**1회차 (1)**

package first;

//import java.lang.\*; //java.lang.\* <- \*이 뜻하는 내용은 java.lang안에 있는 모든 패키지를 사용하겠다.

//import - 내가 필요한 기능들을 사용할 때 불러오기 위해서 적는 명령

//import 패키지경로;

//패키지 - 파일들을 기능별로 구분하여 관리하기 쉽도록 모아놓은 폴더

//java.lang - 프로그램에서 기본적으로 필요한 기능들이 모여있는 패키지

// - 자동으로 import된다.

public class Hello {

//클래스명 첫글자는 대문자 달라지는 단어의 첫글자는 대문자. - 암묵적인 규칙

//클래스명과 파일명은 반드시 일치되어야한다.

//{} - 영역 또는 지역을 표시한다.(데이터의 묶음 처리)

//자바의 프로그래밍 코드는 반드시 클래스 영역 안에만 적어야한다. \*\*

public static void main(String[] args) {

//함수 == 메소드

//메소드 - 어떠한 기능을 실행하기 위한 코드의 집합체

//main메소드 - 프로그램이 실행할때 가장 먼저 실행되는 메소드 \*\*\*\*

//당분간은 main메소드 안에 실행코드를 적을 예정입니다 :)

//하나의 프로그램(class)에서는 반드시 하나의 main메소드

System.out.println("Hello Java");//명령문 - 프로그램의 실질적인 실행 문장

//JAVA에서의 표준 출력 명령

//C - printf("Hello Java");

//python - print("Hello Java");

// ; (종료문자) - 하나의 명령문이 끝나면 사용합니다.

//항상 모든 명령문의 끝에는 ;을 적어주어야한다.

//주석문 - 프로그램의 부연설명을 적는 문장

// - 프로그램 실행에 절대 관여를 하지 않는 문장

// - 한줄주석 : 부분적 코드를 적을때 사용 ( 메모 )

/\*

\* 범위 주석

\* - 여러줄을 주석처리 할 때 사용한다.

\* - 부분설명을 적을때 사용한다.

\*/

//저장 : ctrl + s 실행 : ctrl + F11

}

}

**1회차(2)**

상수

- 리터널 상수, 심볼릭 상수

리터널 상수

- 데이터의 그 자체

- 자동으로 공간이 생성되고 메모리에 저장된다.

- 공간을 구분할 수 있는 방법이 없기 때문에 공간을 다시 이용할 수 없다.

- 단발성 데이터

데이터를 다루는 공간들 ( 변수, 배열, 객체 ..)

- 지속성 데이터

변수

- 데이터 하나를 저장하는 공간

- 용도에 맞는 공간을 만들고 데이터를 저장시켜야 한다.

- 공간을 구분할 수 있는 이름이 있기 때문에 공간을 다시 이용할 수 있다.

- 지속성 데이터

- 선언

- 메모리에 변수 공간을 할당 받고 그 공간에 변수명을 붙이는 작업

- 용도에 맞는 공간을 만들어 주어야 한다.. ( 자료형 )

- 형식

자료형 변수명;

- 변수명 : 할당 받은 변수 공간에 구분하는 이름

- 변수공간을 호출할 때 사용한다.

- 암묵적 규칙

- 데이터의 성격이 연상되는 단어로 이름을 지어준다.

- 소문자를 권장한다.

# 작명 규칙

- 영문자 , 숫자, \_(언더바)로 구성된다. ( 한글 가능 )

- 대소문자 구별이 가능 ex) NAME과 name은 서로 다르다.

- 숫자로 시작할 수 없다. \*\* ex) 1name x, name1 o

- 공백이나 특수기호는 사용할 수 없다. \*\* ex) n ame x name o

- 예약어는 사용할 수 없다.. ex) int x float x while x for x

# 자료형

- 데이터의 형태를 구분 ( 정수, 실수, 문자 ) 하고 데이터의 범위나 부호의

사용 여부등을 결정하는 키워드

- 공간을 형식을 정의한 키워드

- 원시형 ( 기본형 ) - 데이터를 저장하고 관리하는 용도의 공간

- 정수형

byte , short , int (\*) , long

- 실수형

float , double (\*)

- 문자형 ( 단일문자 )

char

- 논리형 : 참과 거짓을 저장하고 관리하는 자료형

boolean

- 참조형 ( 객체형 ) - 실제 데이터 공간이 만들어지고 그 공간의 정보를

저장하는 공간이 함께 만들어지는 형태.. ( 기능이 함께 딸려온다.. )

- 기본형을 제외한 모든 자료형..

- 문자열 String

- 배열

- 초기화

- 할당 받은 변수공간에 값을 처음 넣는 작업

- 할당 받은 공간은 항상 쓰레기 값으로 채워져 있다.. 내가 원하는 데이터로 값을

처음 넣는 것이다.

- 초기화를 진행하지 않으면 공간의 데이터를 이용하려 할 때 error발생

- 형식

1. 선언과 동시에 초기화

자료형 변수명 = 값;

2. 선언 후 초기화

자료형 변수명;

변수명 = 값;

= (대입연산자) : 우측 공간의 데이터가 좌측 공간에 대입하는 연산자

- 호출

- 할당받은 변수공간은 불러오는 작업

- 호출하고 하는 위치에 변수명을 적으면 된다.

- 형식

변수명

**1회차(3)**

package variable;

public class Test01 {

public static void main(String[] args) {

//1. 선언과 동시에 초기화

int a = 20;

//변수명 : a

//자료형 : int ( 정수 )

//데이터 : 20

System.out.println(a);//변수의 호출

//변수의 호출 주의할점

// - 변수명은 "" , '' 감싸면 안된다

a = 10;//값 변경 ( 선언 x 초기화 x )

System.out.println(a);

//2. 선언 후 초기화

int a1;//변수의 선언

a1 = 20;//초기화

System.out.println(a1);//호출

//변수의 선언은 같은 변수명으로 단 한 번만 가능하다. \*\*

//int 1b; - 변수명은 숫자로 시작 x

//int double; - 변수명은 예약어로 사용이 불가능

int A;// - 변수명은 대소문자 구별한다.

A = 15;

System.out.println(a);

System.out.println(A);

}

}

**1회차(4)**

package variable;

public class Test02 {

public static void main(String[] args) {

//정수형

System.out.println(10);//단발성 데이터

//정수형 자료형

//byte 1byte -2^7 ~ 2^7-1 -128 ~ 127

//short 2byte -2^15 ~ 2^15-1 -32768 ~ 32767

//int 4byte -2^31 ~ 2^31-1

//long 8byte -2^63 ~ 2^63-1

//자료형 변수명 = 값;

int a = 20; // a라는 변수를 선언과 동시에 20으로 초기화하겠다

System.out.println(a);

int b;

b = 10; // b라는 변수를 선언 후 10이라는 값으로 초기화

System.out.println(b);

a = b; // b라는 변수공간 안에 있는 10이라는 값을 a라는 변수공간에 대입

System.out.println(a);

//정수형 데이터 사용시 주의 사항

//1. 정수형 데이터는 원형이다.

byte c = 127;

// ++ , --

//증감연산자 : 1증가 또는 1감소

c++;//128 -> -128 //c = c+1;

System.out.println(c);

c++;//129 -> -127

System.out.println(c);

c = -128;

c--;//-129 -> 127

System.out.println(c);

//c = -129; -자료형 값의 범위를 벗어났기때문에 error

//2. 정수끼리의 연산에서는 정수값으로 나온다.

int d = 5, e = 3;//같은 자료형은 ,로 이어서 쓸 수 있다.

System.out.println(d+e);//5 + 3 = 8 ( 더하기 )

System.out.println(d-e);//5 - 3 = 2 ( 빼기 )

System.out.println(d \* e);//5 \* 3 = 15 ( 곱하기 )

System.out.println(d / e);//5 / 3 = 1 ( 나누기 )

// - 정수끼리 연산에서는 몫이 나온다.

System.out.println(d % e);// 5 % 3 = 2 ( 나머지 )

}

}

**1회차(5)**

package variable;

public class Test03 {

public static void main(String[] args) {

//실수

//실수 자료형 유효자리수

//float 4byte 7자리

//double 8byte 15자리

float a = 123.456f;

//실수형 상수는 기본적 double형

//float로 인식시키려면 상수 뒤에 f를 적어주면 된다.

//1. 자료형이 큰 쪽에서 작은 쪽으로 이동하려 error가 발생

// - 같은 크기라면은 정수보다 실수가 더 크다.

// - 조금이라도 데이터의 손실이 발생될 수 있는 상황이라면 우리는 알고

// 적어라 라고 말한다.

// double > long > float > int > short == char > byte

// CAST연산자

// (자료형)상수 , 변수,(연산식)

// - 내가 원하는 데이터 타입으로 데이터를 일시적으로 변환시키는 연산자

System.out.println(a);//float형태

int b = (int)a;

System.out.println(b);

double d = a;//float -> double : double이 크기가 더 크기때문에

System.out.println(d);

b = (int)d;

System.out.println(b);

//2. 실수가 한 개라도 포함된 연산에서는 실수값이 나온다.

int kor = 60;

int eng = 13;

int math = 98;

int sub = 3;//과목의 개수

System.out.println((kor+eng) / 2);//(정수+정수) / 정수 -> 정수

System.out.println((kor+eng) / 2.0);//(정수+정수) / 실수 -> 실수

System.out.println((double)(kor+eng+math)/sub);

//(실수cast)(정수+정수+정수) / 정수 -> 실수

System.out.println((kor+eng+math)/(double)sub);

//(정수+정수+정수) / (실수cast)정수 -> 실수

//(60+13+98) / 3.0 -> 57.0

//3. 정밀한 계산에서는 실수를 사용하지 않는다.

System.out.println(1.3 % 0.2);//나머지 연산은 실수x

//정수 % 정수 o

}

}

**1회차(6)**

package variable;

public class Test04 {

public static void main(String[] args) {

//String - 문자열

// - 참조형

// - 지금은 일반변수처럼 사용

// - 문자열을 구별하기 위해서 ""(쌍따옴표)안에 저장한다.

String name = "홍길동";

System.out.println(name);

//문자열은 연산이 불가능하다.

//단, 문자열 더하기는 가능하다.

//문자열 더하기는 문자열 뒤에 문자열을 이어 붙여준다.

System.out.println("이름 : "+name);

System.out.println(name +"더하기");

//"홍길동" + "더하기" -> "홍길동더하기"

System.out.println(name+10);

//"홍길동" + 10 -> "홍길동10"

System.out.println(10+20+name);

//10 + 20 -> 30

//30 + "홍길동" -> "30홍길동"

System.out.println(name+10+20);

//"홍길동" + 10 -> "홍길동10"

//"홍길동10" + 20 -> "홍길동1020"

System.out.println("내이름은 "+name+"입니다");

//"내이름은 "+"홍길동" -> "내이름은 홍길동"

//"내이름은 홍길동" + "입니다" -> "내이름은 홍길동입니다"

//자바는 문자열 이어서 출력하는 것이 불가능하므로 +연산자로 이어붙이기 해야 한다.

//우선순위로 바꾸어 주기 위해서는 ()안에 먼저 적는다.

}

}