

자바기초

2007. March. 26

자바 기초문법 익히기

류 대 우

davidryu@newtc.co.kr

학습 목표

■ 학습 목표

- 블록의 시작과 끝, 문장의 끝을 처리하는 방법을 설명할 수 있다.
- 자바에서 이름 붙이는 법을 안다.
- 주석의 사용 방법을 안다.
- 자바의 기본적인 데이터 타입과 연산자에 대해 알고 설명할 수 있다.

자바 프로그램의 영역

- ‘영역(block)’ 이란?
 - 클래스와 Method는 항상 시작과 끝이 있으며, 이 시작과 끝 사이에 있는 부분.
 - 서로 짝을 이루지 않으면 에러가 생김
 - 반복문 구조에도 많이 사용.
 - 메서드나 클래스 영역의 시작문장에는 {}를 사용함.
 - 주의 : 모든 실행문의 끝에는 세미콜론(;)을 사용함.
- `public class MyFirst{ //`→클래스영역의 시작
 - `public static String getMessage(){//`→ 메서드 영역의 시작
`return “나의 첫 자바 프로그램”;`
 - `}` //→ 메서드 영역의 끝
- `}` //→클래스영역의 끝

이름 붙이는 규칙(I)

- 클래스, method, 변수의 이름을 정할 때 유의사항
 - 숫자로 시작할 수 없다
 - 밑줄(_), 달러(\$) 이외의 특수 문자 사용 불가
 - Reserved word, 키워드 사용 불가(ex : this, class)
 - 대소문자를 구별함
 - 이해 가능한 프로그램 코드를 생각한다.

이름 붙이는 규칙(II)

- 읽기 쉬운 코드를 만드는 법
 - Class
 - 클래스 이름은 항상 대문자로 시작함
 - 두 개 이상의 단어 조합의 경우 각각 첫 글자를 모두 대문자로 써 줌
 - Method
 - 이름의 첫 글자는 소문자로 시작함
 - 첫 단어 첫 글자는 소문자, 다른 단어 첫 글자는 대문자로 시작함

이름 붙이는 규칙(III)

- 읽기 쉬운 코드를 만드는 법
 - 변수
 - 이름의 첫 글자는 소문자로 시작함
 - 첫 단어 첫 글자는 소문자, 다른 단어 첫 글자는 대문자로 시작함
 - 또는, 두 단어 사이에 _를 넣어준다.
 - 상수
 - 모두 대문자로 나타냄
 - 또는, 두 개 이상의 단어 조합의 경우, 단어 사이에 _를 넣어준다.

주석

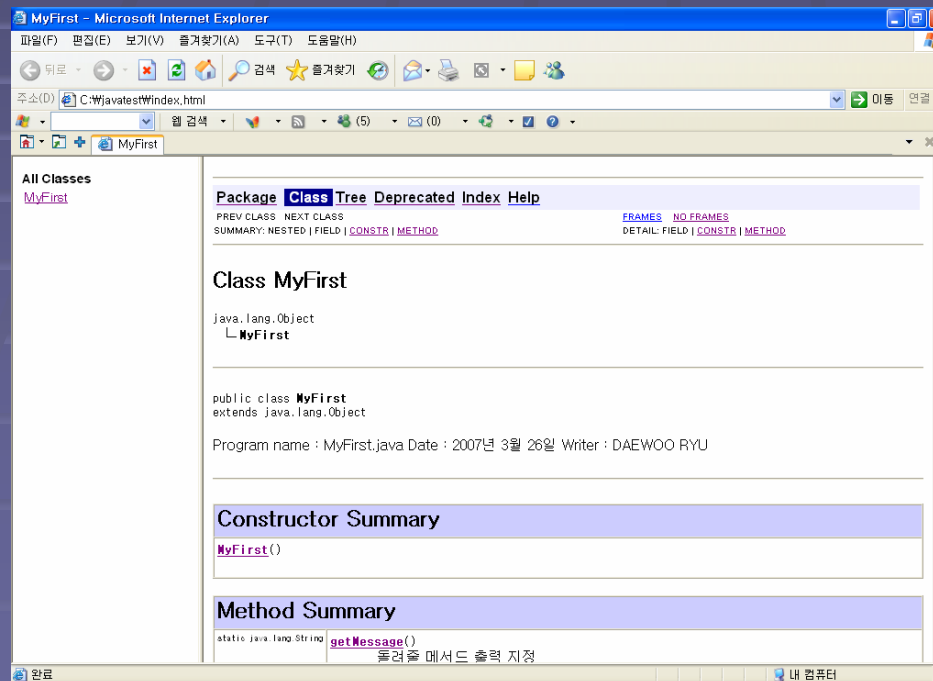
- 주석이란?
 - 프로그램을 설명하기 위해 사용하는 문장
 - 프로그램 수행에 아무런 영향도 미치지 않음
 - 즉, *.java 파일은 파일 크기가 늘어나지만, *.class 파일의 크기는 늘어나지 않음.
- 주석문의 종류
 - // : 한 줄의 문장만 주석처리
 - /* */ : 하나 또는 여러 줄의 문장을 주석 처리
 - /** */ :
 - 하나 또는 여러 줄의 문장을 주석 처리
 - Javadoc Comments라고 함.

주석문을 이용한 API 문서 만들기

```
■ /**
    ■ Program name : MyFirst.java
    ■ Date : 2007년 3월 26일
    ■ Writer : DAEWOO RYU
    ■ */
    ■ public class MyFirst {
    ■ /* 출력 스트링 내용 지정 */
    ■     private static String str = "안녕! 자바야 ";
    ■     /** 돌려줄 메서드 출력 지정 */
    ■     public static String getMessage(){
    ■         return "메서드 출력임";
    ■     }
    ■     /** 프로그램의 시작점인 main 메서드 */
    ■     public static void main(String args[])
    ■     {
    ■         System.out.println(str + getMessage());
    ■     }
    ■ }
```

주석문을 이용한 API 문서 만들기

- javadoc MyFirst.java 를 실행하면, 해당 폴더에 여러 개의 파일이 만들어 짐.
- index.html 명령을 실행해 보시다.

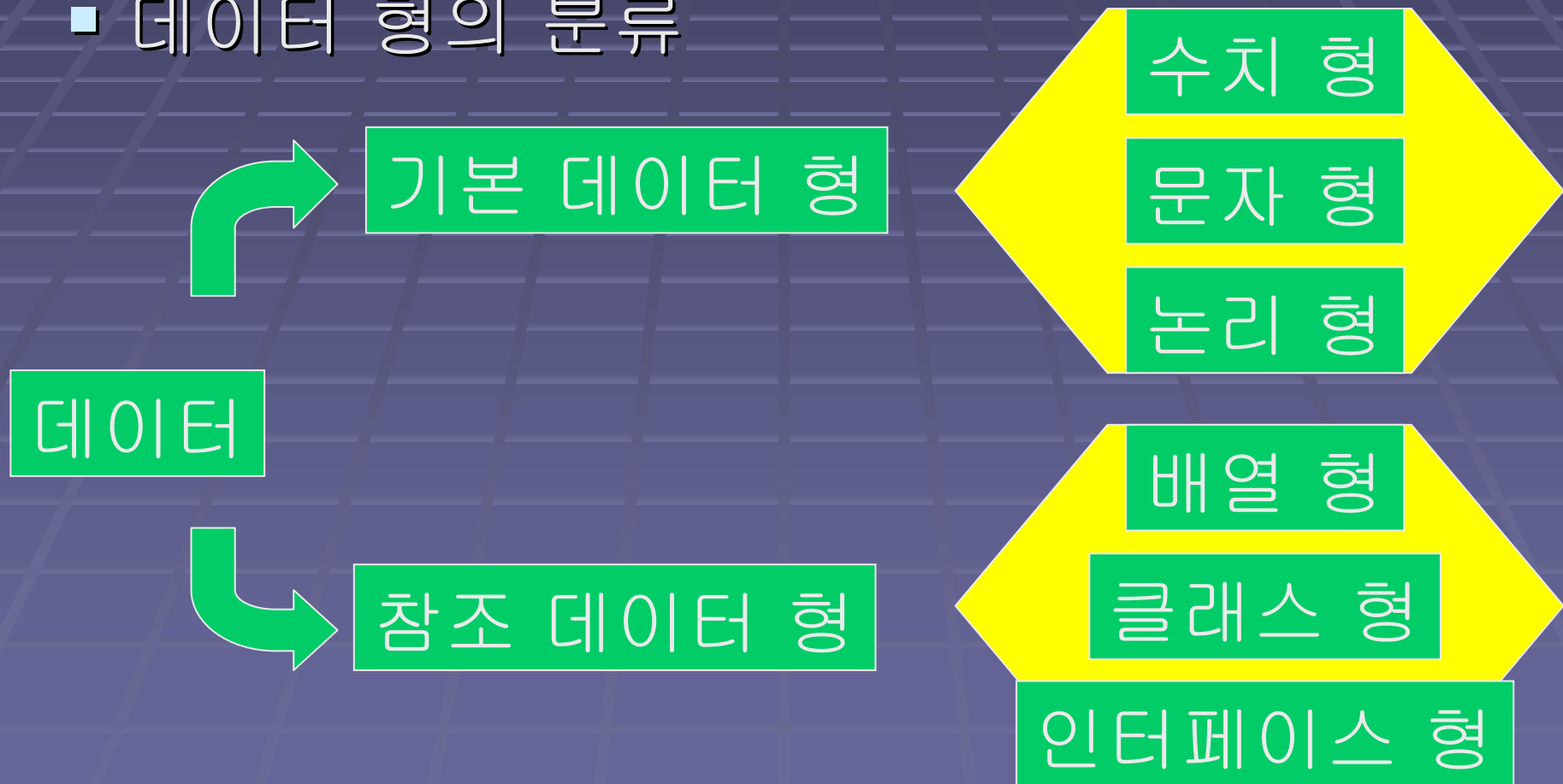


javadoc 태그 목록

- @author 소스 파일의 제작자
- @version 프로그램의 버전
- @see 외부 doc 문서의 클래스를 링크
- @deprecated
 - JDK의 새로운 버전이 등장하면서, 설계나 다른 이유로 폐기된 method나 권장하지 않는 method를 알려줌
- @param 매개 변수 설명
- @return Method의 반환 값을 설명

데이터 타입

- 데이터 형의 분류



기본 데이터 형의 종류별 사용 범위

종류	type	크기(bit)	표현범위
정수형	byte	8	-128~127
	short	16	-32768~32767
	int	32	
	long	64	
실수형	float	32	
	double	64	
문자형	char	16	0~2의 16승
논리형	boolean	1	true or false

연산자

- 정의

- Operator
- 변수와 상수 등 데이터 대상에 대해 연산 동작을 행하는 기호들
- 이 때, operation의 대상이 되는 것을 피 연산자 (Operand)라고 한다.

- 분류

- 단 항 연산자(Unary Operator)
- 이 항 연산자(Binary Operator)
- 삼 항 연산자(Tenary Operator)

연산자

- 우선순위
 - 괄호()안의 식이 가장 먼저 실행
 - 사칙 연산자는 *,/,+,- 연산자 순으로 실행
 - 좌측에서 우측의 순서로 실행
 - 단, = 연산자는 우측에서 좌측으로 실행

단 항 연산자

- 단 항 음수 연산자

연산자	일반 형식	의미
-	-피연산자	피 연산자의 부호를 반대로 바꾸어준다.

단 항 연산자

- 증가/감소 연산자

연산자	일반 형식	의미
++	++피연산자 피연산자++	피 연산자에 1 증가 후 사용 피 연산자 사용 후 1 증가
--	--피연산자 피연산자--	피 연산자에 1 감소 후 사용 피 연산자 사용 후 1 감소

단항 연산자

- public class unary{
 - public static void main(String[] args){
 - int ken=10, yoo=-20, boo;
 - boo=-yoo;
 - System.out.println("***** result *****");
 - System.out.println("ken=" + ken + "," +
 - ("yoo=" + yoo + "," +
 - ("boo=" + boo);
 - System.out.println("++ken=" + (++ken) + ", ken=" + ken);
 - System.out.println("ken++=" + (ken++) + ", ken=" + ken);
 - System.out.println("--boo=" + (--boo) + ", boo=" + boo);
 - System.out.println("boo--=" + (boo--) + ", boo=" + boo);
 - }
- }

이항 연산자

- 산술 연산자
 - Arithmetic Operator

연산자	일반 형식	의미
+	$Op1 + Op2$	Op1 과 Op2의 덧셈
-	$Op1 - Op2$	Op1 과 Op2의 뺄셈
*	$Op1 * Op2$	Op1 과 Op2의 곱셈
/	$Op1 / Op2$	Op1 과 Op2의 나눗셈
%	$Op1 \% Op2$	Op1 과 Op2의 나눈 나머지
^	$Op1 ^ Op2$	Op1의 Op2 승

산술 연산자

- public class Arithmetic{
 - public static void main(String[] args){
 - int x=100, y=30;
 - System.out.println("***** result *****");
 - System.out.println("x + y="+ (x + y));
 - System.out.println("x - y="+ (x - y));
 - System.out.println("x * y="+ (x * y));
 - System.out.println("x / y="+ (x / y));
 - System.out.println("x % y="+ (x % y));
 - System.out.println("y ^ 3="+ (y*y*y));
 - }
- }

이항 연산자

- 관계 연산자

- Relational Operator : 두 수식간의 대소 비교에 사용

연산자	일반 형식	의미
==	Op1 == Op2	두 Op가 같으면 true를 반환함
!=	Op1 != Op2	두 Op가 같지않으면 true를 반환함
>	Op1 > Op2	Op1 이 Op2보다 크면 true를 반환
>=	Op1 >= Op2	Op1 이 크거나 같으면 true를 반환
<	Op1 < Op2	Op1 이 Op2보다 작으면 true 반환
<=	Op1 <= Op2	Op1 이 작거나 같으면 true를 반환

관계 연산자

- public class Relational{
 - public static void main(String[] args){
 - int x=50, y=30;
 - System.out.println("***** result *****");
 - System.out.println("x < y --> "+ (x < y));
 - System.out.println("x <= y --> "+ (x <= y));
 - System.out.println("x > y --> "+ (x > y));
 - System.out.println("x >= y --> "+ (x >= y));
 - System.out.println("x == y --> "+ (x == y));
 - System.out.println("y != 3 --> "+ (x!=y));
 - }
- }

이항 연산자

- 논리 연산자

- Logical Operator : 논리식의 참과 거짓을 판정

연산자	일반 형식	의미
&&	Op1 && Op2	Logical AND
	Op1 Op2	Logical OR
!	!Op1	Logical NOT

논리 연산자

```
■ public class Logical{  
    ■ public static void main(String[] args){  
        ■ boolean x=true, y=true, z=false;  
        ■ System.out.println("***** result *****");  
        ■ System.out.println("boolean x=true, y=true, z=false");  
        ■ System.out.println("--- logical AND---");  
        ■ System.out.println("x && y --> "+ (x && y));  
        ■ System.out.println("y && z --> "+ (y && z));  
        ■ System.out.println("--- logical OR---");  
        ■ System.out.println("x || y --> "+ (x || y));  
        ■ System.out.println("y || z --> "+ (y || z));  
        ■ System.out.println("--- logical NOT---");  
        ■ System.out.println("!x --> "+ (!x));  
    ■ }  
■ }
```

이항 연산자

- 비트 연산자
 - Bitwise Operator

연산자	일반 형식	의미
&	Op1 & Op2	각 비트단위로 AND
	Op1 Op2	각 비트단위로 OR
^	Op1 ^ Op2	각 비트단위로 XOR
~	~Op1	각 bit를 반전 (즉, 1의 보수임)
<<	Op1 << Op2	Op1을 Op2만큼 왼쪽으로 비트이동
>>	Op1 >> Op2	Op1을 Op2만큼 오른쪽으로 비트이동 (이동 전 최상위 비트 값 적용)
>>>	Op1>>>Op2	Op1을 Op2만큼 오른쪽으로 비트이동 (최상위비트 0 적용)

비트 연산자

```
■ public class Bitwise{  
    ■ public static void main(String[] args){  
        ■ int ken=2, boo=7, yoo=-7;  
        ■ System.out.println("***** result *****");  
        ■ System.out.println("ken=2, boo=7, yoo=-7");  
        ■ System.out.println("--- bitwise AND---");  
        ■ System.out.println("ken & boo --> "+ (ken & boo));  
        ■ System.out.println("--- bitwise OR---");  
        ■ System.out.println("ken | boo --> "+ (ken | boo));  
        ■ System.out.println("--- bitwise XOR---");  
        ■ System.out.println("ken ^ boo --> "+ (ken ^ boo));  
        ■ System.out.println("--- COMPLEMENT---");  
        ■ System.out.println("~ken --> "+ (~ken));  
        ■ System.out.println("--- SHIFT---");  
        ■ System.out.println("boo<<2 --> "+ boo<<2 );  
    ■ }  
■ }
```

이항 연산자

- 대입 연산자
 - Assignment Operator

연산자	일반 형식	의미
=	Op1 = Op2	Op2를 Op1에 대입하여 기억시킴 즉, 결합 규칙은 오른쪽에서 왼쪽으로 결합함 전제조건: Op1과 Op2는 항상 변수가 되어야 함.

- 복합 대입 연산자
 - Compound Assignment Operator
 - +=, -=, *=, /=, %=, &=, |=, ^=, <=, >= 등등

대입 연산자

- public class Assignment{
 - public static void main(String[] args){
 - int x=7, y=2;
 - int a, b;
 - a=7;
 - b=2;
 - System.out.println("***** result *****");
 - System.out.println("x= --> "+ x + ",y=" +y);
 - System.out.println("a= --> "+ a + ",b=" +b);
 - System.out.println("x+=y --> "+ x+=y + ",a=a+b →" +(a=a+b));
 - }
- }

삼항 연산자

- 조건 연산자
 - Conditional Operator

연산자	일반 형식	의미
?:	Op1 ? Op2:Op3	Op1이 참이면 Op2를 거짓이면 Op3를 실행

- ```
public class Conditional{
 ■ public static void main(String[] args){
 ■ int x=10, y=20, ken, boo, yoo, soo;
 ■ Ken=boo=30;
 ■ yoo = (ken==boo ? x : y);
 ■ soo = (ken < boo ? x : y);
 ■ System.out.println("***** result *****");
 ■ System.out.println("yoo= --> "+ yoo + ", soo="+ soo);
 }
}
```

# 기타 연산자

- 연결 연산자
  - Concatenation Operator

| 연산자 | 일반 형식         | 의미               |
|-----|---------------|------------------|
| +   | “문자열1”+“문자열2” | “문자열1 문자열2”를 출력함 |

- cast 연산자
  - Cast Operator

| 연산자 | 일반 형식         | 의미                                                |
|-----|---------------|---------------------------------------------------|
| ()  | (데이터형)OPERAND | 임의의 피연산자의 데이터형을 ()안의 데이터 형으로 강제로 바꿀 때 사용하는 연산자이다. |

# CAST 연산자

```
■ public class Cast{
 ■ public static void main(String[] args){
 ■ int x, y;
 ■ float c, d;

 ■ x=10;
 ■ y=3;
 ■ c=x/y;
 ■ d=(float) x/y;

 ■ System.out.println("***** result *****");
 ■ System.out.println("c= --> "+ c + ",d=" +d);
 ■ }
■ }
```

# 연산자의 우선 순위

## ■ 자바 연산자의 우선순위

| 우선순위 | 연산자                                 | 의미            |
|------|-------------------------------------|---------------|
| 1    | (), []                              | 괄호, 대괄호       |
| 2    | !, ++, --                           | 부정, 증감        |
| 3    | *, /, %                             | 산술(승, 제, 나머지) |
| 4    | +, -                                | 산술(가, 감)      |
| 5    | <<, >>, >>>                         | 비트            |
| 6    | <,>, <=, >=                         | 관계            |
| 7    | ==, !=                              | 관계            |
| 8    | &                                   | 비트            |
| 9    | ^                                   | 비트            |
| 10   |                                     | 비트            |
| 11   | &&                                  | 논리            |
| 12   |                                     | 논리            |
| 13   | ? :                                 | 조건            |
| 14   | =, +=, -=, *=, /=, ^=,  =, >>=, <<= | 대입, 복합대입      |