05/27 금

**- 변하지 않게 만들기 -**

StaticVariableAndConst.java

이름을 지을 땐 **모두 대문자**로, 구분자는 밑줄(\_).

Ex) **MY\_NUMBER** = 값;

* 변해서는 안되는 수를 표현 할 때.
* 하나의 약속으로 사용 할 때.
* 실수로 변경 하는 동작을 사용하면 에러!!
* 담아놓고 사용 하기 위해서, 변할 가능성을 차단

final int MY\_NUMBER = 22; // 상수 선언은 똑같음.

MY\_NUMBER++; // 값 변화 → 컴파일 에러.

final double PI = 3.14; → 하나의 약속으로 만들어버림.

final int ASDF;

ASDF = 10; // 변하는 것 같아 보이지만, 최초의 대입연산인 초기화.

// 초기화 부분은 컴파일 에러가 나지 않음. 그 이후는 에러

ASDF = 15; → 컴파일 에러 발생.

--------------------------------------------------------------------

StaticVariableAndConst2java

설명 더 있음.

{

System.out.println(inner);

System.out.println(TEST\_SCOPE);

// final int TEST\_SCOPE;

final int BLOCK = 100; // 지역변수

}

System.out.println(BLOCK); // 지역변수 범위 벗어남 => 컴파일 에러

→ 지역변수와 차이 거의 없음 but 변할 가능성이 있냐 없냐 차이

- 전역 변수 –

StaticVariableAndConst3.java

전역 변수는 어느 측면에서도 변화하기 때문에, 관리 측면에서 매우 힘듦.

→ **현재는 전역 변수 거의 사용 하지 않고 혐오수준**

전역 변수를 넓은 범위에서 똑같은 값으로 사용할 때 파이널 상수 사용

→ final int 사용

Class 안에는 ‘멤버’가 존재한다.

멤버 안에는 < 필드, 메소드 등 > 존재.

Static → 정적인 표현. 객체 표현과는 다르게 존재.

Class 이름으로 호출가능 ex) Math.PI 여기서 Math가 클래스 이름

double a = Math.abs(-372); // 절대값

// static한 메소드 Math.abs; → 기능을 바로 제공해줌.

System.out.println(a);

출력 : 372.0

double r = Math.random(); // 0 ~ 1 사이의 실수를 만들어 실수 형태로 출력

r = (int) (r \* 10); // 곱셈 활용해서 범위 가지는 정수 형태 숫자 만듦.

→ random class에 충분히 있는 기능.

→절차 지향 프로그램에서 만든 것들은 프로그램 확장, 수정에 있어서 힘들어, 객체 지향이 발전하게 됨.

객체 지향 → 현재, 가장 널리 사용.

Java도 객체 지향 언어체계.

**- 객체 지향-**

무조건 참고Box, TestBox.java

책 p.139

모든 현실의 객체(object) -> 객체로 보는 시선에서 시작.

//물건, 아이디어, 세상의 모든 것.

상태와 동작

객체 지향 프로그래밍

객체 -> 코드 - 객체를 코드로 만든다.

코드 -> 객체로

객체들 간에 메시지를 주고 받으며 프로그램의 흐름을 만들어 내는 방식

Ex)강아지

발이 네개다.

눈이 두개다.

털이 있다.

귀엽다.

→ 특징 중 필요한 만큼을 코드로 만들고, 코드를 기반으로 가상의 강아지를 컴퓨터로 만들어 냄. 실제는 아니지만 실제의 본을 딴 객체를 만드는 것.

서로 상호간 소통 하게 만들고(메시지를 주고 받음) 동작하여 프로그램이 흐르게 만드는 것.

추상적인 개념이다.

java에서 객체는 class로 표현.

// 값 / 상태 -> 멤버변수 (member variable) / 필드 (fields)

int length;

int width;

int height;

멤버변수 (member variable) / 필드 (fields)

→ 객체의 구성요소

Instance → 쉽게말해 공간을 할당해 준 것.

TestBox.java에서 new Box();는 인스턴스화 시키는 것.

Box 안에는 Box.java에서 만든 가로, 세로, 높이가 있다.

객체는 필요에 따라 값을 변경하거나 새로 만들어 줌.

Box none = null; // 참조형 변수는 null 을 참조할 수 있다.

→ 없는 것을 참조 → ‘지금 참조 하고 있는 것이 없다’

컴파일은 가능하지만, 결과출력에서 에러.

→ 참조가 제대로 이루어지고 있는지 확인 필수.

System.out.println(box1.color); // color를 나타내는 fields가 존재하지 않음. 컴파일 에러

// 메소드 → 기능 or 동작

box1.printAllState(); // 기능, 동작을 나타내는 메소드에 이름을 붙여서 가져옴. (box.java안에서 메소드 이름 지정 후, 컴파일)

출력 값 : 3

box2.printAllState();

// 같은 동작이지만, instance의 상태에 따라서 값이 달라진다.

// 각각의 상태는 instance마다 고유하게 가지고 있기 때문.

출력 값 : 10

책 p.156

자바 api에서 제공되는 클래스들은 범용적인 상황에서 우리에게 도움을 줄 수 있는 것들이지만,

직접 새로 작성하는 것은 직접 개발자가 개발 할 때 쓰기 때문