



알기 쉽게 해설한
자바 프로그래밍 10판

Chapter 04. 선택문

학습목표

- 선택 논리를 대표하는 if문에 관해서 학습합니다.
- 단순 if문, 이중 if문, 다중 if문, 네포된 if문에 관해 학습합니다.
- 선택문과 반복문에 사용되는 조건문에 대해 학습합니다.
- 다중 선택문인 switch문에 관해 학습합니다.

목차

Section 1. if문

Section 2. 조건식

Section 3. switch문

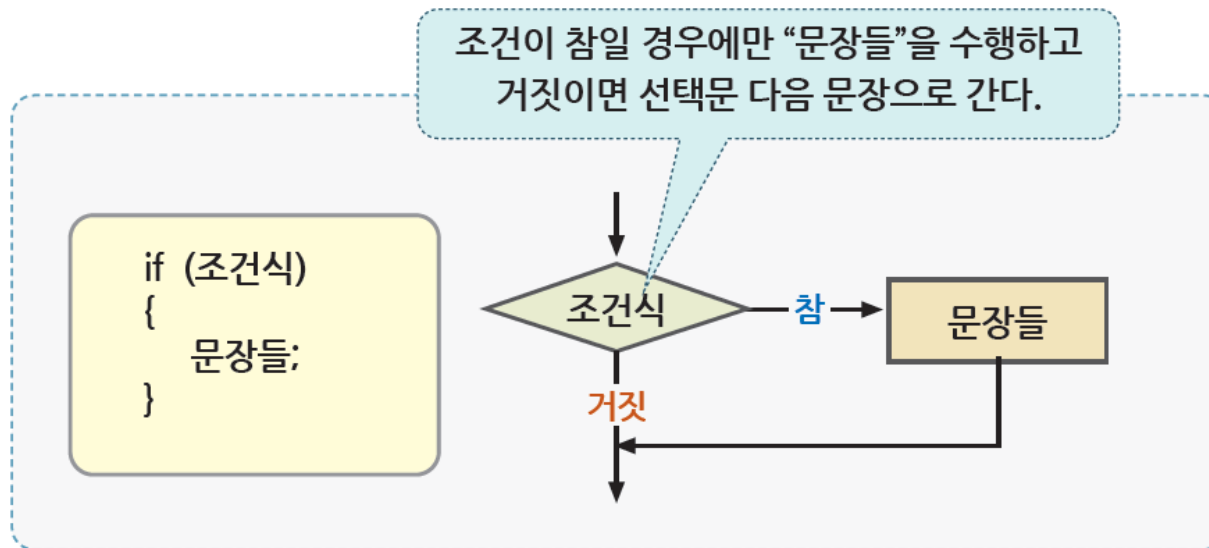
Section 1.

if문

1 if문

1-1 단순 if문(if)

- 우리가 작성하는 프로그램은 순서, 선택, 반복의 논리로 구성
- 대표적인 선택문 if
- 단순 if문(if)





1 if문

1-1 단순 if문(if)

```
if ( num < 0 )  
    System.out.println("음수입니다");
```

조건을 나타내는 문장에는 ";"을 붙이면 안 된다.
문장이 한 문장일 경우 "{}"로 묶지 않아도 된다.

```
if ( num % 2 == 0 )
```

```
{  
    System.out.print("입력된 수 : " + num);  
    System.out.println("은(는) 짝수입니다");  
}
```

조건을 나타내는 문장이 한 문장보다 많으면 반드시 블록("{}")으로 묶어야 한다.



1 if문

1-1 단순 if문(if)

예제 4.1

SimpleIFTest1.java

```
01: import java.util.Scanner;
02: public class SimpleIFTest1 {
03:     public static void main(String args[])
04:     {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("한 개의 숫자를 입력 : ");
07:         int count = stdin.nextInt();
08:         if (count < 0) ← 입력된 값이 0보다 작은지 검사하는 조건문
09:             System.out.println(count + "은(는) 음수입니다 ");
10:     }
11: }
```

↑ 조건이 참일 경우에만 수행

실행 결과 두 번을 실행한 결과

한 개의 숫자를 입력 : 40 ← 아무런 결과를 출력하지 않는다.
한 개의 숫자를 입력 : -22
-22은(는) 음수입니다 ← 음수인 경우 결과를 출력한다.

1 if문

1-1 단순 if문(if)

예제 4.2

SimpleIFTest2.java

```

12: import java.util.Scanner;
13: public class SimpleIFTest2 {
14:     public static void main(String args[])
15:     {
16:         Scanner stdin = new Scanner(System.in);
17:         System.out.print("한 개의 숫자를 입력 : ");
18:         int count = stdin.nextInt();
19:         if (count < 0); ← 조건문 뒤에 ";"을 붙이면 문장 종료
20:             System.out.println(count + "은(는) 음수입니다 ");
21:     }
22: }

```

↑ if문과 상관없이 다음 문장으로 실행

실행 결과

두 번을 실행한 결과

한 개의 숫자를 입력 : 40

40은(는) 음수입니다 ← 조건문과 상관없이 출력

한 개의 숫자를 입력 : -22

-22은(는) 음수입니다 ← 조건문과 상관없이 출력

1 if문

1-1 단순 if문(if)

예제 4.3

SimpleIFTest3.java

```

01: import java.util.Scanner;
02: public class SimpleIFTest3 {
03:     public static void main(String args[])
04:     {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("성적을 입력하세요 : ");
07:         int grade = stdin.nextInt();
08:         if (grade >= 90)
09:         {
10:             System.out.println("축하합니다 ");
11:             System.out.println("A학점을 취득하셨습니다 ");
12:         }
13:         System.out.println("감사합니다");
14:     }
15: }

```

조건문을 90보다 같거나 큰 것으로 지정

한 문장 이상일 경우
"{}"로 묶는다

if문과는 상관이 없는 문장

실행 결과

두 번을 실행한 결과

성적을 입력하세요 : 95
축하합니다
A학점을 취득하셨습니다
감사합니다

성적을 입력하세요 : 89
감사합니다

1 if문

1-1 단순 if문(if)

예제 4.4

SimpleIFTest4.java

```

01: import java.util.Scanner;
02: public class SimpleIFTest4 {
03:     public static void main(String args[])
04:     {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("성적을 입력하세요 : ");
07:         int grade = stdin.nextInt();
08:         if (grade >= 90)
09:             System.out.println("축하합니다 ");
10:             System.out.println("A학점을 취득하셨습니다 ");
11:             System.out.println("감사합니다");
12:     }
13: }
  
```

실행 결과

두 번을 실행한 결과

성적을 입력하세요 : 95

축하합니다 ← 조건문이 참일 경우 출력

A학점을 취득하셨습니다 ← 조건문과 상관없이 출력

감사합니다

성적을 입력하세요 : 89

A학점을 취득하셨습니다 ← 조건문과 상관없이 출력

감사합니다

System.out.println("축하합니다 "); ← 이 문장만 조건절이 참일 경우 수행된다.

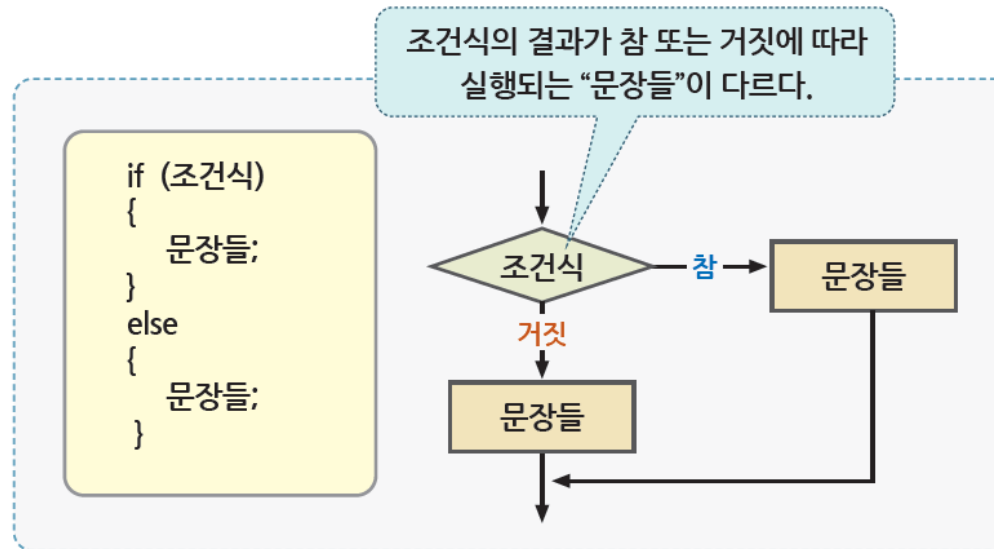
System.out.println("A학점을 취득하셨습니다 "); ← 조건절과 상관없이 무조건 수행된다.

System.out.println("감사합니다"); ←

1 if문

1-2 이중 if문(if-else)

- 조건식의 결과에 따라 특정 작업을 수행해야 하는 경우 사용





1 if문

1-2 이중 if문(if-else)

```
if ( score >= 60 )  
    System.out.println("학점 취득 성공"); ← 참일 경우 수행(한 문장)  
else  
    System.out.println("학점 취득 실패"); ← 거짓일 경우 수행(한 문장)  
System.out.println("학점 취득 여부와 상관없이 무조건 수행됨"); ← 선택문과 상관없는 문장  
  
if ( num % 2 == 0 )  
{  
    System.out.println("입력된 수 : " + num); ← 여러 문장일 경우 묶는다.  
    System.out.println("짝수입니다");  
}  
else  
{  
    System.out.println("입력된 수 : " + num); ← 여러 문장일 경우 묶는다.  
    System.out.println("홀수입니다");  
}
```

1 if문

1-2 이중 if문(if-else)

예제 4.5

DoubleIFTest1.java

```

01: import java.util.Scanner;
02: public class DoubleIFTest1 {
03:     public static void main(String args[])
04:     {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("성적을 입력하세요 : ");
07:         int grade = stdin.nextInt();
08:         if (grade >= 90)
09:             System.out.println("A학점 취득 성공");
10:         else
11:             System.out.println("A학점 취득 실패");
12:         System.out.println("감사합니다");
13:     }
14: }

```

실행 결과

두 번을 실행한 결과

성적을 입력하세요 : 95

A학점 취득 성공 ← 조건문이 참일 경우 출력

감사합니다 ← 조건문과 상관없이 출력

성적을 입력하세요 : 89

A학점 취득 실패 ← 조건문이 거짓일 경우 출력

감사합니다 ← 조건문과 상관없이 출력



1 if문

1-2 이중 if문(if-else)

```
if (grade >= 90)
    System.out.println("A학점 취득 성공");
else
    System.out.println("A학점 취득 실패");
```

위의 문장을 3항 연산자를 이용하면

```
System.out.println(grade >= 90 ? "A학점 취득 성공" : "A학점 취득 실패");
```

한 문장으로 표현할 수 있습니다.

1 if문

1-2 이중 if문(if-else)

예제 4.6

DoubleIFTest2.java

```

01: import java.util.Scanner;
02: public class DoubleIFTest2 {
03:     public static void main(String args[])
04:     {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("숫자를 입력 : ");
07:         int count = stdin.nextInt();
08:         if (count < 0 )
09:         {
10:             System.out.print("입력된 수 " + count);
11:             System.out.println("은(는) 음수입니다");
12:         }
13:         else
14:         {
15:             System.out.print("입력된 수 " + count);
16:             System.out.println("은(는) 음수가 아닙니다");
17:         }
18:     }
19: }

```

조건절이 참일 경우
수행되는 블록

조건절이 거짓일 경우
수행되는 블록

실행 결과

두 번을 실행한 결과

숫자를 입력 : -8

입력된 수 -8은(는) 음수입니다

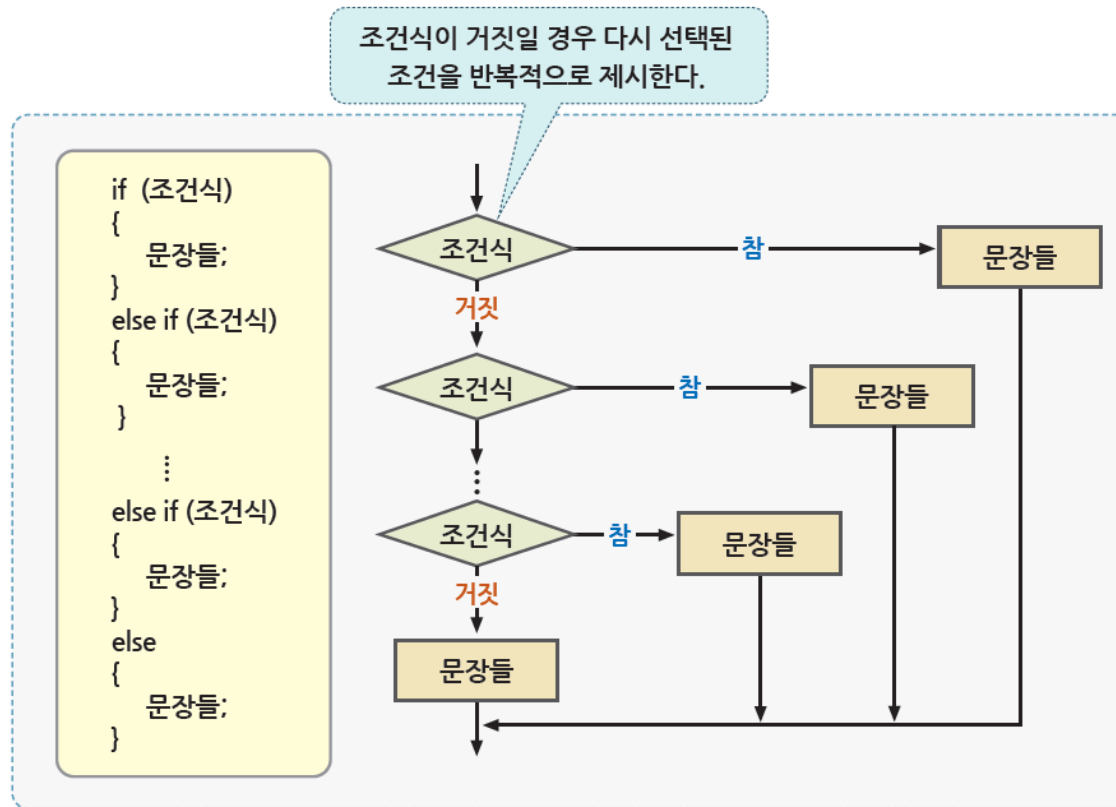
숫자를 입력 : 8

입력된 수 8은(는) 음수가 아닙니다

1 if문

1-3 다중 if문(if-else if)

- 다중 if문은 조건이 거짓일 경우 다시 조건을 제시하는 선택문



1 if문

1-3 다중 if문(if-else if)

예제 4.7

MultilFTest1.java

```

01: import java.util.Scanner;
02: public class MultiIFTest1 {
03:     public static void main(String args[])
04:     {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("성적을 입력하세요 : ");
07:         int grade = stdin.nextInt();
08:         if (grade >= 90)
09:             System.out.println("A학점 취득");
10:         else if (grade >= 80)
11:             System.out.println("B학점 취득");
12:         else if (grade >= 70)
13:             System.out.println("C학점 취득");
14:         else if (grade >= 60)
15:             System.out.println("D학점 취득");
16:         else
17:             System.out.println("학점 취득 실패");
18:     }
19: }
  
```

실행 결과

두 번을 실행한 결과

성적을 입력하세요 : 89
 B학점 취득
 성적을 입력하세요 : 59
 학점 취득 실패

반복적으로 조건을 제시한다.

마지막 if문의 조건이 거짓일 때 수행

1 if문

1-3 다중 if문(if-else if)

예제 4.8

MultiIFTest2.java

```

01: import java.util.Scanner;
02: public class MultiIFTest2 {
03:     public static void main(String args[])
04:     {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("숫자를 입력 : ");
07:         int count = stdin.nextInt();
08:         if (count < 0 )
09:         {
10:             System.out.print("입력된 수 " + count);
11:             System.out.println("은(는) 음수입니다");
12:         }
13:         else if (count == 0) ← 등가 연산자 ==을 사용하여 비교
14:             System.out.print("입력된 수는 0 입니다");
15:         else
16:         {
17:             System.out.print("입력된 수 " + count);
18:             System.out.println("은(는) 양수입니다");
19:         }
20:     }
21: }

```

실행 결과

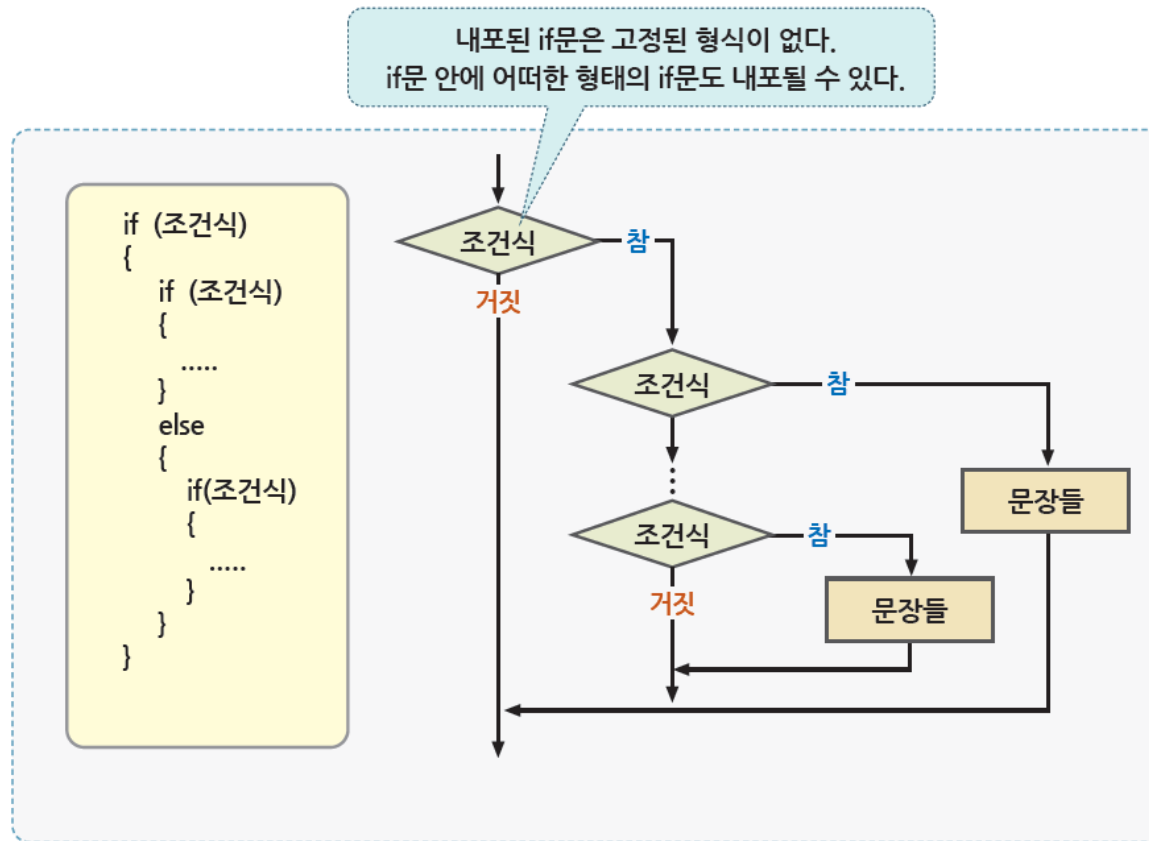
세 번을 실행한 결과

숫자를 입력 : 3
 입력된 수 3은(는) 양수입니다
 숫자를 입력 : 0
 입력된 수는 0입니다
 숫자를 입력 : -3
 입력된 수 -3은(는) 음수입니다

1 if문

1-4 내포된 if문

- 선택문안에 선택문이 내포될 수 있다



1 if문

1-4 내포된 if문

```

if (score >= 90)
    if (score >= 95)
        System.out.println("A+ 학점입니다");
    else
        System.out.println("A- 학점입니다");
else
    if (score >= 60)
        if (score >= 80)
            System.out.println("B 학점입니다");
        else
            System.out.println("C 또는 D 학점입니다");
    else
        System.out.println("학점취득 실패");

```

내포된 관계에서 else는 가장 가까운 if의 else입니다.

내포된 관계의 if문을 사용할 때는 들여쓰기를 명확하게 하는 것이 좋습니다.

```

if(score > 60)
    if(score > 70)
        if(score > 80)
            if(score > 90)
                System.out.println("A학점");
    else
        System.out.println("학점 취득 실패");

```

들여쓰기의 잘못된 예.
첫 if문의 else처럼 보이지만
마지막 if문의 else

1 if문

1-4 내포된 if문

예제 4.9

NestedIFTest1.java

```

01: import java.util.Scanner;
02: public class NestedIFTest1 {
03:     public static void main(String args[])
04:     {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("성적을 입력하세요 : ");
07:         int score = stdin.nextInt();
08:         if (score >= 80)
09:         {
10:             System.out.println("우수 학점군에 속합니다");
11:             if (score >= 90)
12:                 System.out.println("A 학점입니다");
13:             else
14:                 System.out.println("B 학점입니다");
15:         }
16:     else
17:     {

```

Diagram illustrating nested if-else logic:

- Line 08: `if (score >= 80)` is the outer if statement.
- Line 09: `{` starts the outer if block.
- Line 11: `if (score >= 90)` is the inner if statement.
- Line 12: `System.out.println("A 학점입니다");` is the inner if block.
- Line 13: `else` is the inner else statement.
- Line 14: `System.out.println("B 학점입니다");` is the inner else block.
- Line 15: `}` ends the outer if block.
- Line 16: `else` is the outer else statement.
- Line 17: `{` starts the outer else block.

Annotations:

- Line 08: `if (score >= 80)` is labeled "if-else" (referring to the outer structure).
- Line 11: `if (score >= 90)` is labeled "if-else" (referring to the inner structure).

1 if문

1-4 내포된 if문

● 예제 4.9

```

18:      if (score >= 60)
19:      {
20:          System.out.println("보통 학점군에 속합니다");
21:          if (score >= 70)
22:              System.out.println("C 학점입니다");
23:          else
24:              System.out.println("D 학점입니다");
25:      }
26:  else
27:      System.out.println("학점 취득 실패");
28:  }
29:  }
30: }

```

Diagram annotations: Red dashed arrows and labels 'if-else' indicate the nested structure of the conditional statements.

실행 결과 세 번을 실행한 결과

성적을 입력하세요 : 98
 우수 학점군에 속합니다
 A 학점입니다
 성적을 입력하세요 : 67
 보통 학점군에 속합니다
 D 학점입니다
 성적을 입력하세요 : 55
 학점 취득 실패

Section 2.

조건식

2 조건식

2-1 조건식의 추출

● 개발하려는 프로그램의 핵심은 조건식

- 일반적인 문제에서 조건식을 명확하게 추출하는 것이 프로그램의 핵심
- 조건식은 3장에서 학습한 관계 연산자와 논리 연산자로 구성

일반적인 문제	추출된 조건식
입력된 수가 양수인 경우	<code>(input > 0)</code>
두 과목 성적이 모두 80점 이상인 경우	<code>(score1 >= 80 && score2 >= 80)</code>
두 과목 성적 중에서 하나 이상이 80점 이상인 경우	<code>(score1 >= 80 score2 >= 80)</code>
두 과목 성적의 합이 150인 경우	<code>(score1 + score2 == 150)</code>
입력된 두 수가 모두 짝수인 경우	<code>(input1 % 2 == 0 && input2 % 2 == 0)</code>
입력된 점수가 60보다 크고 100보다 작거나 같은 경우	<code>(score > 60 && score <= 100)</code> (<code>60 < score <= 100</code>) <—허용되지 않음
작급이 7 또는 8급이고, 나이가 30대(30-39)인 경우	<code>((grade == 7 grade == 8) && (30 <= age && age <= 39))</code>

2 조건식

2-1 조건식의 추출

드모르간의 법칙

- $a \ \&\& \ b$ 는 $!(\ !a \ || \ !b)$ 과 같습니다.
- $a \ || \ b$ 는 $!(\ !a \ \&\& \ !b)$ 와 같습니다.

표 4-2 조건식과 드모르간 법칙

조건식	동일한 조건식(드모르간 법칙 적용)
$(score1 \geq 80 \ \ score2 \geq 80)$	$!(\ score1 < 80 \ \&\& \ score2 < 80)$
$(score > 60 \ \&\& \ score \leq 100)$	$!(\ score \leq 60 \ \ score > 100)$
$(input1 \% 2 == 0 \ \&\& \ input2 \% 2 == 0)$	$!(input1 \% 2 != 0 \ \ input2 \% 2 != 0)$



2 조건식

2-1 조건식의 추출

● 예제 4.10

예제 4.10

ComConditionTest1.java

```
01: import java.util.Scanner;
02: public class ComConditionTest1 {
03:     public static void main(String args[])
04:     {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("월을 입력하세요 : ");
07:         int month = stdin.nextInt();
08:         if (3 <= month && month <= 5 ) ← 복합 조건식을 사용
09:             System.out.println("봄 입니다");
10:         else if (6 <= month && month <= 8 )
11:             System.out.println("여름 입니다");
12:         else if (9 <= month && month <= 11 )
13:             System.out.println("가을 입니다");
14:         else if (1 == month || month == 2 || month == 12 )
            ↓ 두 개의 논리 연산자를 사용하여
            ↓ 구성한 조건식
```



2 조건식

2-1 조건식의 추출

```
15:         System.out.println("겨울 입니다");
16:     else
17:         System.out.println("해당되는 계절이 없습니다");
18:
19: }
20: }
```

↑ 해당되는 계절이 없을 경우 출력

실행 결과

두 번을 실행한 결과

월을 입력하세요 : 12
겨울 입니다
월을 입력하세요 : 13
해당되는 계절이 없습니다



2 조건식

2-2 단락 평가 연산자

- 논리 연산자와 비트 논리 연산자가 다르게 동작한다

```
int a=10, b=20;  
System.out.println((a >= 20) & (b >= 20)); ←----- false 출력  
System.out.println((a >= 20) && (b >= 20)); ←----- false 출력
```

- 결과는 같지만, 실제 실행은 다르다
- 논리 연산자는 단락 평가 연산자로서 한 쪽을 평가하여 다른 한 쪽을 평가할 필요가 없는 경우 바로 결과를 반환하지만, 비트 논리 연산자는 그 경우에도 남은 부분을 수행한다



2 조건식

2-2 단락 평가 연산자

```
int a=10, b=20;
```

```
if (a >=20 & ++b >= 20) ←----- 두 번째 피연산자도 평가되어 b값이 21로 증가
```

```
    System.out.println("true");
```

```
else
```

```
    System.out.println("false"); ←----- false 출력
```

```
System.out.println(b); ←----- 21이 출력
```

```
int a=10, b=20;
```

```
if (a >=20 && ++b >= 20) ←----- 첫 번째 피연산자가 false이므로 두 번째 피연산자를 평가하지 않는다.
```

```
    System.out.println("true");
```

```
else
```

```
    System.out.println("false"); ←----- false 출력
```

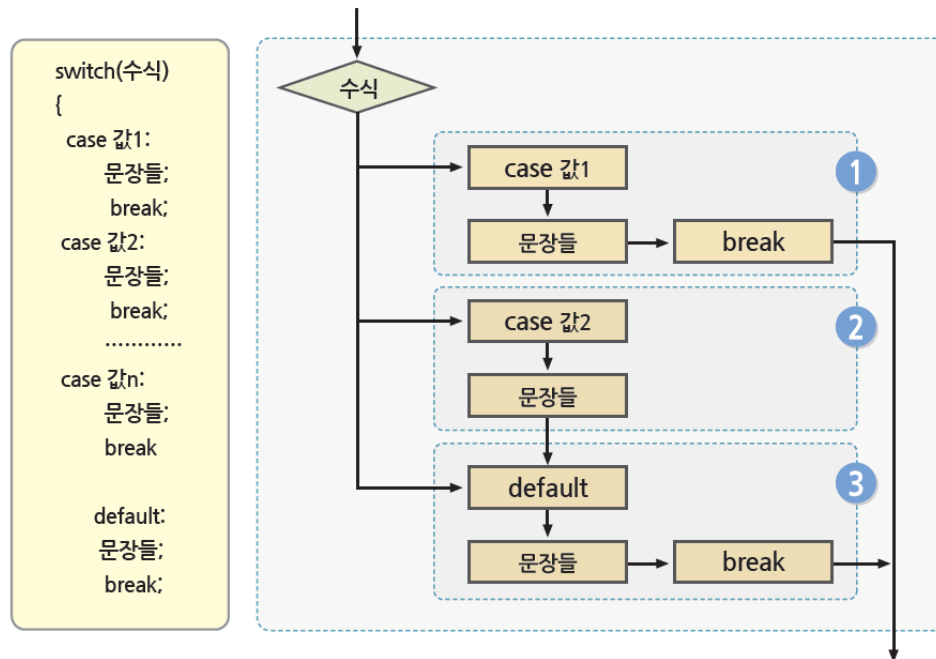
```
System.out.println(b); ←----- 20이 출력
```

Section 3.

switch문

● 다중 선택 기능을 제공하기 위한 switch문

- 정수값을 가지는 정수식에 따라 선택
- case 절의 정수값은 반드시 상숫값. 변수 사용 불가



① 번의 경우 : 가장 일반적인 **case**문으로 수식에 일치하면 문장들이 수행됩니다. **break**문이 있는 경우에는 수행하고 **case**문을 빠져나가게 됩니다.

② 번의 경우 : 수식에 일치하면 문장들이 수행됩니다. **break**문이 없는 경우입니다. 이 경우에는 바로 다음의 **case**문이 수행됩니다.

③ 번의 경우 : **case**문에서 **default**절은 필요에 따라 기술할 수도 있고 생략할 수도 있습니다. **default** 절은 수식의 결과가 앞의 **case**절에 해당되지 않을 경우 수행됩니다.

3 switch문

예제 4.11

SwitchTest1.java

```

01: import java.util.Scanner;
02: public class SwitchTest1 {
03:     public static void main(String args[])
04:     {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("월을 입력하세요 : ");
07:         int month = stdin.nextInt();
08:         String MtoS;
09:         switch (month) ← switch문에 정수식(month) 지정
10:         {
11:             case 12: ←
12:             case 1: ← 하나의 case문과 같은 효과
13:             case 2: ←
14:                 MtoS = "겨울입니다.";
15:                 break; ← switch문을 빠져나간다.
16:             case 3:
17:             case 4:
18:             case 5:
19:                 MtoS = "봄입니다.";

```


3 switch문

```

20:         break;
21:     case 6:
22:     case 7:
23:     case 8:
24:         MtoS = "여름입니다.";
25:         break;
26:     case 9: ←
27:         System.out.print("멋진 9월과 ");
28:     case 10:
29:         System.out.print("아름다운 10월과 "); ←
30:     case 11:
31:         System.out.print("낙엽의 11월은 ");
32:         MtoS = "가을입니다.";
33:         break;
34:     default: ← 1~12의 숫자가 아닐 경우 수행
35:         MtoS = "1~12월을 벗어난 달입니다.";
36:         break;
37: }
38: System.out.println(MtoS);
39: }
40: }
  
```

각각의 case문을 수행하고
아래 case문 실행

실행 결과 여러 번을 실행한 결과

```

월을 입력하세요 : 3
봄입니다.
월을 입력하세요 : 9
멋진 9월과 아름다운 10월과 낙엽의 11월은 가을입니다.
월을 입력하세요 : 11
낙엽의 11월은 가을입니다.
월을 입력하세요 : 13
1~12월을 벗어난 달입니다.
  
```

예제 4.12

SwitchTest2.java

```

01: import java.util.Scanner;
02: public class SwitchTest2 {
03:     public static void main(String args[])
04:     {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("월을 입력하세요(영문자) : ");
07:         String month = stdin.next();
08:         String MtoS;
09:         switch (month) ← switch문의 수식으로 문자열 변수를 지정
10:         {
11:             case "December": ←
12:             case "January": ← case문의 값으로 문자열을 지정
13:             case "February": ←
14:                 MtoS = "겨울입니다.";
15:                 break;
16:             case "March":

```

```

17:         case "April":
18:         case "May":
19:             MtoS = "봄입니다.";
20:             break;
21:         case "June":
22:         case "July":
23:         case "August":
24:             MtoS = "여름입니다.";
25:             break;
26:         case "September":
27:             System.out.print("멋진 9월과 ");
28:         case "October":
29:             System.out.print("아름다운 10월과 ");
30:         case "November":
31:             System.out.print("낙엽의 11월은 ");
32:             MtoS = "가을입니다.";
33:             break;
34:         default:
35:             MtoS = "1~12월을 벗어난 달입니다.";
36:             break;
37:     }
38:     System.out.println(MtoS);
39: }
40: }
  
```

실행 결과

2번 실행

월을 입력하세요(영문자) : January
겨울입니다.

월을 입력하세요(영문자) : march
1~12월을 벗어난 달입니다.

3 switch문

예제 4.13

SwitchTest3.java

```

01: public class SwitchTest3 {
02:     public static void main(String args[])
03:     {
04:         char c = 'A';  ← char형의 변수 선언과 초기화
05:         //char c = 'a'
06:         switch (c)  ← switch문의 수식으로 char형 변수를 지정
07:         {
08:             case 'A':  ← case문의 값으로 문자를 지정
09:             case 'B':  ←
10:                 System.out.print("우수한 성적입니다");
11:                 break;
12:             case 'C':
13:             case 'D':
14:                 System.out.print("분발하십시오");
15:                 break;
16:             case 'F':
17:                 System.out.print("많은 노력이 필요합니다");
18:                 break;
19:             default:
20:                 System.out.print("정확한 성적이 아닙니다");
21:         }
22:     }
23: }
  
```

실행 결과

2번 실행

우수한 성적입니다

정확한 성적이 아닙니다

Thank You!