**12회차(1)**

< 객체지향 프로그래밍 >

- Object Oriented Programming

- Object를 '객체' 또는 '사물'로 해석한다.

- 객체지향 프로그래밍이 나온 가장 큰 이유는?

- 코드의 재사용성

- 객체지향 이전에는 함수(메소드) 단위로 프로그램을 작성

- 함수(메소드)만 가지고는 늘어나는 소스코드의 양을 감당할 수 없게 됨

- 그래서, 결과적으로 대규모 소프트웨어 개발의 어려움을 해소하기 위해 나온 개념.

< 클래스 >

- 메소드는 기능을 중심으로 해서 소프트웨어를 설계하고 구현

- 객체지향은 클래스를 중심으로 해서 소프트웨어를 설계하고 구현

- 클래스는 어떠한 대상을 표현하기 위해 정보(속성)을 저장할 수 있는 공간과,

그 공간을 제어할 수 있는 기능(메소드)를 하나의 블록으로 묶어서 정의한 사용자 정의 자료형

- 즉, 클래스는 변수와 메소드를 하나의 자료형을 관리하기 위한 방법

- 클래스를 정의하는 기본적인 형태는 다음과 같다

class 클래스이름{

// 클래스 블록

}

< 객체 >

- 클래스는 사용자가 직접 정의한 타입 정도로 이해

- 기존에 이용했던 자바를 기준으로 생각해보면

- int, Array, String 등이 자바에서 미리 정의된 클래스

- 객체란?

- 클래스 타입으로 만들어진 변수 정도로 이해

- 그냥 단순히 변수라고 표현하기에는 부족하기 때문에 '객체' 또는 '인스턴스'라고 표현

- 기본적으로는 'new' 연산자를 통해서 생성되는 모든 변수는 객체가 된다

- 타입이 같은 거지, 객체가 같다고 볼 수는 없다

**12회차(2)**

package Test01;

public class TE1 {

// 1) 1부터 5까지의 합을 출력하는 메소드

// 2) 정수 3개를 배열에 입력 받아서 최댓값을 출력하는 메소드

static int sum(int a, int b) { // return 값 존재 -> int (1)

int sum = 0;

for (int i = a; i <= b; i++) {

sum += i;

}

return sum;

}

static void test01(){ // return 값 존재 X -> void (2)

int cumsum = 0; // 누적합!

for(int i = 0; i < 6; i++){

cumsum += i;

}

System.out.println(cumsum);

}

public static void main(String[] args) {

System.out.println("문제 1) 1부터 5까지의 합 : " + sum(1, 5)); // (1)

test01(); // (2)

}

}

**12회차(3)**

package Test01;

import java.util.Scanner;

public class TE2 {

// 문제 2) 1부터 n까지의 합을 돌려주는 메소드

static int add(int n) {

int sum = 0;

for(int i = 1; i <= n; i++) {

sum += i;

}

return sum;

}

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

System.out.print("n을 입력하시오 : ");

int n = sc.nextInt();

System.out.println("문제 2) 1부터 n까지의 합 : "+add(n));

}

}

**12회차(4)**

package Test01;

import java.util.Scanner;

public class TE3 {

// 문제 3) 정수 3개를 배열에 입력 받아서 최대값을 출력

static void test03() {

Scanner sc = new Scanner(System.in);

int[] arr = new int[3]; // 객체 생성

System.out.println("정수 3개 입력 : ");

// (1) 값 3개를 입력 받는데 가장 쉽게는

// 배열의 각 원소에 직접 하나씩 입력 받기

arr[0] = sc.nextInt();

arr[1] = sc.nextInt();

arr[2] = sc.nextInt();

// (2) 배열의 크기만큼 값을 입력 받고 싶다면

// 반복문 사용

for(int i =0; i < arr.length; i++) { // arr.length : 배열의 크기

arr[i] = sc.nextInt();

}

// 배열에서 가장 큰 값을 찾는 방법은?

// 변수 max 를 두고 해당 변수가 항상 큰 값이 유지되게 하는 방법

int max = 0;

for(int i = 0; i < arr.length; i++){

// 현재 max의 값보다 배열이 더 크면

// max의 값을 배열의 값으로 교체, 대입

if(max < arr[i]) max = arr[i];

}

System.out.println("문제 3) 정수 3개 중 가장 큰 값 : "+max);

}

public static void main(String[] args) {

test03();

}

}

**12회차(5)**

package Test01;

import java.util.Scanner;

public class TE4 {

// 문제 4) 배열 1개를 입력 받아서 배열의 가장 큰 값을 돌려주는 메소드

// 입력은 main메소드에서, 입력 받은 배열을 메소드에 전달

// 메소드는 전달받은 배열에서 가장 큰 값을 돌려주도록 합니다

static int test04( int [] arr ) {

int max = 0;

for(int i = 0; i < arr.length; i++){

if(max < arr[i]) max = arr[i];

}

return max;

}

public static void main(String[] args) {

// 배열의 값을 입력 받은 후 test04 메소드를 호출

Scanner sc = new Scanner(System.in);

int [] arr = new int[3];

System.out.print("배열 1개 입력 : ");

for(int i = 0; i < arr.length; i++) {

arr[i] = sc.nextInt();

}

System.out.println("문제 2) 배열 1개 중 가장 큰 값 : "+test04(arr));

}

}

**12회차(6)**

package Test01;

// 국영수 점수를 관리하기 위한 클래스 정의

class Student {

// 국영수 점수를 관리하기 위한 변수 필요

// 아래를 같은 클래스의 멤버라고 해서 '멤버변수'라고 부른다

int kor;

int eng;

int math;

// 평균을 구하는 메소드를 정의

// 마찬가지로 '멤버 메소드'라고 부른다

void getAve() {

// 멤버 메소드는 같은 멤버 변수를 언제나 참조할 수 있기 때문에

// 멤버 변수에 대한 평균을 구하기만 하면 된다

double ave = (kor + eng + math) / 3.0;

System.out.println("국영수 평균 점수는 : "+ave+"입니다.");

}

}

// 학생 한 명의 국영수 점수를 관리하고 싶다면,

// 해당 클래스 타입으로 변수(객체)를 만들어주면 된다

// 만약에 여러 학생의 국영수 점수를 관리하고 싶다면,

// 동일한 타입의 여러 객체를 만들어서 관리해주면 된다

public class TE5{

public static void main(String [] args) {

// 학생 한 명의 점수를 관리하기 위한 변수를 생성 (객체)

// 배열을 생성하는 것과 동일하게 new 연산자를 이용해서 정의

Student stu01 = new Student(); // stu01의 타입은 Student

// 객체를 생성한 이후에는 각 멤버에 접근

// 변수를 참조할 수도 있고 멤버메소드를 호출할 수도 있다

// 이때 참조 연산자는 .(dot)입니다 // -> 객체.멤버

// Q. 그럼 1번 학생의 평균을 구해보자

stu01.getAve();

// 점수가 입력되지 않았기 때문에 평균이 제대로 구해지지 않는다. -> (0.0점으로 나옴)

// 그럼 각 멤버변수에 접근해서 점수를 입력해보자 -> 평균이 제대로 나온다!

stu01.kor = 77;

stu01.eng = 55;

stu01.math = 50;

stu01.getAve();

}

}