

Java



자바의 기초 문법(1)

자바 프로그램 작성의 기초
로컬 변수의 선언/이용
대입문의 작성
배열의 선언/생성/이용



자바의 기초 문법

01. 자바 프로그램 작성의 기초

컴퓨터가 일을 처리하는 방법

- 명령대로 수행하는 컴퓨터



Hello, Jane



Hello, Tom

Hello, Java라고
출력해라.



`System.out.println("Hello, Java");`

Hello, Java

네, 시키는대로
하겠습니다.



자바의 기초 문법

01. 자바 프로그램 작성의 기초

컴퓨터가 일을 처리하는 방법

- 문법이 조금만 어긋나도 못 알아듣는 컴퓨터



안녕하3



안녕하세요라는
뜻인가보군



`System.out.println("Hello, Java");`



???

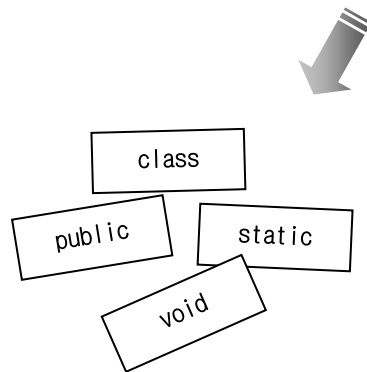
자바의 기초 문법

01. 자바 프로그램 작성의 기초

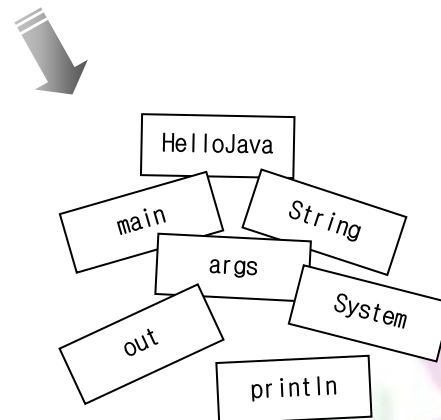
자바 프로그램의 구성요소

- 단어: 키워드(keyword), 식별자(identifier), 상수를 표현하는 단어

```
class HelloJava {  
    public static void main(String args[]) {  
        System.out.println("Hello, Java");  
    }  
}
```



키워드(keyword)



식별자(identifier)

Identifier & Keyword

Identifier

- Identifier? 보고 느낄 수 있는 모든 사물(객체)들을 각각 구별할 수 있는 것을 의미,

❖ Identifier Definition Rule

구분	정의 규칙	사용 예
클래스	<ul style="list-style-type: none">■ 첫 문자는 항상 대문자로 표현■ 하나 이상의 단어가 합쳐질 때는 각 단어의 첫 문자들만 대문자로 표현	<pre>class JavaTest{ ...; }</pre>
변수와 메서드	<ul style="list-style-type: none">■ 첫 문자는 항상 소문자로 표현■ 하나 이상의 단어가 합쳐질 때는 두 번째부터 오는 단어의 첫 문자들만 대문자로 표현	<pre>String itLand; public void getTest(){ ...; }</pre>
상수	<ul style="list-style-type: none">■ 모든 문자를 대문자로 표현■ 하나 이상의 단어가 합쳐질 때 공백 필요 시 <code>under score()</code>를 사용하여 연결한다.	<pre>int JAVATEST = 10; int JAVA_TEST = 20;</pre>

자바의 기초 문법

01. 자바 프로그램 작성의 기초

자바 프로그램의 구성요소

- 자바의 키워드들

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

- 상수 값을 표현하는 단어들

true	false	null
------	-------	------

Identifier & Keyword

● Keyword

- ▶ Java Programming을 하는데 있어 특정한 의미가 부여되어 이미 만들어진 Identifier를 말한다.
- ▶ Keyword에 등록되어 있는 것을 Programming에서 Identifier로 사용할 수 없다.

(const와 goto는 예약어로 등록만 되어 있을 뿐 사용되지 않는 예약어이다.)

※ 예약어의 종류

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

자바의 기초 문법

01. 자바 프로그램 작성의 기초

자바 프로그램의 구성요소

- 여러가지 기호: 대괄호, 중괄호, 소괄호, 마침표, 세미콜론(;) 등

```
class HelloJava {  
    public static void main(String args[]) {  
        System.out.println("Hello, Java");  
    }  
}
```

자바의 기초 문법

01. 자바 프로그램 작성의 기초

자바 프로그램의 구성요소

- 데이터: 문자열(string), 문자(character), 정수, 소수 등

```
class HelloJava {  
    public static void main(String args[]) {  
        System.out.println("Hello, Java");  
    }  
}
```

문자열

자바의 기초 문법

01. 자바 프로그램 작성의 기초 변수의 선언문

- 변수 : 데이터를 담는 일종의 그릇
- 변수를 사용하기 위해서는 먼저 선언을 해야 함
- 변수의 선언문(declaration statement)

`int num;` ← 세미콜론(;)은 명령문의 끝을 표시하는 문자

↑ ↑

정수(int) 타입의 값을 num이라는 이름의 변수를
담을 수 있는 선언하는 선언문

Data Type

● Java의 DataType

▶ primitive data type

- Java compiler의해서 해석되는 data type

▶ reference data type

- Java API에서 제공되거나 Programmer에 의해서 만들어진
class를 data type으로 Declaration하는 경우



Data Type

● Primitive Data Type의 종류

※ 기본 자료형의 종류

자료형	키워드	크기	기본값	표현 범위
논리형	boolean	1bit	false	true 또는 false(0과 1이 아니다)
문자형	char	2byte	\u0000	0 ~ 65,535
정수형	byte	1byte	0	-128 ~ 127
	short	2byte	0	-32,768 ~ 32,767
	int	4byte	0	-2,147,483,648 ~ 2,147,483,647
	long	8byte	0	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
실수형	float	4byte	0.0	-3.4E38 ~ +3.4E38
	double	8byte	0.0	-1.7E308 ~ +1.7E308

Data Type

● DataType Conversion(Casting)

※ 형 변환의 종류

종류	설 명	코딩 예
프로모션	<ul style="list-style-type: none">▪ 더 큰 자료형으로의 변환(자동)▪ 정보의 손실 없음	<pre>short a, b; a = b = 10; int c = a + b; // 형 변환</pre>
		<pre>short s = 10; float f = 10 + 3.5f; // 형 변환</pre>
디모션	<ul style="list-style-type: none">▪ 더 작은 자료형으로의 변환(명시)▪ 정보의 손실이 발생할 가능성 있음	<pre>short a, b, c; a = b = 10; c = (short)(a + b); // 형 변환</pre>
		<pre>int c = 0; short s = 10; c = (int)(10 + 3.5f); // 형 변환</pre>

Operator

- ➡ Operator란 Data의 가공을 위해 정해진 방식에 따라 계산하고 결과를 얻기 위한 행위를 의미하는 기호들의 총칭이다, 그리고 각 Operator들은 연산을 하기 위해 인식하는 DataType들이 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

Operator

● Arithmetic Operator

➡ 4칙 연산(+, -, *, /)과 나머지 값을 구하는 Operator(%)를 말한다.

※ 산술 연산자의 종류

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

Operator

● Substitution Operator

- ▶ 특정한 상수 값이나 변수 값 또는 Object를 변수에 전달하여 기억시킬 때 사용하는 Operator이다.

※ 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.

01. 자바 프로그램 작성의 기초

대입문

- 대입문(*assignment statement*) : 변수에 데이터를 담는 명령문
- 기호 =를 이용해서 만들 수 있음
- 대입문의 예

num = 10 + 20;

↑ ↑

num이라는 10 + 20의 계산 결과를
이름의 변수에 대입하는 대입문

Operator

● Comparison Operator

- ▶ 변수나 상수의 값을 비교할 때 쓰이는 Operator로서 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.

Operator

Logic Operator

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

※ 논리 연산자의 종류 (1)

구분	연산자	의미	설명
논리 연산자	&&	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.

※ 논리 연산자의 종류 (2)

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

Operator

● Bit Operator

- ▶ 피연산자 즉 연산의 대상이 되는 값들을 내부적으로 bit단위로 변경한 후 연산을 수행하는 Operator이다.

※ 비트 연산자의 종류

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

Operator

Shift Operator

- bit단위의 연산처리를 하며 자료의 가공을 위해 오른쪽 또는 왼쪽으로 이동하여 값에 대한 변화를 일으키는 Operator이다.

※ 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.

Operator

● Increase & Decrease Operator

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

※ 증감 연산자의 종류

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

Operator

● Condition Operator(삼항 연산자)

- ▶ 하나의 조건을 정의하여 만족 시에는 'true' 를 Return하고 만족하지 못할 시에는 'false' 을 Return하여 단순 비교에 의해 변화를 유도하는 Operator이다.

※ 조건 연산자의 종류

구분	연산자	의미	구성
조건 연산자	? :	제어문의 단일 비교문과 유사하다.	조건식 ? 참값 : 거짓값

자바의 기초 문법

01. 자바 프로그램 작성의 기초 두번째 프로그램

- 선언문과 대입문을 포함하는 프로그램

```
1    class SimpleAdder {  
2        public static void main(String args[]) {  
3            int num;  
4            num = 10 + 20;  
5            System.out.println(num);  
6        }  
7    }
```

자바의 기초 문법

01. 자바 프로그램 작성의 기초

두번째 프로그램

- 컴파일 방법 : 명령 프롬프트 창에서 다음 명령을 실행

```
javac SimpleAdder.java
```

↑
소스 코드 파일의 이름

- 실행 방법 : 명령 프롬프트 창에서 다음 명령을 실행

```
java SimpleAdder
```

↑
클래스의 이름

자바의 기초 문법

01. 자바 프로그램 작성의 기초

두번째 프로그램

- 실행 결과



```
E:\work\chap2\2-1>java SimpleAdder
30
E:\work\chap2\2-1>
```

자바의 기초 문법

01. 자바 프로그램 작성의 기초 조건문

- 조건문(conditional statement) : 조건에 따라 주어진 일을 하는 명령문
- if 키워드를 이용해서 만들 수 있음
- 조건문의 예

```
if (num > 10) —————  
    System.out.println("계산 결과가 10보다 큼니다.");
```

num 변수의
값이 10보다 크면
"계산 결과가
10보다 큼니다."라고
출력합니다.

01. 자바 프로그램 작성의 기초

세번째 프로그램

- 조건문을 포함하는 프로그램

```
1    class SimpleAdder2 {  
2        public static void main(String args[]) {  
3            int num;  
4            num = 10 + 20;  
5            if (num > 10)  
6                System.out.println("계산 결과가 10보다 큼니다.");  
7        }  
8    }
```

자바의 기초 문법

01. 자바 프로그램 작성의 기초 세번째 프로그램

- 실행 결과



```
명령 프롬프트
E:\work\chap2\2-1>java SimpleAdder2
계산 결과가 10보다 큼니다.
E:\work\chap2\2-1>
```

자바의 기초 문법

01. 자바 프로그램 작성의 기초

용어 설명

- 부명령문(substatement) : 다른 명령문 안에 포함된 명령문

```
if (num > 10)
```

```
    System.out.println("계산 결과가 10보다 큼니다.");
```

if 조건문

부명령문
(substatement)

자바의 기초 문법

01. 자바 프로그램 작성의 기초 반복문

- 반복문(iterative statement) : 주어진 일을 반복하는 명령문
- while, do, for 키워드를 이용해서 만들 수 있음
- 반복문의 예

```
while (num < 10) {  
    System.out.println("Hello, Java");  
    num = num + 1;  
}
```

num < 10 라는
조건을 만족하는 동안
이 부분을 반복 실행합니다.

01. 자바 프로그램 작성의 기초

네번째 프로그램

- 반복문을 포함하는 프로그램

```
1    class HelloJava10 {  
2        public static void main(String args[]) {  
3            int num = 0;  
4            while (num < 10) {  
5                System.out.println("Hello, Java");  
6                num = num + 1;  
7            }  
8        }  
9    }
```

자바의 기초 문법

01. 자바 프로그램 작성의 기초

네번째 프로그램

- 실행 결과



```
E:\work\chap2\2-1>java HelloJava10
Hello, Java
Hello, Java
Hello, Java
Hello, Java
Hello, Java
Hello, Java
Hello, Java
Hello, Java
Hello, Java
Hello, Java
Hello, Java
Hello, Java

E:\work\chap2\2-1>
```

자바의 기초 문법

01. 자바 프로그램 작성의 기초 용어 설명

- 블록(block) : 명령문들을 중괄호로 둘러싼 것

```
while (num < 10) {  
    System.out.println("Hello, Java");  
    num = num + 1;  
}
```

블록(block)

while 문

자바의 기초 문법

01. 자바 프로그램 작성의 기초

주석

- 주석(comment) : 프로그램의 실행에 영향을 미치지 않게 만들어진 텍스트

```
1  /* 프로그램 이름 : HelloJava10
2     프로그램 설명 : Hello, Java를 10번 출력하는 프로그램
3     작성일 : 2006/2/11
4     작성자 : 이영애 */
5  class HelloJava10 {
6      public static void main(String args[]) {
7          int num = 0;           // num: 반복 회수를 카운트하는 변수
8          while (num < 10) {
9              System.out.println("Hello, Java");
10             num = num + 1;
11         }
12     }
13 }
```

/*부터 */까지가 주석

//부터 줄 끝까지가 주석

자바의 기초 문법

01. 자바 프로그램 작성의 기초 공백

- 공백문자 : 스페이스(SP), 탭(HT), 줄바꿈 문자(CR, LF), 새페이지 문자(FF)
- 공백을 제거한 Hello, Java 프로그램

```
class HelloJava {  
    public static void main(String args[]) {  
        System.out.println("Hello, Java");  
    }  
}
```



```
class HelloJava{public static void main(String args[]){System.out.println("Hello,  
Java");}}
```

02. 로컬 변수의 선언과 이용

로컬 변수

- 로컬 변수(local variable) : 메소드 안에 선언한 변수

```
class SimpleAdder {  
    int var = 3; —————  
    public static void main(String args[]) {  
        int num; —————  
        num = 10 + 20;  
        System.out.println(num);  
    }  
}
```

로컬 변수가 아님

로컬 변수

자바의 기초 문법

02. 로컬 변수의 선언과 이용

로컬 변수의 선언문

• 기본 형식(1)

타입 식별자;
↑ ↑
변수의 타입 변수의 이름

[예]

```
int       num;  
float     f1;  
String    str;
```

• 기본 형식(2)

타입 식별자 = 초기값;
↑ ↑ ↑
변수의 타입 변수의 이름 변수의 초기값을 계산하는 식

[예]

```
int num1 = 5;  
int num2 = num1 + 10;  
String str = "Hello, Java";
```

02. 로컬 변수의 선언과 이용

로컬 변수의 선언문

[여러 변수를 한꺼번에 선언한 예]

```
short s1, s2;  
int num1=10, num2=20;  
double pi=3.14, radius;
```


02. 로컬 변수의 선언과 이용

변수의 타입

타입 이름	설명
byte	정수
short	정수
int	정수
long	정수
float	소수
double	소수
char	문자 하나
boolean	참 또는 거짓
String	문자열
그밖의 타입들	* 나중에 배울 것임

프리미티브 타입
(기본형 데이터 타입)

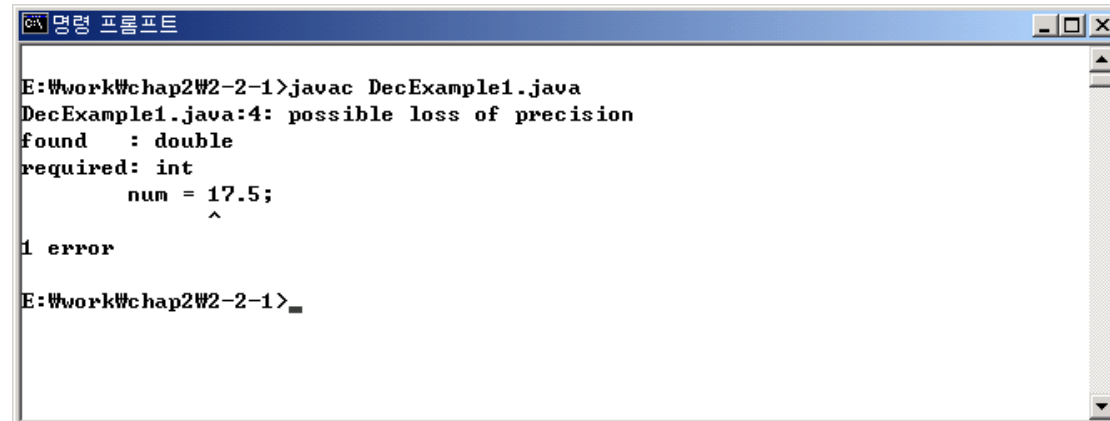
자바의 기초 문법

02. 로컬 변수의 선언과 이용 변수의 타입

[예제 2-7] 변수에 타입과 맞지 않는 값을 대입한 잘못된 예

```
1  class DecExample1 {  
2      public static void main(String args[]) {  
3          int num;  
4          num = 17.5;  
5          System.out.println(num);  
6      }  
7  }
```

int 타입의 변수에 소수를 대입하는 잘못된 명령문입니다.



```
C:\>명령 프롬프트  
E:\work\chap2\2-1>javac DecExample1.java  
DecExample1.java:4: possible loss of precision  
found   : double  
required: int  
    num = 17.5;  
        ^  
1 error  
E:\work\chap2\2-1>
```

02. 로컬 변수의 선언과 이용

식별자 명명 규칙

- 하나 이상의 글자로 이루어져야 한다.
- 첫 번째 글자는 문자이거나 '\$', '_'여야 한다.
- 두 번째 이후의 글자는 숫자, 문자, '\$', '_'여야 한다.
- '\$', '_' 외의 특수 문자는 사용할 수 없다.
- 길이의 제한은 없다.
- 키워드는 식별자로 사용할 수 없다.
- 상수 값을 표현하는 단어 **true**, **false**, **null**은 식별자로 사용할 수 없다.

[예]

br1	title	\$date
_data	dataTable	actionPerformed
MAX_NUM	CartViewer	i77

자바의 기초 문법

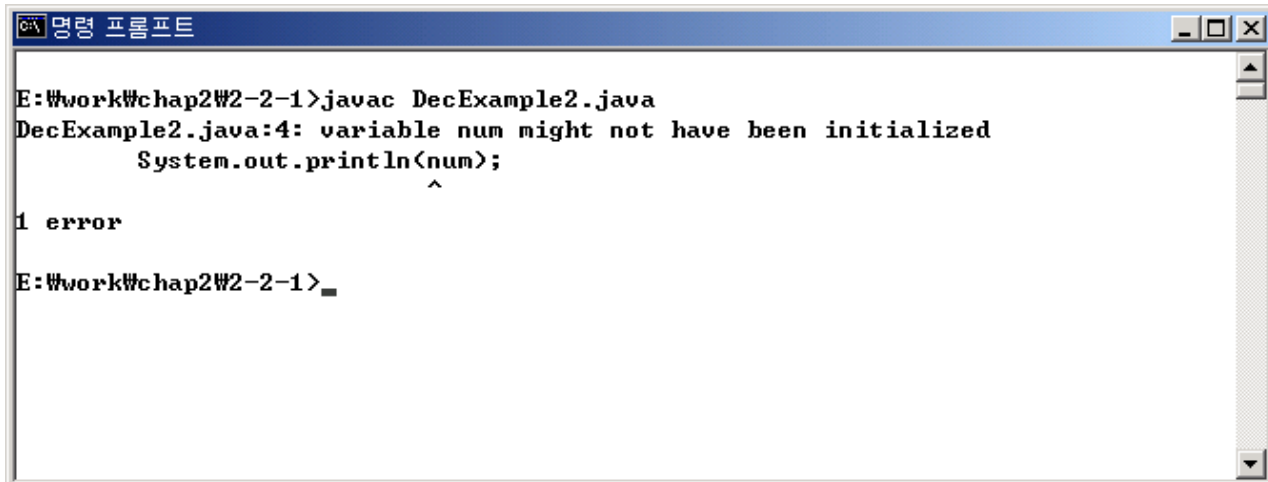
02. 로컬 변수의 선언과 이용

로컬 변수의 사용 방법

[예제 2-8] 변수에 아무런 값도 넣지 않고 사용하려고 한 잘못된 예

```
1  class DecExample2 {  
2      public static void main(String args[]) {  
3          int num;  
4          System.out.println(num);  
5      }  
6  }
```

변수에 아무 값도 넣지 않고 사용하는 것은 잘못입니다.



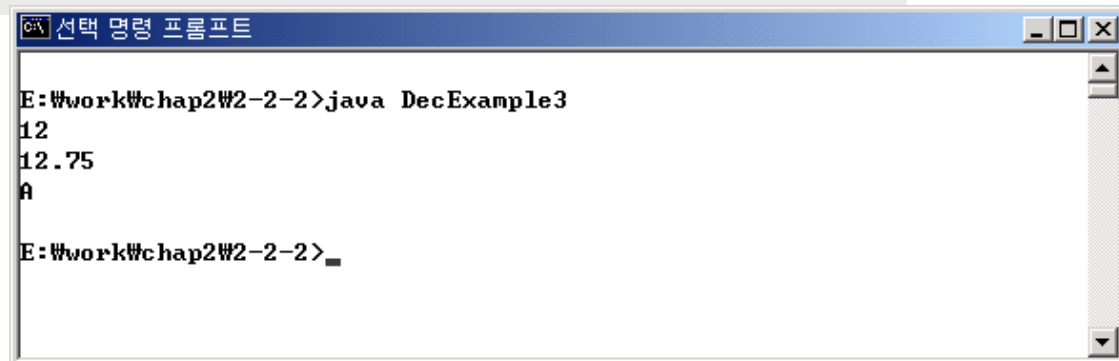
```
E:\work\chap2\2-1>javac DecExample2.java  
DecExample2.java:4: variable num might not have been initialized  
    System.out.println(num);  
                        ^  
1 error  
E:\work\chap2\2-1>
```

자바의 기초 문법

02. 로컬 변수의 선언과 이용 로컬 변수의 사용 범위

[예제 2-9] 변수의 선언문이 메소드 중간에 나오는 예

```
1  class DecExample3 {  
2      public static void main(String args[]) {  
3          short num1 = 12;           // num1 변수의 선언문  
4          System.out.println(num1);  
5          double num2 = 12.75;       // num2 변수의 선언문  
6          System.out.println(num2);  
7          char ch = 'A';             // ch 변수의 선언문  
8          System.out.println(ch);  
9      }  
10 }
```



```
선택 명령 프롬프트  
E:\work\chap2\2-2>java DecExample3  
12  
12.75  
A  
E:\work\chap2\2-2>
```

자바의 기초 문법

02. 로컬 변수의 선언과 이용

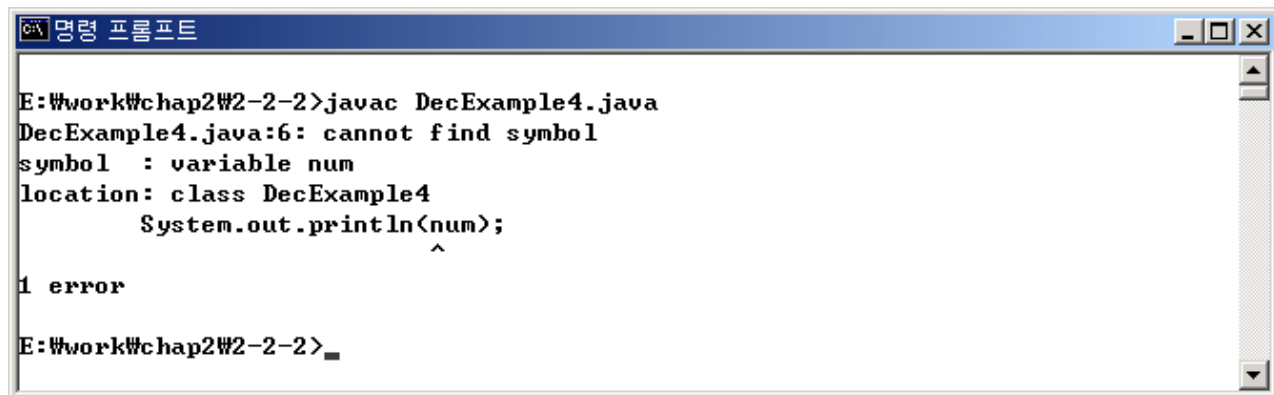
로컬 변수의 사용 범위

[예제 2-10] 블록 안에서 선언된 변수를 잘못 사용한 예

```
1    class DecExample4 {  
2        public static void main(String args[]) {  
3            {  
4                int num = 10;  
5            }  
6            System.out.println(num);  
7        }  
8    }
```

블록 안에 선언된 변수를 했습니다.

블록 안에 선언된 변수를 블록 밖에서
사용하려고 했으므로 잘못된 명령문입니다.



```
명령 프롬프트  
E:\work\chap2\2-2>javac DecExample4.java  
DecExample4.java:6: cannot find symbol  
symbol   : variable num  
location : class DecExample4  
            System.out.println(num);  
                        ^  
1 error  
E:\work\chap2\2-2>
```

자바의 기초 문법


02. 로컬 변수의 선언과 이용

로컬 변수의 사용 범위

[예제 2-11] 블록 밖에서 선언된 변수를 블록 안에서 사용하는 예

```
1  class DecExample5 {  
2      public static void main(String args[]) {  
3          int num = 10;  
4          {  
5              num = 30;  
6          }  
7          System.out.println(num);  
8      }  
9  }
```

----- 블록 밖에 선언된 변수를 블록 안에서 사용하는 것은 가능합니다.



```
명령 프롬프트  
E:\work\chap2\2-2>java DecExample5  
30  
E:\work\chap2\2-2>
```

02. 로컬 변수의 선언과 이용

final 변수

- final 변수 : 변수에 값을 딱 한번만 대입할 수 있는 변수

[예제 2-12] final 변수의 사용 예

```
1  class FinalExample1 {  
2      public static void main(String args[]) {  
3          final double pi = 3.14;          ----- final 변수를 선언합니다  
4          double radius = 2.0, circum;  
5          circum = 2 * pi * radius;  
6          System.out.println(circum);----- final 변수를 사용합니다  
7      }  
8  }
```



```
E:\work\chap2\2-2-3>java FinalExample1  
12.56  
  
E:\work\chap2\2-2-3>
```

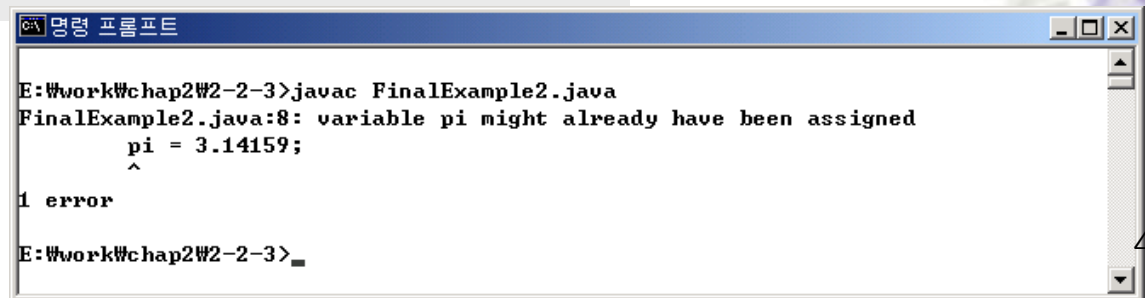

자바의 기초 문법

02. 로컬 변수의 선언과 이용

final 변수

[예제 2-13] final 변수의 잘못된 사용 예

```
1  class FinalExample2 {  
2      public static void main(String args[]) {  
3          final double pi;  
4          double radius = 2.0; ----- final 변수를 선언합니다.  
5          pi = 3.14;  
6          double circum = 2 * pi * radius; ----- final 변수에 초기값을 대입합니다.  
7          System.out.println(circum);  
8          pi = 3.14159;  
9          double area = pi * radius * radius;  
10         System.out.println(area); ----- final 변수에 또 다른 값을 대입합니다. (잘못된 명령문)  
11     }  
12 }
```



```
명령 프롬프트  
E:\work\chap2\2-3>javac FinalExample2.java  
FinalExample2.java:8: variable pi might already have been assigned  
    pi = 3.14159;  
    ^  
1 error  
E:\work\chap2\2-3>
```

자바의 기초 문법

03. 여러가지 대입문 단순 대입문

- 기본 형식

변수 = 식;

↑ ↑
변수의 이름 변수에 대입할 값을
 계산하는 식

[예]

```
num1 = 1;  
num2 = 2 + 3;  
num3 = num1 + num2;
```

사칙 연산에 사용되는 연산자들

연산자	설명
+	덧셈
-	뺄셈
*	곱셈
/	나눗셈 몫
%	나눗셈 나머지

03. 여러가지 대입문

대입문이 처리되는 방식

- 우변의 식이 좌변의 변수에 대입되기 전에 모두 계산됨
- num 변수의 값이 2라고 가정하면;

num = num + 3;

↑ ↑
2 + 3가 계산되어 5가 산출되고,
그 결과가 다시 num 변수에 대입됨

03. 여러가지 대입문

복합 대입 연산자

- += : 덧셈(+) 기능과 대입(=) 기능이 복합된 연산자

num += 3;



num = num + 3; 과 똑같은 일을 하는 명령문

- 사칙연산과 대입을 함께 수행하는 복합대입 연산자

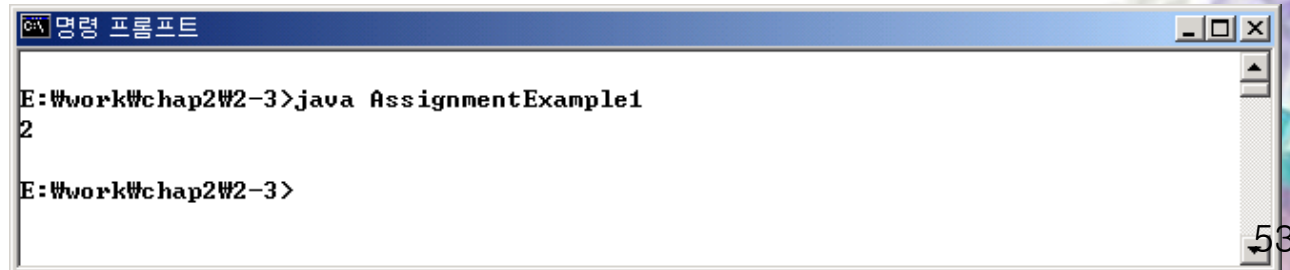
연산자	설명
+=	+와 = 기능의 복합
-=	-와 = 기능의 복합
*=	*와 = 기능의 복합
/=	/와 = 기능의 복합
%=	%와 = 기능의 복합

자바의 기초 문법

03. 여러가지 대입문 복합 대입 연산자

[예제 2-14] 복합 대입 연산자의 사용 예

```
1  class AssignmentExample1 {  
2      public static void main(String args[]) {  
3          int num = 17;  
4          num += 1;      // num = num + 1; 과 동일  
5          num -= 2;      // num = num - 2; 과 동일  
6          num *= 3;      // num = num * 3; 과 동일  
7          num /= 4;      // num = num / 4; 과 동일  
8          num %= 5;      // num = num % 5; 과 동일  
9          System.out.println(num);  
10     }  
11 }
```



```
C:\>명령 프롬프트

E:\work\chap2\2-3>java AssignmentExample1
2

E:\work\chap2\2-3>
```

03. 여러가지 대입문

증가 연산자와 감소 연산자

- 증가 연산자 ++ : 변수의 기존 값에 1을 더하는 연산자

num++;

↑
num = num + 1;와 동일

- 감소 연산자 -- : 변수의 기존 값에서 1을 빼는 연산자

num--;

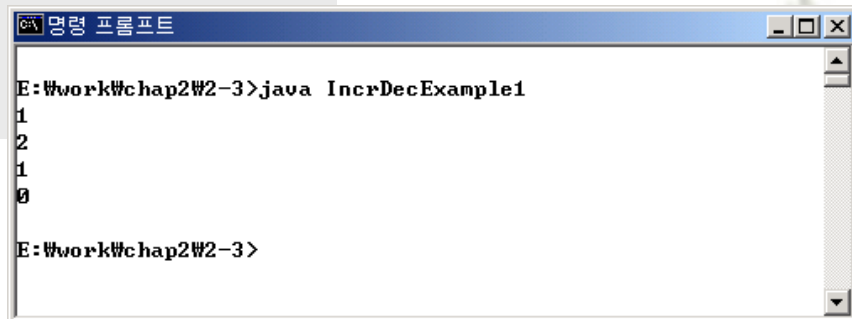
↑
num = num - 1;와 동일

자바의 기초 문법

03. 여러가지 대입문 증가 연산자와 감소 연산자

[예제 2-15] ++ 연산자와 -- 연산자의 사용 예

```
1  class IncrDecExample1 {  
2      public static void main(String args[]) {  
3          int num = 0;  
4          num++;  
5          System.out.println(num);  
6          ++num;  
7          System.out.println(num);  
8          num--;  
9          System.out.println(num);  
10         --num;  
11         System.out.println(num);  
12     }  
13 }
```



```
E:\work\chap2\2-3>java IncrDecExample1  
1  
2  
1  
0  
E:\work\chap2\2-3>
```

오늘 보다 내일이 더 기대되는~ 님들아!