한다.

```
enum class Direction(val no: Int) {
    NORTH(0), SOUTH(1), WEST(2), EAST(3)
}

fun main(args: Array<String>) {
    val direction: Direction = Direction.NORTH
    println(Direction.NORTH.no) //0
}
```

```
enum class Direction(var no: Int, val str: String) {
    NORTH(0, "north"), SOUTH(1, "south"), WEST(2, "west"), EAST(3, "east")
}

fun main(args: Array<String>) {
    val direction: Direction = Direction.NORTH

    println("no : ${direction.no}, ${direction.str}") //no : 0, north

    direction.no=10

    println("no : ${direction.no}, ${direction.str}") //no : 10, north
}
```

Enum 클래스

개발자 임의 데이터 삽입

- 열거 상수는 객체이다.
- 열거 상수는 enum 예약어로 선언한 클래스를 상속받는 클래스의 객체입니다.
- 이때 서브 클래스의 이름은 없으며 이러한 클래스를 익명 클래스(Anonymous class)라고 합니다.
- 예를 들어, enum class Direction { NORTH }라고 작성하면 NORTH는 Direction이라는 클래스를 상속받는 클래스의 객체입니다.
- 따라서 enum class Direction (val no: Int)라고 열거형 클래스를 선언하면 하위 클래스에서는 상위 클래스의 생성자에 맞추어 호출해야 하므로 NORTH(0)이라고 표현한다.

Enum 클래스

개발자 임의 데이터 삽입

- 열거 상수가 익명 클래스의 객체이므로 원한다면 익명 클래스를 직접 정의해서 이용할 수 있다.
- 이름없는 클래스를 직접 정의하여 다양하게 이용
- 세미콜론 (;) 앞부분이 문자열의 나열이고 뒷 부분이 클래스에 선언하고자 하는 프로퍼티와 함수 선언 부분

```
enum class Direction {
    NORTH {
        override val data1: Int = 10
        override fun myFun(){
            println("north myFun....")
        }
    },
    SOUTH {
        override val data1: Int = 20
        override fun myFun(){
            println("south myFun....")
        }
    };
    abstract val data1: Int
    abstract fun myFun()
}
fun main(args: Array<String>) {
    val direction: Direction = Direction.NORTH
    println(direction.data1)
    direction.myFun()
}
```

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare