열거형 (Enum)

상수 데이터들의 집합

- 열거형이 갖는 값뿐만 아니라 타입까지 관리하여, 논리적인 오류 줄일 수 있음

  → 실제 값이 같아도 타입이 다르면 조건식 결과가 false
- 상수의 값이 바뀌면, 해당 상수를 참조하는 모든 소스를 다시 컴파일해야 하지만, 열거형 상수를 사용하면 다시 컴파일 X
- 상수를 단순히 정수로 치부하지말고 **객체 지향적으로 객체화해서 관리**하자는 취지
- 비교
  - '==' 사용 가능. (equals() 보다 빠른 성능)
  - '<', '>' 사용 불가. compareTo() 사용 가능
- switch문에 사용 가능 (이때, case문에 열거형이름은 적지 않고, 상수 이름만 적어야 함)

열거형 사용

```
// 열거형 정의
enum 열거형이름 { 상수명1, 상수명2, ...}

// ex)
enum Direction { EAST, SOUTH, WEST, NORTH }
```

```
// 열거형 상수 사용
열거형이름.상수명

// ex)
class Unit {
    int x, y;
    Direction dir;

    void init() {
        dir = Direction.EAST;
```

```
    }
}
```

메서드

```
// values() : 열거형의 모든 상수 반환
Direction[] dArr = Direction.values();
for (Direction direction : dArr) {
    System.out.println(direction);
}

// 결과
EAST
SOUTH
WEST
NORTH
```

메서드	설명
EnumType[] values()	열거형의 모든 상수 반환
Class<E> getDeclaringClass()	열거형의 Class객체 반환
String name()	열거형 상수의 이름을 문자열로 반환
int ordinal()	열거형 상수가 정의된 순서를 반환(0부터 시작)
EnumType valueOf(String name)	지정된 열거형에서 name과 일치하는 열거형 상수 반환

열거형 멤버 추가

```
enum Direction {
    EAST(1, ">"), SOUTH(2, "V"), WEST(3, "<"), NORTH(4, "^");
    // 반드시 끝에 ';' 붙여야 함

    private final int value; // 정수를 저장할 필드
    private final String symbol; // 문자 저장할 필드

    // 생성자
    Direction(int valule, String symbol) {
        this.value = value;
```

```
        this.symbol = symbol;

    }
    // 열거형 생성자는 제어자가 묵시적으로 private이므로 외부에서 생성자
}
```

열거형 추상 메소드 추가

상수별 추상 메서드 구현해야 함 → 익명 클래스와 유사

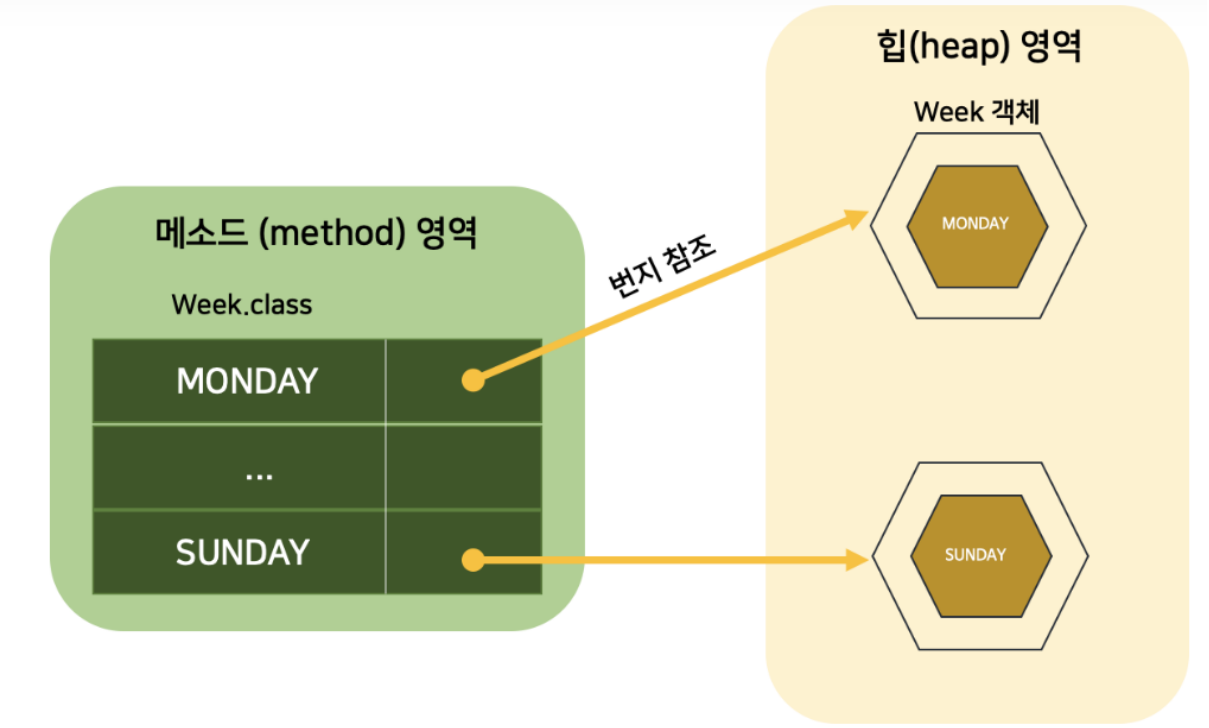
```
public enum Shape {
    SQUARE(100) {
        @Override
        public double calculateArea(double sideLength) {
            return sideLength * sideLength;
        }
    },
    CIRCLE(200) {
        @Override
        public double calculateArea(double radius) {
            return Math.PI * radius * radius;
        }
    },
    TRIANGLE(300) {
        @Override
        public double calculateArea(double base, double heigh
            return 0.5 * base * height;
        }
    };

    // 추상 메서드 선언
    public abstract double calculateArea(double arg1);

    // 추가로 다른 메서드도 선언 가능
    public void printDescription() {
        System.out.println("This is a shape.");
    }
}
```

참조 방식

primitive 타입이 아닌 referece 타입으로 분류되며, 그래서 enum 상수값은 **힙(heap)** 영역에 저장됨

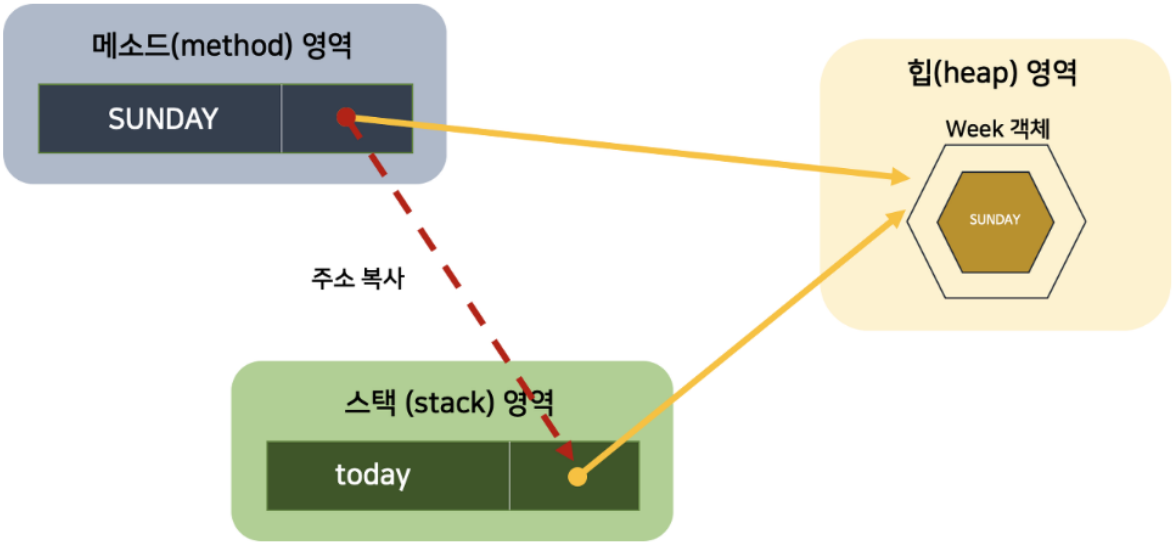


String 처럼 스택 영역에 있는 변수들이 힙 영역에 있는 데이터의 주소값을 저장함으로써 참조 형태를 띄게 됨

→ 같은 enum 타입 변수 끼리 같은 상수 데이터를 바라봄으로써 둘이 주소를 비교하는 == 연산 결과는 true가 됨

```
Week today = null; // 참조 타입이기 때문에 null도 저장 가능
today = Week.SUNDAY;

// 주소값 비교
System.out.println(today == Week.SUNDAY); // true
```



배열도 마찬가지로, 각 배열 원소들마다 참조 주소값들이 저장되어 힙 영역의 상수 데이터 가리킴

```
// enum Week 의 모든 상수값들을 배열로 변환
Week[] days = Week.values();
```