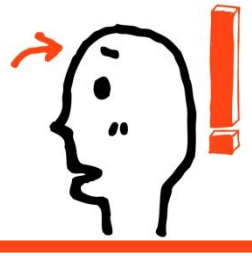
Android Java

4장 실행 흐름의 컨트롤(조건문 & 반복문)

자바에서 사용하는 조건문 & 반복문



01 논리적인 흐름 제어를 위한 if 조건문 만들기

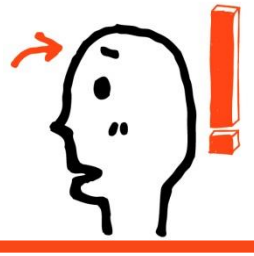
02 논리 흐름 제어를 위한 또 다른 방법, switch-case 조건문

03 for, while, do-while 반복문 비교하기

04 continue와 break 키워드로 흐름 제어하기

05 반복문의 중첩

논리적인 흐름 제어를 위한 if 조건문 만들기



■ 기본 if 문

- 조건이 참일 때와 거짓일 때 각각 다른 일을 수행

```
if (조건식)
    실행할 문장;
```

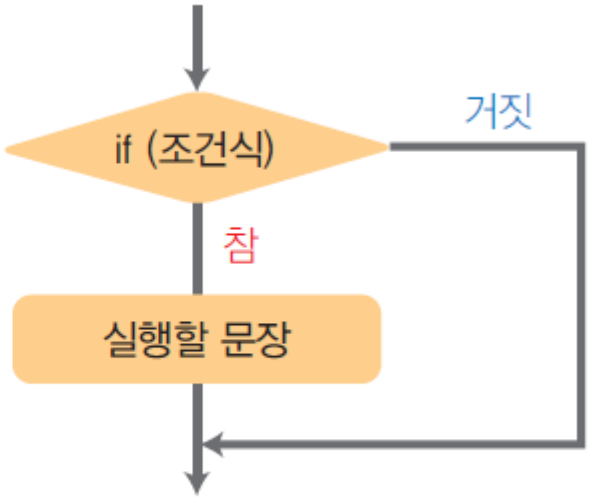
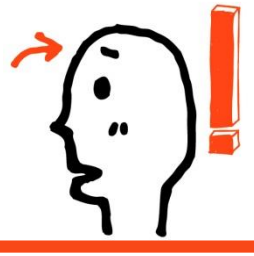


그림 5-1 if 문의 형식과 순서도

논리적인 흐름 제어를 위한 if 조건문 만들기



실습 5-1 기본 if 문 사용 예 1

```
01 public class Ex05_01 {
02    public static void main(String[] args) {
03       int a = 99;
04
05       if (a < 100)          ----- a가 100보다 작으므로 참이다.
06          System.out.printf("100보다 작군요..₩n");
07    }
08 }
```

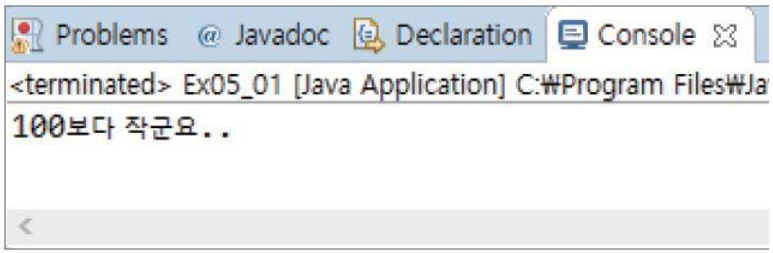


그림 5-2 실행 결과

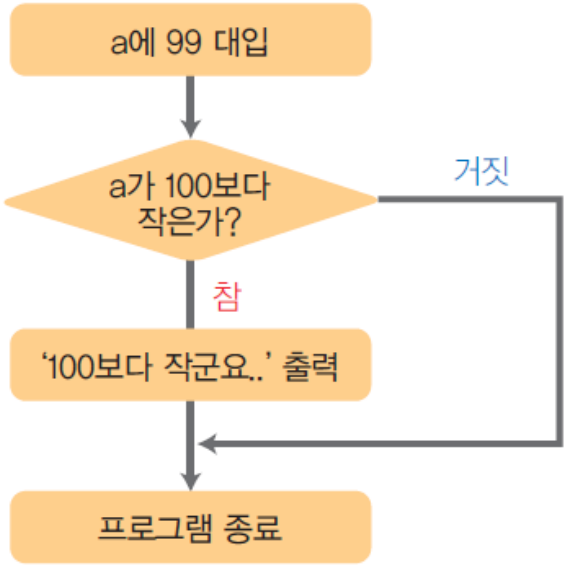
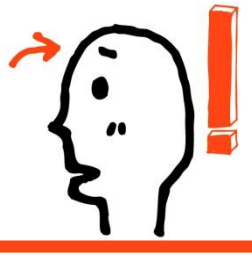


그림 5-3 [실습 5-1]의 실행 과정

논리적인 흐름 제어를 위한 if 조건문 만들기



실습 5-2 기본 if 문 사용 예 2

```
01 public class Ex05_02 {
02    public static void main(String[] args) {
03       int a = 200;
04
05       if (a < 100)
06          System.out.printf("100보다 작군요..₩n");
07          System.out.printf("거짓이므로 이 문장은 안보이겠죠?₩n");
08
09       System.out.printf("프로그램 끝! ₩n");
10    }
11 }
```

6행이 참이면 수행할 것으로 예상된다.

9행이 거짓이면 6, 7행을 수행하지 않고 9행을 수행할 것으로 예상된다.

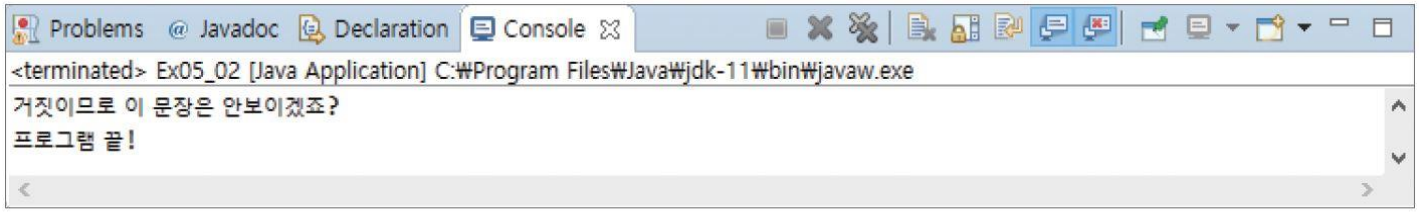
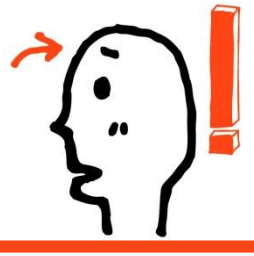


그림 5-4 실행 결과

논리적인 흐름 제어를 위한 if 조건문 만들기



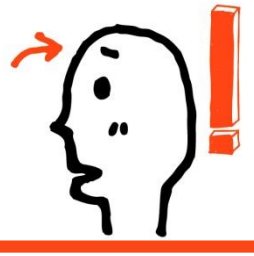
- 줄바꿈 함정 – [실습5-2]에서 7행이 실행 된 이유

  다음과 같이 줄바꿈을 수정하여 실행

```
if (a < 100)
    System.out.printf("100보다 작군요..₩n");

System.out.printf("거짓이므로 이 문장은 안보이겠죠?₩n");
System.out.printf("프로그램 끝! ₩n");
```

즉 5행의 조건식이 거짓이므로 그 아래 문장인 6행만 건너뛰고 7행부터 실행함

논리적인 흐름 제어를 위한 if 조건문 만들기



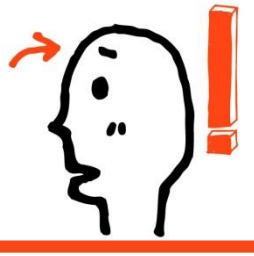
실습 5-3 기본 if 문 사용 예 3

```
01 public class Ex05_03 {
02    public static void main(String[] args) {
03       int a = 200;
04
05       if (a < 100) {
06          System.out.printf("100보다 작군요..₩n");
07          System.out.printf("거짓이므로 앞의 문장은 안보이겠죠?₩n");
08       }
09
10       System.out.printf("프로그램 끝! ₩n");
11    }
12 }
```

5행이 참이면 중괄호로 묶인 부분이 모두 수행된다.

그림 5-5 실행 결과

논리적인 흐름 제어를 위한 if 조건문 만들기



■ if~else 문

```
if (조건식)
    실행할 문장 1;
else
    실행할 문장 2;
```

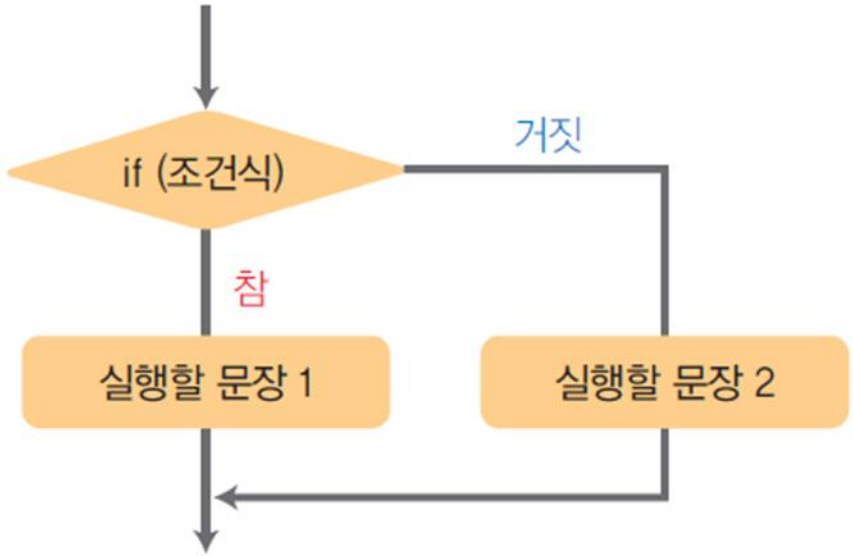
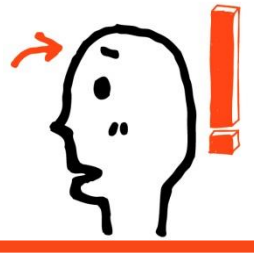


그림 5-6 if~else 문의 형식과 순서도

논리적인 흐름 제어를 위한 if 조건문 만들기



실습 5-4 if~else 문 사용 예

```
01 public class Ex05_04 {
02    public static void main(String[] args) {
03       int a = 200;
04
05       if (a < 100)
06          System.out.printf("100보다 작군요..₩n");----- 5행이 참이면(a가 100보다 작으면) 실행한다.
07       else
08          System.out.printf("100보다 크군요..₩n");----- 5행이 거짓이면(a가 100보다 크거나 같으면) 실행한다.
09    }
10 }
```

Problems @ Javadoc Declaration Console

```
<terminated> Ex05_04 [Java Application] C:₩Program Files₩Ja
100보다 크군요..
```

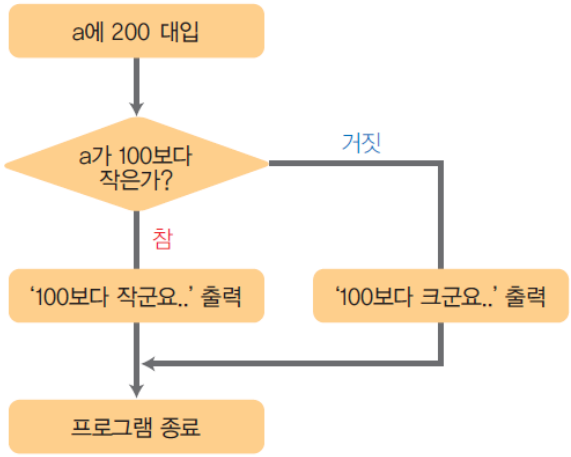
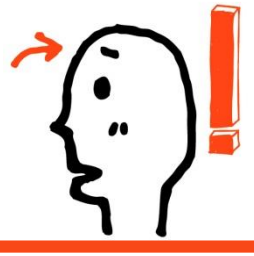


그림 5-8 [실습 5-4]의 실행 과정

논리적인 흐름 제어를 위한 if 조건문 만들기



실습 5-5 중괄호를 사용한 if~else 문 사용 예 1

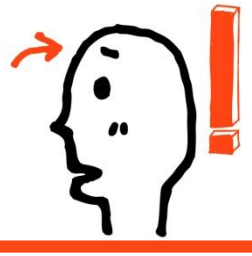
```
01 public class Ex05_05 {
02    public static void main(String[] args) {
03       int a = 200;
04
05       if (a < 100) {
06          System.out.printf("100보다 작군요..₩n");
07          System.out.printf("참이면 이 문장도 보이겠죠?₩n");
08       } else {
09          System.out.printf("100보다 크군요..₩n");
10          System.out.printf("거짓이면 이 문장도 보이겠죠?₩n");
11       }
12
13       System.out.printf("프로그램 끝! ₩n");
14    }
15 }
```

5행이 참이면(a가 100보다 작으면) 실행한다.

5행이 거짓이면(a가 100보다 크거나 같으면) 실행한다.

```
Problems  @ Javadoc  Declaration
<terminated> Ex05_05 [Java Application] C:
100보다 크군요..
거짓이면 이 문장도 보이겠죠?
프로그램 끝!
```

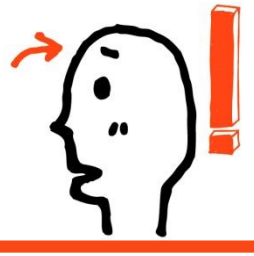
논리적인 흐름 제어를 위한 if 조건문 만들기



if 조건문

- 상황에 따라서 조건식이 많아지는 경우에는 else if 또는 else 키워드를 추가해서 사용
- else if 키워드를 사용할 때는 조건식이 필요하지만 else는 조건식이 필요 없음
- if와 else if에서 선언된 실행문은 각 키워드와 연결되는 조건식이 참인 경우에 실행되지만 else의 실행문은 앞서 선언한 모든 조건문이 false인 경우 실행

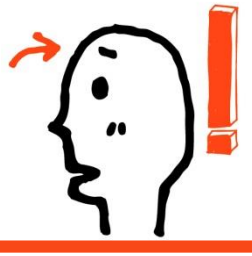
논리적인 흐름 제어를 위한 if 조건문 만들기



if 조건문

```
if(BooleanExpressionA)
{
    statement1;  //BooleanExpressionA가 true인 경우에만 실행되는 구문
}
else if(BooleanExpressionB)
{
    statement2;  //BooleanExpressionB가 true인 경우에만 실행되는 구문
}
else if(BooleanExpressionC)
{
    statement3;  //BooleanExpressionC가 true인 경우에만 실행되는 구문
}
else
{
    statement4;  //위의 모든 조건식이 false인 경우 실행되는 구문
}
```

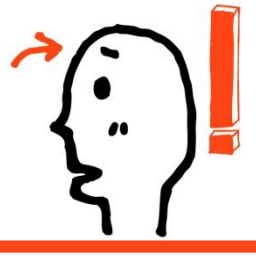
논리적인 흐름 제어를 위한 if 조건문 만들기



if 조건문

```
if(BooleanExpressionA)
{
    statement1;  //BooleanExpressionA가 true인 경우에만 실행되는 구문
}
else if(BooleanExpressionB)
{
    statement2;  //BooleanExpressionB가 true인 경우에만 실행되는 구문
}
else if(BooleanExpressionC)
{
    statement3;  //BooleanExpressionC가 true인 경우에만 실행되는 구문
}
else
{
    statement4;  //위의 모든 조건식이 false인 경우 실행되는 구문
}
```

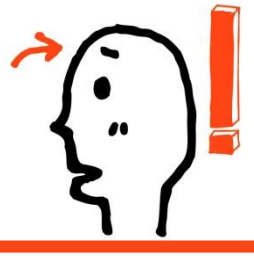
논리적인 흐름 제어를 위한 if 조건문 만들기



```
public static void main(String[] args) {
    int num = 120;

    if(num < 0)
        System.out.println("0 미만");
    else if(num < 100)
        System.out.println("0 이상 100 미만");
    else
        System.out.println("100 이상");
}
```

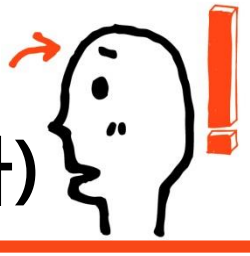
```
명령 프롬프트
C:₩JavaStudy>java IEIE
100 이상

C:₩JavaStudy>_
```

논리적인 흐름 제어를 위한 if 조건문 만들기



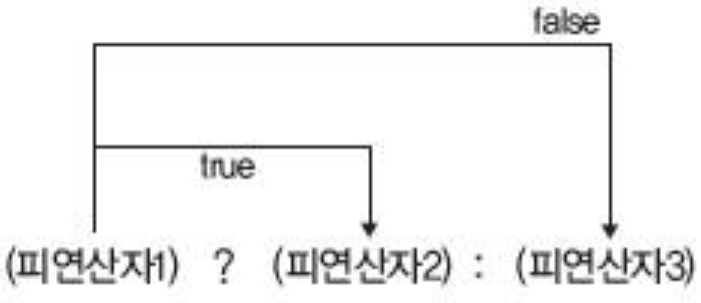
```
if (a >= 90)
    System.out.printf ("A");
else if (a >= 80)
    System.out.printf ("B");
else if (a >= 70)
    System.out.printf ("C");
else if (a >= 60)
    System.out.printf ("D");
else
    System.out.printf ("F");
```

if~else와 유사한 성격의 조건 연산자(3항 연산자)



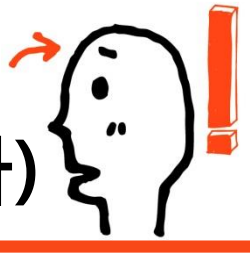
조건 연산자(3항 연산자)

- **3항 연산자란?** 피연산자의 개수가 3개인 것을 의미
- 3항 연산자는 조건 비교를 하여 그 결과를 반환하는 연산자



- 3항 연산자는 두 개의 연산자인 **물음표(?)**와 **콜론(:)** 그리고 **3개의 피연산자**로 구성
- 피연산자1의 결과값이 true이면 피연산자2의 값 반환,
- 피연산자1의 값이 false이면 피연산자 3의 값 반환

if~else와 유사한 성격의 조건 연산자(3항 연산자)

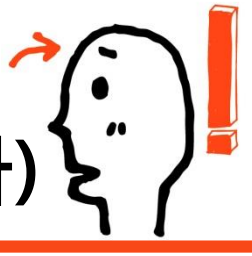


조건식을 간단하게 해주는 3항 연산자

3항 연산자	사용예	설명
case 1	boolean isRound = false; boolean rt = (isRound) ? true : false;	isRound 변숫값에 따라서 변수 rt에 Boolean 값이 설정된다. isRound 값이 true이면 rt도 true, isRound 값이 false이면 rt도 false이다.
case 2	boolean isEnable = true; int rt = (isEnable) ? 1 : 2;	case 1과 비슷하지만 isEnable 변숫값에 따라서 int rt 변수에 다른 값이 설정된다. isEnable이 true이면 rt는 1이 할당되고 isEnable이 false이면 rt는 2가 된다.
case 3	int a = 3; boolean rt = (a > 4) ? false : true;	첫 번째 피연산자 위치에 있는 연산식의 결과에 따라서 변수 rt에 값이 설정된다. 변수 a가 4를 초과하면 rt에 false를, 변수 a가 4 이하이면 rt에는 true가 설정된다.
case 4	int a = 3; int rest = (a > 0) ? (a % 2) : 0;	case 3과 비슷하게 첫 번째 연산식의 결과에 따라서 rest 변수의 값이 변경된다. 변수 a가 0을 초과하면 a % 2의 결과값이 rest 변수에 입력된다. a가 0 이하의 값이면 rest 변수에는 0이 입력된다.

△ 표 3-11 3항 연산자를 활용한 예제

if~else와 유사한 성격의 조건 연산자(3항 연산자)



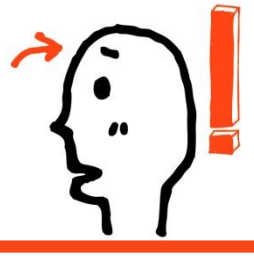
조건식을 간단하게 해주는 3항 연산자

코드 3-8 package com.gilbut.chapter3;

```
1   public class TenaryOperator
2   {
3      public static void main(String[] args)
4      {
5         if (args == null || args.length != 2)
6         {
7            System.out.println("help : java TenaryOperator number1 number2");
8            return;
9         }
10
11        int a = Integer.parseInt(args[0]);
12        int b = Integer.parseInt(args[1]);
13        int c = (a > b) ? (a - b) : (b - a);
14
15
16        System.out.println((a > b) ? "A > B" : "B >= A");
17        System.out.println("difference : " + c);
18     }
19  }
```

3항 연산자는 여러분이 실제로 개발할 때 많이 사용하는 구문으로 앞으로 배울 if-else 구문과 비슷한 역할을 합니다.

간단한 계산기 프로그램



- 중복 if문을 활용하여 두 수의 +, -, *, /, % 연산을 수행하는 프로그램을 작성해보자.

Problems @ Javadoc Declaration Console

<terminated> test_test [Java Application] C:₩Program Files₩Java₩jre1.8.0_65₩bin₩javaw.exe (2016. 10. 21. 오전 8:08:18)

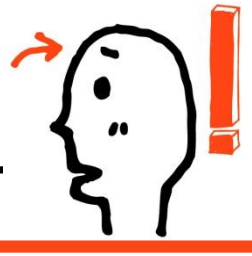
```
첫번째 수를 입력하세요 : 12
계산할 연산자를 입력하세요 : /
두번째 수를 입력하세요 : 5
12/5=2.4
```

Problems @ Javadoc Declaration Console

<terminated> test_test [Java Application] C:₩Program Files₩Java₩jre1.8.0_65₩bin₩javaw.exe (2016. 10. 21. 오전 8:12:44)

```
첫번째 수를 입력하세요 : 12
계산할 연산자를 입력하세요 : &
두번째 수를 입력하세요 : 5
연산자를 잘못 입력했습니다.
```

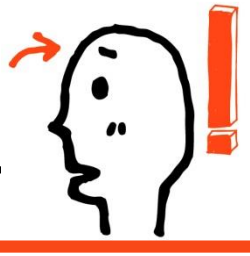
논리 흐름 제어를 위한 또 다른 방법, switch-case 조건문



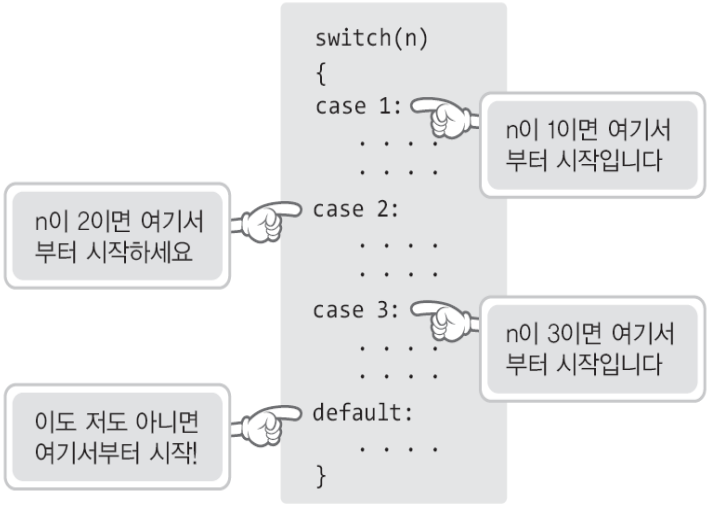
switch 구문의 사용 방법

- **switch-case** 조건문과 관련된 키워드는 **switch**와 **case** 그리고 **default**
- switch, case 이 두 개의 키워드는 반드시 같이 사용되어야만 정상적으로 동작
- **조건식**과 **실행 구문**으로 구성
- 조건식은 괄호로 둘러싸여 있고 실행 구문은 중괄호로 둘러싸여 있음
  (실행 구문을 한 줄로 표현할 수 있다면 중괄호는 생략 가능)

논리 흐름 제어를 위한 또 다른 방법, switch-case 조건문

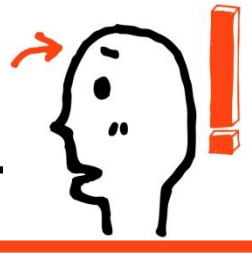


switch 구문의 사용 방법




```
switch(selectExpression)
{
    case valueA :
        statementA  //조건식이 valueA와 같을 경우 실행되는 구문
        break;
    case valueB :
    case valueC :
        statementB  //조건식이 valueB 또는 valueC와 같을 경우 실행되는 구문
        break;
    default :
        statementC  //조건식이 위에 열거한 값과 같지 않는 경우 실행되는 구문
}
```

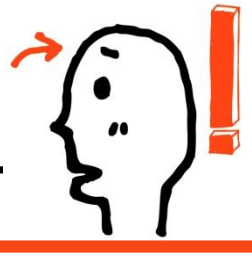
논리 흐름 제어를 위한 또 다른 방법, switch-case 조건문



switch 구문의 사용 방법

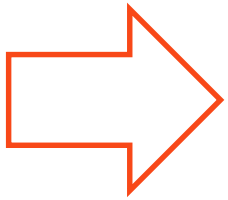
- switch-case 기본 문법에서 조건식에는 정수나 문자열을 나타내는 조건식 혹은 변수가 위치
- 사용 가능한 데이터형은 **byte, short, char, int, String**
- case 구문 바로 뒤에는 조건식에서 반환할 수 있는 데이터를 입력
- 조건식에서 반환하는 값과 일치하는 값이 case 키워드에 있다면 그 case 키워드와 연결된 실행 구문이 실행
- 실행 구문을 끝내기 위해서는 반드시 **break** 키워드를 사용

논리 흐름 제어를 위한 또 다른 방법, switch-case 조건문



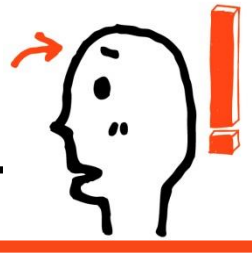
switch 구문의 사용 방법

```
switch (30-28)
{
    case 0:
        System.out.println("Result 0")
        break;
    case 1:
    case 2:
        System.out.println("Result 1 or 2");
    case 3:
        System.out.println("Result 3");
        break;
    default;
        System.out.println("Result unknown")
}
```



- Result 1 or 2
- Result 3

논리 흐름 제어를 위한 또 다른 방법, switch-case 조건문



switch 구문의 사용 방법

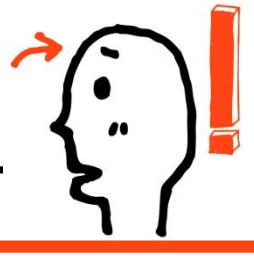
1. switch 바로 뒤에 위치한 조건식 (30 – 28)을 실행

2. 첫 번째 case 바로 뒤에 위치한 값 0과 30 - 28 조건식의 결과인 정수 2와 비교, 첫 번째 case값 0과 30 - 28의 결과값 2는 서로 같지 않으므로 두 번째 case 구문으로 이동

3. case 2와 조건식의 결과값이 일치하기 때문에 System.out.println("Result 1 or 2"); 구문을 실행, 그리고 break; 구문을 만날 때까지 계속 실행 구문을 실행

4. case 3에서 break; 구문을 만나면 switch-case 구문에서 탈출

논리 흐름 제어를 위한 또 다른 방법, switch-case 조건문



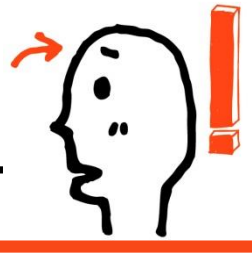
```
public static void main(String[] args) {
    int n = 3;

    switch(n) {
    case 1:
        System.out.println("Simple Java");
        break;
    case 2:
        System.out.println("Funny Java");
        break;
    case 3:
        System.out.println("Fantastic Java");
        break;
    default:
        System.out.println("The best programming language");
    }

    System.out.println("Do you like Java?");
}
```

```
명령 프롬프트
C:₩JavaStudy>java SwitchBreak
Fantastic Java
Do you like Java?

C:₩JavaStudy>
```

switch + break 구성의 다른 예

```
switch(n) {
case 1:
case 2:
case 3:
    System.out.println("case 1, 2, 3");
    break;
default:
    System.out.println("default");
}
```

논리 흐름 제어를 위한 또 다른 방법, switch-case 조건문



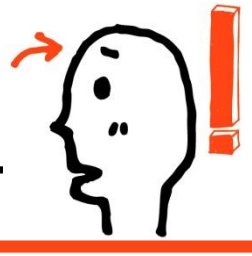
실습 5-9 switch~case 문 사용 예 1

```
01 import java.util.Scanner;
02
03 public class Ex05_09 {
04    public static void main(String[] args) {
05       Scanner s = new Scanner(System.in);
06       int a;
07
08       System.out.printf("1 ~ 4 중에 선택하세요 : ");
09       a = s.nextInt( );
10
11       switch (a) {
12       case 1:
13          System.out.printf("1을 선택했다₩n");
14          break;
```

11 ----- 입력한 a 값에 따라서 분기한다.

12~14 a가 1이면 13행을 수행하고, 14행에서 switch 블록을 빠져나간다.

논리 흐름 제어를 위한 또 다른 방법, switch-case 조건문



```
15      case 2:
16         System.out.printf("2를 선택했다₩n");
17         break;
18      case 3:
19         System.out.printf("3을 선택했다₩n");
20         break;
21      case 4:
22         System.out.printf("4를 선택했다₩n");
23         break;
24      default:
25         System.out.printf("이상한걸 선택했다.₩n");
26      }
27   }
28 }
```

a가 1, 2, 3, 4에 해당되지 않을 경우에 수행한다.

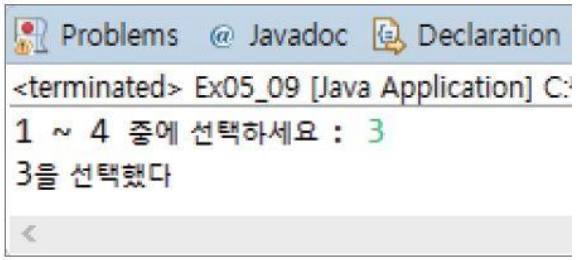
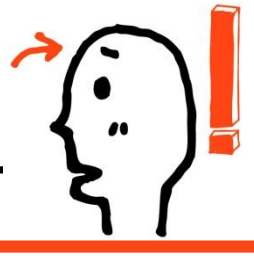


그림 5-16 실행 결과

논리 흐름 제어를 위한 또 다른 방법, switch-case 조건문



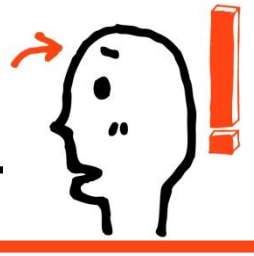
- a가 3일 때

  switch~case 문의 흐름도

```
switch (a)
{
case 1:
    printf("1을 선택했다₩n");
    break;
case 2:
    printf("2를 선택했다₩n");
    break;
case 3:
    printf("3을 선택했다₩n");
    break;
case 4:
    printf("4를 선택했다₩n");
    break;
default:
    printf("이상한걸 선택했다.₩n");
}
```

그림 5-17 [실습 5-9]에서 a가 3일 때 switch~case 문의 흐름도

논리 흐름 제어를 위한 또 다른 방법, switch-case 조건문



- [실습 5-9]에서 break를 빼고 실행

```
switch (a) {
case 1:
   System.out.printf("1을 선택했다₩n");
case 2:
   System.out.printf("2를 선택했다₩n");
case 3:
   System.out.printf("3을 선택했다₩n");
case 4:
   System.out.printf("4를 선택했다₩n");
default:
   System.out.printf("이상한걸 선택했다.₩n");
}
```

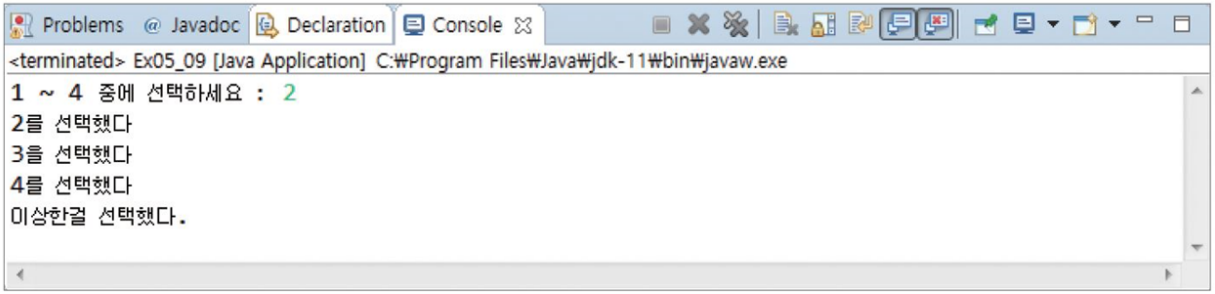
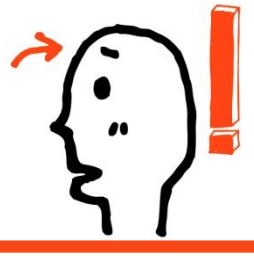


그림 5-18 실행 결과

for, while, do-while 반복문 비교하기



for 반복문

- **반복문(루프(loop)문)** : 반복된 작업을 처리하는데 사용
- 각각 반복할 조건을 설정할 수 있는 부분과 반복할 실행 구문으로 구성
- 자바에서 제공하는 세 가지 반복문 중에서 가장 많이 사용하는 구문은 **for 구문**
- for 키워드 다음에 위치한 괄호에는 세 개의 식들이 위치
- 각각의 결과에 의해서 반복 작업을 실행
- 괄호 안에 위치한 세 개의 식들은 세미콜론(;)으로 **초기식, 조건식, 반복식**으로 구분

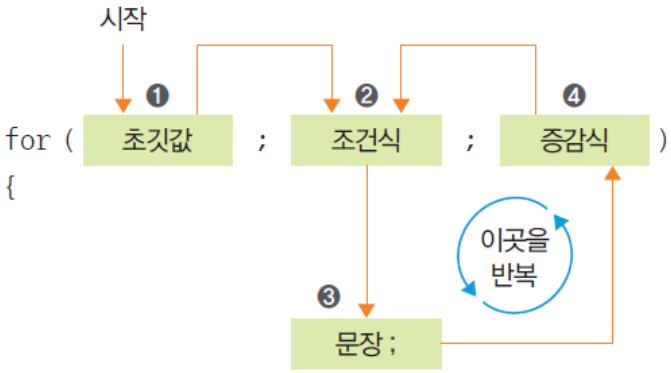
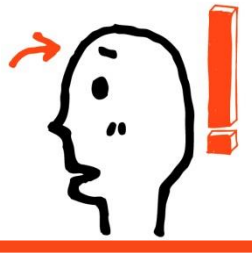


그림 6-3 for 문의 실행 순서

for, while, do-while 반복문 비교하기



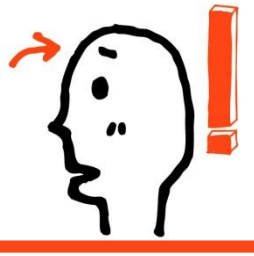
for 반복문

- for 루프문을 실행하기 전 가장 먼저 **초기식**을 실행
- **조건식**의 반환값이 true라면 실행 구문을 한 번 실행
- 그리고 마지막으로 **반복식**을 실행
- 1번의 반복이 종료되면 for 구문은 다시 조건식을 확인하여 결과값에 따라서 실행 구문의 실행 여부를 결정

초기식	for 구문을 시작하기 앞서 가장 먼저 실행하는 식
조건식	조건식이 true이면 반복해서 실행 구문을 실행한다. 즉, for 구문의 반복 조건을 제어하는데 사용되는 식
반복식	실행 구문이 실행되고 실행되는 식

△ 표 4-1 for 구문을 제어하는 세 가지 식

for, while, do-while 반복문 비교하기



for 반복문

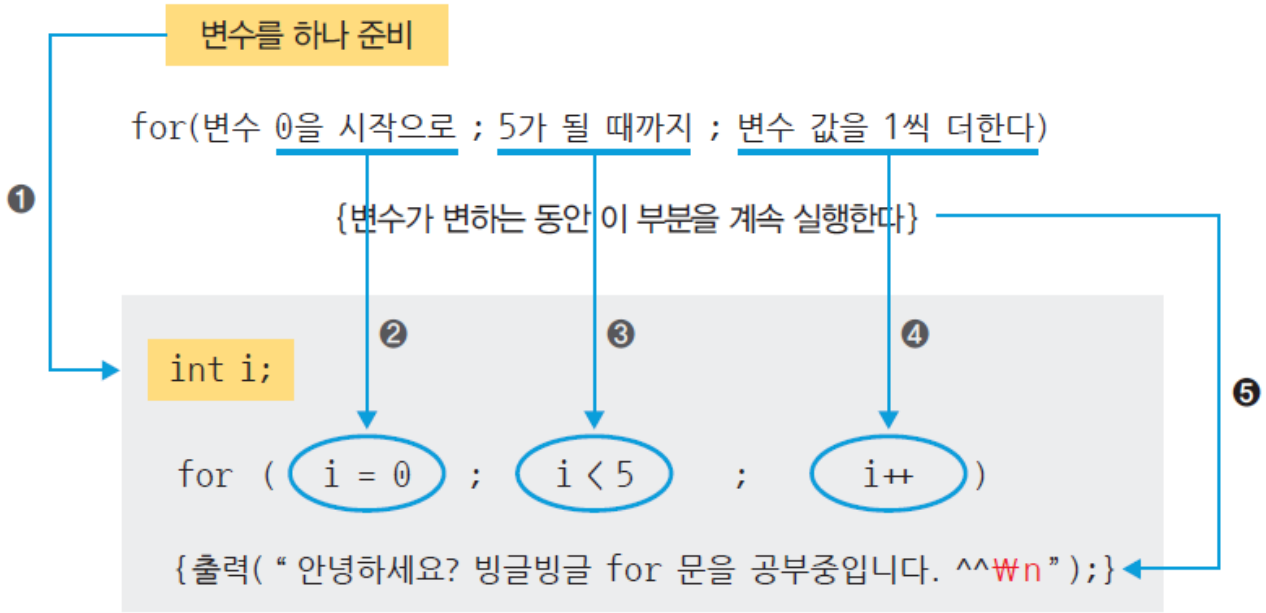
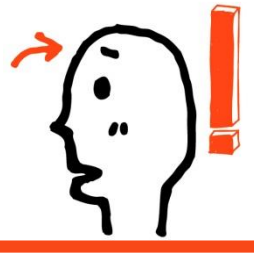


그림 6-4 for 문의 개념과 실제 사용

초깃값 → 조건식 → 반복할 문장 → 증감식 → 조건식 → 반복할 문장 → 증감식 → 조건식 → 반복할 문장 → 증감식 → 조건식 …

for, while, do-while 반복문 비교하기

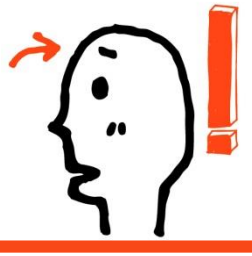


for 반복문

```
for(int i = 0; i < 10; i++)
{
    System.out.println("Loop count : " + i);
}
```

- 위의 구문은 초기식(int i = 0), 조건식(i < 10) 그리고 반복식(i ++)으로 구성
- 초기식에서는 변숫값 0을 갖는 변수 i를 선언하고, 변수 i를 조건식과 반복식에서 사용
- 루프를 한 번 돌 때마다 변수 i의 값은 단항 증가 연산자(++)에 의해서 1씩 증가
- 그래서 변수 i는 조건식에 의해 0~9까지 총 10번의 루프를 돌게 됨

for, while, do-while 반복문 비교하기



for 반복문

첫 번째 루프의 흐름
1 ⇒ 2 ⇒ 3 ⇒ 4 [i=1]

두 번째 루프의 흐름
2 ⇒ 3 ⇒ 4 [i=2]

세 번째 루프의 흐름
2 ⇒ 3 ⇒ 4 [i=3]

네 번째 루프의 흐름
2 [i=3]따라서 탈출!

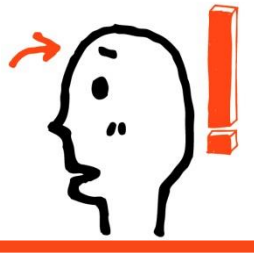
```
for( 1. int i=0 ; 2. i<3 ; 4. i++ )
{
    3. System.out.println("...");
}
```

```
class ForBasic
{
    public static void main(String[] args)
    {
        for(int i=0; i<3; i++)
            System.out.println("I love Java " + i);
    }
}
```

실행 결과

```
I love Java 0
I love Java 1
I love Java 2
```

for, while, do-while 반복문 비교하기



실습 6-3 for 문과 중괄호 사용 예

```
01 public class Ex06_03 {
02    public static void main(String[] args) {
03       int i;
04       for (i = 0; i < 3; i++)
05       {
06          System.out.printf("안녕하세요? ₩n");
07          System.out.printf("##또 안녕하세요?## ₩n");
08       }
09
10       System.out.printf("₩n₩n");
11
12       for (i = 0; i < 3; i++)
13          System.out.printf("안녕하세요? ₩n");
14          System.out.printf("##또 안녕하세요?## ₩n");
15
16    }
17 }
```

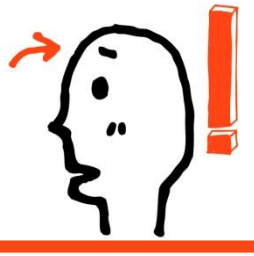
for 문에 중괄호를 사용했다.

for 문에 중괄호를 사용하지 않았다.

```
Problems @ Javadoc
<terminated> Ex06_03 [Java Ap
안녕하세요?
##또 안녕하세요?##
안녕하세요?
##또 안녕하세요?##
안녕하세요?
##또 안녕하세요?##


안녕하세요?
안녕하세요?
안녕하세요?
##또 안녕하세요?##
```

그림 6-6 실행 결과

for, while, do-while 반복문 비교하기



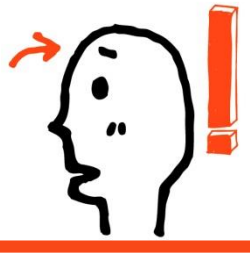
실습 6-12 for 문을 사용한 구구단 프로그램

```
01 import java.util.Scanner;
02
03 public class Ex06_12 {
04    public static void main(String[] args) {
05       Scanner s = new Scanner(System.in);
06       int i;
07       int dan;                    ----- 계산할 단을 입력받을 변수를 선언한다.
08
09       System.out.printf(" 몇 단 ? ");
10       dan = s.nextInt( );         ----- 계산할 단을 입력받는다.
11
12       for (i = 1; i <= 9; i++) {
13          System.out.printf(" %d X %d = %d ₩n", dan, i, dan * i);
14       }                           입력한 단에 대한 구구단을 1부터 9까지 반복해서 출력한다.
15    }
16 }
```

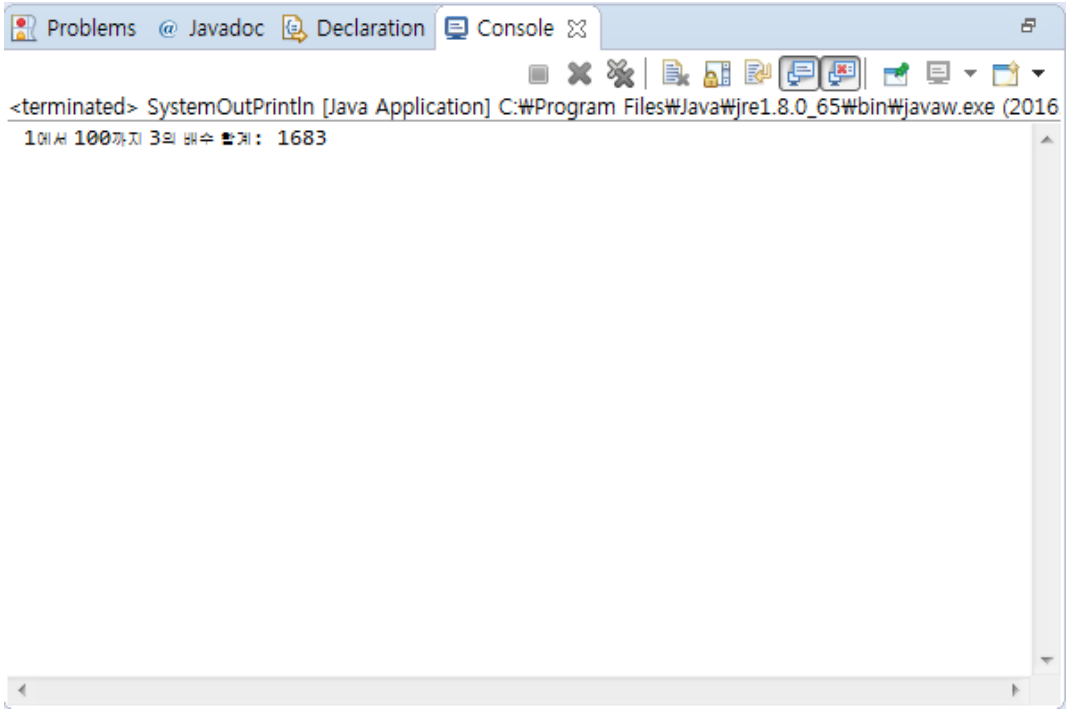
```
Problems  @ Javadoc
<terminated> Ex06_12 [Java
몇 단 ? 7
7 X 1 = 7
7 X 2 = 14
7 X 3 = 21
7 X 4 = 28
7 X 5 = 35
7 X 6 = 42
7 X 7 = 49
7 X 8 = 56
7 X 9 = 63
```

그림 6-16 실행 결과

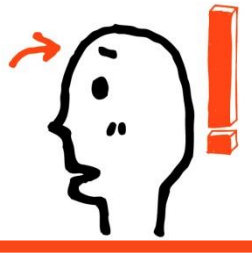
3의 배수 합계 프로그램



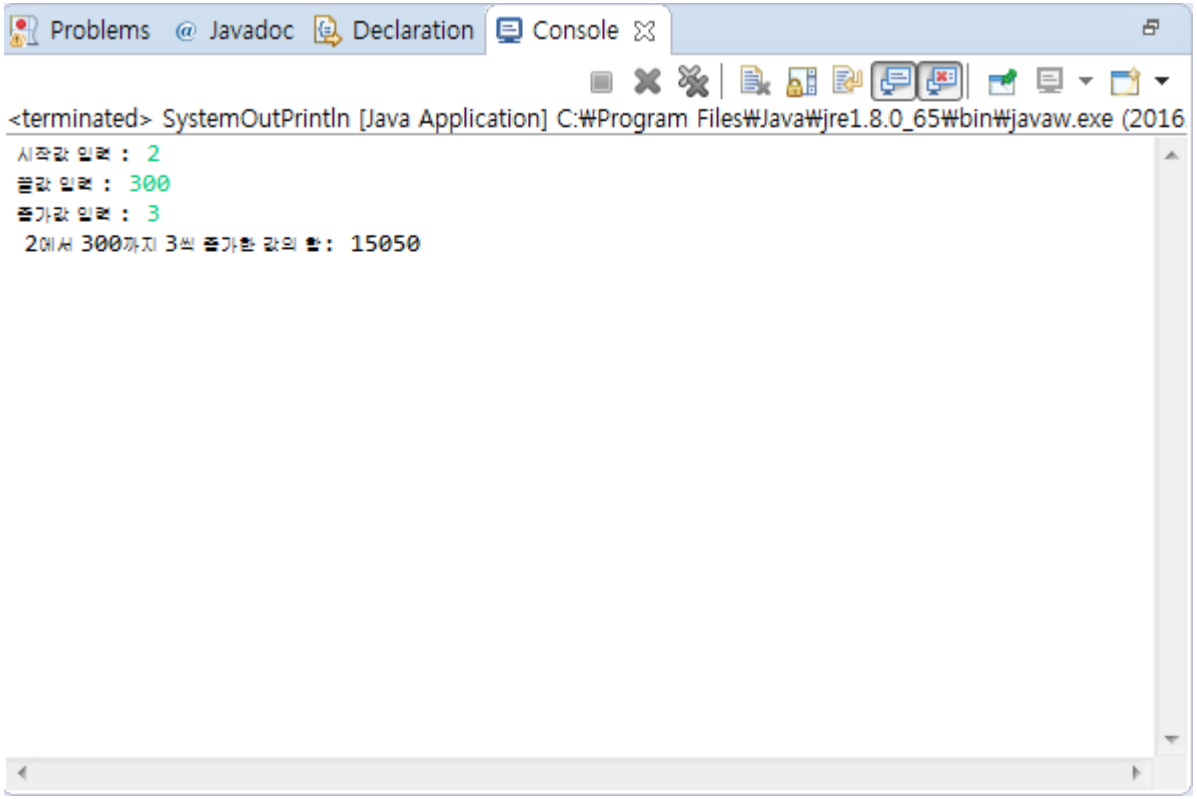
- For문을 이용하여 1~100중에서 3의 배수의 합계를 구하는 프로그램을 작성해보자.



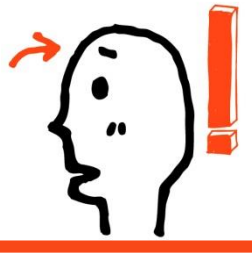
시작값, 끝값, 증가값 입력 합계 프로그램



- 시작값과 끝값, 증가값을 입력하여 시작값과 끝값까지의 합계를 구하는 프로그램을 작성해보자.(for문 이용)



for, while, do-while 반복문 비교하기



while과 do-while 구문

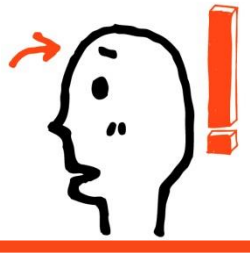
```
while (조건문)
{
        실행 구문
}
```

실행 순서 ①
조건문이 true이면, 다시 실행 순서 ②로 이동한다.

실행 순서 ②
실행 구문 실행

- 조건문이 true이면 반복을 계속 진행
- 맨 처음 while 반복문을 실행할 때 그 조건문의 값이 false라면 while문은 그대로 종료
- 맨 처음 조건문이 true이면 조건문이 false가 될 때까지 계속 반복

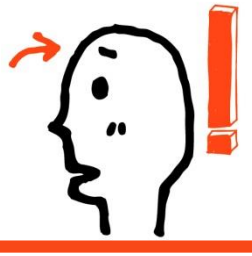
for, while, do-while 반복문 비교하기



while과 do-while 구문

- do-while이 실행되면 do 키워드와 중괄호 안에 위치한 실행 구문이 먼저 시작, 그 다음에 조건문의 결과값을 확인
- do-while 구문의 조건문이 false를 반환할 때까지 실행 구문을 실행
- **while 구문과의 차이점**은 먼저 실행 구문을 실행시키고 그 다음 조건문의 결과에 따라서 반복 여부를 확인

for, while, do-while 반복문 비교하기



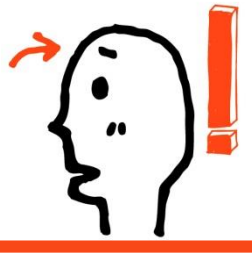
while과 do-while 구문

```
class WhileBasic
{
    public static void main(String[] args)
    {
        int num=0;

        while(num<5)
        {
            System.out.println("I like Java " + num);
            num++;
        }
    }
}
```

실행 결과

```
I like Java 0
I like Java 1
I like Java 2
I like Java 3
I like Java 4
```

for, while, do-while 반복문 비교하기



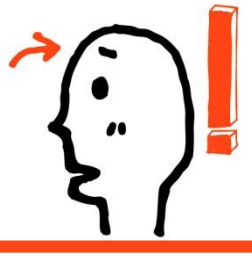
while과 do-while 구문

```
class DoWhileBasic
{
    public static void main(String[] args)
    {
        int num=0;

        do
        {
            System.out.println("I like Java " + num);
            num++;
        }while(num<5);
    }
}
```

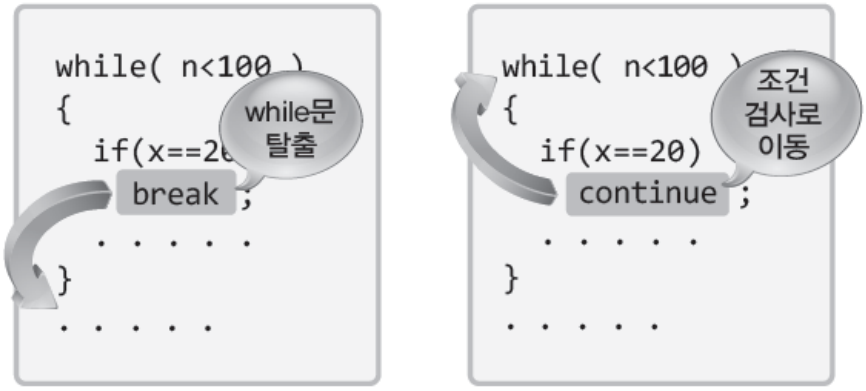
실행결과

```
I like Java 0
I like Java 1
I like Java 2
I like Java 3
I like Java 4
```

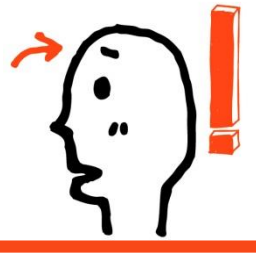
continue와 break 키워드로 흐름 제어하기



- **continue와 break** : 제어문이나 반복문이 나오면 변하는 흐름을 제어하는 역할
- **continue** 키워드는 반복문(for문, while문 그리고 do-while문)에서만 사용 가능
- continue 키워드의 역할은 바로 다음 반복으로 넘어가도록 흐름을 제어하는 것
- 반복문에서 continue를 만나면 실행하지 못한 구문들이 남아 있어도 다음 반복문으로 넘어감
- **break** 키워드는 반복문과 switch 구문에서 사용 가능
- 반복문과 switch 구문에서 break을 만나면, 조건문 값이 false가 아니더라도 즉각 반복문을 빠져 나옴



continue와 break 키워드로 흐름 제어하기

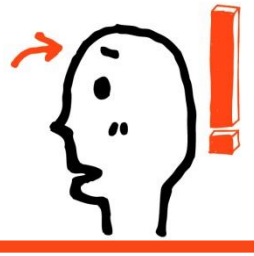


무한 반복 탈출을 위한 break 키워드

코드 4-9 package com.gilbut.chapter4;

```
1   public class FlowControlSample
2   {
3       public static void main(String[] args)
4       {
5           int i = 0;
6           while (true)
7           {
8               System.out.print(i ++);
9               if (i > 100)
10                  break;
11
12              System.out.print(“, “);
13          }
14      }
15  }
```

continue와 break 키워드로 흐름 제어하기



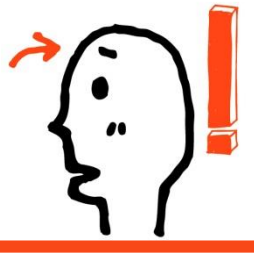
무한 반복 탈출을 위한 break 키워드

실행결과

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64,
65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85,
86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100
```

- 조건식이 항상 true이기 때문에 while 구문만 봤을 때는 무한 반복에서 빠져 나올 수 없는 상태
- 하지만 중간에 break를 사용해서 변수 i의 크기가 100을 초과하면 while 구문을 빠져 나오도록 함

continue와 break 키워드로 흐름 제어하기

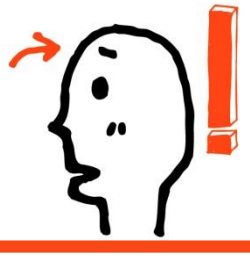


다음 루프로 넘어가는 continue 키워드

코드 4-10 package com.gilbut.chapter4;

```
1   public class FlowControlSample2
2   {
3       public static void main(String[] args)
4       {
5           for (int i = 1; i <= 10; i++)
6           {
7               if (i % 3 == 0)
8                   continue;
9
10              System.out.print(i + "\t");
11          }
12      }
13  }
```

continue와 break 키워드로 흐름 제어하기



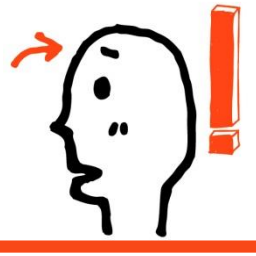
다음 루프로 넘어가는 continue 키워드

실행결과

```
1       2       4       5       7       8       10
```

- for 구문의 초기식에 선언된 변수 i가 1부터 10까지 증가
- 실행 구문에서는 변수 i를 3으로 나눈 나머지(3의 배수를 찾는 방법)가 0이면 다음 루프로 넘어가고 나머지가 0이 아니면 i를 화면에 출력

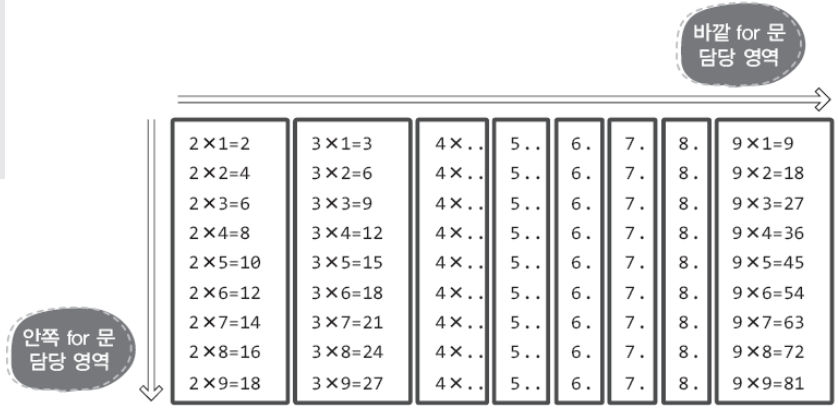
반복문의 중첩



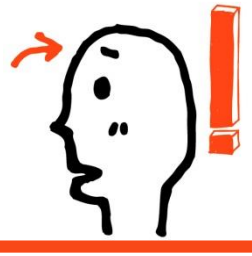
가장 많이 등장하는 for문의 중첩

```
class ByTimes
{
    public static void main(String[] args)
    {
        for(int i=2; i<10; i++)
        {
            for(int j=1; j<10; j++)
                System.out.println(i + " x " + j + " = " + i*j);
        }
    }
}
```

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
/* ~중간생략~ */
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
```



반복문의 중첩



중첩 for 문의 개념

- for 문 내부에 또 다른 for 문이 들어 있는 형태

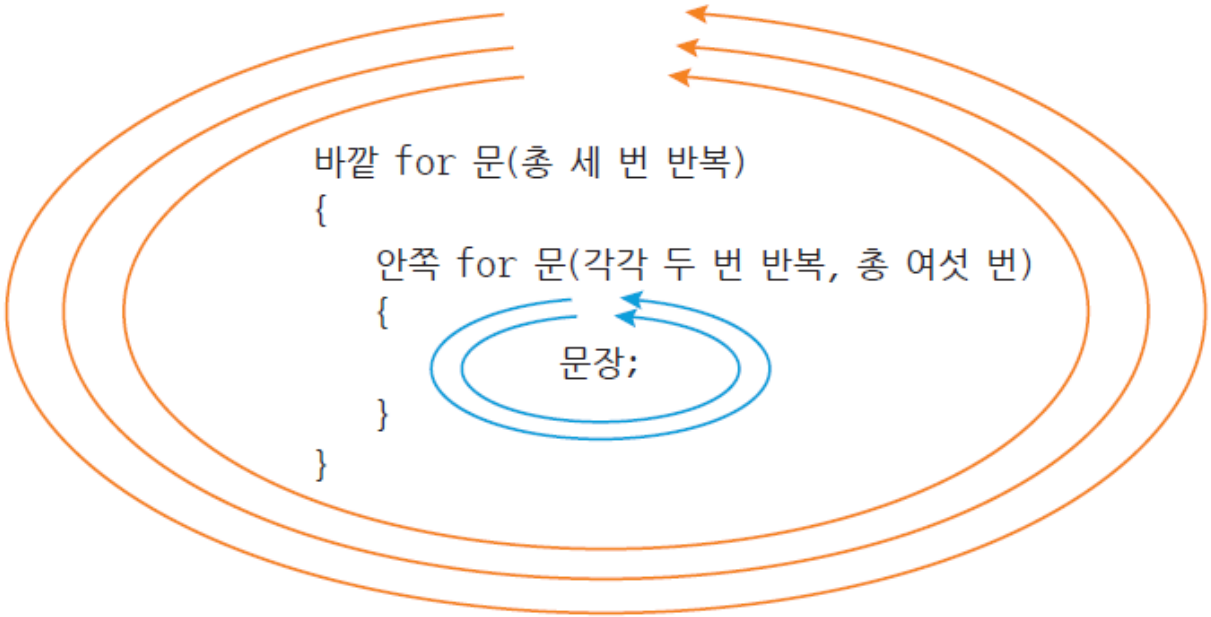


그림 6-17 중첩 for 문의 동작 개념

반복문의 중첩

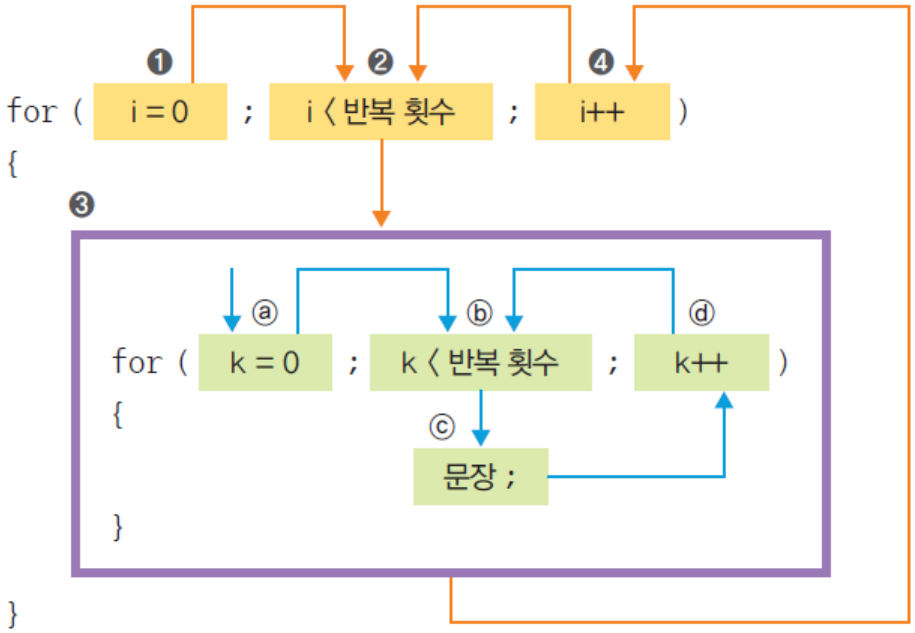
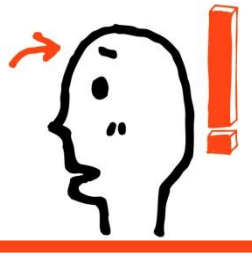


그림 6-18 중첩 for 문의 작동 방식

❶→❷→❸→(ⓐ→ⓑ→ⓒ→ⓓ→ⓑ→ⓒ→ⓓ→ⓑ→ 안쪽 for 문을 빠져나감)→❹→❷→❸
→(ⓐ→ⓑ→ⓒ→ⓓ→ⓑ→ⓒ→ⓓ→ⓑ→ 안쪽 for 문을 빠져나감)→❹→❷→❸→(ⓐ→ⓑ→ⓒ
→ⓓ→ⓑ→ⓒ→ⓓ→ⓑ→ 안쪽 for 문을 빠져나감)→❹→❷→ 바깥 for 문을 빠져나감

반복문의 중첩



실습 6-13 중첩 for 문 사용 예 1

```
01 public class Ex06_13 {
02    public static void main(String[] args) {
03       int i, k;                                  ----- 반복할 변수 i, k를 선언한다.
04
05       for (i = 0; i < 3; i++)                    ----- 바깥 for 문을 세 번 반복한다.
06       {
07          for (k = 0; k < 2; k++)                 ----- 안쪽 for 문을 두 번 반복한다.
08          {
09             System.out.printf("중첩 for 문입니다. (i값: %d, k값: %d)₩n", i, k);
                                                    i와 k 값을 총 여섯 번(=3×2) 출력한다.
10          }
11       }
12
13    }
14 }
```

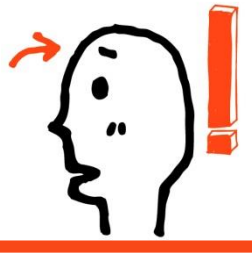
Problems @ Javadoc Declaration C

<terminated> Ex06_13 [Java Application] C:₩Progr

```
중첩 for 문입니다. (i값: 0, k값: 0)
중첩 for 문입니다. (i값: 0, k값: 1)
중첩 for 문입니다. (i값: 1, k값: 0)
중첩 for 문입니다. (i값: 1, k값: 1)
중첩 for 문입니다. (i값: 2, k값: 0)
중첩 for 문입니다. (i값: 2, k값: 1)
```

그림 6-19 실행 결과

반복문의 중첩



- 실습[6-13]의 중첩 for 문에서 i와 k 값의 변화

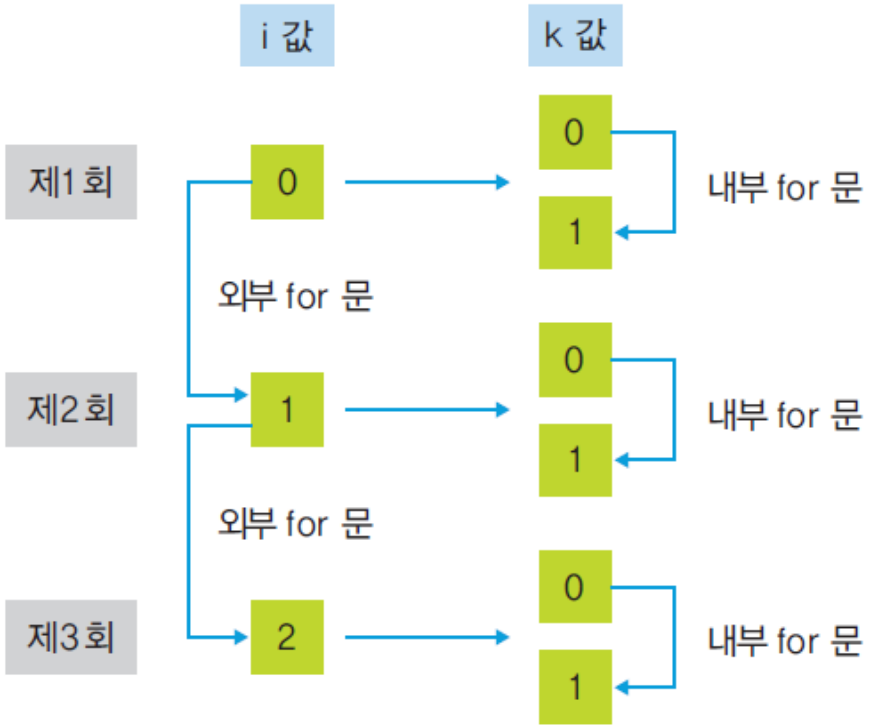
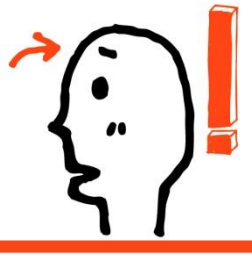


그림 6-20 중첩 for 문에서 i와 k 값의 변화

반복문의 중첩

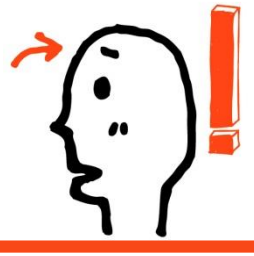


■ 중첩 for 문의 활용

- 구구단 2단~9단

그림 6-21 구구단에서 변수 i와 k의 추출

반복문의 중첩

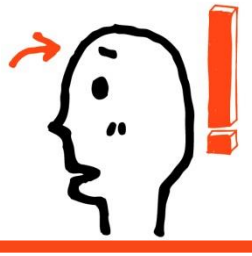


가장 많이 등장하는 for문의 중첩

```
class ByTimes
{
    public static void main(String[] args)
    {
        for(int i=2; i<10; i++)
        {
            for(int j=1; j<10; j++)
                System.out.println(i + " x " + j + " = " + i*j);
        }
    }
}
```

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
/* ~중간생략~ */
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
```

반복문의 중첩



실습 6-15 중첩 for 문 사용 예 3

```
01 public class Ex06_15 {
02    public static void main(String[] args) {
03       int i, k;
04
05       for (i = 1; i <= 9; i++) {        ----- 각 단의 뒷자리 숫자 1~9를 반복한다.
06          for (k = 2; k <= 9; k++) {     ----- 2~9단을 반복한다.
07             System.out.printf("%3dX%d=%2d",          );   ----- 각 단별로 한 줄씩 출력한다.
08          }
09          System.out.printf("₩n");       ----- 각 단의 한 줄을 출력한 후 다음 줄로 넘긴다.
10       }
11
12    }
13 }
```

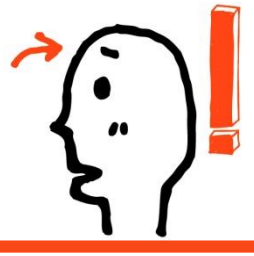
Problems | Javadoc | Declaration | Console

<terminated> Ex06_15 [Java Application] C:₩Program Files₩Java₩jre1.8.0_60₩bin₩javaw.exe (2016. 1. 3. 오후 11:52:03)

```
 2X1= 2  3X1= 3  4X1= 4  5X1= 5  6X1= 6  7X1= 7  8X1= 8  9X1= 9
 2X2= 4  3X2= 6  4X2= 8  5X2=10  6X2=12  7X2=14  8X2=16  9X2=18
 2X3= 6  3X3= 9  4X3=12  5X3=15  6X3=18  7X3=21  8X3=24  9X3=27
 2X4= 8  3X4=12  4X4=16  5X4=20  6X4=24  7X4=28  8X4=32  9X4=36
 2X5=10  3X5=15  4X5=20  5X5=25  6X5=30  7X5=35  8X5=40  9X5=45
 2X6=12  3X6=18  4X6=24  5X6=30  6X6=36  7X6=42  8X6=48  9X6=54
 2X7=14  3X7=21  4X7=28  5X7=35  6X7=42  7X7=49  8X7=56  9X7=63
 2X8=16  3X8=24  4X8=32  5X8=40  6X8=48  7X8=56  8X8=64  9X8=72
 2X9=18  3X9=27  4X9=36  5X9=45  6X9=54  7X9=63  8X9=72  9X9=81
```

그림 6-24 실행 결과

반복문의 중첩



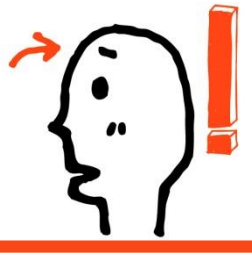
While문도 중첩해봅시다!

```
class ByTimes2
{
    public static void main(String[] args)
    {
        int i=2, j;
        while(i<10)
        {
            j=1;
            while(j<10)
            {
                System.out.println(i + " x " + j + " = " + i*j);
                j++;
            }
            i++;
        }
    }
}
```

실행 결과

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
/* ~중간생략~ */
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
```

기타 for문



■ 여러 개의 초깃값과 증감식을 사용하는 for 문

```
for (초깃값 1, 초깃값 2; 조건식; 증감식 1, 증감식 2)
```

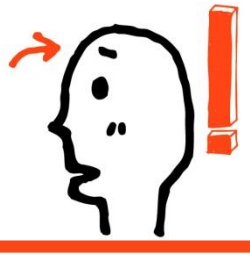
실습 6-16 다양한 for 문의 형태 1

```
01 public class Ex06_16 {
02    public static void main(String[] args) {
03       int i, k;                          ----- 반복할 변수 i와 k를 선언한다.
04
05       for (i = 1, k = 1; i <= 9; i++, k++)  ----- 초깃값과 증감식이 2개이다.
06          System.out.printf(" %d X %d = %d ₩n", i, k, i * k);
07
08    }
09 }
```

```
Problems  @ Javado
<terminated> Ex06_16 [Ja
1 X 1 = 1
2 X 2 = 4
3 X 3 = 9
4 X 4 = 16
5 X 5 = 25
6 X 6 = 36
7 X 7 = 49
8 X 8 = 64
9 X 9 = 81
```

그림 6-25 실행 결과

기타 for문



■ 초깃값과 증감식이 없는 for 문

- 0~9를 출력하는 3가지 for 문

❶ 기본 형식

```
int i;
for (i = 0; i < 10; i ++)
{
   출력 ("%d ₩n", i) ;
}
```

==

❷ 초깃값 빼기

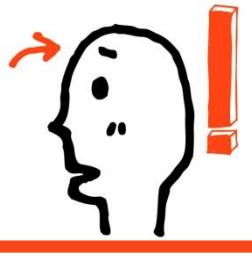
```
int i;
i = 0;
for ( ____ ; i < 10; i ++)
{
   출력 ("%d ₩n", i) ;
}
```

==

❸ 초깃값과 증감식 빼기

```
int i;
i = 0;
for ( ____ ; i < 10; ____ )
{
   출력 ("%d ₩n", i) ;
   i ++ ;
}
```

기타 for문



실습 6-17 다양한 for 문의 형태 2

```
01 public class Ex06_17 {
02    public static void main(String[] args) {
03       int i;
04       i = 0;
05       for ( ; ; ) {                    ----- 초깃값, 조건식, 증감식이 없다.
06          System.out.printf("%d  ₩n", i);
07          i++;
08       }
09
10    }
11 }
```

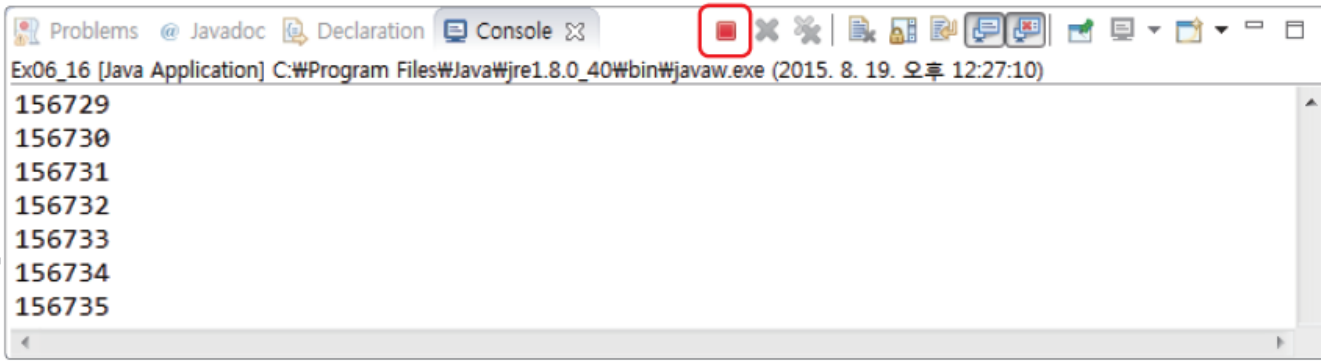
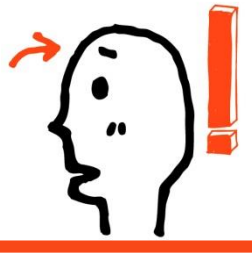


그림 6-27 실행 결과

반복문의 중첩



break 레이블

```
class LabeledBreak
{
    public static void main(String[] args)
    {
        outerLoop :
        for(int i=1; i<10; i++)
        {
            for(int j=1; j<10; j++)
            {
                System.out.println("[" + i +", " + j + "]");
                if(i%2==0 && j%2==0)
                    break outerLoop;
            }
        }
    }
}
```

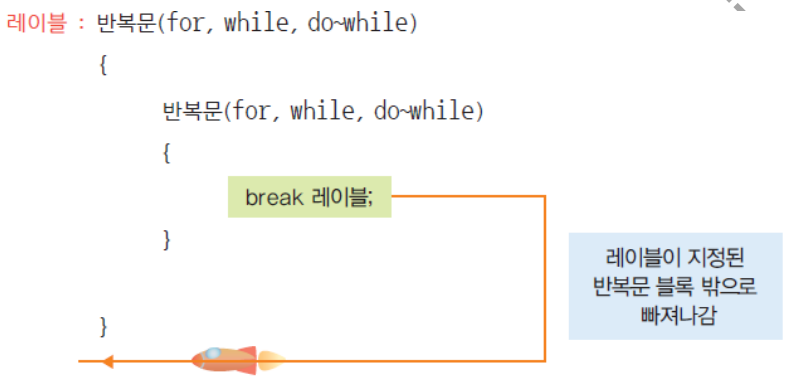
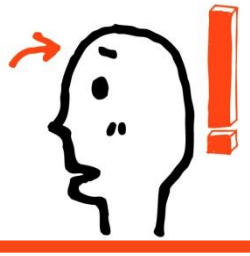


그림 7-18 break 레이블문의 작동

break문은 자신을 감싸는 반복문을 하나밖에 벗어나지 못한다. 때문에 둘 이상의 반복문을 벗어날 때는 break 레이블을 사용을 고려할 수 있다! 하지만 빈번한 사용은 바람직하지 못하다!

반복문의 중첩



- 현재 메소드를 불렀던 곳으로 돌아가는 return 문

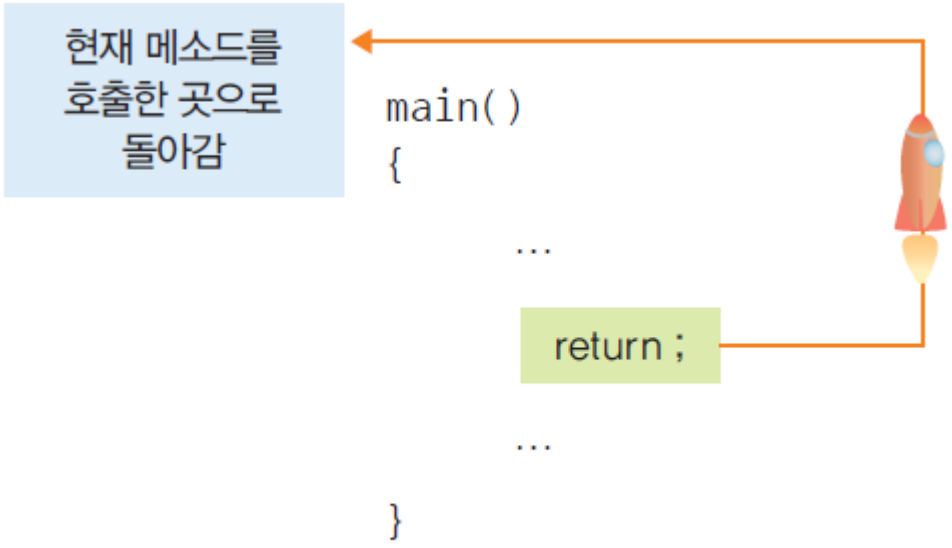
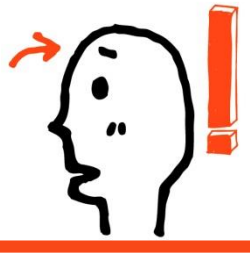


그림 7-20 return 문의 작동

반복문의 중첩



실습 7-13 return 문 사용 예

```
01 public class Ex07_13 {
02    public static void main(String[] args) {
03       int hap = 0;
04       int i;
05
06       for (i = 1; i <= 100; i++)
07          hap += i;                    ---- 1부터 100까지 합계를 누적한다.
08
09       System.out.printf("1부터 100까지의 합은 %d 입니다.₩n", hap);  ----- 합계를 출력한다.
10
11       if (hap > 5000)
12          return;                      ----- 현재 메소드를 호출한 곳으로 복귀한다.
13
14       System.out.printf("프로그램의 끝입니다.");  ----- 한 번도 실행되지 않는다.
15    }
16 }
```

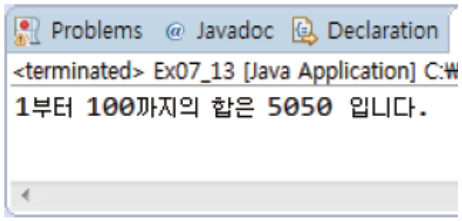
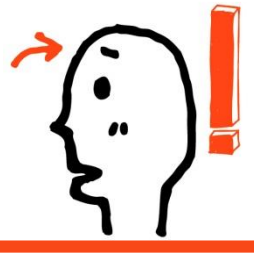


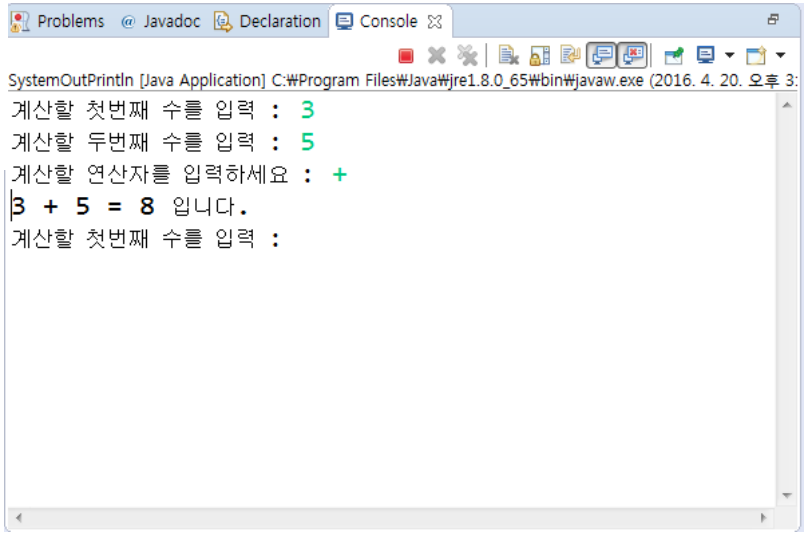
그림 7-21 실행 결과

무한루프를 이용한 계산기 프로그램

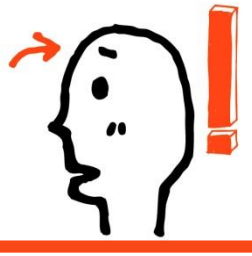


- 다음과 같이 while문을 이용해 무한루프를 만들고 두 개의 수와 하나의 연산자를 입력 받아 계산기 프로그램을 작성해보자

```
while (true) {
    System.out.printf("계산할 첫번째 수를 입력 : ");
    a = s.nextInt();
    System.out.printf("계산할 두번째 수를 입력 : ");
    b = s.nextInt();
    String ch= s.nextLine();
    System.out.printf("계산할 연산자를 입력하세요 : ");
    ch= s.nextLine();
```

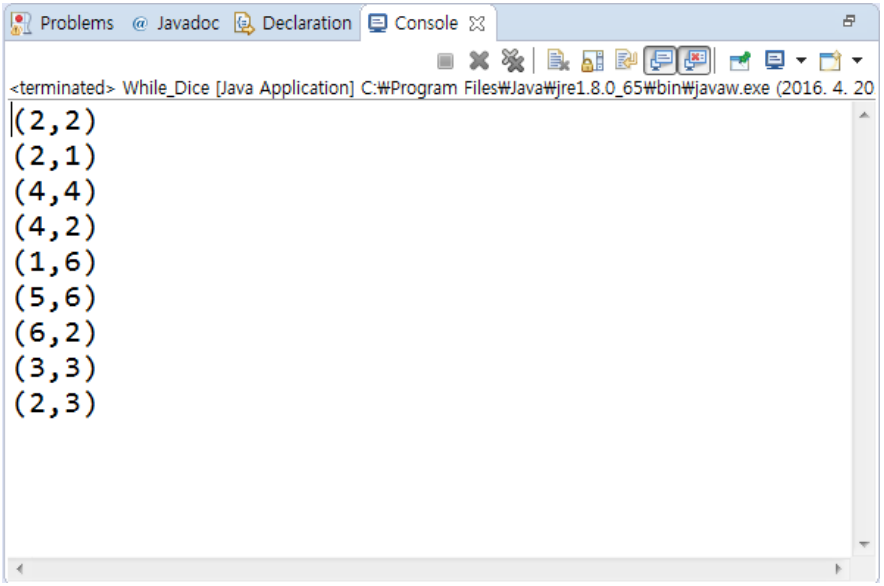


주사위 게임

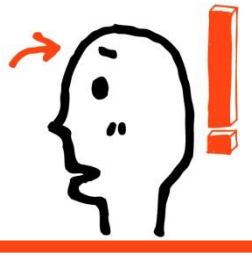


- While문과 Math.random() 메소드를 이용해서 두 개의 주사위를 던졌을 때 나오는 눈을(눈1, 눈2) 형태로 출력하고, 눈의 합이 5가 아니면 계속 주사위를 던지고, 눈의 합이 5이면 실행을 멈추는 코드를 작성해보시오.
  눈의 합이 5가 되는 조합은 (1,4), (4,1), (2,3), (3,2)입니다.

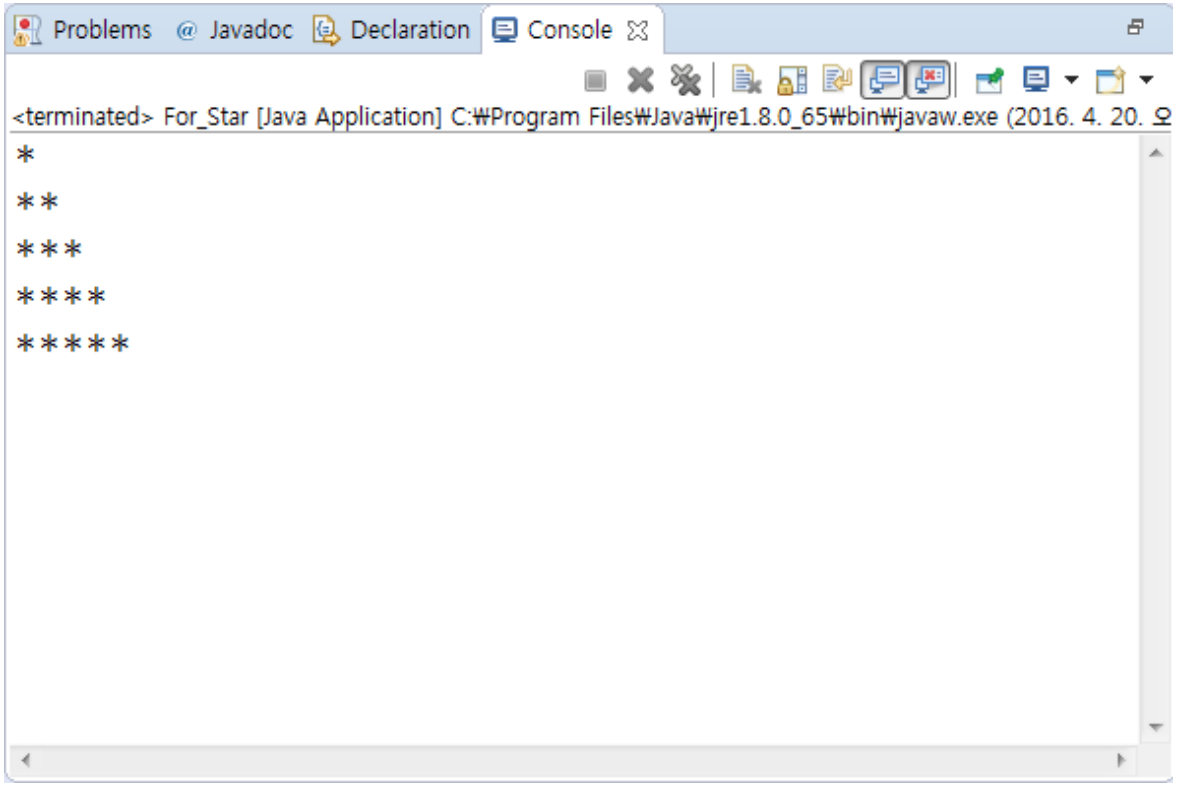
```
int num1 = (int) (Math.random()*6)+1;
```



별 출력 프로그램



- 중첩 For문을 이용해서 실행결과와 같은 삼각형을 출력하는 코드를 작성하시오.



THANK YOU

실무에서 알아야 할 기술은 따로 있다! 자바를 다루는 기술