

03. Hello, Java 프로그램

소스 코드 다시 살펴보기

클래스

```
1 class HelloJava {  
2     public static void main(String args[]) {  
3         System.out.println("Hello, Java");  
4     }  
5 }
```

메소드

03. Hello, Java 프로그램

소스 코드 다시 살펴보기

클래스 이름

```
1 class HelloJava {  
2     public static void main(String args[]) {  
3         System.out.println("Hello, Java");  
4     }  
5 }
```

클래스 본체
(class body)

03. Hello, Java 프로그램

소스 코드 다시 살펴보기

메소드 이름

```
1  class HelloJava {  
2      public static void main(String args[]) {  
3          System.out.println("Hello, Java");  
4      }  
5  }
```

메소드 본체
(method body)

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

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    }
```

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

05. 조건문

if 조건문

- if-else 조건문의 기본 형식

if (조건식)

실행부분 ————— 조건식이 true일 때만 실행되는 부분

else

실행부분 ————— 조건식이 false일 때만 실행되는 부분

[예]

```
if (num1 > num2)
    System.out.println("num1 = " + num1);
else
    System.out.println("num2 = " + num2);
```

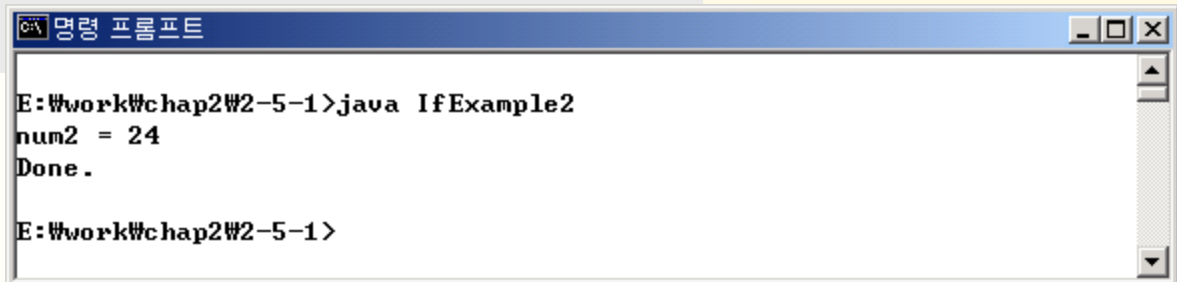
05. 조건문

if 조건문

[예제 2-22] if- else 문의 사용 예

```
1  class IfExample2 {  
2      public static void main(String args[]) {  
3          int num1 = 12;  
4          int num2 = 24;  
5          if (num1 > num2)  
6              System.out.println("num1 = " + num1);  
7          else  
8              System.out.println("num2 = " + num2);  
9          System.out.println("Done.");  
10     }  
11 }
```

num1과 num2 중 큰 값을 출력합니다.



```
명령 프롬프트  
E:\work\chap2\2-5-1>java IfExample2  
num2 = 24  
Done.  
E:\work\chap2\2-5-1>
```

05. 조건문

switch 조건문

- switch 조건문의 전형적인 형식

```
switch (식) {  
    case 값1: {  
        명령문들  
        break;  
    case 값2: {  
        명령문들  
        break;  
    case 값3: {  
        명령문들  
        break;  
    default : {  
        명령문들  
        break;  
    }  
}
```

정수나 char 타입의 값을 산출할 수 있는 식

1회 이상 여러 번 반복 가능한 부분

생략 가능한 부분

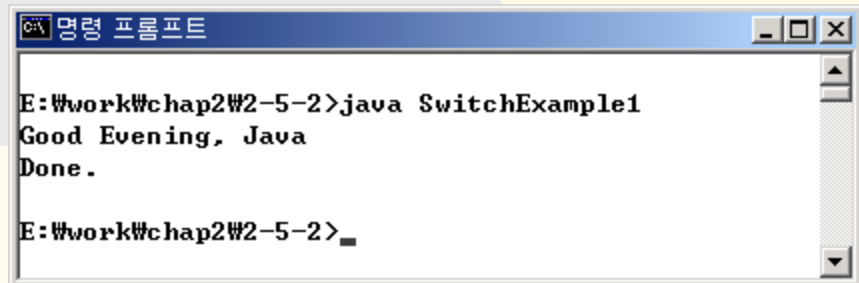
05. 조건문

switch 조건문

[예제 2-27] switch 문의 전형적인 사용 예

```
1  class SwitchExample1 {
2      public static void main(String args[]) {
3          int num = 3;
4          switch (num) {
5              case 1 :
6                  System.out.println("Good Morning, Java");
7                  break;
8              case 2 :
9                  System.out.println("Good Afternoon, Java");
10                 break;
11             case 3 :
12                 System.out.println("Good Evening, Java");
13                 break;
14             default :
15                 System.out.println("Hello, Java");
16                 break;
17         }
18         System.out.println("Done.");
19     }
20 }
```

num의 값에 따라 다른 메시지를 출력합니다.



```
E:\work\chap2\2-5-2>java SwitchExample1
Good Evening, Java
Done.

E:\work\chap2\2-5-2>
```

06. 반복문

while 반복문

- while 문의 기본 형식

```
while ( 조건식 )  
    실행부분
```

└────────── true 또는 false 값을 산출할 수 있는 식

────────── 조건식이 true일 동안 반복 실행되는 부분

06. 반복문

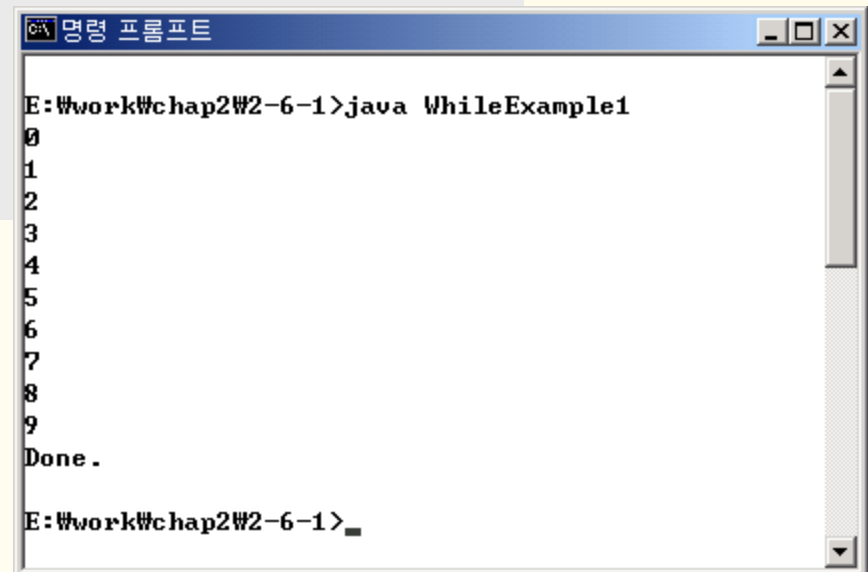
while 반복문

[예제 2-30] while 문의 사용 예

```
1  class WhileExample1 {  
2      public static void main(String args[]) {  
3          int cnt = 0;  
4          while (cnt < 10) {  
5              System.out.println(cnt);  
6              cnt++;  
7          }  
8          System.out.println("Done.");  
9      }  
10 }
```

cnt가 10보다 작을 동안

이 부분을 반복합니다.



```
명령 프롬프트  
E:\work\chap2\2-6-1>java WhileExample1  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
Done.  
E:\work\chap2\2-6-1>
```

06. 반복문

do-while 반복문

- do-while 문의 기본 형식

```
do
    실행부분
while ( 조건식 );
```

조건식이 true일 동안 반복 실행되는 부분

true 또는 false 값을 산출할 수 있는 식


- while 문과의 차이점
 - 조건식을 검사하기 전에 무조건 실행 부분을 한 번 실행
 - 마지막에 세미콜론(;)을 반드시 써야 함

06. 반복문

do-while 반복문

[예제 2-32] do-while 문의 사용 예

```
1  class DoWhileExample1 {  
2      public static void main(String args[]) {  
3          int cnt = 0;  
4          do {  
5              System.out.println(cnt);  
6              cnt++;  
7          } while (cnt < 10);  
8          System.out.println("done");  
9      }  
10 }
```

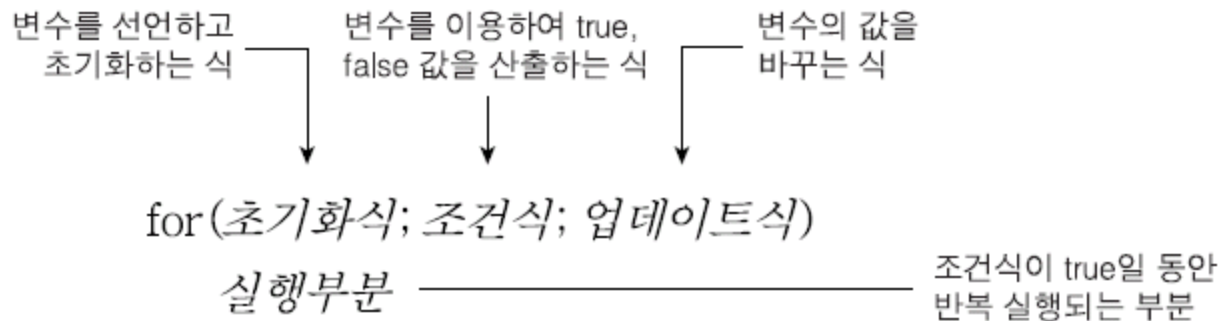


```
명령 프롬프트  
E:\work\chap2\2-6-2>java DoWhileExample1  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
Done.  
E:\work\chap2\2-6-2>
```

06. 반복문

for 반복문

- for 문의 기본 형식



[예]

```
for (int cnt = 0; cnt < 10; cnt++)  
    System.out.println(cnt);
```

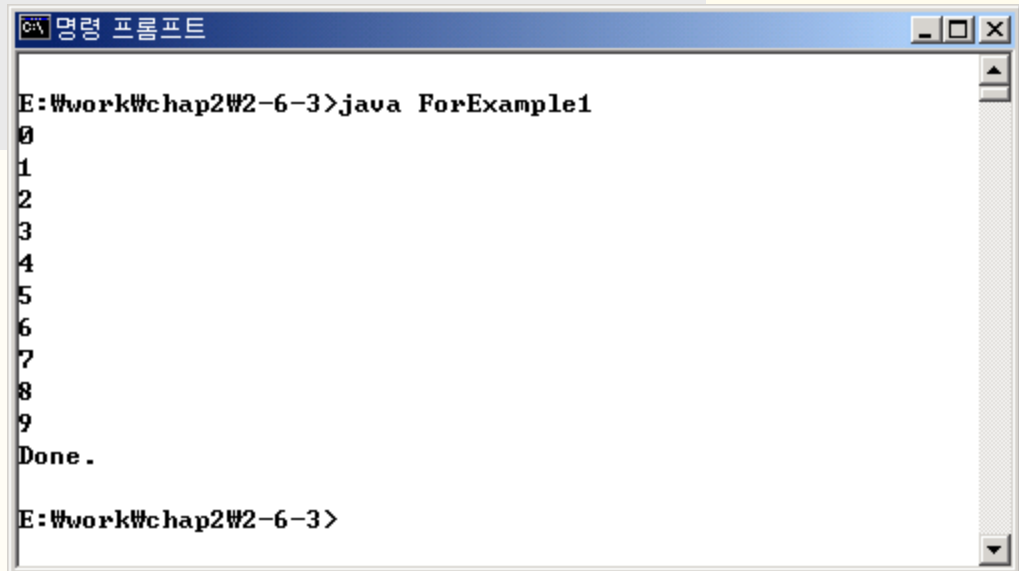
06. 반복문

for 반복문

[예제 2-33] for 문의 전형적인 사용 예

```
1  class ForExample1 {  
2      public static void main(String args[]) {  
3          for (int cnt = 0; cnt < 10; cnt++)  
4              System.out.println(cnt);  
5          System.out.println("Done.");  
6      }  
7  }
```

0부터 9까지의 정수를 순서대로 출력합니다.

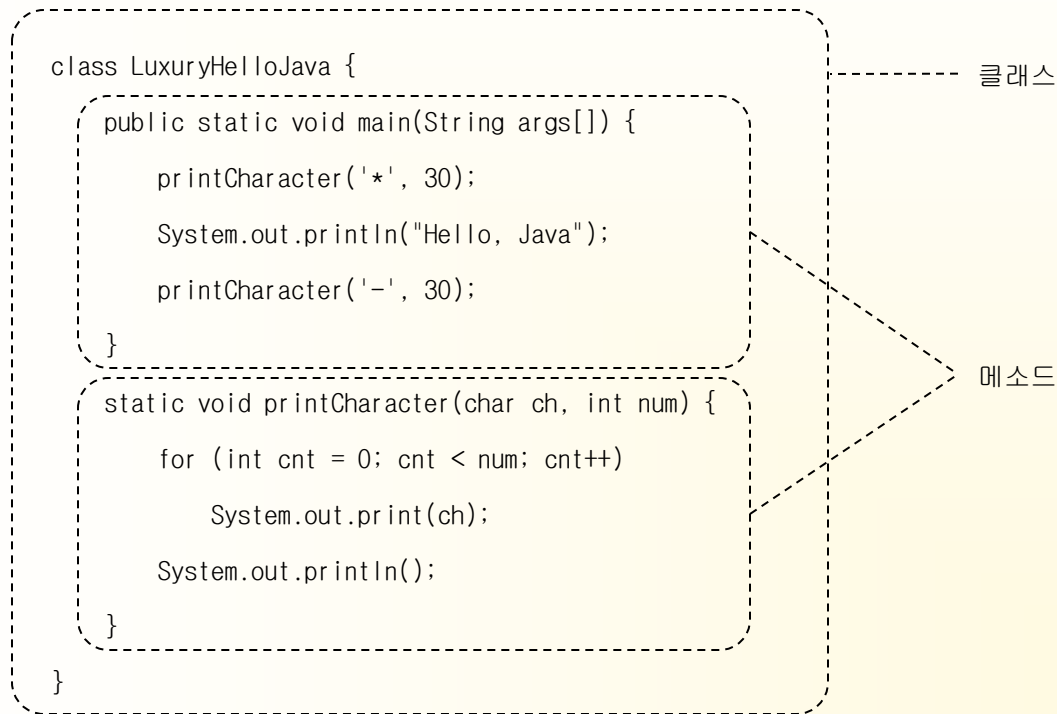


```
명령 프롬프트  
E:\work\chap2\2-6-3>java ForExample1  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
Done.  
E:\work\chap2\2-6-3>
```

07. 메소드 호출문

메소드 호출문

- 여러 개의 메소드가 포함된 클래스



- main이 아닌 메소드는 자동으로 실행되지 않음

07. 메소드 호출문

메소드 호출문

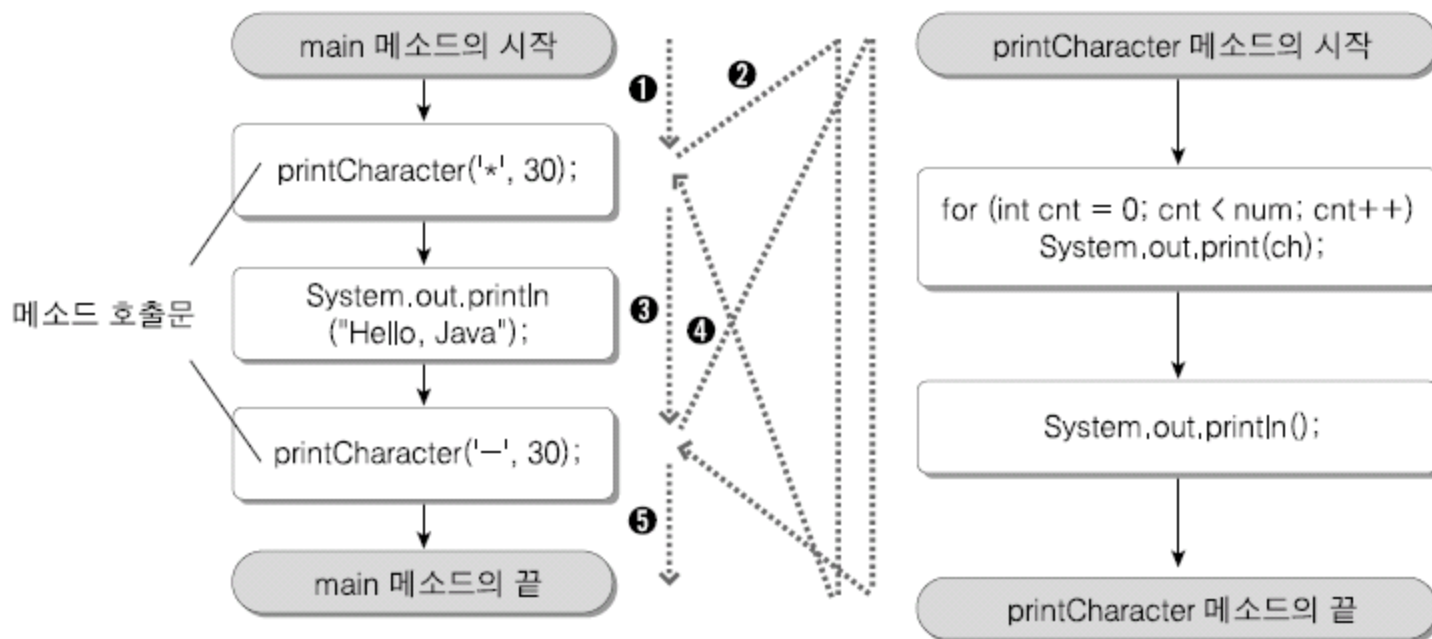
- 여러 개의 메소드가 포함된 클래스

```
class LuxuryHelloJava {  
    public static void main(String args[]) {  
        printCharacter('*', 30); ----- 메소드 호출문  
        System.out.println("Hello, Java");  
        printCharacter('-', 30); ----- 메소드 호출문  
    }  
    static void printCharacter(char ch, int num) {  
        for (int cnt = 0; cnt < num; cnt++)  
            System.out.print(ch);  
        System.out.println();  
    }  
}
```

- main이 아닌 메소드는 호출해야 실행됨

07. 메소드 호출문

프로그램의 실행 흐름



07. 메소드 호출문

파라미터(parameter)

```
printCharacter('*', 30);
```



파라미터

07. 메소드 호출문

파라미터 변수

```
class LuxuryHelloJava {  
    public static void main(String args[]) {  
        printCharacter('*', 30);  
        .  
        .  
    }  
    static void printCharacter(char ch, int num) {  
        .  
        .  
        .  
    }  
}
```

메서드 호출문에 있는 파라미터는
메서드의 파라미터 변수에 대입됩니다.

파라미터 변수

07. 메소드 호출문

메소드 호출문의 작성 방법

- 기본 형식

메소드이름(파라미터1, 파라미터2, 파라미터3);

↑
파라미터는 하나도 없을 수도 있고,
1개 이상 여러 개 있을 수도 있음

[예]

```
System.out.println("Hello, Java");  
printCharacter('A', 10);
```

07. 메소드 호출문

메소드 호출문의 작성 방법

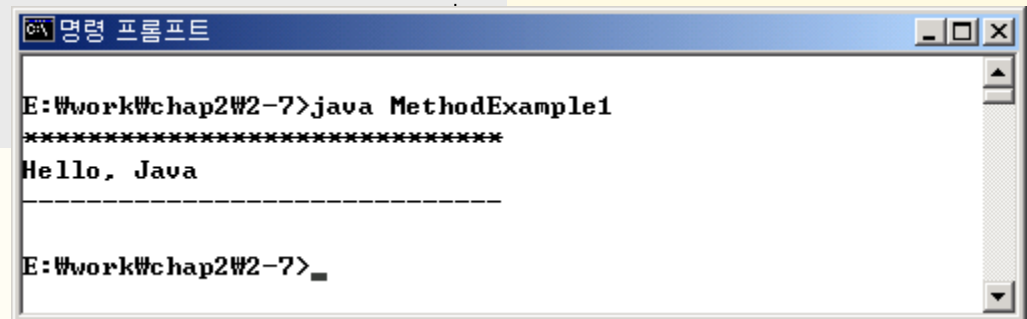
[예제 2-44] 메소드 호출 예

```
1  class MethodExample1 {  
2      public static void main(String args[]) {  
3          printCharacter('*', 30); -----  
4          System.out.println("Hello, Java");  
5          printCharacter('-', 30); -----  
6      }  
7      static void printCharacter(char ch, int num)  
8          for (int cnt = 0; cnt < num; cnt++)  
9              System.out.print(ch);  
10         System.out.println();  
11     }  
12 }
```

메소드 호출문

메소드 호출문

호출되는 메소드



```
C:\>명령 프롬프트  
E:\work\chap2\2-7>java MethodExample1  
*****  
Hello, Java  
-----  
E:\work\chap2\2-7>
```

07. 메소드 호출문

결과를 리턴하는 메소드

- 리턴 값(return value) : 메소드가 호출한 쪽으로 넘겨주는 메소드의 실행 결과
- 리턴 값을 리턴하는 메소드 호출문의 형식

변수 = 메소드이름(파라미터1, 파라미터2, 파라미터3);

↑
메소드의 리턴 값을
대입할 변수의 이름

↑
파라미터는 하나도 없을 수도 있고,
1개 이상 여러 개 있을 수도 있음

[예]

```
sum = add(1, 2);
```

07. 메소드 호출문

결과를 리턴하는 메소드

[예제 2-45] 리턴 값을 리턴하는 메소드의 호출 예

```
1  class MethodExample2 {  
2      public static void main(String args[]) {  
3          int result;  
4          result = add(3, 4);  
5          System.out.println(result);  
6      }  
7      static int add(int num1, int num2) {  
8          int sum;  
9          sum = num1 + num2;  
10         return sum;  
11     }  
12 }
```

리턴 값을 받는 메소드 호출문

호출되는 메소드



```
명령 프롬프트  
E:\work\chap2\2-7>java MethodExample2  
7  
E:\work\chap2\2-7>
```

07. 메소드 호출문

return 문

- 기본 형식 (1)

return 식;

↑
메서드의 리턴 값을
계산하는 식

[예]

```
return sum;  
return num1 + num2;
```

- 기본 형식 (1)

return;

[예]

```
return;
```

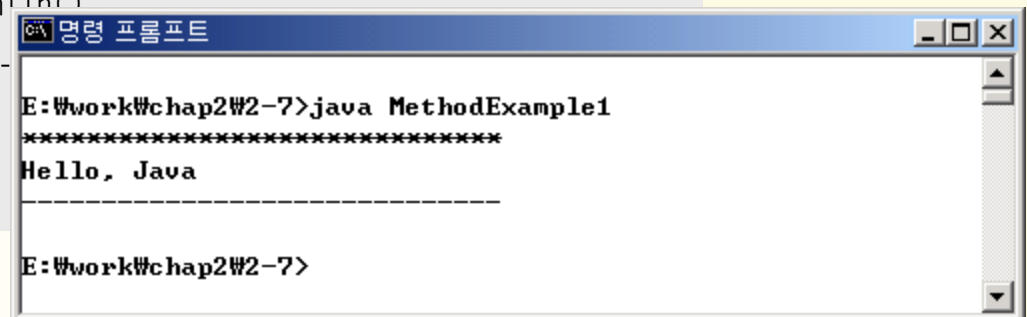
07. 메소드 호출문

return 문

[예제 2-46] 리턴 값이 없는 메소드 호출 예

```
1  class MethodExample1 {
2      public static void main(String args[]) {
3          printCharacter('*', 30);
4          System.out.println("Hello, Java");
5          printCharacter('-', 30);
6      }
7      static void printCharacter(char ch, int num) {
8          for (int cnt = 0; cnt < num; cnt++)
9              System.out.print(ch);
10         System.out.println();
11         return;
12     }
13 }
```

리턴 값이 없는 메소드임을 표시하는 키워드



```
명령 프롬프트
E:\work\chap2\2-7>java MethodExample1
*****
Hello, Java
-----
E:\work\chap2\2-7>
```


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

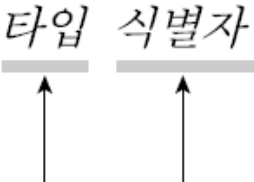
b) 10개의 데이터를 저장할 수 있는 배열

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)

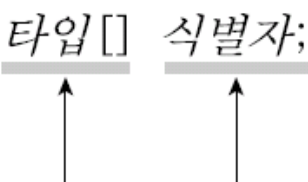
- 배열 변수 선언문의 형식 (1)

`타입 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 (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

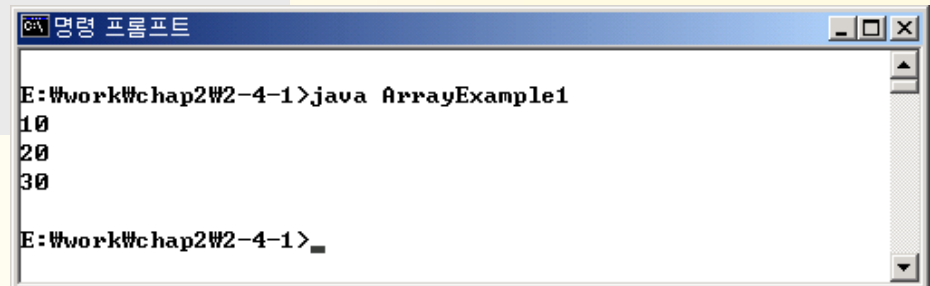
[예제 2-16] 1차원 배열의 사용 예

```
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.

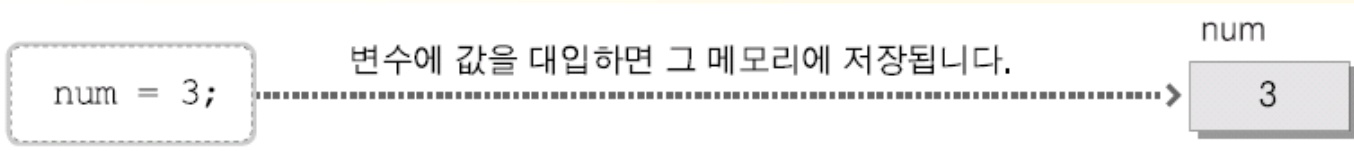
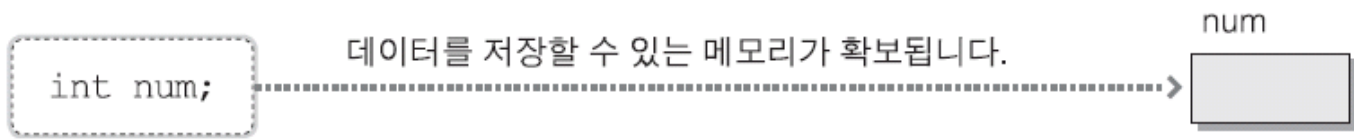


```
명령 프롬프트  
E:\work\chap2\2-4-1>java ArrayExample1  
10  
20  
30  
E:\work\chap2\2-4-1>
```

04. 배열의 선언, 생성, 이용

배열을 생성해야 하는 이유

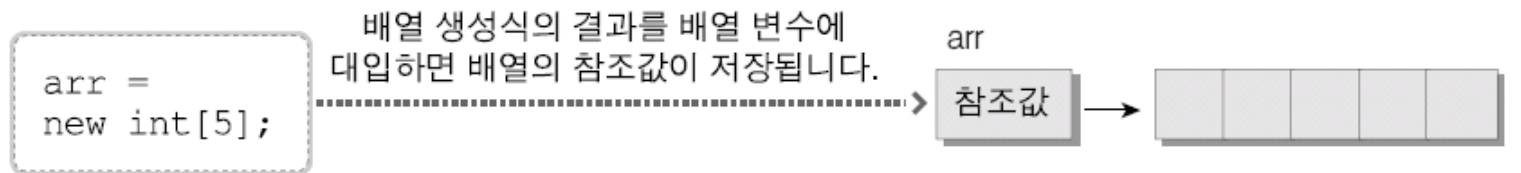
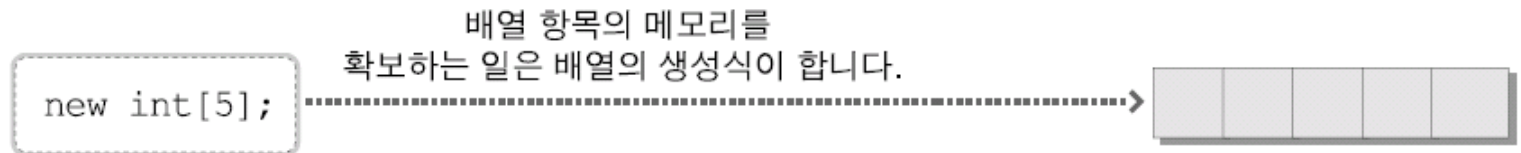
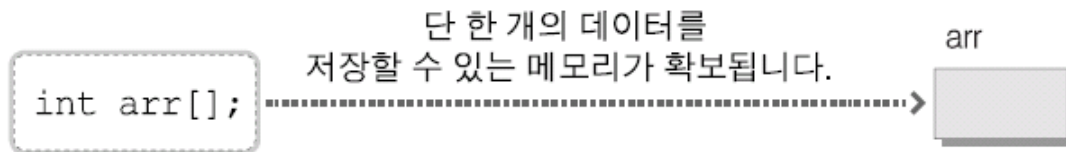
- 단일 변수의 메모리



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]

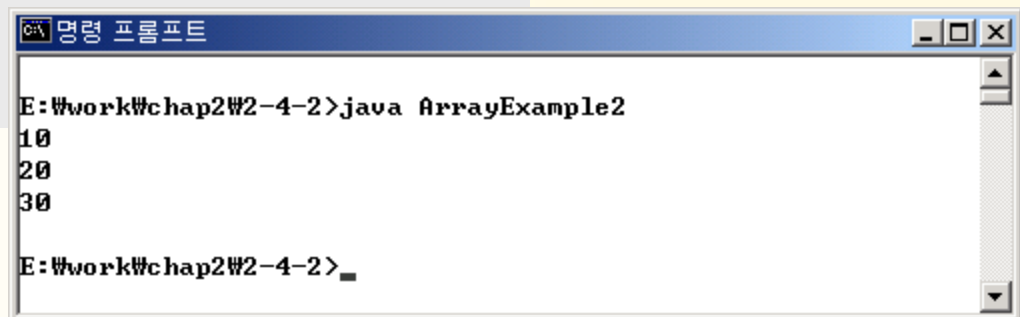
```
arr = new int[10][10];  
num = new float[5][2];  
strArr = new String[3][15];
```

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;
```



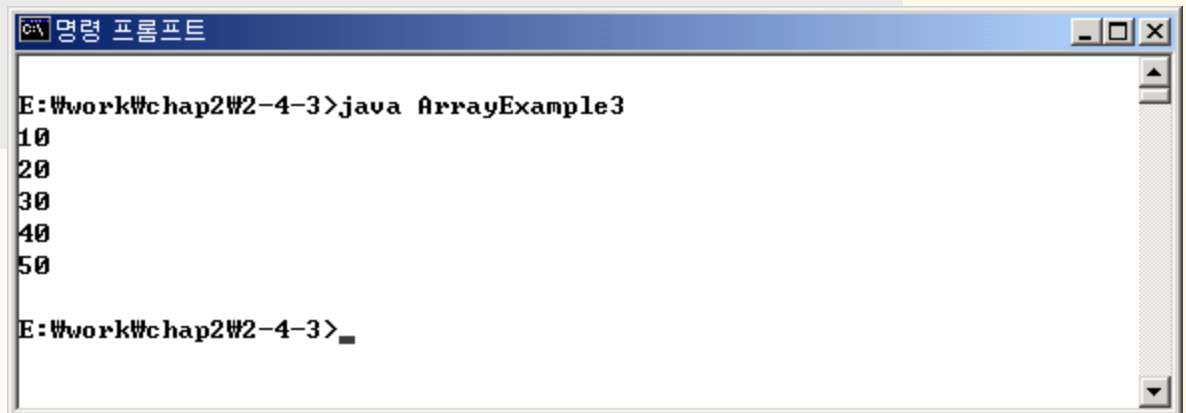
```
int arr[] = { 10, 20, 30, 40, 50,  
             60, 70, 80, 90, 100 };
```

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>
```

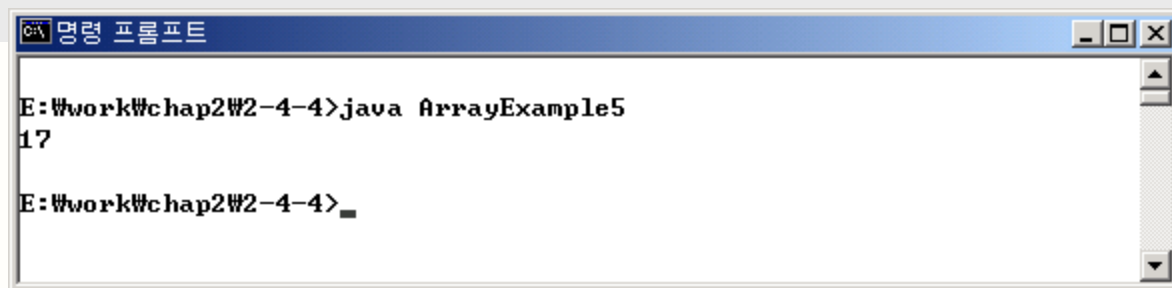
04. 배열의 선언, 생성, 이용

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>
```