

2장 자바의 기초 문법(1)

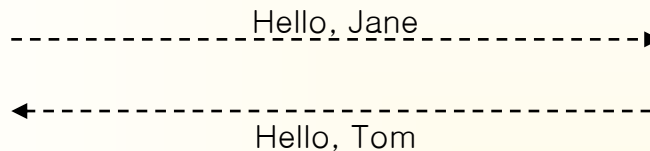
학습 목표

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

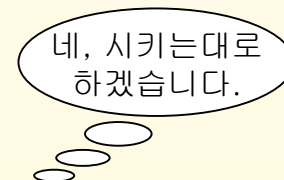
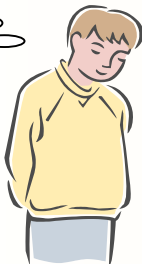
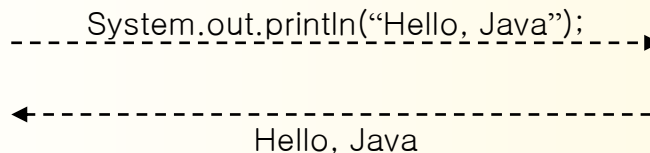
01. 자바 프로그램 작성의 기초

컴퓨터가 일을 처리하는 방법

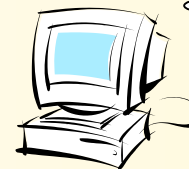
- 명령대로 수행하는 컴퓨터



Hello, Java라고
출력해라.



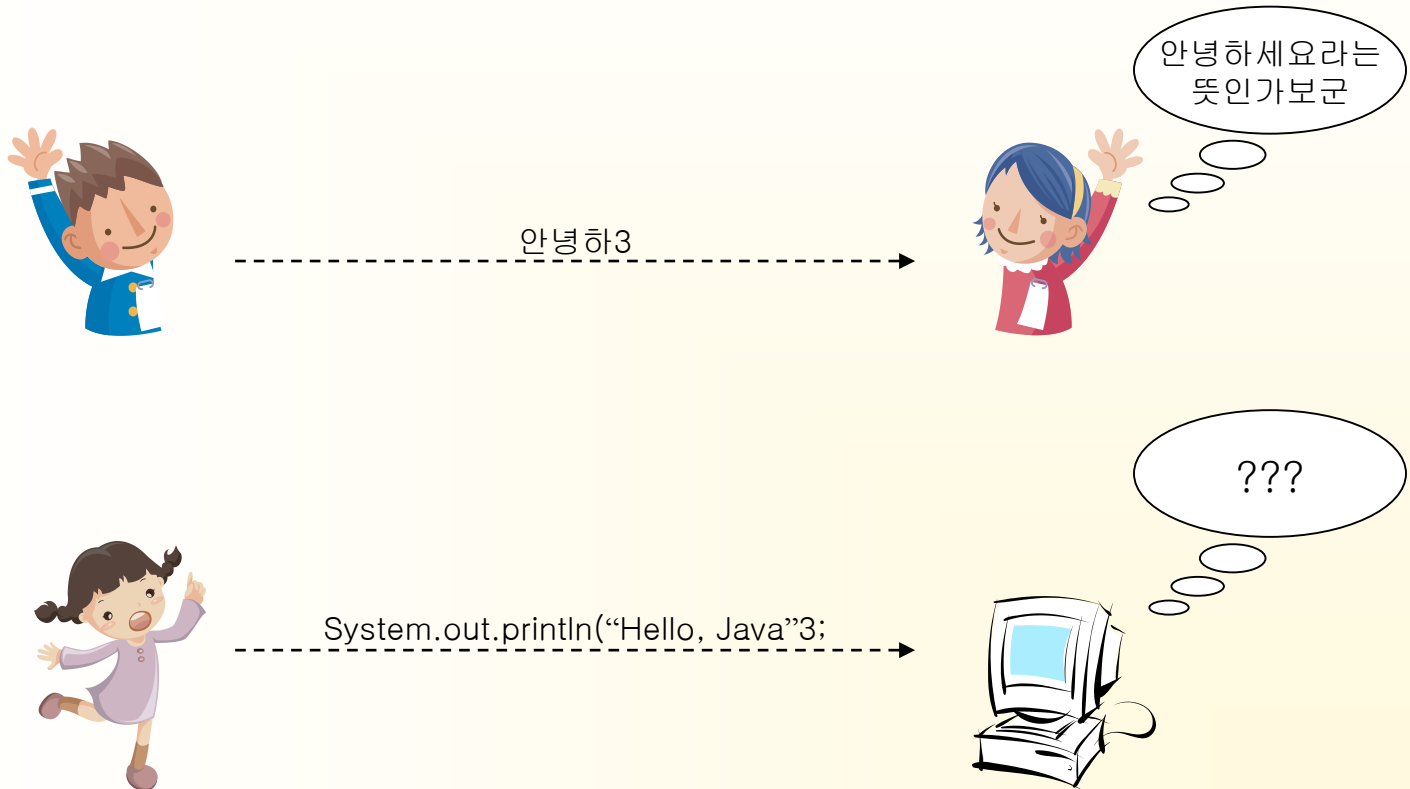
네, 시키는대로
하겠습니다.



01. 자바 프로그램 작성의 기초

컴퓨터가 일을 처리하는 방법

- 문법이 조금만 어긋나도 못 알아듣는 컴퓨터

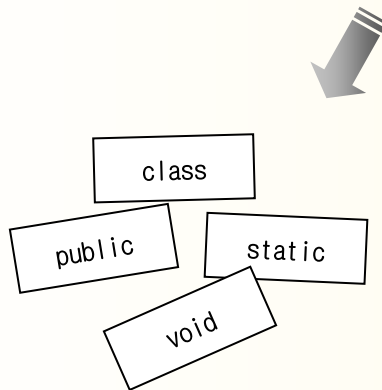


01. 자바 프로그램 작성의 기초

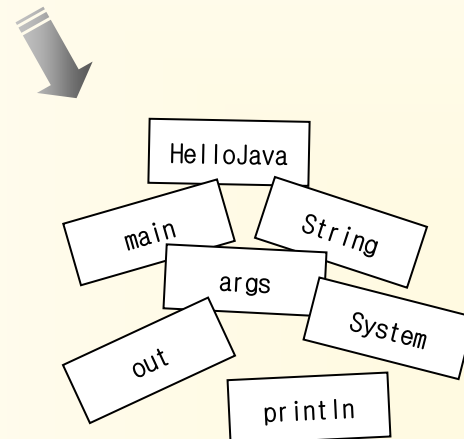
자바 프로그램의 구성요소

- 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)

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
------	-------	------

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");  
    }  
}
```


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



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

01. 자바 프로그램 작성의 기초

대입문

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

The diagram shows the code `num = 10 + 20;` with two horizontal lines underlining `num` and `10 + 20`. An arrow points from the text "num이라는 이름의 변수에" (to the variable named num) to the underlined `num`. Another arrow points from the text "10 + 20의 계산 결과를 대입하는 대입문" (assignment statement that assigns the calculation result of 10 + 20) to the underlined `10 + 20`.

```
num = 10 + 20;
```

num이라는 이름의 변수에

10 + 20의 계산 결과를 대입하는 대입문

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    }
```

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

01. 자바 프로그램 작성의 기초

두번째 프로그램

- 실행 결과



```
C:\> 명령 프롬프트

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

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

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 큽니다.");
```

num 변수의
값이 10보다 크면
"계산 결과가
10보다 큽니다."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    }
```

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>
```

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)

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    }
```

01. 자바 프로그램 작성의 기초

네번째 프로그램

- ## ● 실행 결과

A screenshot of a Windows Command Prompt window. The title bar at the top is blue and contains the text "명령 프롬프트" (Command Prompt) along with standard window control icons (minimize, maximize, close). The main area of the window has a white background and displays the following text:

```
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
```


The prompt "E:\work\chap2\2-1>" appears again at the bottom of the visible text. On the right side of the window, there are vertical scroll bars indicating the content can be scrolled up or down.

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

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

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

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

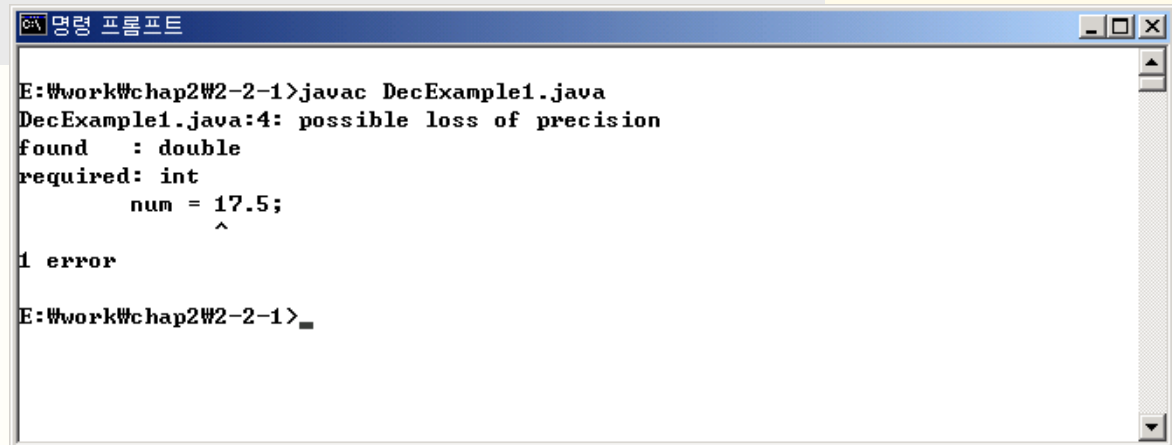
프리티미브 타입

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; ----- int 타입의 변수에 소수를 대입하는 잘못된 명령문입니다.  
5          System.out.println(num);  
6      }  
7  }
```



```
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

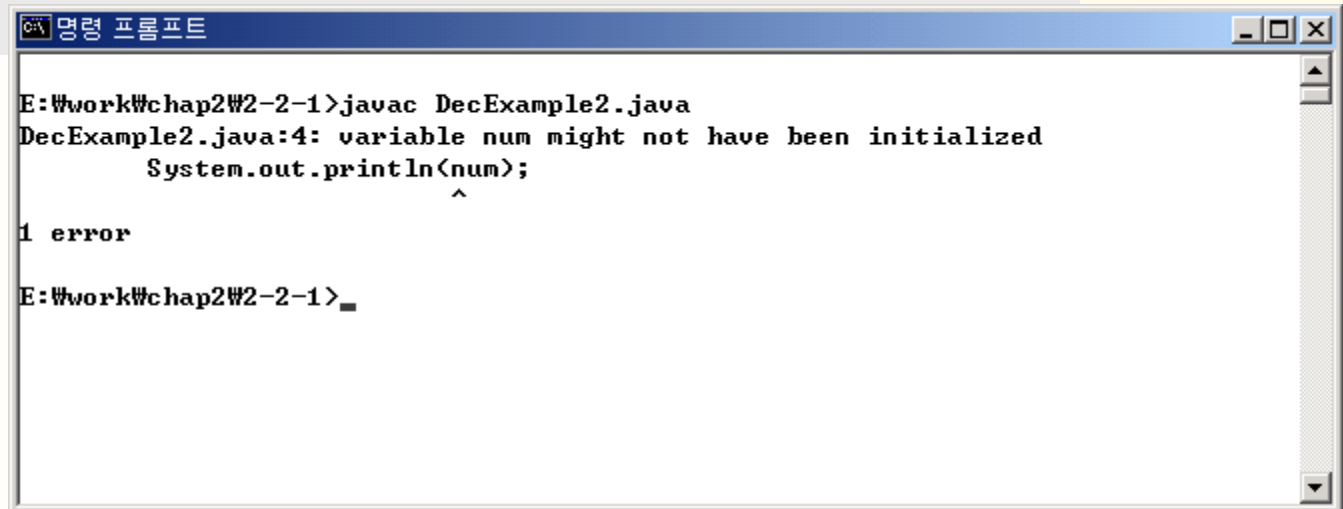
02. 로컬 변수의 선언과 이용

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.



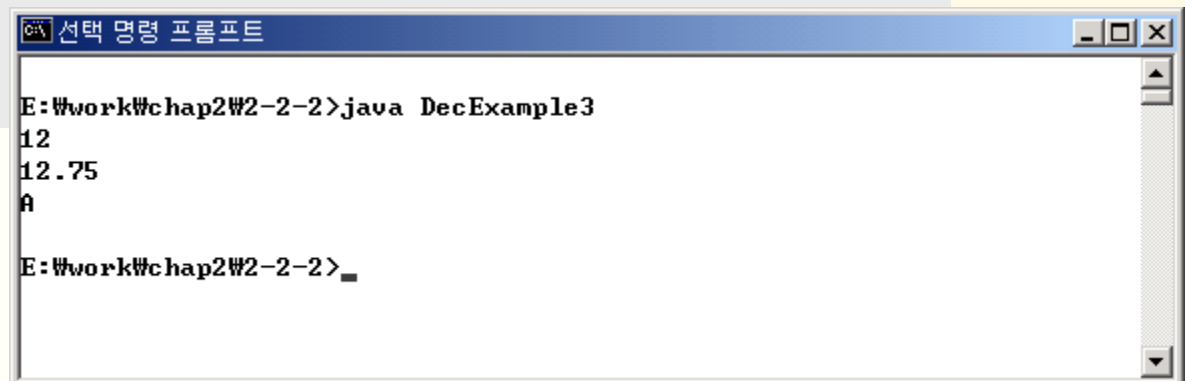
```
명령 프롬프트  
E:\work\chap2\2-2-1>javac DecExample2.java  
DecExample2.java:4: variable num might not have been initialized  
    System.out.println(num);  
                        ^  
1 error  
E:\work\chap2\2-2-1>
```

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>
```

02. 로컬 변수의 선언과 이용

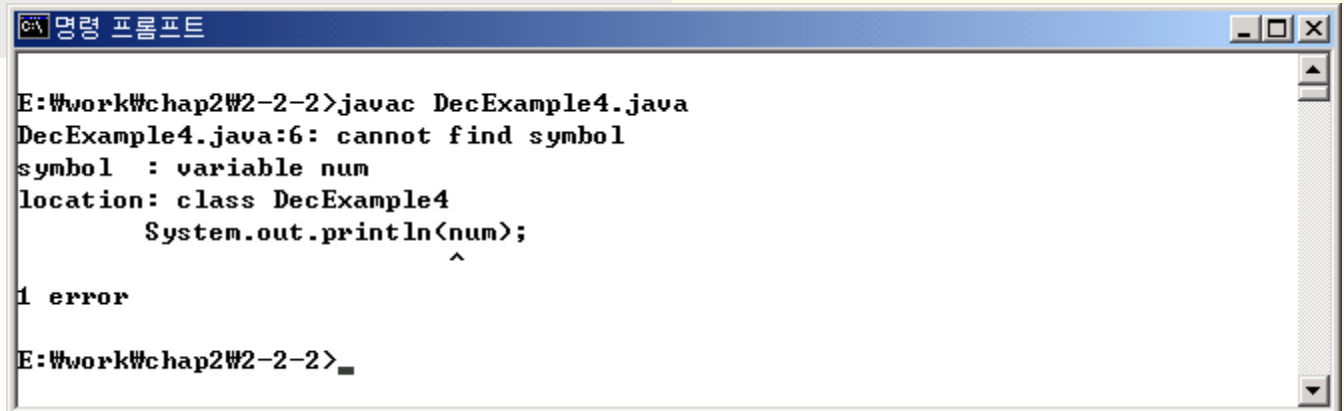
로컬 변수의 사용 범위

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


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


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; ----- final 변수를 사용합니다  
6          System.out.println(circum);  
7      }  
8  }
```



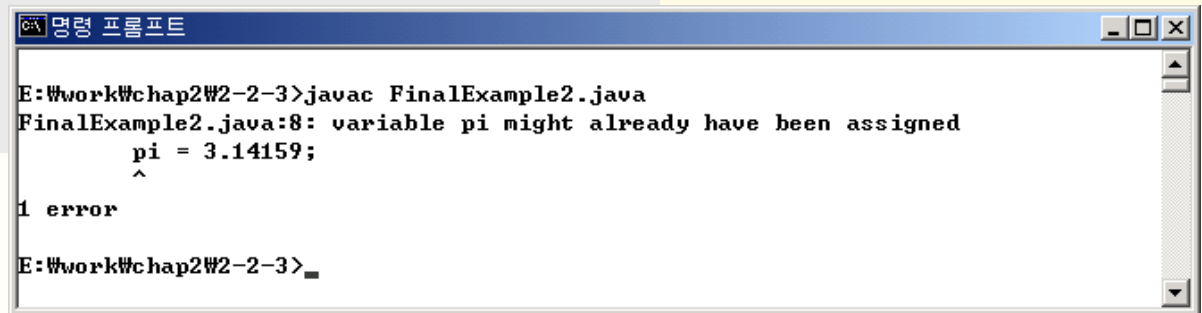
```
명령 프롬프트  
E:\work\chap2\2-2-3>java FinalExample1  
12.56  
E:\work\chap2\2-2-3>
```

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; ----- final 변수를 선언합니다.  
4          double radius = 2.0;  
5          pi = 3.14; ----- final 변수에 초기값을 대입합니다.  
6          double circum = 2 * pi * radius;  
7          System.out.println(circum);  
8          pi = 3.14159; ----- final 변수에 또 다른 값을 대입합니다. (잘못된 명령문)  
9          double area = pi * radius * radius;  
10         System.out.println(area);  
11     }  
12 }
```



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

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
<code>+=</code>	+와 = 기능의 복합
<code>-=</code>	-와 = 기능의 복합
<code>*=</code>	*와 = 기능의 복합
<code>/=</code>	/와 = 기능의 복합
<code>%=</code>	%와 = 기능의 복합

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 }
```



```
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--;`



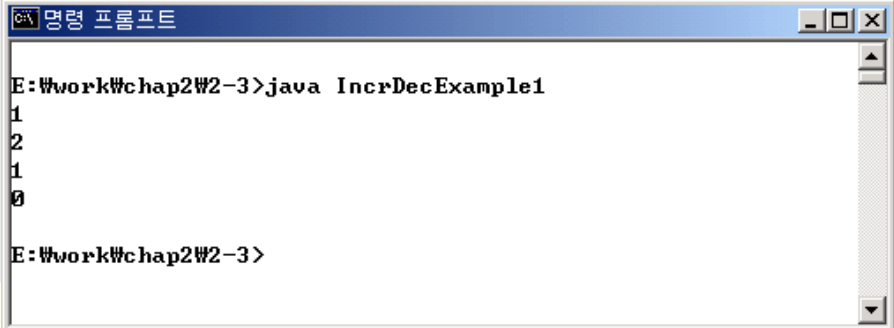
`num = num - 1;`와 동일

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 }
```



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


04. 배열의 선언, 생성, 이용

배열의 필요성

- 개별적인 변수와 배열

a) 10개의 다른 이름을 갖는 변수들

이름:	a	b	c	d	e	f	g	h	i	j
	7	9	100	32	5	8	6	72	27	81


b) 10개의 데이터를 저장할 수 있는 배열

이름:	num									
	7	9	100	32	5	8	6	72	27	81
인덱스:	0	1	2	3	4	5	6	7	8	9

04. 배열의 선언, 생성, 이용

배열의 선언 (1차원 배열)


- 배열 변수 선언문의 형식 (1)

타입 $\text{식별자}[];$

 배열 항목의 타입 배열 변수의 이름

[예]

```
int      arr[];
float    num[];
String   strArr[];
```

- 배열 변수 선언문의 형식 (2)

$\text{타입}[]$ $\text{식별자};$

 배열 항목의 타입 배열 변수의 이름

[예]

```
int[]    arr;
float[]  num;
String[] strArr;
```

04. 배열의 선언, 생성, 이용

배열의 생성 (1차원 배열)

- 배열은 선언뿐만 아니라 생성을 해야만 사용할 수 있음
- 배열 생성식의 형식

`new 타입[크기];`

배열 항목의 타입 배열 항목의 수

[예]

```
arr = new int[10];  
num = new float[5];  
strArr = new String[3];
```

04. 배열의 선언, 생성, 이용

배열의 이용 (1차원 배열)

- 배열 이름과 인덱스를 이용하면 배열 항목을 단일 변수처럼 사용 가능
- 배열 항목을 가리키는 식

배열이름[인덱스];

배열 변수의 이름 배열 항목의 위치

[예]

```
arr[0] = 12;  
num[3] = num[1] + num[2];  
System.out.println(strArr[2]);
```

04. 배열의 선언, 생성, 이용

배열의 선언, 생성, 이용

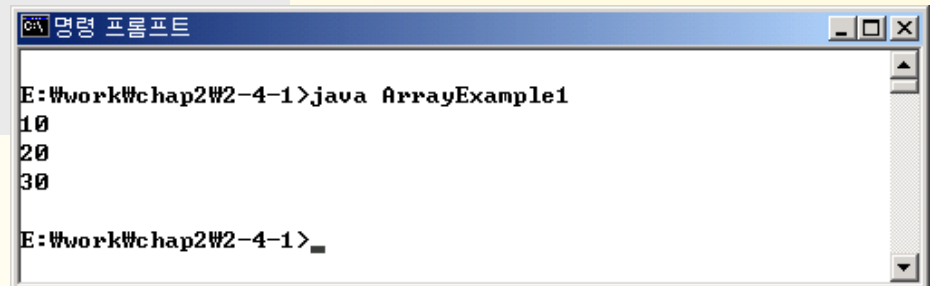
[예제 2-16] 1차원 배열의 사용 예

```
1  class ArrayExample1 {  
2      public static void main(String args[]) {  
3          int arr[]; -----  
4          arr = new int[10]; -----  
5          arr[0] = 10;  
6          arr[1] = 20;  
7          arr[2] = arr[0] + arr[1];  
8          System.out.println(arr[0]);  
9          System.out.println(arr[1]);  
10         System.out.println(arr[2]);  
11     }  
12 }
```

배열 변수를 선언합니다.

배열을 생성합니다.

배열 항목을 사용합니다.

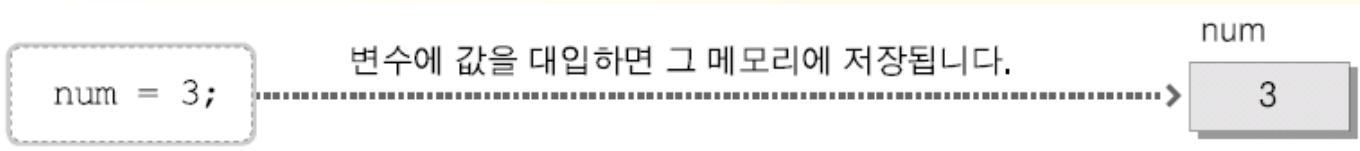
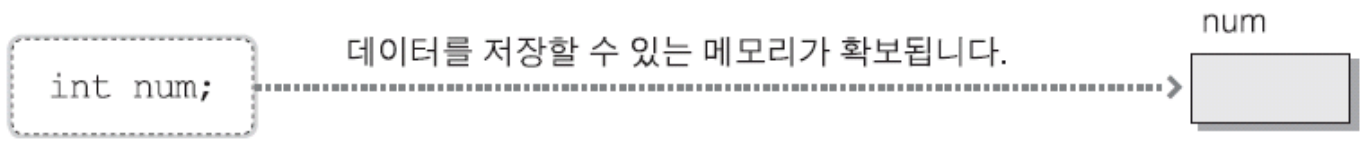


```
명령 프롬프트  
E:\work\chap2\2-4-1>java ArrayExample1  
10  
20  
30  
E:\work\chap2\2-4-1>
```

04. 배열의 선언, 생성, 이용

배열을 생성해야 하는 이유

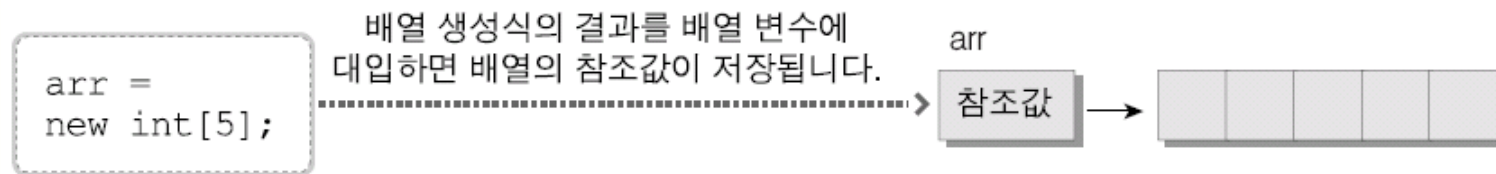
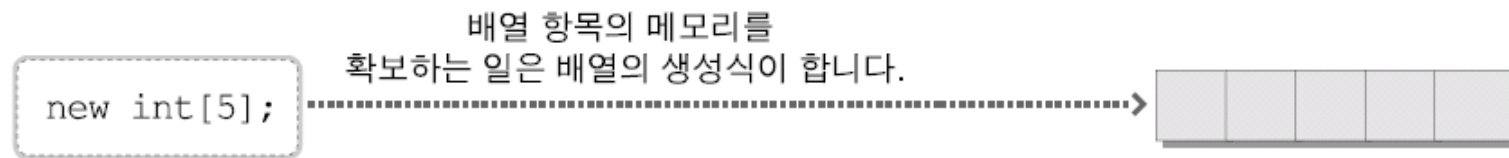
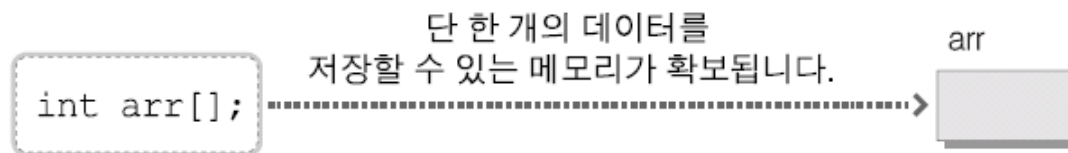
- 단일 변수의 메모리



04. 배열의 선언, 생성, 이용

배열을 생성해야 하는 이유

- 배열의 메모리



04. 배열의 선언, 생성, 이용

배열의 선언 (2차원 배열)

- 배열 변수 선언문의 형식 (1)

타입 식별자[] [] ;
 ↑ ↑
 배열 항목의 타입 배열 변수의 이름

[예]

```
int        arr[] [] ;
float     num[] [] ;
String    strArr[] [] ;
```

- 배열 변수 선언문의 형식 (2)

타입[] [] 식별자 ;
 ↑ ↑
 배열 항목의 타입 배열 변수의 이름

[예]

```
int[] []        arr ;
float[] []      num ;
String[] []    strArr ;
```


04. 배열의 선언, 생성, 이용

배열의 생성 (2차원 배열)

- 배열 생성식의 형식

`new 타입[크기1][크기2];`

배열 항목의 타입 행의 수 열의 수

[예]

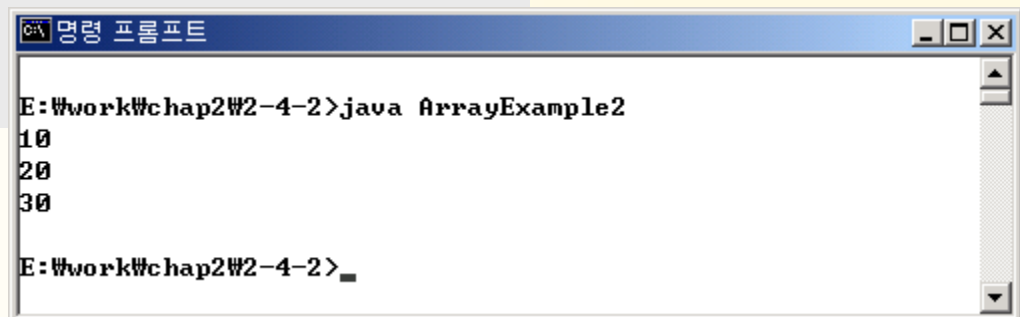
```
arr = new int[10][10];  
num = new float[5][2];  
strArr = new String[3][15];
```

04. 배열의 선언, 생성, 이용

배열의 선언, 생성, 이용

[예제 2-17] 2차원 배열의 사용 예

```
1  class ArrayExample2 {  
2      public static void main(String args[]) {  
3          int table[][] = new int[3][4];  
4          table[0][0] = 10;  
5          table[1][1] = 20;  
6          table[2][3] = table[0][0] + table[1][1];  
7          System.out.println(table[0][0]);  
8          System.out.println(table[1][1]);  
9          System.out.println(table[2][3]);  
10     }  
11 }
```



```
명령 프롬프트  
E:\work\chap2\2-4-2>java ArrayExample2  
10  
20  
30  
E:\work\chap2\2-4-2>
```

04. 배열의 선언, 생성, 이용

효율적인 코딩 방법

- 배열 선언과 초기화를 함께 하는 명령문

```
int arr[];  
arr = new int[10];  
arr[0] = 10;  
arr[1] = 20;  
arr[2] = 30;  
arr[3] = 40;  
arr[4] = 50;  
arr[5] = 60;  
arr[6] = 70;  
arr[7] = 80;  
arr[8] = 90;  
arr[9] = 100;
```



```
int arr[] = new int[10];  
arr[0] = 10;  
arr[1] = 20;  
arr[2] = 30;  
arr[3] = 40;  
arr[4] = 50;  
arr[5] = 60;  
arr[6] = 70;  
arr[7] = 80;  
arr[8] = 90;  
arr[9] = 100;
```



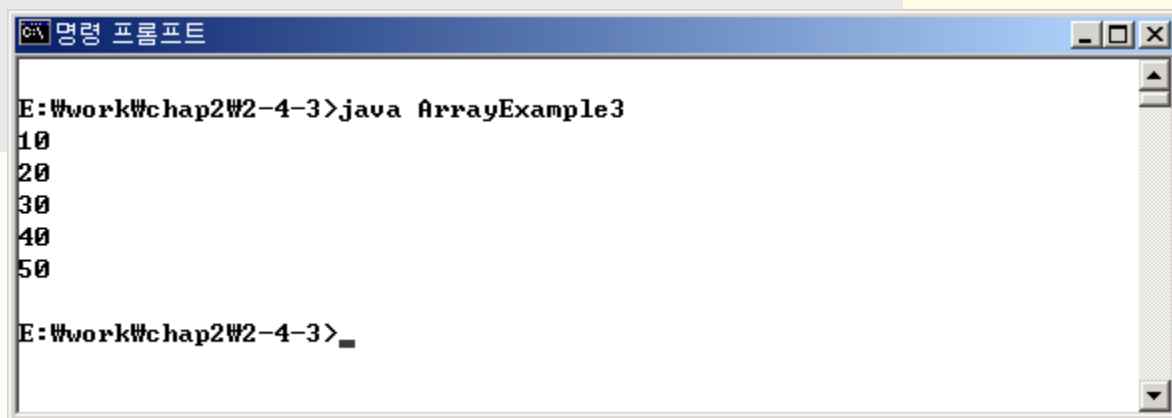
```
int arr[] = { 10, 20, 30, 40, 50,  
             60, 70, 80, 90, 100 };
```

04. 배열의 선언, 생성, 이용

효율적인 코딩 방법

[예제 2-18] 배열 항목을 초기화하는 배열 변수 선언문 (1차원 배열)

```
1  class ArrayExample3 {  
2      public static void main(String args[]) {  
3          int arr[] = { 10, 20, 30, 40, 50 };  
4          System.out.println(arr[0]);  
5          System.out.println(arr[1]);  
6          System.out.println(arr[2]);  
7          System.out.println(arr[3]);  
8          System.out.println(arr[4]);  
9      }  
10 }
```



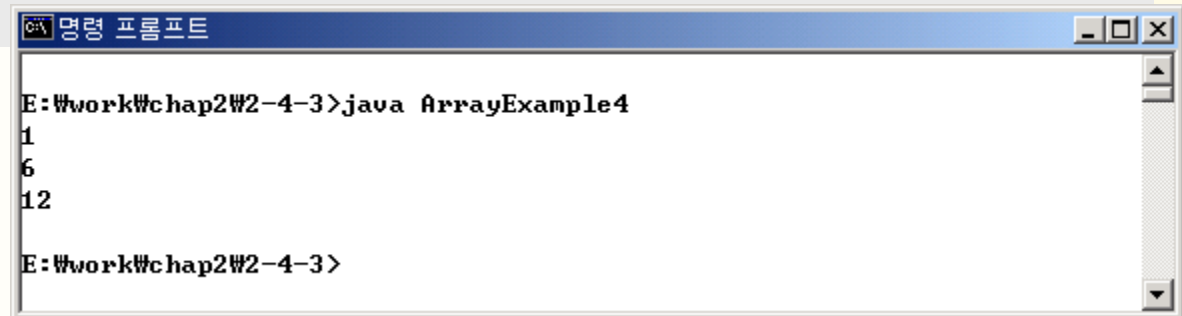
```
명령 프롬프트  
E:\work\chap2\2-4-3>java ArrayExample3  
10  
20  
30  
40  
50  
E:\work\chap2\2-4-3>
```

04. 배열의 선언, 생성, 이용

효율적인 코딩 방법

[예제 2-19] 배열 항목을 초기화하는 배열 변수 선언문 (2차원 배열)

```
1  class ArrayExample4 {  
2      public static void main(String args[]) {  
3          int table[][] = { { 1, 2, 3, 4 }, { 5, 6, 7, 8 }, { 9, 10, 11, 12 } };  
4          System.out.println(table[0][0]);  
5          System.out.println(table[1][1]);  
6          System.out.println(table[2][3]);  
7      }  
8  }
```



```
명령 프롬프트  
E:\work\chap2\2-4-3>java ArrayExample4  
1  
6  
12  
E:\work\chap2\2-4-3>
```

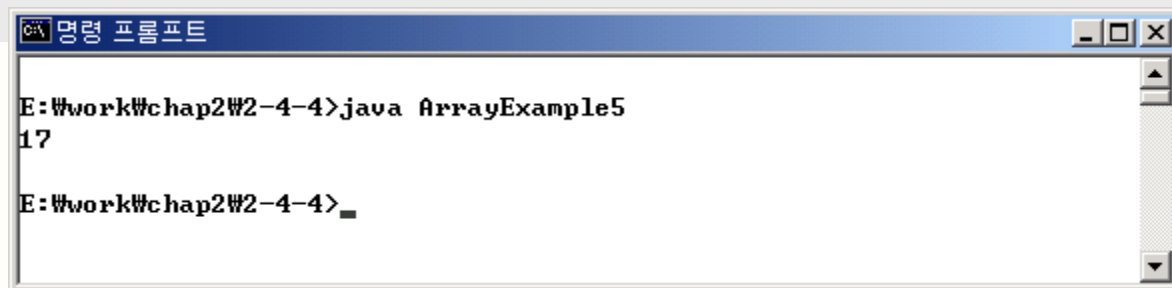
04. 배열의 선언, 생성, 이용

배열의 항목 수

- 배열의 항목수는 <배열이름>.length라는 식으로 알 수 있음

[예제 2-20] 배열의 항목 수를 출력하는 프로그램

```
1  class ArrayExample5 {  
2      public static void main(String args[]) {  
3          int arr[] = { 21, 2, 4, 3, 4, 65, 7, 178, 3, 5, 45, 78, 2, 0, 32, 75, 92 };  
4          System.out.println(arr.length);  
5      }  
6  }
```



```
명령 프롬프트  
E:\work\chap2\2-4-4>java ArrayExample5  
17  
E:\work\chap2\2-4-4>
```