훈련교사 : 전 은 석
email : euns_jun@naver.com

Java II

❖ 자바 프로그램에서 사용하는 단어의 종류

예약어(키워드), 상수, 식별자

◆ 식별자

식별(識別)자란?

- 보고 느낄 수 있는 모든 사물(객체)들을 각각 구별할 수 있는 것을 의미 한다.
- 클래스, 메소드, 변수, 상수에 프로그래머가 정하는 이름
  (프로그래밍에서 사용하는 모든 이름)

* 대소문자가 구분되며 길이에 제한이 없다. (Abcd와 abcd는 다른것으로 간주)
* 숫자로 시작해서는 안된다.
* 식별자에는 예약어(키워드), 상수를 쓸수 없다.
* 식별자가 겹쳐져서 구분해줘야 할 경우 마침표(.)로 연결해서 소속을 표시

◆ 상수 : true, false, null

◆ 예약어 (키워드) : 이미 약속된 의미를 갖는 단어

abstract	assert	boolean	break	byte	case	catch	char	class	const
continue	default	do	double	else	enum	extends	final	finally	float
for	goto	if	implements	import	instanceof	int	interface	long	native
new	package	private	protected	public	return	short	static	strictfp	super
switch	synchronized	this	throw	throws	transient	try	void	volatile	while

클래스 정의 규칙

- 첫 글자는 대문자로 한다.
- 두 단어 이상으로 이루어지는 경우 각 단어의 첫글자는 대문자로 한다.(Camel 표기법)
- 한글도 쓸수 있다.(가능하면 피할 것)
- 특수 문자는 _, $ 는 쓸 수 있다(첫 글자에도 가능). 나머지는 쓸 수 없다.
- 첫 글자로 숫자가 올 수 없다.

변수, 메소드 정의 규칙

- 첫 글자는 소문자로 한다.
- 두단어 이상으로 이루어지는 경우 두번째 단어 이후 첫글자는 대문자로 한다. (Camel 표기법)

상수 정의 규칙

- 모든 문자는 대문자를 사용한다.
- 두 단어 이상이 될 경우 각 단어를 언더 스코어(_) 로 연결한다.

자바에서의 기호들

- 소괄호, 중괄호, 대괄호 (( ), { }, [ ]) : 영역을 구분
- 작은 따옴표 : 문자를 표시
- 큰 따옴표 : 문자열을 표시
- 세미콜론(;) : 명령문의 끝을 표시하는 기호

❖ 자바 프로그램의 구조

자바프로그램은 반드시 모든 내용이 하나의 class 안에 포함되어야 한다.

◆ class를 만드는 방법

```
[접근지정자] [속성] class 클래스이름 {

}
```

- 예제

```
public class Test {

}
```

* 형식에 있어서 [ ] 안의 내용은 생략해도 무방하다.

훈련교사 : 전 은 석
email : euns_jun@naver.com

◈ 파일이름 지정하는 방법

1. 하나의 파일에는 하나의 class 만 만드는 것을 원칙으로 한다.
그러므로 파일의 이름은 class 의 이름과 동일하게 만들어야 한다.

2. 어쩔수 없이 하나의 파일에 두개 이상의 class를 만들게 되는 경우
   1) 그중에 public 클래스의 이름으로 정한다.
   (하나의 파일에는 public 클래스는 0개 ~ 1개까지만 허락한다.)
   2) 만약 public class가 없으면
      a. main 함수가 있는 클래스이름으로 한다.
      b. main 함수가 없거나 main() 가 여러개 있으면
      아무 클래스이름으로 하면된다.

훈련교사 : 전 은 석
email : euns_jun@naver.com

진입점 함수

클래스가 실행되기 위해서는 JVM에 의해서 실행되는 함수가 있어야 한다.
JVM은 오직 "public static void main(String[] args)" 함수만 사용한다.

따라서 만약 어떤 클래스를 만들고 그 클래스가 실행되기 위해서는 반드시
public static void main(String[] args)가 존재해야만 한다.

이처럼 어떤 프로그램을 시작하는 함수를 **진입점 함수**라고 부른다.

* class 안에는 필요한 함수나 필요한 변수를 포함할 수 있다.

❖ 자료형

◆ 기본 자료형 (Primitive Data Type)

자바에서 사용하는 기본 데이터 타입

* : 숫자형 기본 입력 타입

구 분	자료형	byte	범 위
논리형	boolean	1	true, false
문자형	char	2	0 ~ 2^{16} - 1(0 ~ 65535)
정수형	byte	1	-128 ~ 127 (-2^7 ~ 2^7 - 1)
	short	2	-32,768 ~ 32,767 (-2^{15} ~ 2^{15} - 1)
	int *	4	-2147483648 ~ 2147483647 (-2^{31} ~ 2^{31} - 1)
	long	8	-9223372036854775808 ~ 9223372036854775807 (-2^{63} ~ 2^{63} - 1)
실수형	float	4	1.4E-45 ~ 3.4028235E38
	double *	8	4.9E-324 ~ 1.79769313486231570E308

* 실수형은 양의 값만 표시 됨

변수

- 아직 알려지지 않거나 어느 정도까지만 알려져 있는 양이나 정보에 대한 상징적인 이름
- 다른 언어에서는 스칼라(scalar)라 부르기도 한다.
- 데이터를 담는 일종의 그릇

종류

- 기본형 (Primitive Type)
  - 기본 자료형 8개를 저장
  - 실제 값이 저장
- 참조형(Reference Type)
  - 객체 주소를 저장
  - 기본형을 제외한 나머지 타입
  - 변수의 타입으로 클래스의 이름을 사용

형식

- 기본자료형 변수이름; → 변수 선언
- 기본자료형 변수이름 = 데이터; → 변수 선언 & 초기화
- 클래스이름 변수이름;
- 클래스이름 변수이름 = new 클래스이름();

상수

- 알려져 있는 양이나 정보에 대한 상징적인 이름
- 한번 정하면 수정할 수 없다.
- 상수명은 대문자로 한다.
- 상수는 선언과 동시에 초기화 해야한다.(값을 변경할 수 없으므로...)

◆ 형식

- final 데이터타입 상수명(대문자) = 데이터; → 상수 선언 & 초기화

변수 선언의 의미

변수 선언은 단순히 데이터를 담는 것만이 아니다.
데이터의 형태와 데이터의 크기, 필요한 메모리의 크기도 지정한다는 의미

int num ; => num이라는 변수에
int 형의 데이터 &
4바이트의 데이터 &
메모리 4바이트 할당받는다.

❖ 형변환

값의 자료형을 원하는 자료형으로 변환하는 작업

❖ 리터럴 형변환

데이터를 입력하면 리터럴 영역에 저장이 먼저 된다.
이때 테이터의 타입이 자동으로 결정되고 크기도 정해진다. 이것을 리터럴 타입이라 한다.
정수는 int로 자동 결정되고 실수는 double 타입으로 자동으로 결정이 된다.

이때 메모리의 크기를 정해줄 수도 있는데 이때 쓰는 방법이 리터럴 형변환이다.

＊ 방법

입력할 데이터뒤에 알파벳을 붙여준다.

- 데이터L; → long타입
- 데이터F; → float타입

📌 정수형 데이터는 리터럴 형변환 없이 사용할 수 있다.

❖ 형변환

❖ 자동 형변환

지정하지 않아도 자동적으로 형태를 바꿔서 사용되는 경우
: 작은 형태의 데이터가 큰 형태의 데이터로 필요한 경우 자동 형변환이 일어난다.

예제

```
float num1 = 10;     // 성공
float num2 = 10.;    // 실패
```

❖ 강제 형변환

자동 형변환이 불가능한 경우 개발자가 강제로 형 변환을 할 필요가 있다.

◆ 형식

```
(변환할 데이터형) 데이터;
```

예제

```
(float) 10.0;
```

❖ 연산자

연산자란
자료의 가공을 위해 정해진 방식에 따라 계산하고 결과를 얻기 위한 행위를 의미하는 기호들의 총칭이다.
각 연산자들은 연산을 하기 위해 인식하는 자료형들이 정해져 있다.

❖ 연산자의 종류와 우선순위

종류	연산자	우선순위
증감 연산자	++, --	1순위
산술 연산자	+, -, *, /, %	2순위
시프트 연산자	〉〉, 〈〈, 〉〉〉	3순위
비교 연산자	〉, 〈, 〉=, 〈=, = =, !=	4순위
비트 연산자	&, |, ^, ~	~만 1순위, 나머지는 5순위
논리 연산자	&&, ||, !	!만 1순위, 나머지는 6순위
조건(삼항) 연산자	?, :	7순위
대입 연산자	=, *=, /=, %=, +=, -=	8순위

❖ 대입 연산자

특정한 상수 값이나 변수 값 또는 객체를 변수에 전달하여 기억시킬 때 사용하는 연산자이다

❖ 대입 연산자의 종류

구분	연산자	의미
대입 연산자	=	연산자를 중심으로 오른쪽 변수값을 왼쪽 변수에 대입한다.
	+=	왼쪽 변수에 더하면서 대입한다.
	-=	왼쪽 변수값에서 빼면서 대입한다.
	*=	왼쪽 변수에 곱하면서 대입한다.
	/=	왼쪽 변수에 나누면서 대입한다.
	%=	왼쪽 변수에 나머지 값을 구하면서 대입한다.

예제

```
int no1 = 10;
no1 += 10;
System.out.println( no1 );
```

❖ 산술연산자

4칙 연산(+, -, *, /)과 나머지 값을 구하는 연산자(%)

❖ 산술연산자의 종류

구분	연산자	의미
산술 연산자	+	더하기
	–	빼기
	*	곱하기
	/	나누기
	%	나머지 값 구하기

- 예제

```
int no1 = 10 / 3 ;
System.out.println("no1 = " + no1);
```

```
int no1 = 10%3;
System.out.println("no1 = " + no1);
```

❖ 증감연산자

1씩 증가 또는 감소시키는 연산자이다.
무엇보다 중요한 것은 ++ 또는 --와 같은
연산자가 변수 앞에 위치하느냐? 아니면 변수 뒤에 위치하느냐?가 더 중요한 연산자이다.

❖ 증감연산자의 종류

구분	연산자	의미
증감 연산자	++	1씩 증가시킨다.
	--	1씩 감소시킨다.

예제

```
int no1 = 10;
System.out.println("no1 = " + (++no1));
```

```
int no1 = 10;
System.out.println("no1 = " + (no1++));
System.out.println("no1 = " + no1);
```

예제

```
int     no1 = 10;
int     no2 = no1++ + no1++;
System.out.println( no2 );
```

예제

```
int no1 = 10;
int no2 = ++no1 + no1++;

System.out.println( no2 );
```

❖ 비교 연산자

변수나 상수의 값을 비교할 때 쓰이는 연산자로서
결과가 항상 true 또는 false인 논리값(boolean)이어야 한다.

❖ 비교 연산자의 종류

구분	연산자	의미
비교 연산자	〉	크다.
	〈	작다.
	〉=	크거나 같다.
	〈=	작거나 같다.
	= =	피연산자들의 값이 같다.
	!=	피연산자들의 값이 같지 않다.

예제

```
System.out.println(1 > 2 );
```

❖ 논리 연산자

true나 false인 논리 값을 가지고 다시 한번 조건 연산하는 연산자이다.
하나 이상의 처리 조건이 있어야 하며 먼저 처리되는 조건에 따라
다음의 처리 조건을 처리할지 안 할지를 결정하는 말 그대로 논리적인 연산자이다.

❖ 논리 연산자의 종류

연산자	의미	설명
&	and(논리곱)	주어진 조건들이 모두 true일 때만 true를 나타낸다.
&&		
|	or(논리합)	주어진 조건들 중 하나라도 true이면 true를 나타낸다.
||		
!	not(부정)	true는 false로 false는 true로 나타낸다.

✻ **&& 와 || 는 절삭 연산을 한다.**

연산자	설명
&&	선조건이 true일 때만 후조건을 실행하며 선조건이 false이면 후조건을 실행하지 않는다.
||	선조건이 true이면 후조건을 실행하지 않으며 선조건이 false일 때만 후조건을 실행한다.

예제

```
System.out.println(true & false );
System.out.println(true || false );
```

예제

```
int no1 = 10;
int no2 = 11;
int no3 = 11;

System.out.println(no1++ > no2 && no1 < no3 );
```

❖ 시프트 연산자

시프트 연산은 대상 필드의 값을 비트로 바꾼 후 비트 수만큼 이동시켜서 값을 얻는 연산이다.
이연산은 boolean, float, double 형의 경우는 사용할 수 없다.

많은 CPU에서는 상수 곱셈 등의 연산을 시프트 연산자로 처리하는 것이
산술 논리 장치를 거치는 것보다 빠르기 때문에,
컴파일러에서는 곱셈/나눗셈 연산을 자동적으로 산술 시프트 명령어로 변환한다.

❖ 시프트 연산자의 종류

구분	연산자	의미
시프트 연산자	>>	bit값을 오른쪽으로 이동(빈 자리는 부호값으로 대입)한다.
	<<	bit값을 왼쪽으로 이동(빈 자리는 0으로 대입)한다.
	>>>	bit값을 오른쪽으로 이동(빈 자리는 0으로 대입)한다.

예제

```
System.out.println(11 << 2 );
```

← 11×2^2

❖ 비트 연산자

피연산자 즉 연산의 대상이 되는 값들을
내부적으로 bit단위로 변경한 후 연산을 수행하는 연산자이다.

❖ 비트 연산자의 종류

구분	연산자	의미
비트 연산자	&	비트 단위의 AND
	|	비트 단위의 OR
	^	XOR(배타적 OR)

예제

```
int no1 = 168 ;
int no2 = 245 ;
System.out.println( no1 & no2);
System.out.println( no1 | no2);
System.out.println( no1 ^ no2);
```

	128	64	32	16	8	4	2	1	
no1	1	0	1	0	1	0	0	0	168
no2	1	1	1	1	0	1	0	1	245
&	1	0	1	0	0	0	0	0	160
|	1	1	1	1	1	1	0	1	253
^	0	1	0	1	1	1	0	1	93