

# 2장 자바의 기초 문법(2)

## 학습 목표

- 조건문의 작성
- 반복문의 작성
- 메소드 호출문의 작성
- 익셉션을 처리하는 try 문

## 05. 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

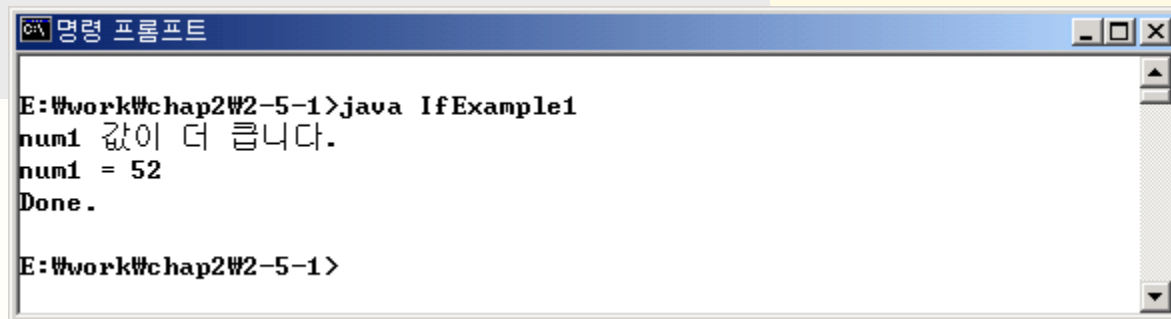
```
if (num1 > num2) {  
    System.out.println("num1 값이 더 큼니다.");  
    System.out.println(num1);  
}
```

## 05. 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>
```

## 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);
```

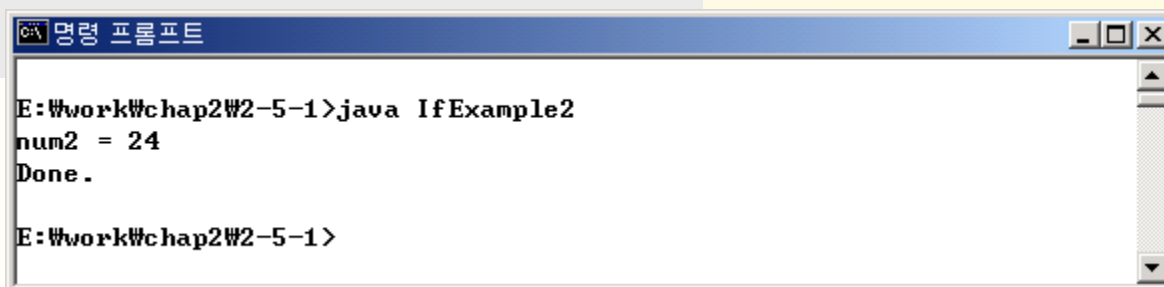
## 05. 조건문

### 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.



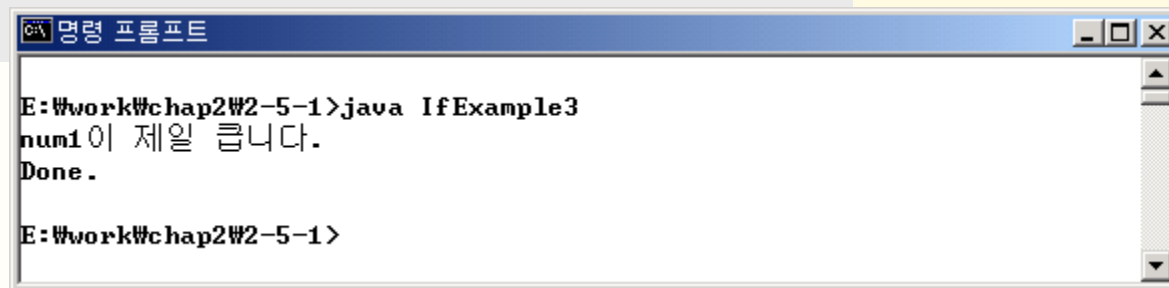
```
명령 프롬프트  
E:\work\chap2\2-5-1>java IfExample2  
num2 = 24  
Done.  
E:\work\chap2\2-5-1>
```

## 05. 조건문

### 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>
```

## 05. 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);
```

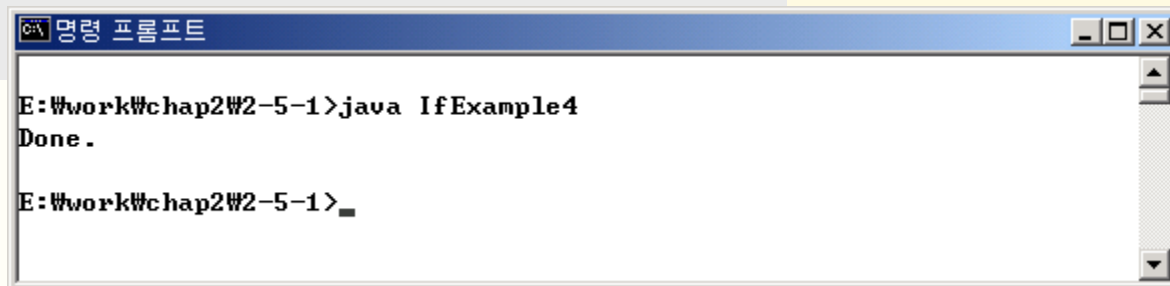
- 자바의 dangling else 규칙  
“dangling else는 가장 가까이 있는 if 키워드와 짝을 이룬다.”

## 05. 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     }
```



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

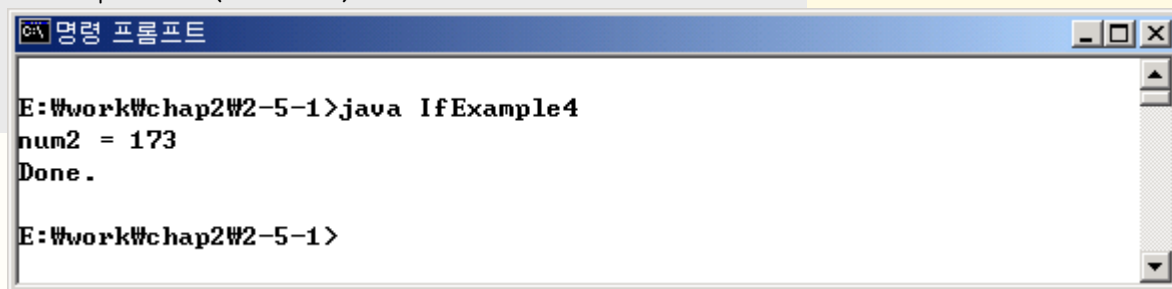


## 05. 조건문

### 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 }
```



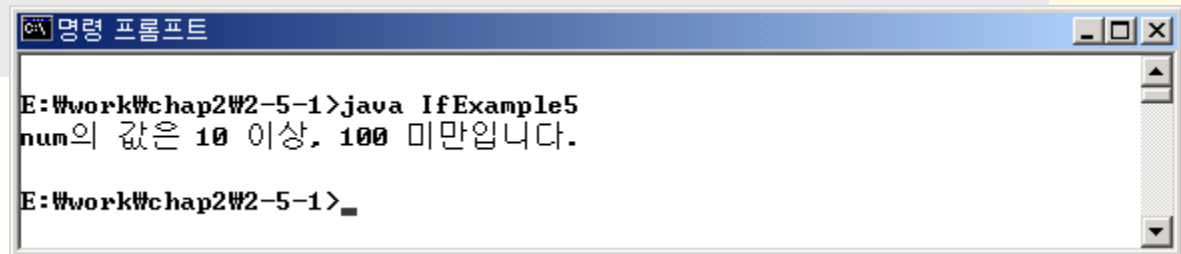
```
명령 프롬프트  
E:\work\chap2\2-5-1>java IfExample4  
num2 = 173  
Done.  
E:\work\chap2\2-5-1>
```

## 05. 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>
```

## 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

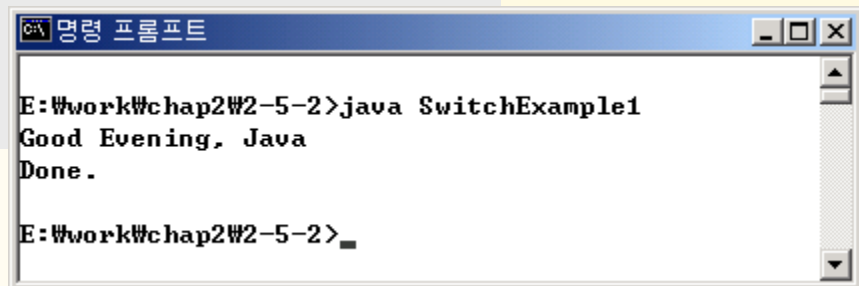
## 05. 조건문

### 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>
```

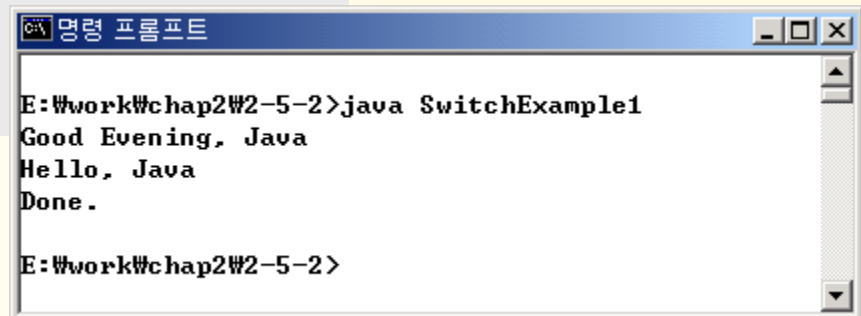
## 05. 조건문

### 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 }
```

break 문이 없는 switch 문



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

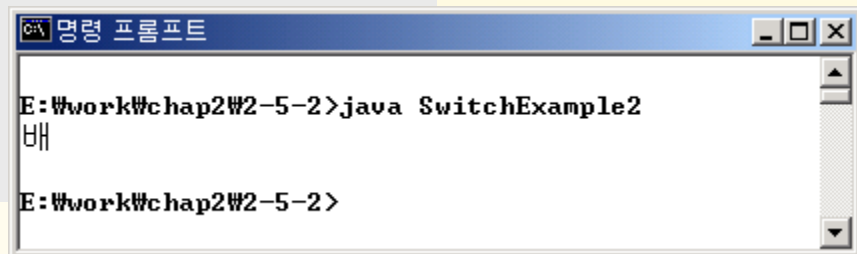
## 05. 조건문

### 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.



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

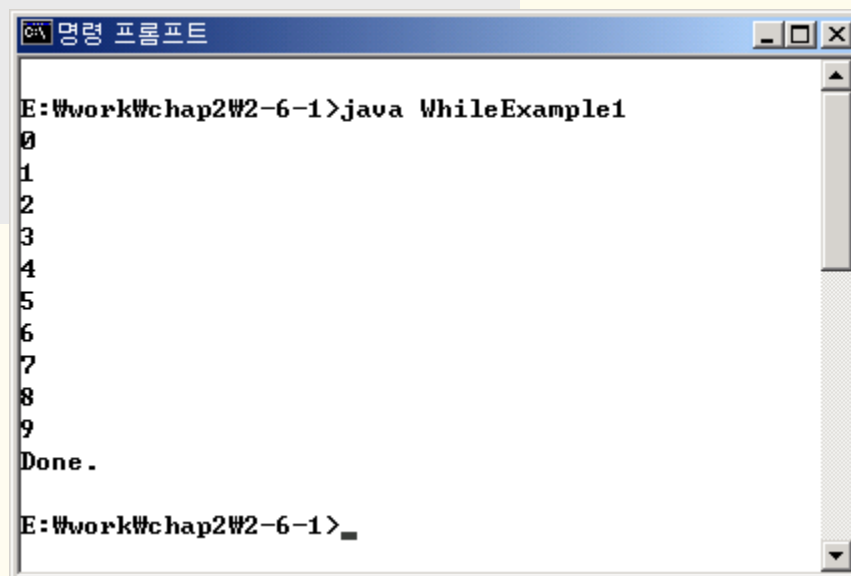
## 06. 반복문

### 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>
```




## while 반복문

```

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

```



A screenshot of a Windows command prompt window titled "명령 프롬프트" (Command Prompt). The window shows the command "E:\work\chap2\2-6-1>java WhileExample2" being entered.

[illegible]

## 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 }
```



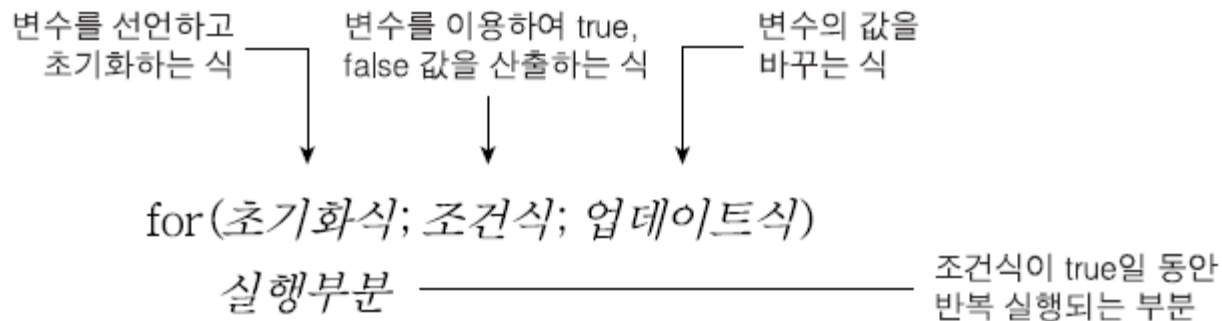
명령 프롬프트

```
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>
```

## 06. 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 }
```



```
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>
```

## 06. 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 }
```

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\2-6-3>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
```

The output consists of 15 identical lines, each displaying "Hello, Java".

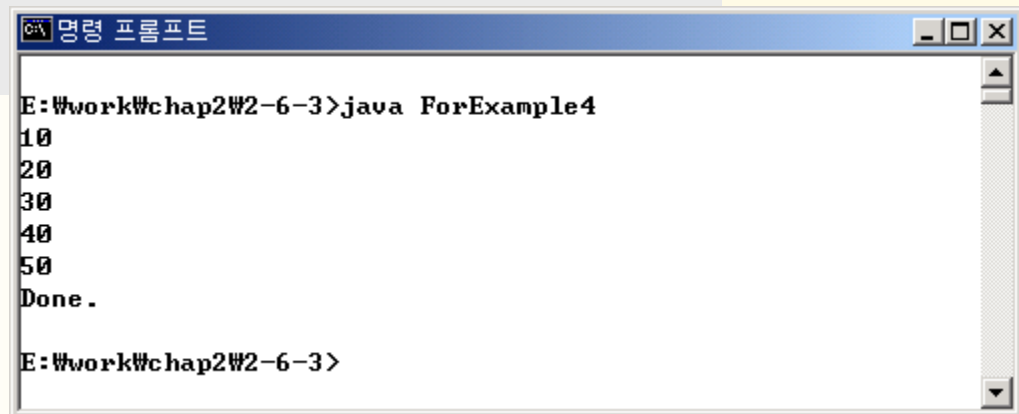
## 06. 반복문

### 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.



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



## 06. 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);
```

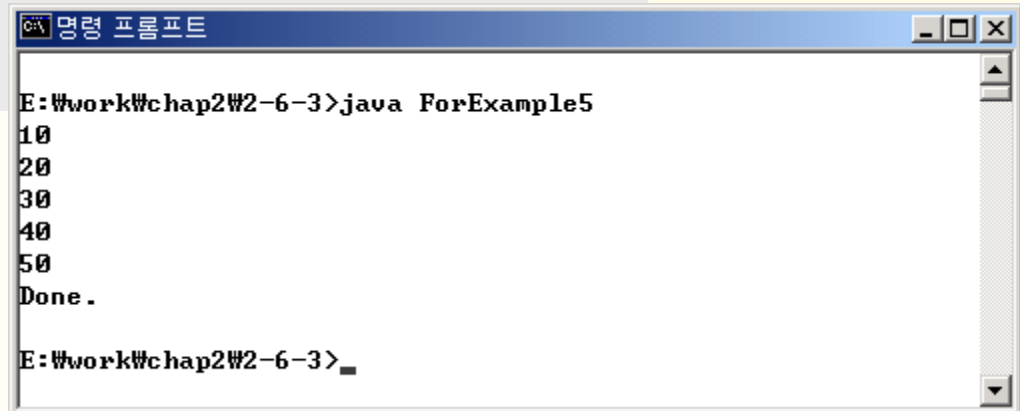
## 06. 반복문

### 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>
```

## 06. 반복문

### break 문

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

```
break;
```

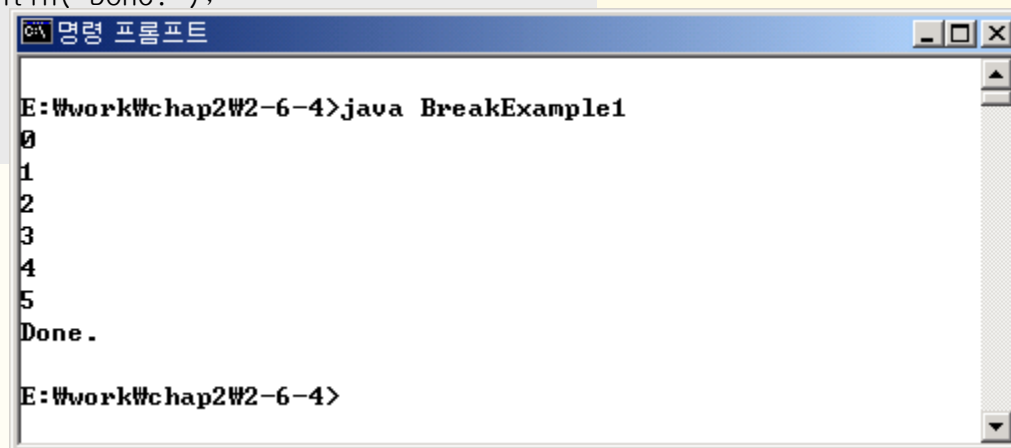
## 06. 반복문

### 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>
```

## 06. 반복문

## 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 < 4; col++) {
5                      System.out.print(" ");
6                      if ((row + col) % 2 == 0)
7                          break;
8                  }
9              }
10             System.out.println();
11         }
12     }

```





```
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>
Done.
E:\work\chap2\2-6-4>
```

## 06. 반복문

### 중첩된 반복문과 break 문

- 중첩된 반복문을 한꺼번에 빠져나가는 방법
  - 반복문에 라벨을 붙인다
  - 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

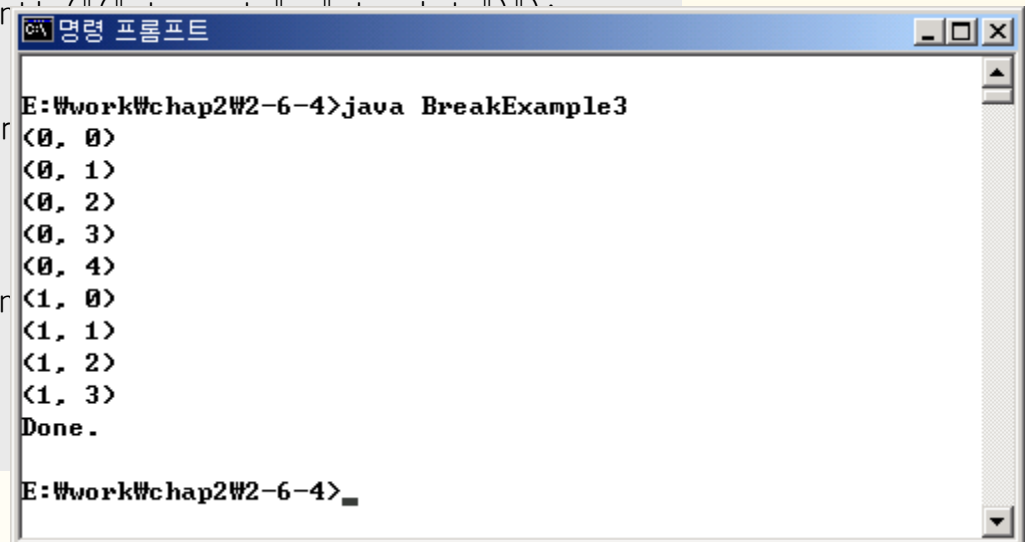
라벨을 지정한 break 문

## 06. 반복문

### 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.print(" ");  
                if ((row == 1) && (col == 0))  
                    break outerLoop;  
                System.out.print("<0, " + col + ">");  
                if (col == 4) System.out.println();  
            }  
        }  
        System.out.println("Done.");  
    }  
}
```



```
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>
```

## 06. 반복문

### continue 문

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

```
continue;
```



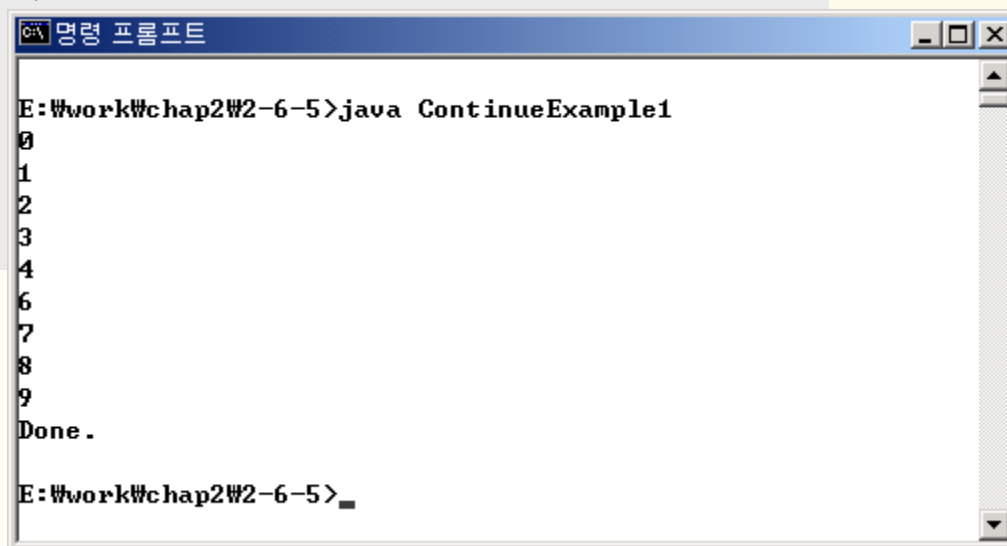
## 06. 반복문

### 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      }  
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>
```

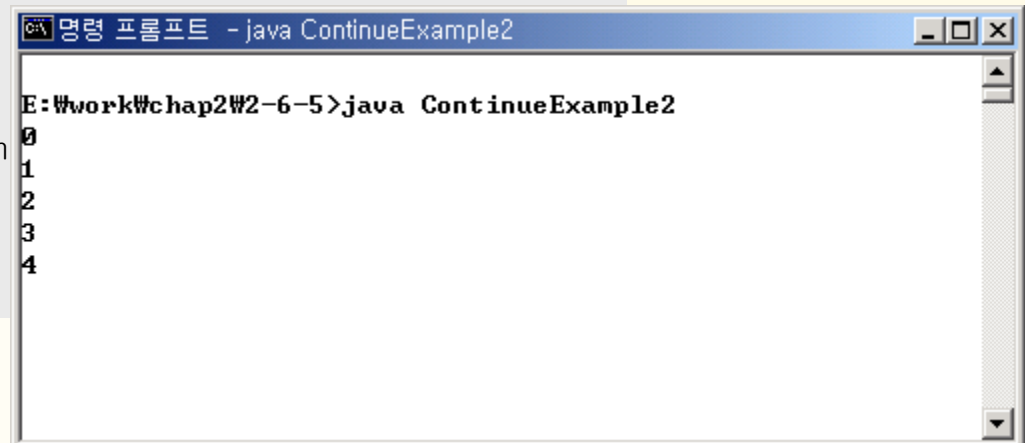
## 06. 반복문

### 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(cnt);  
11         }  
12     }
```

cnt가 5이면 while 문의 다음번 반복 과정을 계속합니다.



## 06. 반복문

### 중첩된 반복문과 continue 문

- 중첩된 반복문의 바깥쪽 반복을 계속하는 방법
  - 반복문에 라벨을 붙인다
  - continue 문에 라벨을 지정한다

```
loop:
    for (int cnt = 0; cnt < 100; cnt++) {
        System.out.println(cnt);
        if (cnt == 5)
            continue loop;
    }
```

for 문에 붙여진 라벨

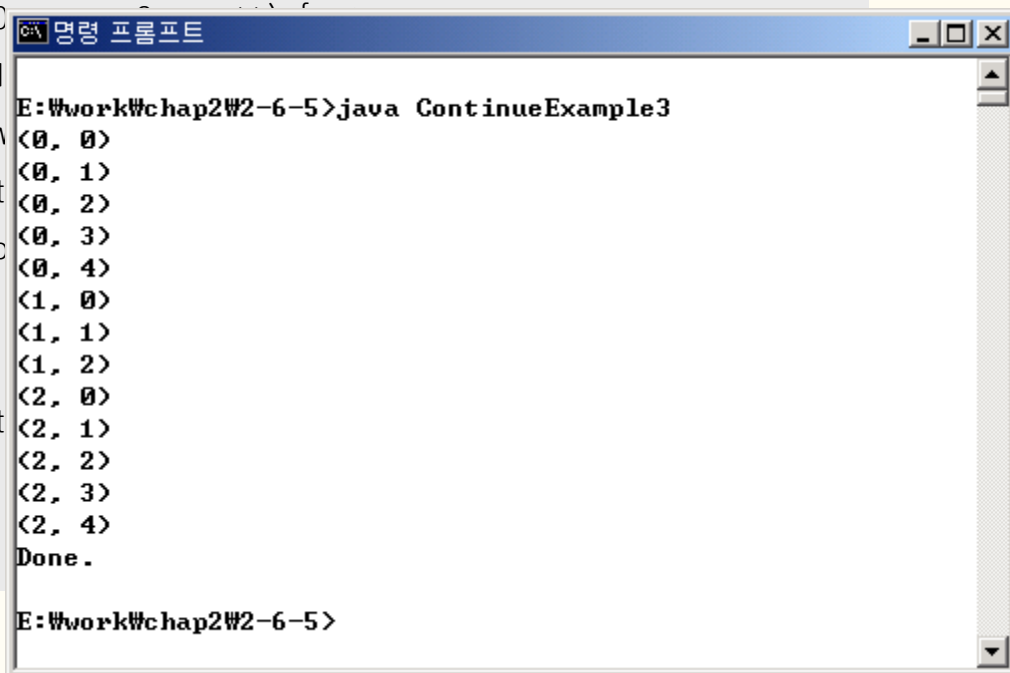
라벨을 지정한 continue 문

## 06. 반복문

### 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 + col) % 2 == 0)  
7                          continue outerLoop;  
8                      System.out.print(row + " " + col + " ");  
9                  }  
10             }  
11             System.out.println();  
12         }  
13     }
```

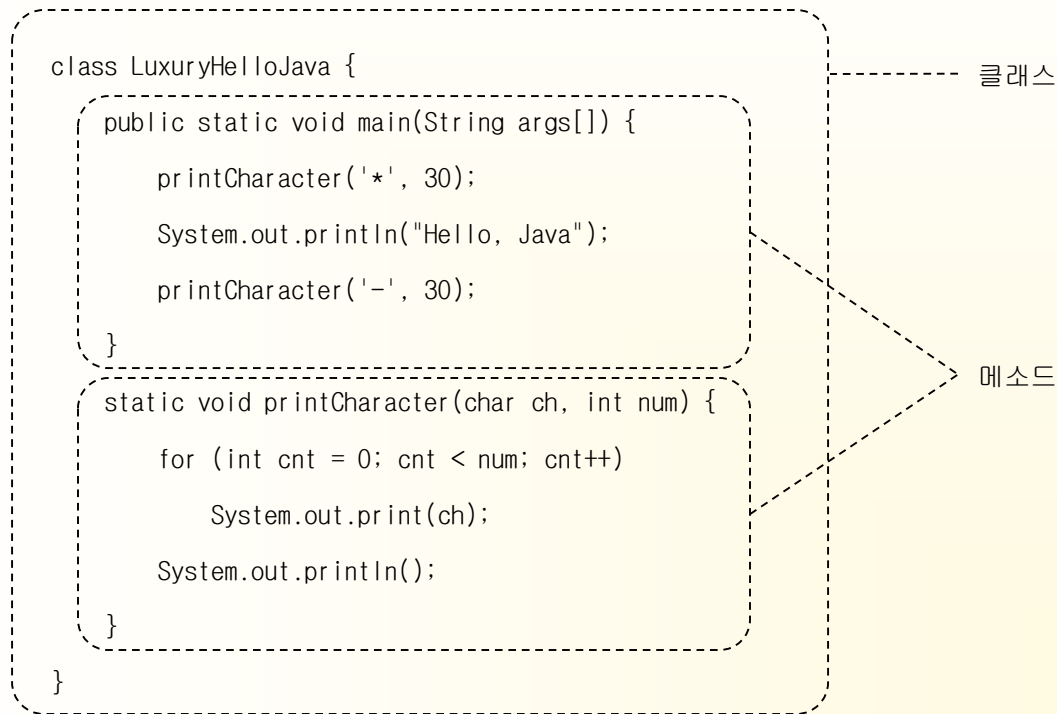


```
E:\work\chap2\2-6-5>java ContinueExample3  
<0, 1>  
<0, 3>  
<1, 0>  
<1, 2>  
<1, 4>  
<2, 0>  
<2, 2>  
<2, 4>  
Done.  
E:\work\chap2\2-6-5>
```

## 07. 메소드 호출문

### 메소드 호출문

- 여러 개의 메소드가 포함된 클래스



- main이 아닌 메소드는 자동으로 실행되지 않음

## 07. 메소드 호출문

### 메소드 호출문

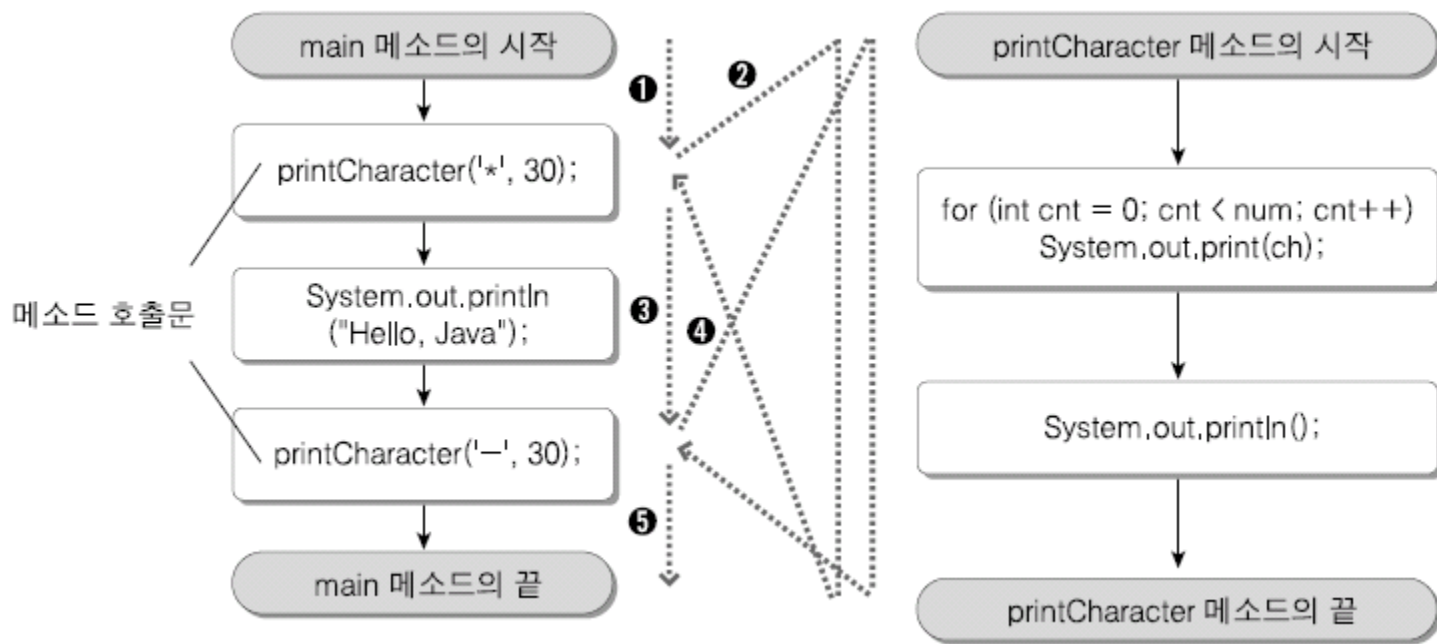
- 여러 개의 메소드가 포함된 클래스

```
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

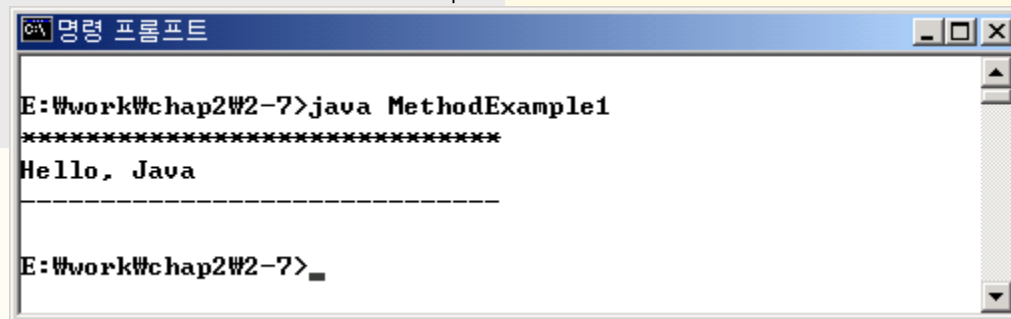
[예제 2-44] 메소드 호출 예

```
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



```
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]

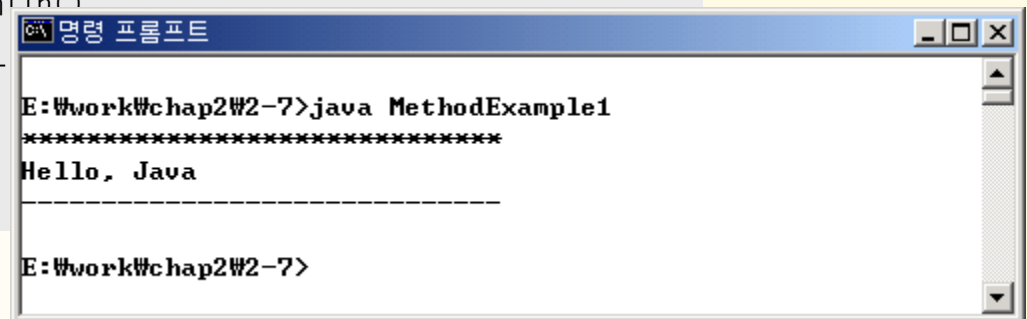
```
return;
```

## 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; ----- 리턴 값이 없는 메소드임을 표시하는 키워드  
12     }  
13 }
```

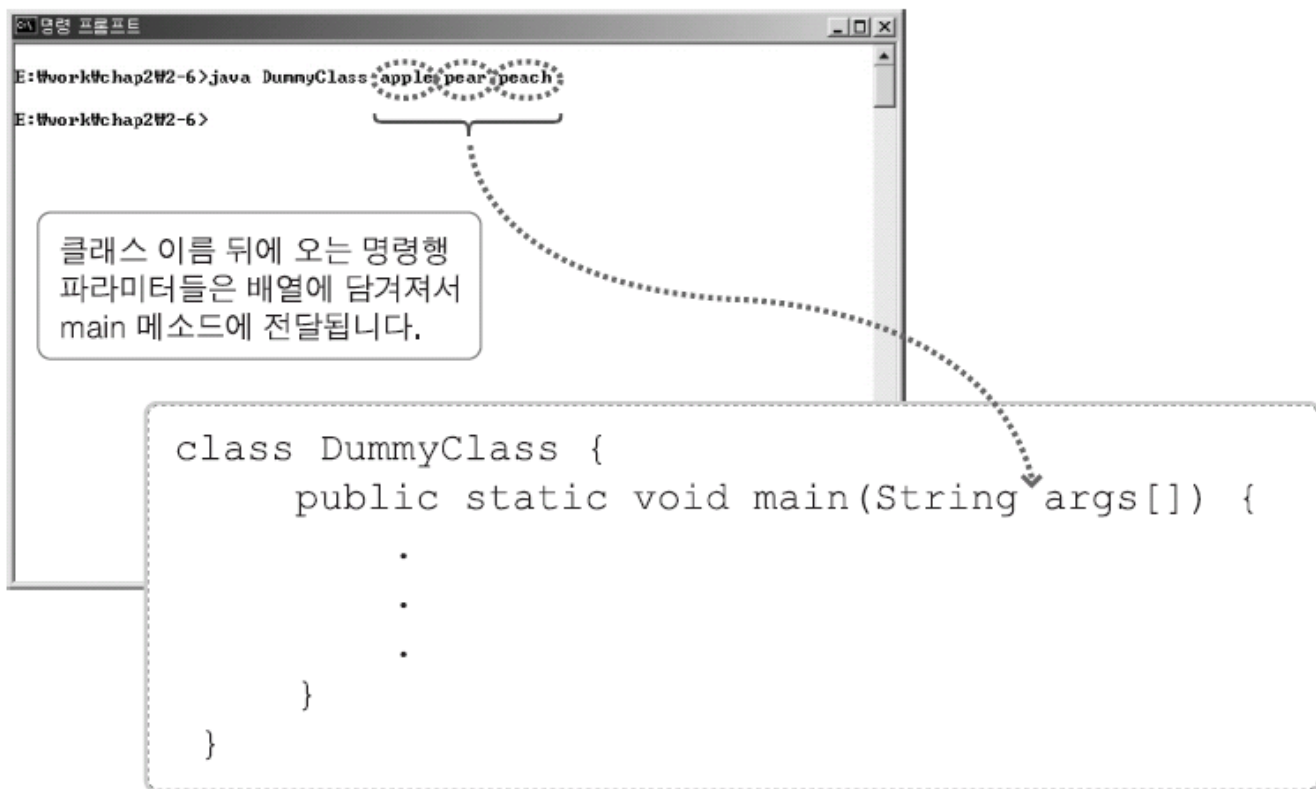


```
명령 프롬프트  
E:\work\chap2\2-7>java MethodExample1  
*****  
Hello, Java  
-----  
E:\work\chap2\2-7>
```

## 07. 메소드 호출문

### main 메소드의 파라미터

- main 메소드의 파라미터 변수가 하는 일





## 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) }  
4              System.out.println(str);  
5          System.out.println("args.length=" + args.length);  
6      }  
7  }
```

args 배열의 항목 값을 순서대로 출력합니다.

args 배열의 항목 수를 출력합니다.

## 07. 메소드 호출문

### main 메소드의 파라미터

[예제 2-47] 명령행 파라미터를 출력하는 프로그램 (실행 결과)

```
명령 프롬프트
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.

X  
O  
공백문자를 포함하는 값을 전달하려면 큰 따옴표로 묶어서 표시해야 합니다.

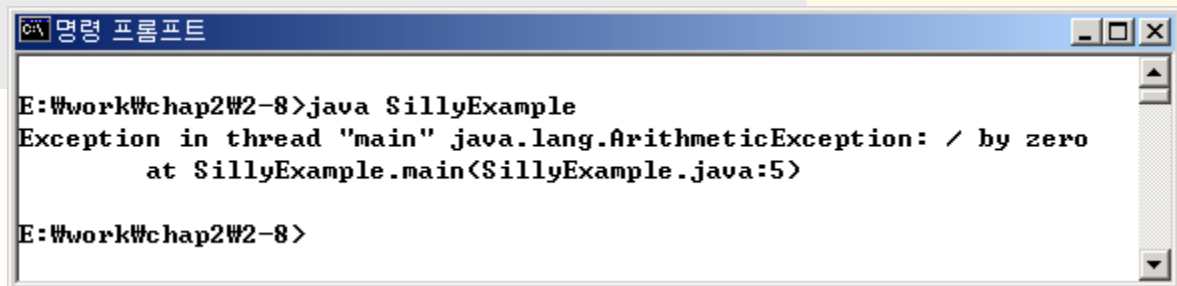
## 08. 익셉션 처리에 사용되는 try 문

### 익셉션이란?

- 익셉션(exception, 예외) : 자바에서 에러를 지칭하는 용어

[예제 2-48] 익셉션을 발생하는 프로그램의 예

```
1    class SillyExample {  
2        public static void main(String args[]) {  
3            int a = 3, b = 0;  
4            int result;  
5            result = a / b;  
6            System.out.println(result);  
7            System.out.println("Done.");  
8        }  
9    }
```



```
명령 프롬프트  
E:\work\chap2\2-8>java SillyExample  
Exception in thread "main" java.lang.ArithmeticException: / by zero  
    at SillyExample.main(SillyExample.java:5)  
E:\work\chap2\2-8>
```

## 08. 익셉션 처리에 사용되는 try 문

### try 문

- try 문 : 익셉션 처리에 사용되는 명령문
- 기본 형식

```
try
    try블록
catch (익셉션타입 익셉션변수) } ----- catch 절(catch clause). 반복 가능
    catch블록
finally } ----- finally 절(finally clause). 생략 가능
    finally블록
```

- try 블록, catch 블록, finally 블록은 모두 중괄호로 둘러싸인 블록이어야 함

## 08. 익셉션 처리에 사용되는 try 문

### try 문

[예제 2-49] try 문의 사용 예

```
1  class SmartExample {
2      public static void main(String args[]) {
3          int a = 3, b = 0;
4          int result;
5          try {
6              result = a / b;
7              System.out.println(result);
8          }
9          catch (java.lang.ArithmeticException e) {
10             System.out.println("잘못된 연산입니다.");
11         }
12         finally {
13             System.out.println("Done.");
14         }
15     }
16 }
```

이 부분을 실행하다가  
익셉션이 발생하면

이 부분을 실행합니다.

이 부분은 익셉션 발생 유무와



```
명령 프롬프트
E:\work\chap2\2-8>java SmartExample
잘못된 연산입니다.
Done.
E:\work\chap2\2-8>
```

## 08. 익셉션 처리에 사용되는 try 문

### try 문

[예제 2-50] finally 블록이 없는 try 문의 예

```
1  class SmartExample {  
2      public static void main(String args[]) {  
3          int a = 3, b = 0;  
4          int result;  
5          try {  
6              result = a / b;  
7              System.out.println(result);  
8          }  
9          catch (java.lang.ArithmeticException e) {  
10             System.out.println("잘못된 연산입니다.");  
11         }  
12         System.out.println("Done.");  
13     }  
14 }
```



```
명령 프롬프트  
E:\work\chap2\2-8>java SmartExample  
잘못된 연산입니다.  
Done.  
E:\work\chap2\2-8>
```

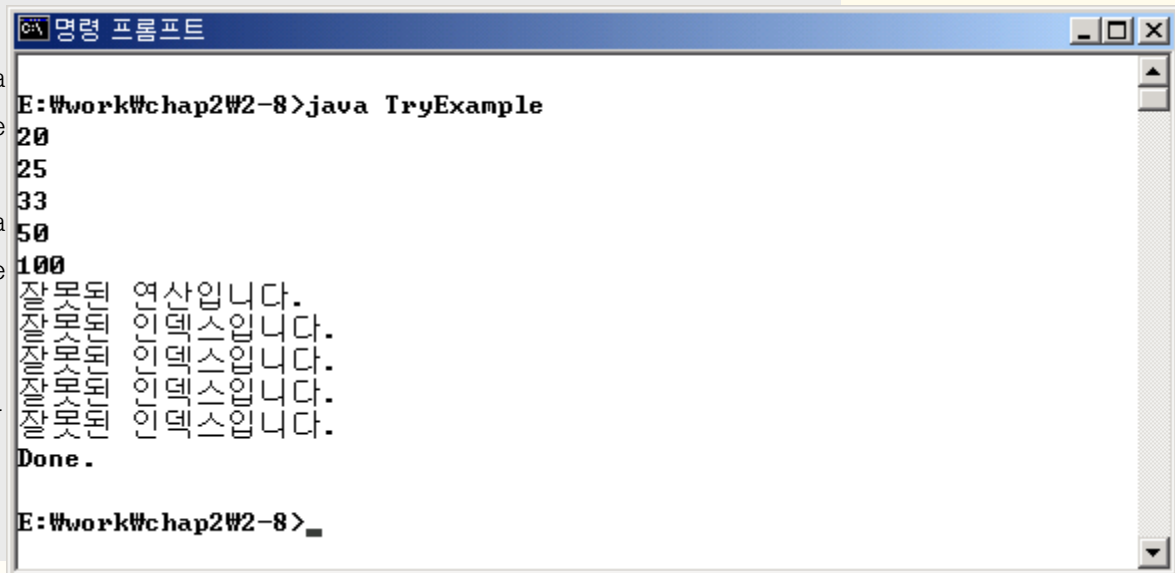
## 08. 익셉션 처리에 사용되는 try 문

### try 문

[예제 2-51] 두 종류의 익셉션을 처리하는 try 문의 예

```
1  class TryExample {
2      public static void main(String args[]) {
3          int divisor[] = { 5, 4, 3, 2, 1, 0 };
4          for (int cnt = 0; cnt < 10; cnt++) {
5              try {
6                  int share = 100 / divisor[cnt];
7                  System.out.println(share);
8              }
9              catch (java.lang.ArithmeticException e) {
10                 System.out.println("잘못된 연산입니다.");
11             }
12             catch (java.lang.IndexOutOfBoundsException e) {
13                 System.out.println("잘못된 인덱스입니다.");
14             }
15         }
16         System.out.println("Done.");
17     }
18 }
```

이 명령문은 0으로 나눌 때와 인덱스를 잘못 썼을 때 익셉션을 발생할 수 있습니다.



```
명령 프롬프트
E:\work\chap2\2-8>java TryExample
20
25
33
50
100
잘못된 연산입니다.
잘못된 인덱스입니다.
잘못된 인덱스입니다.
잘못된 인덱스입니다.
잘못된 인덱스입니다.
Done.
E:\work\chap2\2-8>
```