

# Java



# 자바의 기초 문법(2)

조건문의 작성

반복문의 작성

메소드 호출문의 작성

익셉션을 처리하는 try 문



# Comment

- Comment는 실제 Program에 영향을 주지 않으며 단지 Source code의 기능이나 동작을 설명하기 위해 사용된다.

## ※ 주석문의 종류

주석 종류	의미	설명
// 주석문	단행 주석처리	현재 행에서 //의 뒷문장부터 주석으로 처리된다.
/* 주석문 */	다행 주석처리	/* 에서 */ 사이의 문장이 주석으로 처리된다.
/** 주석문 */	HTML 문서화 주석처리	/** 에서 */ 사이의 문장이 주석으로 처리된다. 장점은 HTML 문서화로 주석이 처리되므로 API와 같은 도움말 페이지를 만들 수 있다.

# 제어문

Program의 흐름에 영향을 주고 때에 따라 제어가 가능하도록 하는 것이 바로 '제어문' 이다.

## ❖ 제어문의 종류

### ■ 조건문

: 주어진 조건의 결과에 따라 실행 문장을 다르게 하여 전혀 다른 결과를 얻기 위해 사용되는 제어문이다,

- if문, switch문

### ■ 반복문

: 특정한 문장을 정해진 규칙에 따라 반복처리하기 위한 제어문이다,

- for문, while문, do~while문

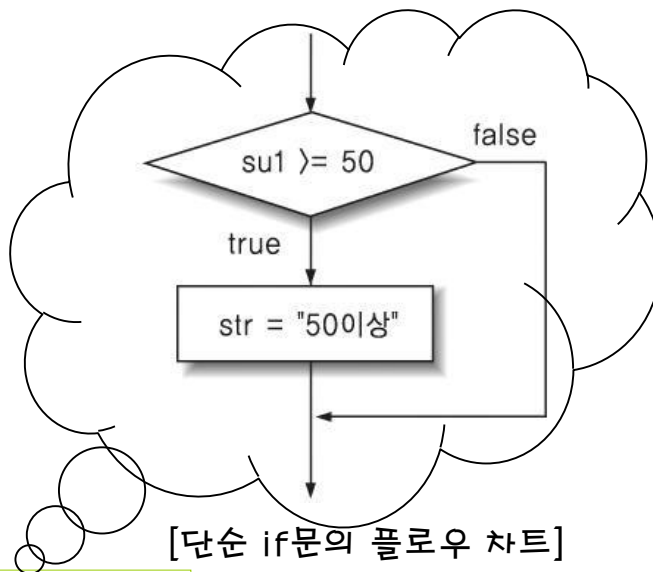
■ break문 : 반복문 내에서 쓰이며 반복문을 빠져나갈 때 쓰이는 제어문이다,

■ continue문 : 현재 진행되는 반복 회차를 포기하고 다음 회차로 이동 한다,

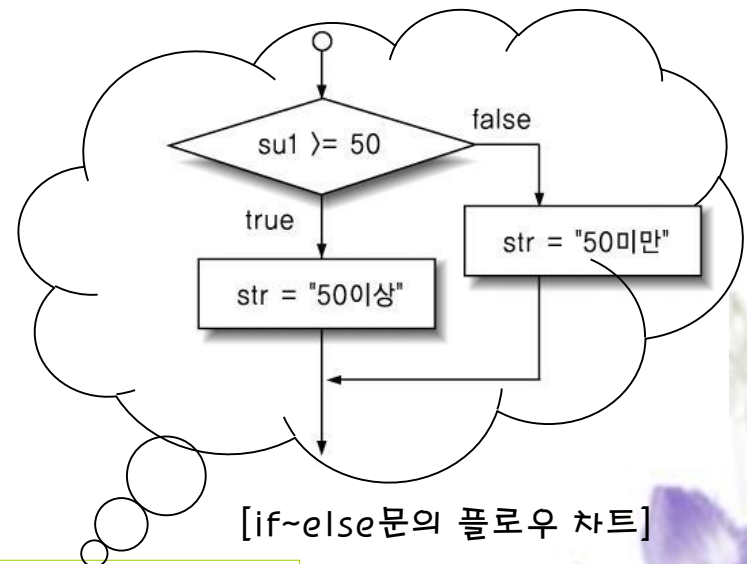
# 제어문

## if문

- boolean형의 결과를 가지는 조건식이 있어야 하며 그 조건식의 결과로 수행하는 문장을 결정하게 되는 조건문이다.



```
if(su1 >= 50)  
    str  
    = "50이상" ;
```



```
if(su1 >= 50)  
    str =  
    "50이상" ;  
else  
    str =  
    "50미만" ;
```

# 조건문

## ● if 조건문

### • if 조건문의 기본 형식 (1)

if ( 조건식 )  
    명령문

```
if (num1 > num2)  
    System.out.println("num1 값이 더 큼니다.");
```

### • if 조건문의 기본 형식 (2)

if ( 조건식 )  
    블록

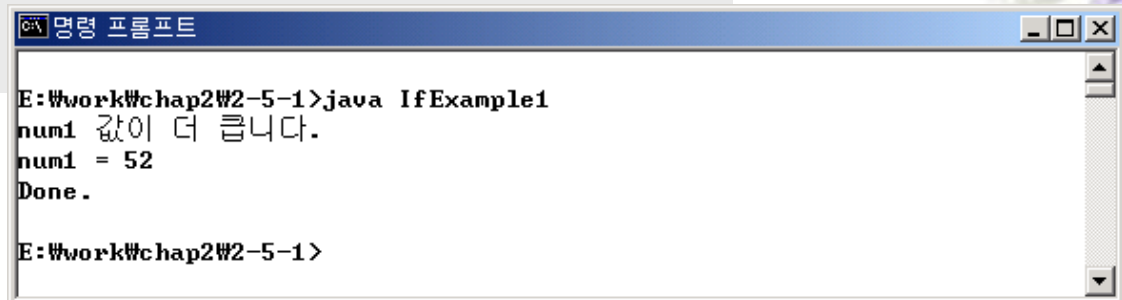
```
if (num1 > num2) {  
    System.out.println("num1 값이 더 큼니다.");  
    System.out.println(num1);  
}
```

# 조건문

## if 조건문

- [예제 2-21] if 문의 사용 예

```
1  class IfExample1 {  
2      public static void main(String args[]) {  
3          int num1 = 52;  
4          int num2 = 24;  
5          if (num1 > num2) {  
6              System.out.println("num1 값이 더 큼니다.");  
7              System.out.println("num1 = " + num1);  
8          }  
9          System.out.println("Done.");  
10     }  
11 }
```



```
명령 프롬프트  
E:\work\chap2\2-5-1>java IfExample1  
num1 값이 더 큼니다.  
num1 = 52  
Done.  
E:\work\chap2\2-5-1>
```

# 조건문

## ● if 조건문

### • if-else 조건문의 기본 형식

if ( 조건식 )

실행부분 ————— 조건식이 true일 때만 실행되는 부분

else

실행부분 ————— 조건식이 false일 때만 실행되는 부분

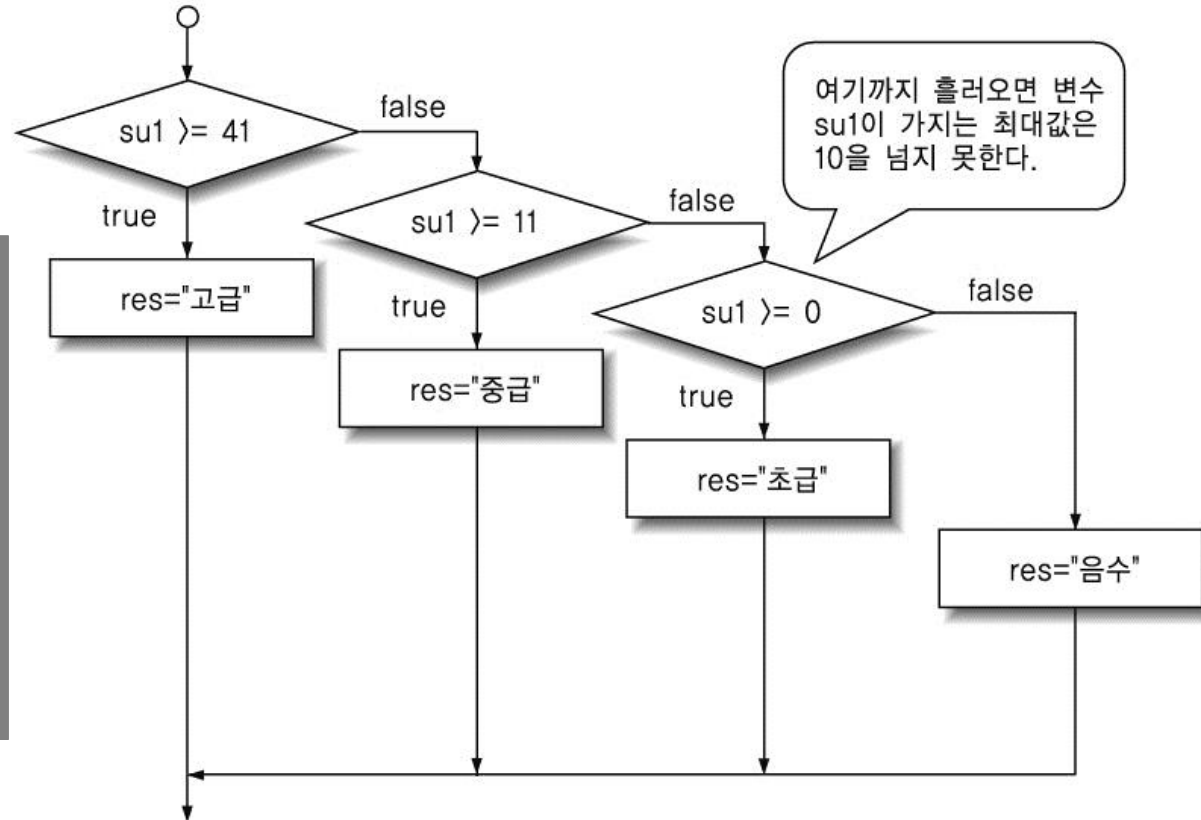
### • [예]

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



# 제어문

```
if(su1 >= 41)
    res = "고급" ;
else if(su1 >= 11)
    res = "중급" ;
else if(su1 >= 0)
    res = "초급" ;
else
    res = "음수" ;
```



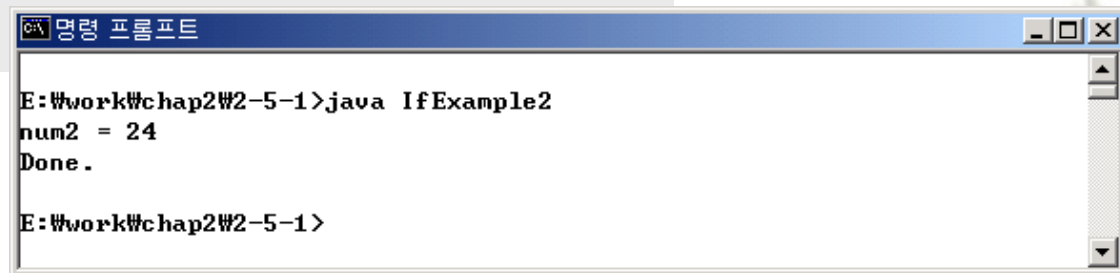
# 조건문

## 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.



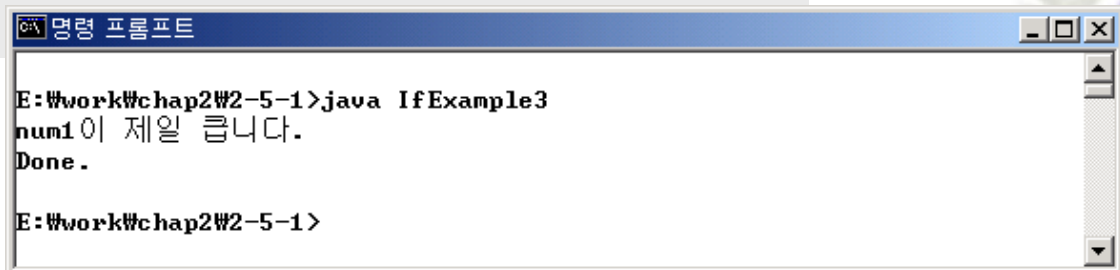
```
E:\work\chap2\2-5-1>java IfExample2  
num2 = 24  
Done.  
E:\work\chap2\2-5-1>
```

# 조건문

## if 조건문

- [예제 2-23] if 문을 포함하는 if 문의 예

```
1  class IfExample3 {
2      public static void main(String args[]) {
3          int num1 = 52;
4          int num2 = 24;
5          int num3 = 32;
6          if (num1 > num2) ----- num1보다 num2가 크면
7              if (num1 > num3) ----- num1과 num3를 비교해서
8                  System.out.println("num1이 제일 큼니다."); ----- num1이 클 때만
9          System.out.println("Done."); ----- 메시지를 출력합니다.
10     }
11 }
```



```
명령 프롬프트
E:\work\chap2\2-5-1>java IfExample3
num1이 제일 큼니다.
Done.
E:\work\chap2\2-5-1>
```

# 조건문

## ● if 조건문

- dangling else : 어느 if 키워드와 짝을 이루는지 모호한 else 키워드

```
if (num1 > num2)
    if (num1 > num3)
        System.out.println("num1 = " + num1);
    else
        System.out.println("num2 = " + num2);
```

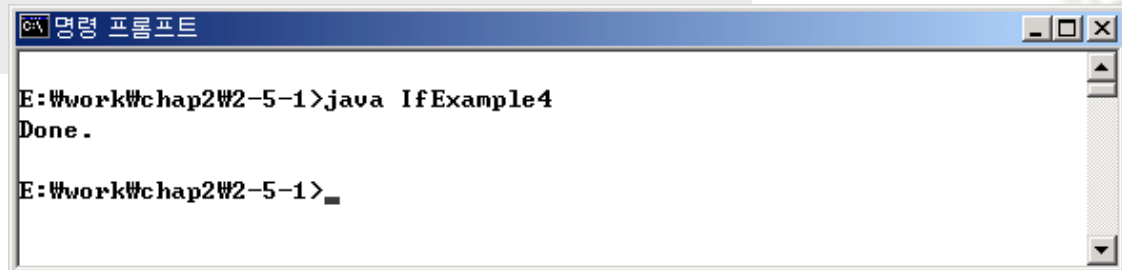
- 자바의 dangling else 규칙
- "dangling else는 가장 가까이 있는 if 키워드와 짝을 이룬다."

# 조건문

## if 조건문

- [예제 2-24] *dangling else* 규칙을 잘 모르고 작성한 프로그램

```
1  class IfExample4 {  
2      public static void main(String args[]) {  
3          int num1 = 152;  
4          int num2 = 173;  
5          if (num1 > num2)  
6              if (num1 > 100)  
7                  System.out.println("num1 = " + num1);  
8          else  
9              if (num2 > 100)  
10                 System.out.println("num2 = " + num2);  
11          System.out.println("Done.");  
12      }  
13  }
```



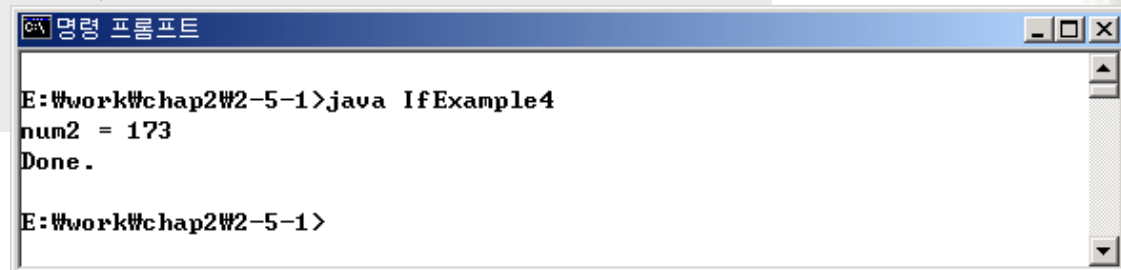
```
명령 프롬프트  
E:\work\chap2\2-5-1>java IfExample4  
Done.  
E:\work\chap2\2-5-1>
```

# 조건문

## if 조건문

- [예제 2-25] 수정된 IfExample4 프로그램

```
1  class IfExample4 {  
2      public static void main(String args[]) {  
3          int num1 = 152;  
4          int num2 = 173;  
5          if (num1 > num2) {  
6              if (num1 > 100)  
7                  System.out.println("num1 = " + num1);  
8          }  
9          else {  
10             if (num2 > 100)  
11                 System.out.println("num2 = " + num2);  
12         }  
13         System.out.println("Done.");  
14     }  
15 }
```



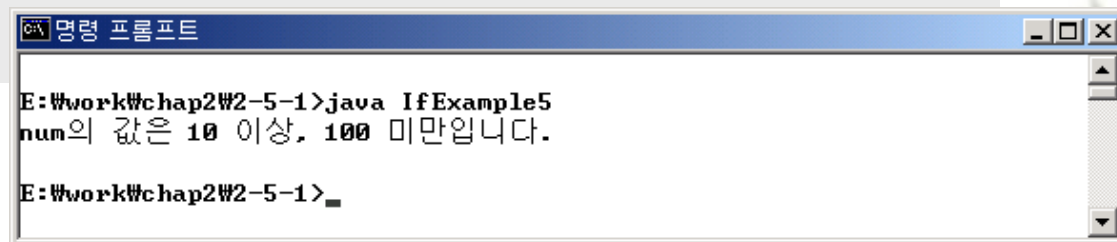
```
명령 프롬프트  
E:\work\chap2\2-5-1>java IfExample4  
num2 = 173  
Done.  
E:\work\chap2\2-5-1>
```

# 조건문

## if 조건문

- [예제 2-26] *dangling else* 규칙을 잘 활용한 프로그램

```
1  class IfExample5 {  
2      public static void main(String args[]) {  
3          int num = 74;  
4          if (num < 10)  
5              System.out.println("num의 값은 10 미만입니다.");  
6          else if (num < 100)  
7              System.out.println("num의 값은 10 이상, 100 미만입니다.");  
8          else if (num < 1000)  
9              System.out.println("num의 값은 100 이상, 1000 미만입니다.");  
10         else  
11             System.out.println("num의 값은 1000 이상입니다.");  
12     }  
13 }
```



```
명령 프롬프트  
E:\work\chap2\2-5-1>java IfExample5  
num의 값은 10 이상, 100 미만입니다.  
E:\work\chap2\2-5-1>
```

## switch문

- ▶ If문의 조건값은 boolean형인데 비해 switch문의 조건값은 long형을 제외한 정수형(byte, short, int) 또는 char형인 것이 다르다.

```
Switch(argument value) {  
    case 조건값1 : ← semicolon이 아닌 colon,  
        수행문; break;  
    case 조건값2 :  
        수행문; break; ← break문은 하나의 조건 값마다 넣어주는 것이 적당.  
    case 조건값3 :          만약 없을 시에는 다음 break문을 만날 때까지 모든  
        수행문; break      수행문을 처리,  
    default :              ← 받은 argument value가 case문의 조건값1에서 조건  
        수행문;           값3까지 일치하는 것이 단 하나도 없다면 default를 수행.  
}
```



# 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

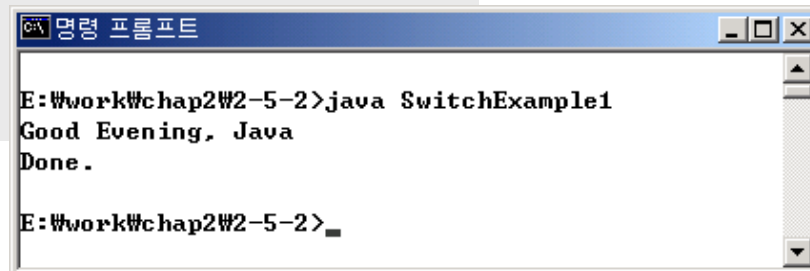
# 조건문

## 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>
```

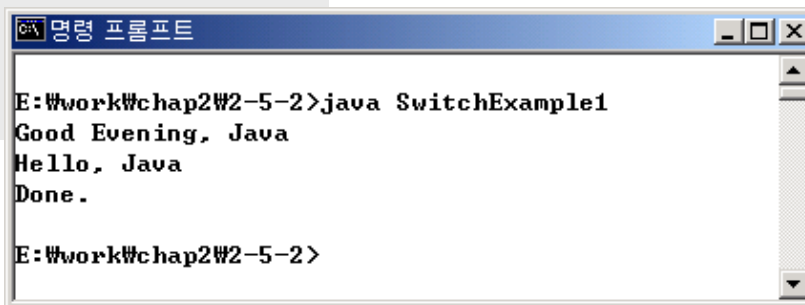
# 조건문

## switch 조건문

- [예제 2-28] break 문을 모두 없앤 SwitchExample1 프로그램

```
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              case 2 :  
8                  System.out.println("Good Afternoon, Java");  
9              case 3 :  
10                 System.out.println("Good Evening, Java");  
11                 default :  
12                     System.out.println("Hello, Java");  
13             }  
14             System.out.println("Done.");  
15         }  
16     }
```

break문이 없는 switch문



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

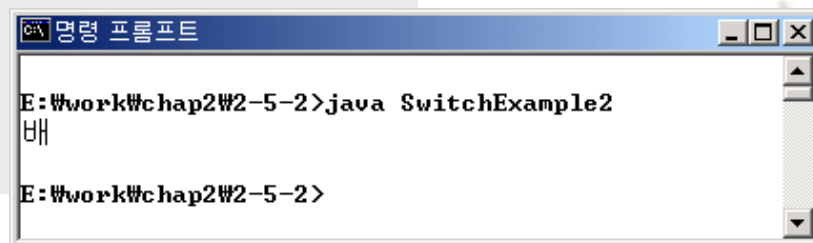
# 조건문

## switch 조건문

- [예제 2-29] 둘 이상의 값에 대해 같은 처리를 하는 switch 문의 예

```
1  class SwitchExample2 {  
2      public static void main(String args[]) {  
3          char ch = 'p';  
4          switch (ch) {  
5              case 'A' :  
6              case 'a' :  
7                  System.out.println("사과");  
8                  break;  
9              case 'P' :  
10             case 'p' :  
11                 System.out.println("배");  
12                 break;  
13             case 'G' :  
14             case 'g' :  
15                 System.out.println("포도");  
16                 break;  
17             default :  
18                 System.out.println("?");  
19                 break;  
20         }  
21     }  
22 }
```

A와 a, P와 p, G와 g에 대해  
똑같은 처리를 하는 switch 문입니다.



```
C:\> 명령 프롬프트  
E:\work\chap2\2-5-2> java SwitchExample2  
배  
E:\work\chap2\2-5-2>
```

# 제어문

## for문

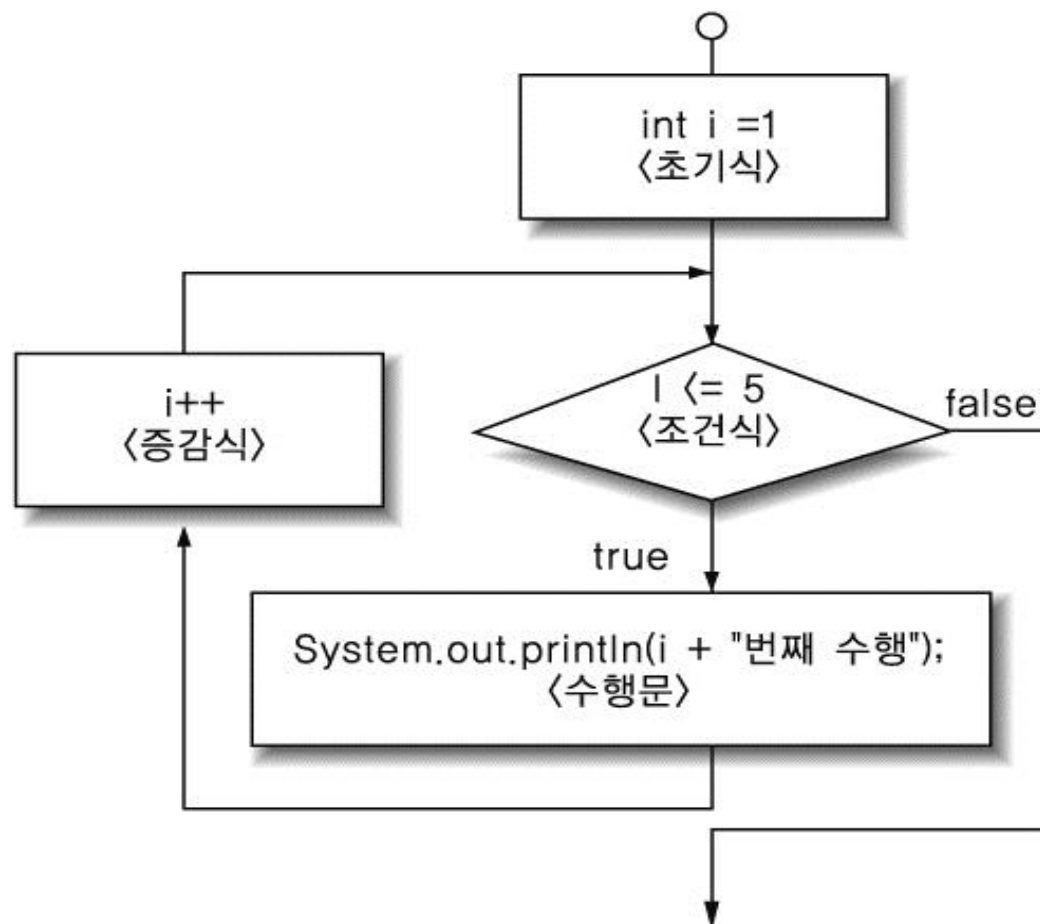
- ▶ 특정한 명령들을 정해진 규칙에 따라 반복처리 할 때 사용하는 제어문이다. 다음은 for문의 구성이다.

```
for(초기식 ; 조건식 ; 증감식){  
    수행문1;  
    수행문2;  
}
```

초기식	가장 먼저 수행하는 부분이며 두 번 다시 수행하지 않는다.(다중 for문에서는 예외)
조건식	초기식 다음으로 수행하는 부분이며 loop가 돌 때마다 한번씩 비교하여 반복을 수행해야 할지 반복을 벗어나야 할지를 결정한다.
증감식	증감식은 loop를 수행할 때마다 조건식에서 비교하기 전에 항상 수행하며 조건식에 사용되는 변수의 값을 증가 시키거나 감소 시켜 loop를 원활하게 수행하거나 무한 루프를 피하는데 바탕이 되는 부분이다.

# 제어문

```
04     for(int i = 1 ; i <= 5 ; i++)  
05         System.out.println(i+"번째 수행");  
06
```



# 제어문

## 다중 for문

- ▶ 단일 for문에서 끝나는 것이 아니라 그것을 다시 여러 번 반복하는 제어문이다. 다시 말해서 for문 안에 for문이 있는 경우를 다중 for문이라 한다.
- ▶ 예문 : 애국가 1절~4절까지를 3번 부르세요!

```
for(초기식1 ; 조건식1 ; 증감식1) {
```

```
    명령어1;
```

```
    for(초기식2 ; 조건식2 ; 증감식2){  
        명령어2;  
    }
```

3번  
부르기

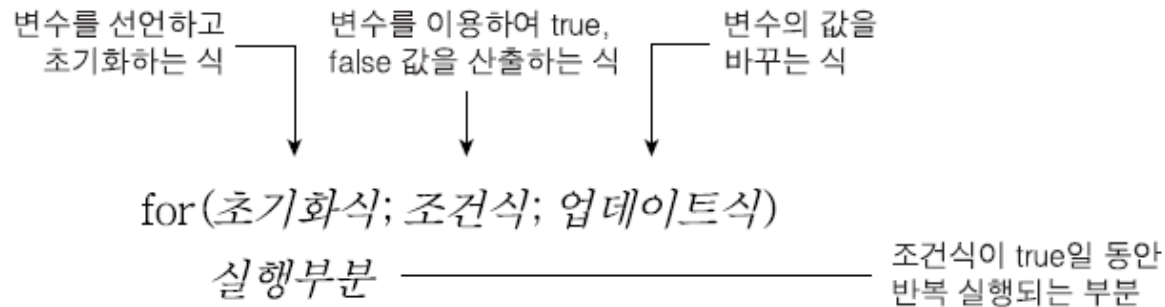
1절부터 4절까지  
부르기

```
}  
    명령어3;
```

## 반복문

for 반복문

•• for 문의 기본 형식



- [예]

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



# 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.



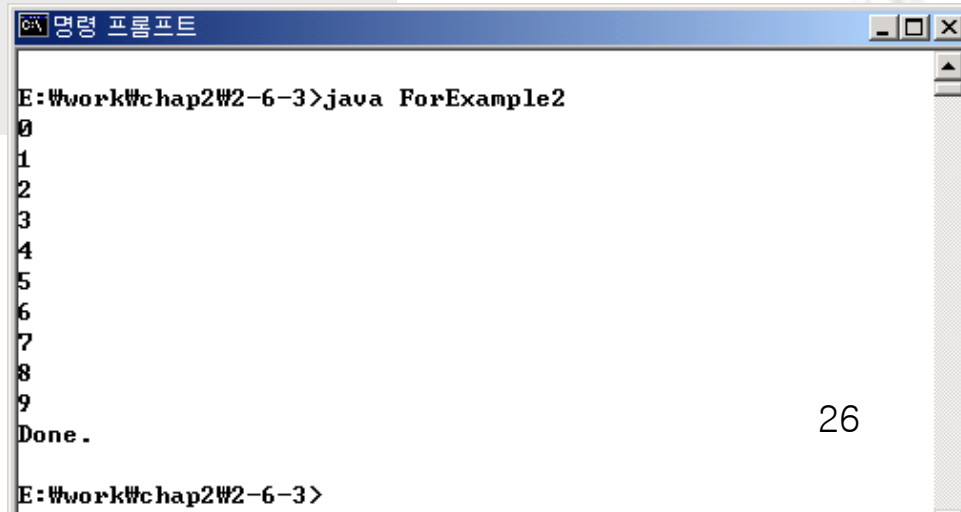
```
명령 프롬프트  
E:\work\chap2\2-6-3>java ForExample1  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
Done.  
E:\work\chap2\2-6-3>
```

# 반복문

## for 반복문

- [예제 2-34] 전형적이지 않은 for 문의 예

```
1    class ForExample2 {  
2        public static void main(String args[]) {  
3            int cnt = 0;  
4            for ( ; cnt < 10; ) {  
5                System.out.println(cnt);  
6                cnt++;  
7            }  
8            System.out.println("Done.");  
9        }  
10    }
```



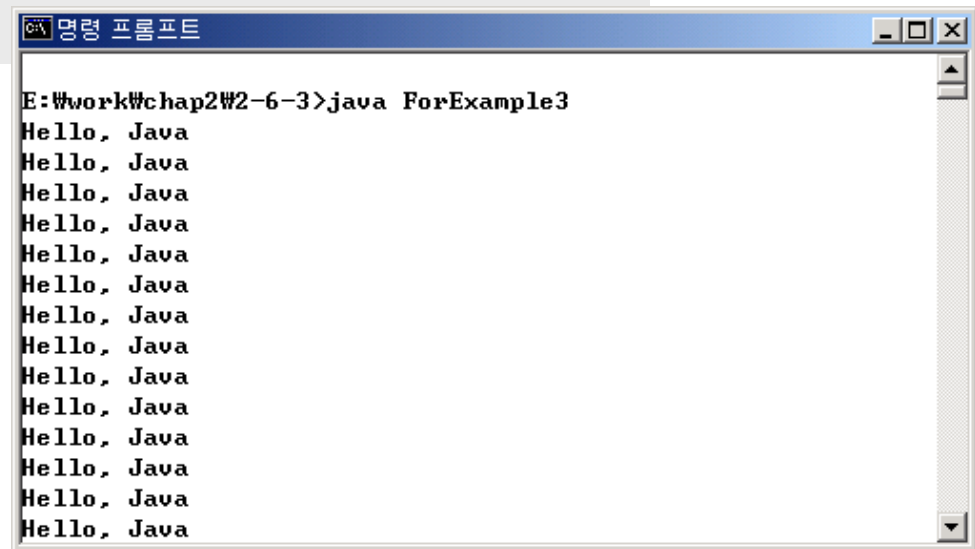
```
C:\명령 프롬프트  
E:\work\chap2\2-6-3>java ForExample2  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
Done.  
E:\work\chap2\2-6-3>
```

# 반복문

## for 반복문

- [예제 2-35] for 문을 이용한 무한 루프 프로그램

```
1    class ForExample3 {  
2        public static void main(String args[]) {  
3            for (;;)   
4                System.out.println("Hello, Java");  
5        }  
6    }
```



```
명령 프롬프트  
E:\work\chap2\2-6-3>java ForExample3  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java
```

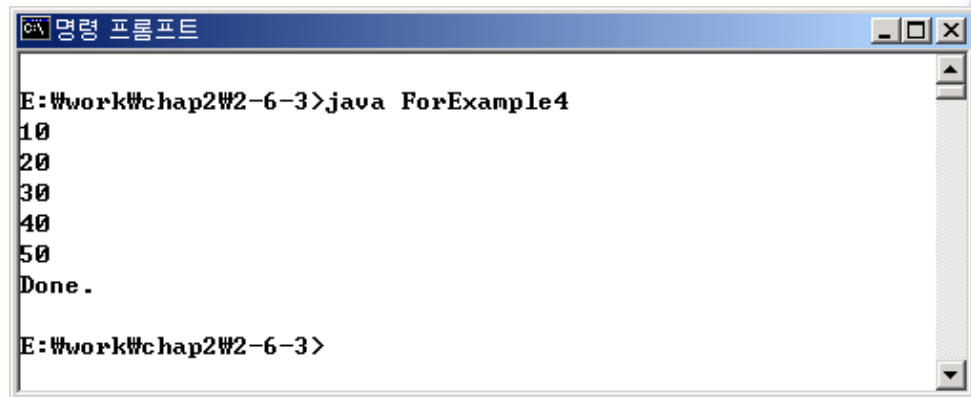
# 반복문

## for 반복문

- [예제 2-36] for 문으로 배열 항목을 순서대로 처리하는 프로그램

```
1  class ForExample4 {  
2      public static void main(String args[]) {  
3          int arr[] = { 10, 20, 30, 40, 50 };  
4          for (int cnt = 0; cnt < arr.length; cnt++) {  
5              System.out.println(arr[cnt]);  
6          }  
7          System.out.println("Done.");  
8      }  
9  }
```

배열의 항목을  
순서대로 출력합니다.



```
C:\>명령 프롬프트  
E:\work\chap2\2-6-3>java ForExample4  
10  
20  
30  
40  
50  
Done.  
E:\work\chap2\2-6-3>
```

# 반복문

## ● for 반복문

• 향상된 for 문의 형식

for(변수타입 변수이름 : 배열이름)

실행부분 ————— 반복 실행되는 부분

- 변수 타입 : 배열 항목과 동일한 타입
- 변수 이름 : 프로그래머가 나름대로 정할 수 있음

### • [예]

```
for (int num : arr)
    System.out.println(num);
```

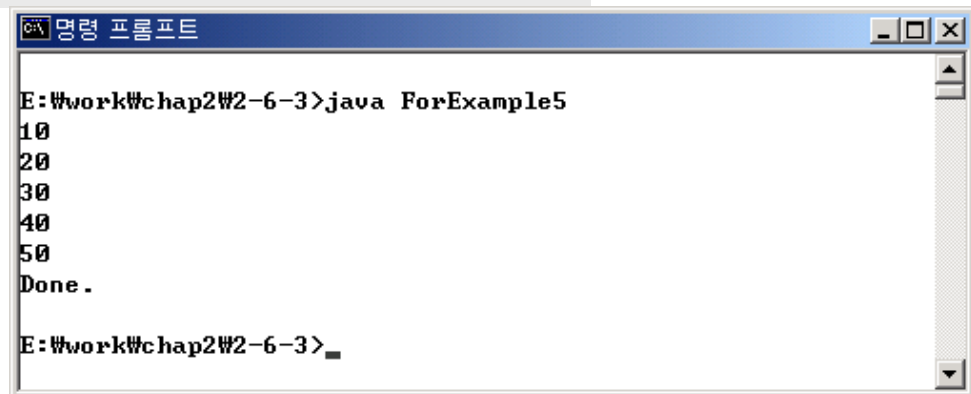
# 반복문

## for 반복문

- [예제 2-37] 향상된 for 문의 예

```
1  class ForExample5 {  
2      public static void main(String args[]) {  
3          int arr[] = { 10, 20, 30, 40, 50 };  
4          for (int num : arr) {  
5              System.out.println(num);  
6          }  
7          System.out.println("Done.");  
8      }  
9  }
```

배열의 항목을 순서대로 출력합니다.

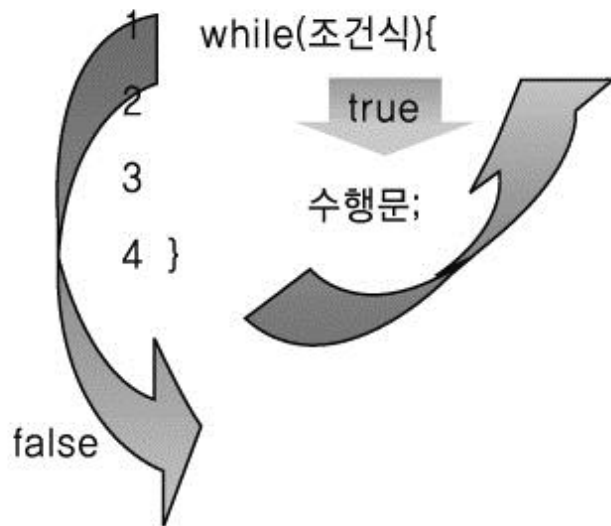


```
명령 프롬프트  
E:\work\chap2\2-6-3>java ForExample5  
10  
20  
30  
40  
50  
Done .  
E:\work\chap2\2-6-3>
```

# 제어문

## while문

- ▶ while문은 for문과 유사하며 조건비교에 만족 할 때에만 반복 처리하는 제어문이다, 다음은 구성과 동작이다



:: First비교, Last처리

# 반복문

## ● while 반복문

•• while 문의 기본 형식

while ( 조건식 )  
실행부분

———— true 또는 false 값을 산출할 수 있는 식

———— 조건식이 true일 동안 반복 실행되는 부분



# 반복문

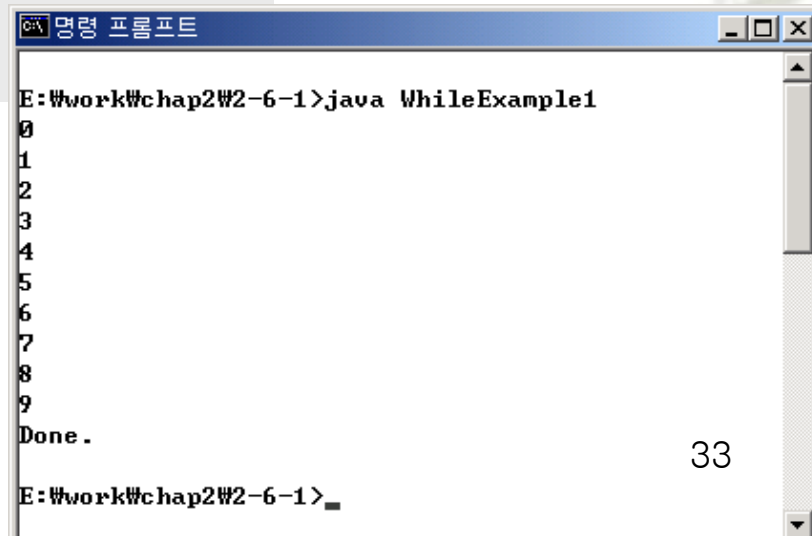
## while 반복문

- [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.



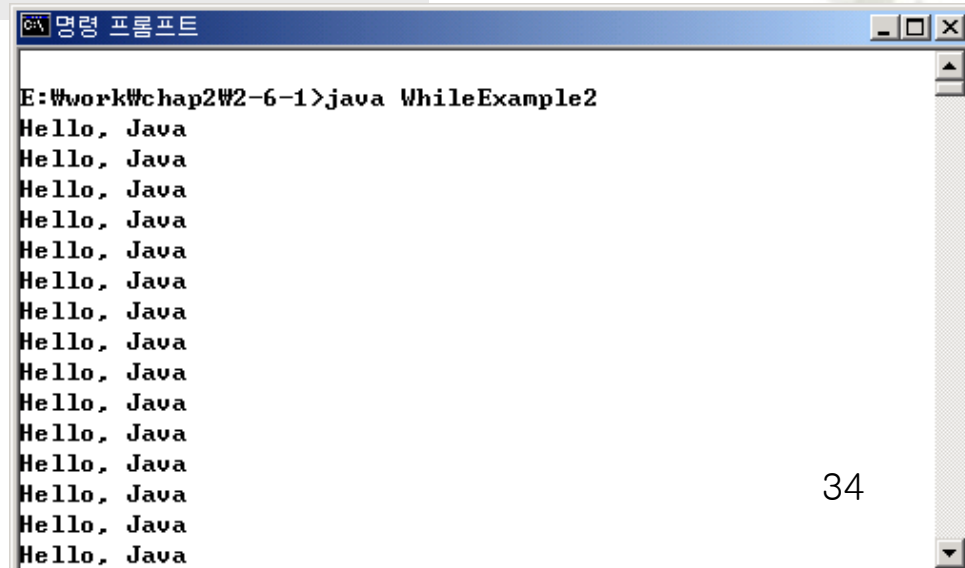
```
명령 프롬프트  
E:\work\chap2\2-6-1>java WhileExample1  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
Done.  
E:\work\chap2\2-6-1>
```

# 반복문

## • while 반복문

- [예제 2-31] while 문을 이용한 무한 루프 프로그램

```
1    class WhileExample2 {  
2        public static void main(String args[]) {  
3            while (true)  
4                System.out.println("Hello, Java");  
5        }  
6    }
```

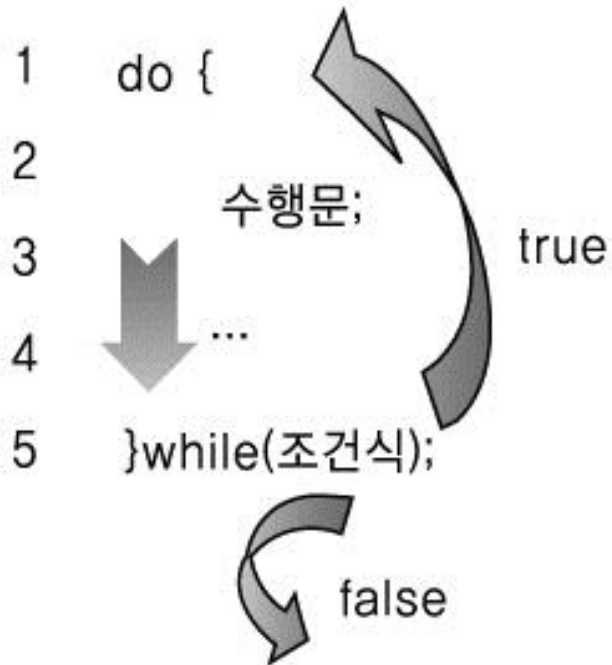


```
명령 프롬프트  
E:\work\chap2\2-6-1>java WhileExample2  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java  
Hello, Java
```

# 제어문

## do~while문

- ▶ while문이 [First 비교, Last 처리]라 하면 do~while문은 [First 처리, Last 비교]이다, 즉 조건비교에 불 만족해도 무조건 한번은 수행하게 되어 있음!



:: 先 처리, 後 비교

# 반복문

## ● do-while 반복문

•• do-while 문의 기본 형식

do

실행부분 ————— 조건식이 true일 동안 반복 실행되는 부분

while ( 조건식 );

————— 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

# 반복문

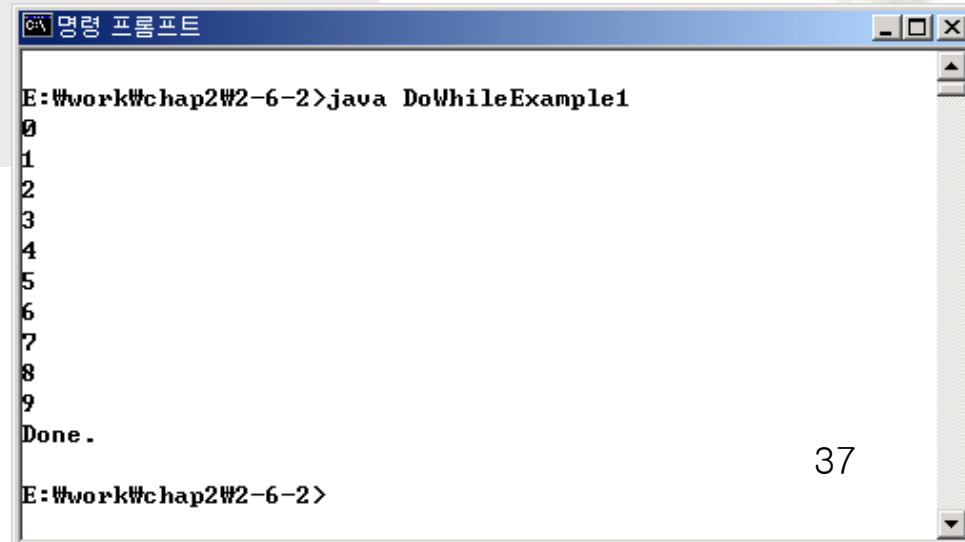
## do-while 반복문

- [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 }
```

이 부분을 한 번 실행하고 나서

cnt가 10보다 작으면 계속 반복 실행합니다.



```
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>
```

## break문

➡ 가장 가까운 반복문을 탈출할 때 쓰이는 제어문.

다음은 내부 for문에서 break를 사용 했으므로 내부 for문만 탈출한다는 뜻의 그림이다



# 제어문

## ● break 문

- while, do, for 문 안에서 사용되면 반복문을 빠져나가는 기능
- switch 문 안에서 사용되면 switch 문을 빠져나가는 기능
- break 문의 기본 형식

break;

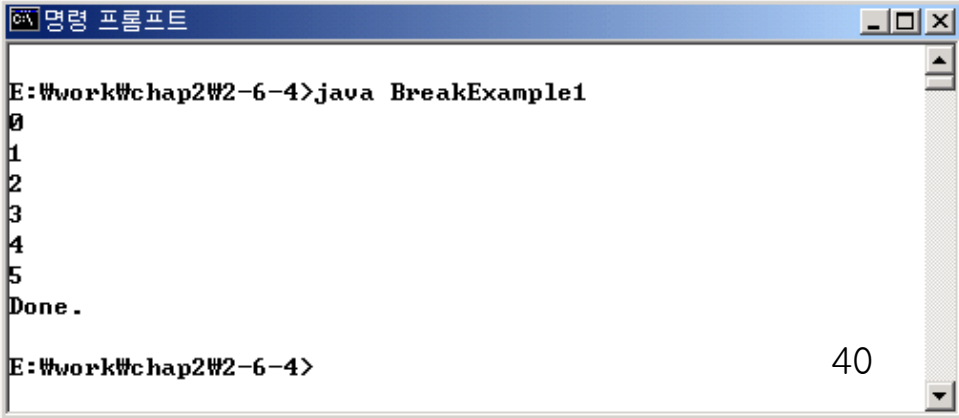
# 제어문

## break 문

- [예제 2-38] `break` 문을 이용하여 반복문을 빠져나가는 예

```
1  class BreakExample1 {  
2      public static void main(String args[]) {  
3          for (int cnt = 0; cnt < 10; cnt++) {  
4              System.out.println(cnt);  
5              if (cnt == 5) {  
6                  break; }  
7          }  
8          System.out.println("Done.");  
9      }  
10 }
```

cnt 값이 5이면 for 반복문을 빠져나갑니다.



```
명령 프롬프트  
E:\work\chap2\2-6-4>java BreakExample1  
0  
1  
2  
3  
4  
5  
Done.  
E:\work\chap2\2-6-4>
```



# 제어문

## 중첩된 반복문과 break 문

- [예제 2-39] 중첩된 반복문을 빠져나가는 break 문

```
1  class BreakExample2 {
2      public static void main(String args[]) {
3          for (int row = 0; row < 3; row++) {
4              for (int col = 0; col < 5; col++) {
5                  System.out.println("(" + row + ", " + col + ")");
6                  if ((row == 1) && (col == 3)) {
7                      break;
8                  }
9              }
10         System.out.println("Done.");
11     }
12 }
```

row가 1이고 col이 3이면  
안쪽 for 반복문을 빠져나갑니다.

```
C:\ 명령 프롬프트
E:\work\chap2\2-6-4>java BreakExample2
<0, 0>
<0, 1>
<0, 2>
<0, 3>
<0, 4>
<1, 0>
<1, 1>
<1, 2>
<1, 3>
<2, 0>
<2, 1>
<2, 2>
<2, 3>
<2, 4>
```

## 중첩된 반복문과 break 문

• 중첩된 반복문을 한꺼번에 빠져나가는 방법

- 1) 반복문에 라벨을 붙인다
- 2) break 문에 라벨을 지정한다

```
loop: for (int cnt = 0; cnt < 100; cnt++) {  
    System.out.println(cnt);  
    if (cnt > 10)  
        break loop;  
}
```

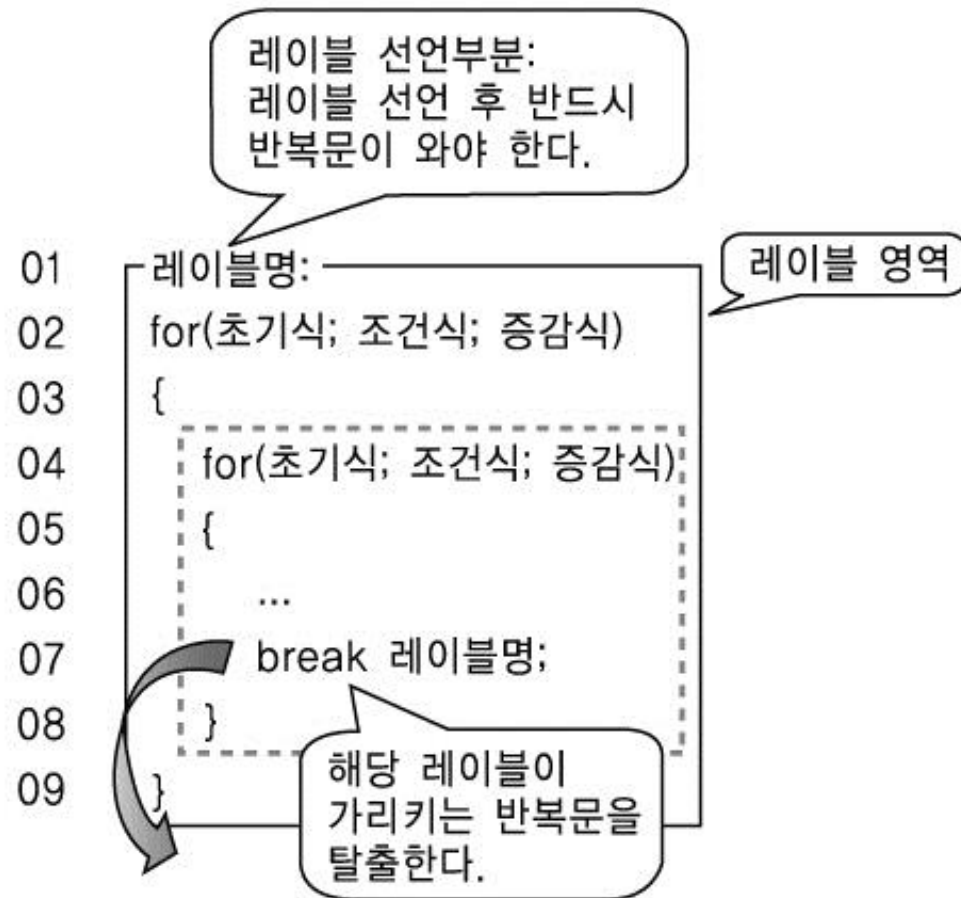
for 문에 붙여진 라벨

라벨을 지정한 break 문

# 제어문

## break label문

- ➡ `break label`은 `break`문과 같지만 다중 반복문에서 한번에 바깥쪽 반복문을 탈출할 때 많이 쓰이는 제어문.



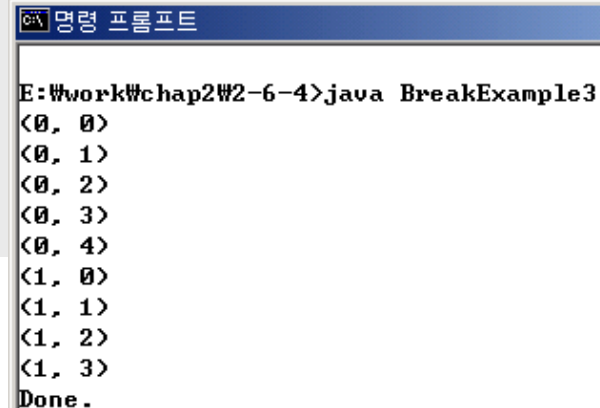
# 제어문

## 중첩된 반복문과 break 문

- [예제 2-40] 중첩된 반복문을 빠져나가는 break 문

```
class BreakExample3 {  
    public static void main(String args[]) {  
        outerLoop:  
        for (int row = 0; row < 3; row++) {  
            for (int col = 0; col < 5; col++) {  
                System.out.println("(" + row + ", " + col + ")");  
                if ((row == 1) && (col == 3))  
                    break outerLoop;  
            }  
        }  
        System.out.println("Done.");  
    }  
}
```

row가 1이고 col이 3이면  
바깥쪽 반복문을 빠져나갑니다.




```
명령 프롬프트  
E:\work\chap2\2-6-4>java BreakExample3  
<0, 0>  
<0, 1>  
<0, 2>  
<0, 3>  
<0, 4>  
<1, 0>  
<1, 1>  
<1, 2>  
<1, 3>  
Done.  
E:\work\chap2\2-6-4>
```

# 제어문

## continue문

- 반복문을 탈출하기 위해 사용되는 것이 아니라 continue문 이하의 수행 문들을 포기하고 다음 회차의 반복을 수행하기 위한 제어문.

```
01
02  for(초기식; 조건식; 증감식)
03  {
04      for(초기식; 조건식; 증감식)
05      {
06          ...
07          continue;
08          수행문 1;
09      }
10  }
```



continue문을 만나면 다음의 수행문들을 수행하지 않고 다음 반복을 위해 증감식으로 넘어간다.

# 제어문

## ● continue 문

- 반복문 안에서만 사용 가능
- 반복문의 다음번 반복을 계속하는 기능
- `continue` 문의 기본 형식

`continue;`

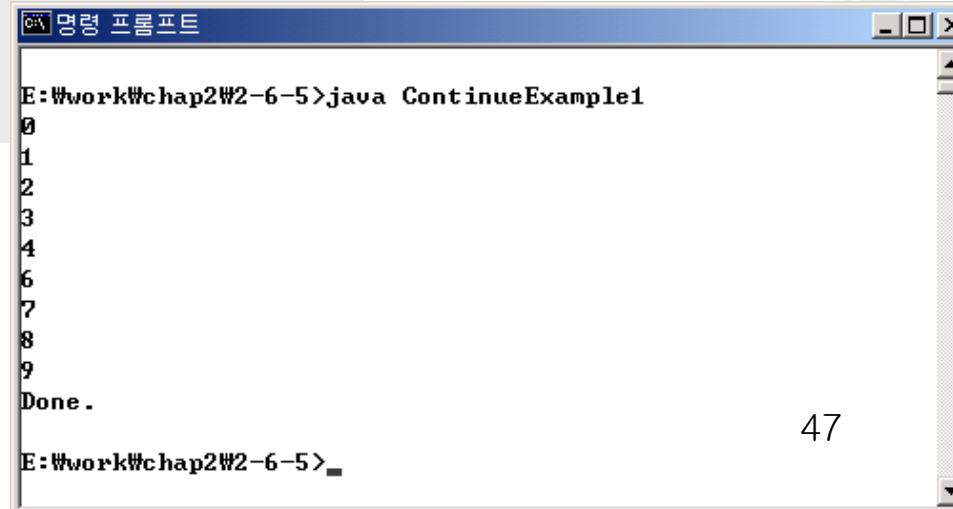
# 제어문

## continue 문

- [예제 2-41] continue 문의 사용 예

```
1  class ContinueExample1 {  
2      public static void main(String args[]) {  
3          for (int cnt = 0; cnt < 10; cnt++) { <-  
4              if (cnt == 5)  
5                  continue; -----  
6              System.out.println(cnt);  
7          }  
8          System.out.println("Done.");  
9      }  
10 }
```

cnt가 5이면 for 문의 다음번  
반복 과정을 계속합니다.



```
명령 프롬프트  
E:\work\chap2\2-6-5>java ContinueExample1  
0  
1  
2  
3  
4  
6  
7  
8  
9  
Done.  
E:\work\chap2\2-6-5>
```

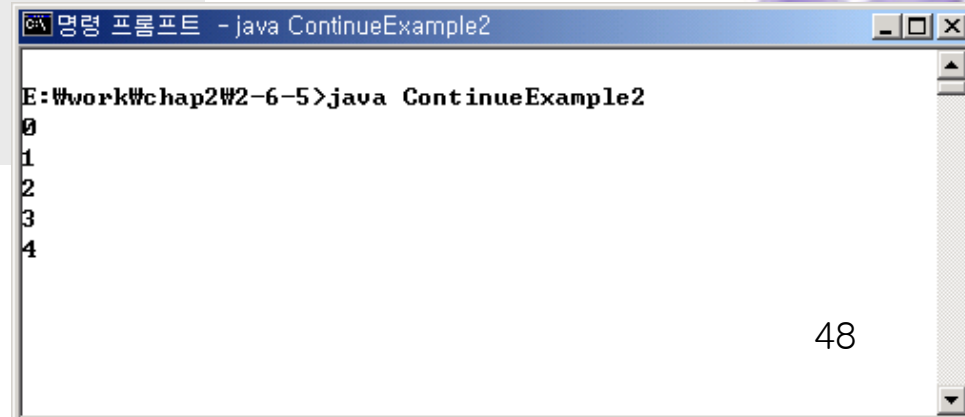
# 제어문

## continue 문

- [예제 2-42] `continue` 문의 잘못된 사용 예

```
1  class ContinueExample2 {  
2      public static void main(String args[]) {  
3          int cnt = 0;  
4          while (cnt < 10) {  
5              if (cnt == 5)  
6                  continue;  
7              System.out.println(cnt);  
8              cnt++;  
9          }  
10         System.out.println("Done.");  
11     }  
12 }
```

cnt가 5이면 while 문의 다음번 반복 과정을 계속합니다.



```
명령 프롬프트 - java ContinueExample2  
E:\work\chap2\2-6-5>java ContinueExample2  
0  
1  
2  
3  
4
```



# 제어문

## continue label문

- ▶ 레이블을 가지는 continue문은 레이블이 지칭하는 반복문의 조건식 또는 증감식으로 Program상 수행 시점(제어권)이 이동.

```
01 레이블명:  
02 for(초기식; 조건식; 증감식)  
03 {  
04     for(초기식; 조건식; 증감식)  
05     {  
06         ...  
07         continue 레이블명;  
08         수행문 1;  
09     }  
10 }
```

내부 반복문을 중단하고 외부 반복문의 다음 반복회차를 수행하기 위한 제어문이다.  
continue 문 아래의 수행문 1은 수행하지 못한다.

[그림 3-27] continue label문의 구성과 동작

## 중첩된 반복문과 continue 문

• 중첩된 반복문의 바깥쪽 반복을 계속하는 방법

- 1) 반복문에 라벨을 붙인다
- 2) continue 문에 라벨을 지정한다

```
loop: ————— for 문에 붙여진 라벨
    for (int cnt = 0; cnt < 100; cnt++) {
        System.out.println(cnt);
        if (cnt == 5)
            continue loop; ————— 라벨을 지정한 continue 문
    }
```

# 제어문

## 중첩된 반복문과 continue 문

- [예제 2-43] 중첩된 반복문의 바깥쪽 반복을 계속하는 `continue` 문

```
1  class ContinueExample3 {
2      public static void main(String args[]) {
3          outerLoop:
4              for (int row = 0; row < 3; row++) { <-----
5                  for (int col = 0; col < 5; col++) {
6                      if ((row == 1) && (col == 3))
7                          continue outerLoop; -----
8                      System.out.println("(" + row + ", " + col + ")");
9                  }
10             }
11             System.out.println("Done.");
12         }
13     }
```

row가 1이고 col이 3이면  
바깥쪽 for 문의 다음번  
반복 과정을 계속합니다.

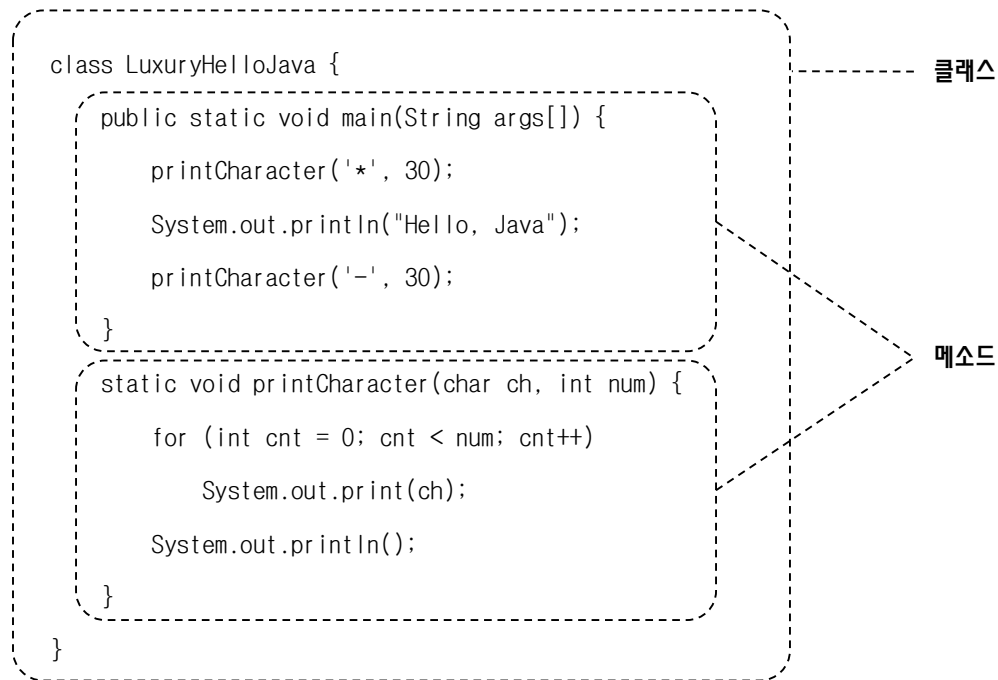
명령 프롬프트

```
E:\work\chap2\2-6-5>java ContinueExample3
<0, 0>
<0, 1>
<0, 2>
<0, 3>
<0, 4>
<1, 0>
<1, 1>
<1, 2>
<2, 0>
<2, 1>
<2, 2>
<2, 3>
<2, 4>
Done.
```

## 07. 메소드 호출문

### 메소드 호출문

- 여러 개의 메소드가 포함된 클래스



- - main이 아닌 메소드는 자동으로 실행되지 않음

## 07. 메소드 호출문

### 메소드 호출문

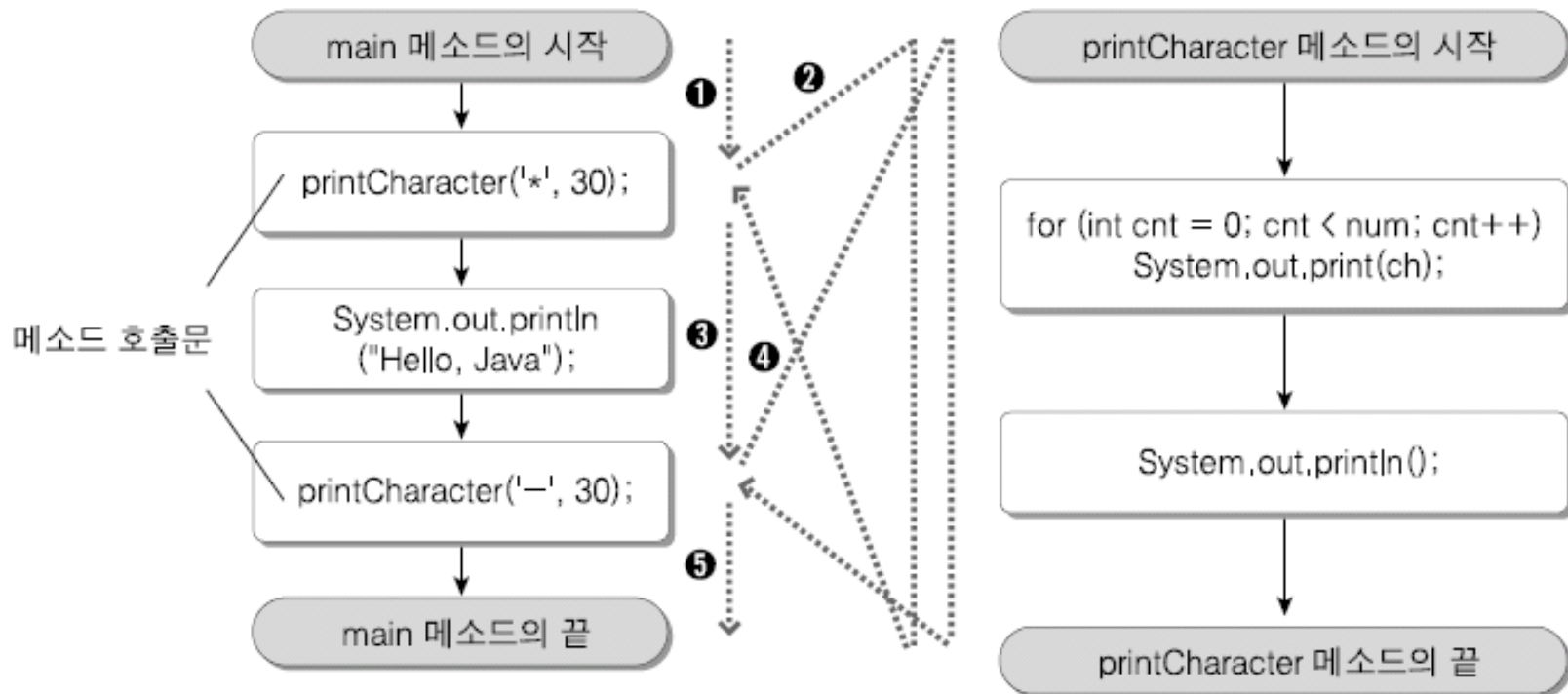
- 여러 개의 메소드가 포함된 클래스

```
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);
```

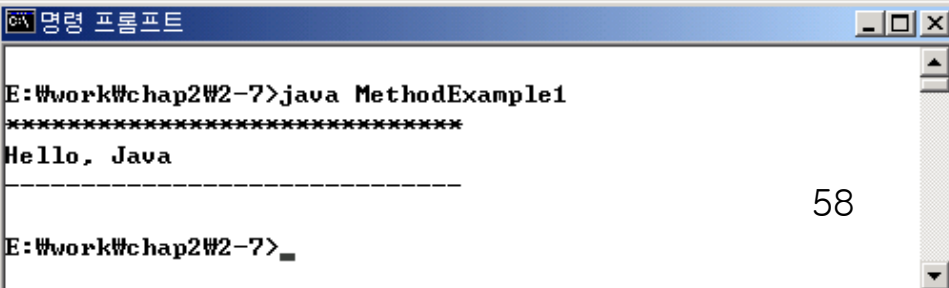
## 07. 메소드 호출문

### 메소드 호출문의 작성 방법

- [예제 2-44] 메소드 호출 예

```
1  class MethodExample1 {  
2      public static void main(String args[]) {  
3          printCharacter('*', 30); ----- 메소드 호출문  
4          System.out.println("Hello, Java");  
5          printCharacter('-', 30); ----- 메소드 호출문  
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 }
```

호출되는 메소드



```
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

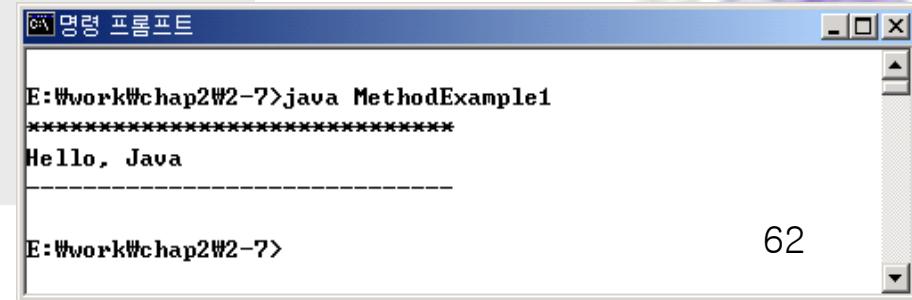
### return 문

- [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

리턴 값이 없는 return 문



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

E:\work\chap2\2-7>
```

## 07. 메소드 호출문

### main 메소드의 파라미터

- main 메소드의 파라미터 변수가 하는 일

클래스 이름 뒤에 오는 명령행 파라미터들은 배열에 담겨져서 main 메소드에 전달됩니다.

```
class DummyClass {  
    public static void main(String args[]) {  
        .  
        .  
        .  
    }  
}
```

## 07. 메소드 호출문

### main 메소드의 파라미터

- [예제 2-47] 명령행 파라미터를 출력하는 프로그램

```
1    class ParamExample1 {  
2        public static void main(String args[]) {  
3            for (String str : args) }----- args 배열의 항목 값을 순서대로 출력합니다.  
4                System.out.println(str);  
5            System.out.println("args.length=" + args.length);----- args 배열의 항목 수를  
6        }                                           출력합니다.  
7    }
```



## 07. 메소드 호출문

### main 메소드의 파라미터

- [예제 2-47] 명령행 파라미터를 출력하는 프로그램 (실행 결과)

```
E:\work\chap2\2-6>java ParamExample1
args.length=0

E:\work\chap2\2-6>java ParamExample1 apple pear peach
apple
pear
peach
args.length=3

E:\work\chap2\2-6>java ParamExample1 사과 배 복숭아 참외
사과
배
복숭아
참외
args.length=4

E:\work\chap2\2-6>java ParamExample1 Hello, Java
Hello,
Java
args.length=2

E:\work\chap2\2-6>java ParamExample1 "Hello, Java"
Hello, Java
args.length=1

E:\work\chap2\2-6>_
```

이렇게 실행하면 main 메소드에는 크기 0인 배열이 전달됩니다.

이렇게 실행하면 클래스 이름 뒤의 명령행 파라미터들이 main 메소드에 전달됩니다.

공백문자를 포함하는 값을 전달하려면 큰 따옴표로 묶어서 표시해야 합니다.

X  
0

오늘 보다 내일이 더 기대되는~ 님들아!