



알기 쉽게 해설한
자바 프로그래밍 10판

Chapter 05. 반복문

학습목표

- 반복 논리를 대표하는 3가지 종류의 반복문을 학습합니다.
- while문, do-while문, for문에 대해 자세하게 학습합니다.
- 3가지 반복문의 비교와 중첩 사용에 관해 학습합니다.
- 제어를 이동시키기 위한 break문과 continue문을 학습합니다.
- 레이블 블록을 사용한 제어 이동에 관해 학습합니다.

목차

Section 1. 반복문의 개요

Section 2. while문

Section 3. do-while문

Section 4. for문

Section 5. 반복문의 비교와 중첩

Section 6. 제어의 이동

Section 1.

반복문의 개요



1 반복문의 개요

● 어떤 작업을 반복적으로 수행할 때

- 예 : 학생 성적의 평균을 구한다, 1부터 100까지의 합을 구한다

```
total = st1_koscore+st2_koscore+...생략....+st179_koscore+st180_koscore;
```

```
sum = 1+2+3+4+5+6+7+...생략...+97+98+99+100;
```

- 위와 같은 방법으로는 불가능

● 반복 논리를 제공하는 반복문을 사용

- 대부분의 프로그래밍 언어는 반복 논리를 표현할 수 있는 반복문 제공
- 대표적인 반복문 : While문, do-while문, for문

Section 2.

while문



2 While문

- while문 : 특정 조건이 만족하는 동안 지정된 영역을 반복적으로 수행할 때

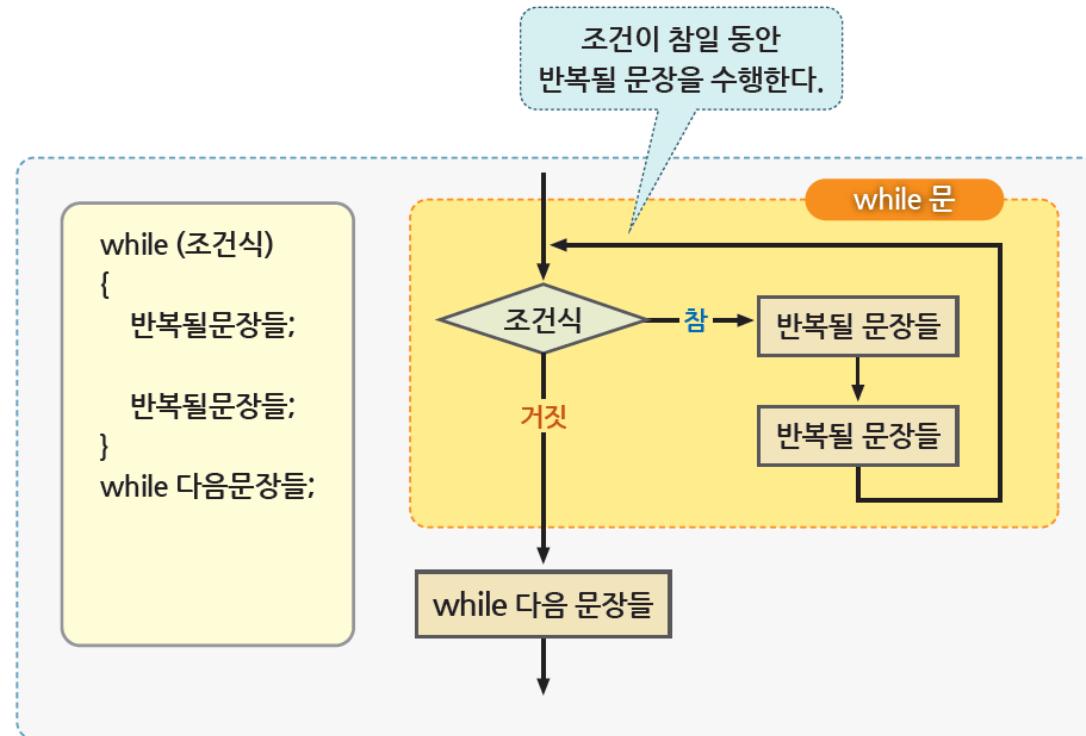


그림 5-1 while문



2 While문

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

```
while(a > b)
```

```
System.out.println("이 문장은 영원히 나타나지 않는다");
```

조건이 항상 거짓이기 때문에 수행될 수 없다.

```
while ( number <= 100 )
```

```
{
```

```
    sum = sum + number;
```

```
    number = number + 1;
```

조건을 변화시키는 문장이 반복 부분에
포함되는 것이 일반적이다.

```
    System.out.println(number);
```

```
}
```

```
boolean flag = true;
```

```
while ( flag )
```

```
{
```

```
    System.out.println("이 부분은 영원히 반복된다");
```

무한 반복된다.

```
}
```



2 While문

예제 5.1

WhileTest1.java

```
01: public class WhileTest1 {  
02:     public static void main(String args[]) {  
03:         {  
04:             int hap=0, count=1;  
05:             while (count <= 10)  
06:             { ←  
07:                 hap = hap + count; ← 조건이 참인 동안 수행되는 반복 블록  
08:                 count = count + 1; ← 조건을 변화시킨다.  
09:             } ←  
10:             System.out.println("1부터 10까지의 합은 "+ hap + " 입니다");  
11:     }  
12: }
```

실행 결과

1부터 10까지의 합은 55 입니다



2 While문

예제 5.2

WhileTest2.java

```
01: import java.util.Scanner;
02: public class WhileTest2 {
03:     public static void main(String args[])
04:     {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("원하는 단을 입력하세요 : ");
07:         int dan = stdin.nextInt();
08:         int x = 1;
09:         while (x <= 9)
10:         {
11:             System.out.println(dan + " * " + x + " = " + dan * x);
12:             x++;
13:         }
14:     }
15: }
```

실행 결과

34를 입력하여 실행

원하는 단을 입력하세요 : 34

34 * 1 = 34

34 * 2 = 68

34 * 3 = 102

34 * 4 = 136

34 * 5 = 170

34 * 6 = 204

34 * 7 = 238

34 * 8 = 272

34 * 9 = 306

반복하면서 구구단을 출력



2 While문

JAVA

예제 5.3

WhileTest3.java

```
01: import java.util.Scanner;
02: public class WhileTest3 {
03:     public static void main(String args[])
04:     {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("원하는 팩토리얼값을 입력 : ");
07:         int fac = stdin.nextInt();
08:         int facValue = fac;
09:         while (fac > 1)◀ 1보다 크면 계속 반복
10:         {
11:             System.out.print(fac +"*");
12:             facValue *= --fac;◀ 단축 연산자와 증감 연산자를 사용하여
13:             하나의 문장으로 축약
14:         }
15:         System.out.println("1="+ facValue);
16:     }
}
```

실행 결과 7을 입력한 경우

원하는 팩토리얼값을 입력 : 7
7*6*5*4*3*2*1=5040

Section 3.

do-while문

3 do-while문

- while문과 유사하지만, 반복을 먼저 수행하고 조건을 검사
(최소한 한번은 실행)

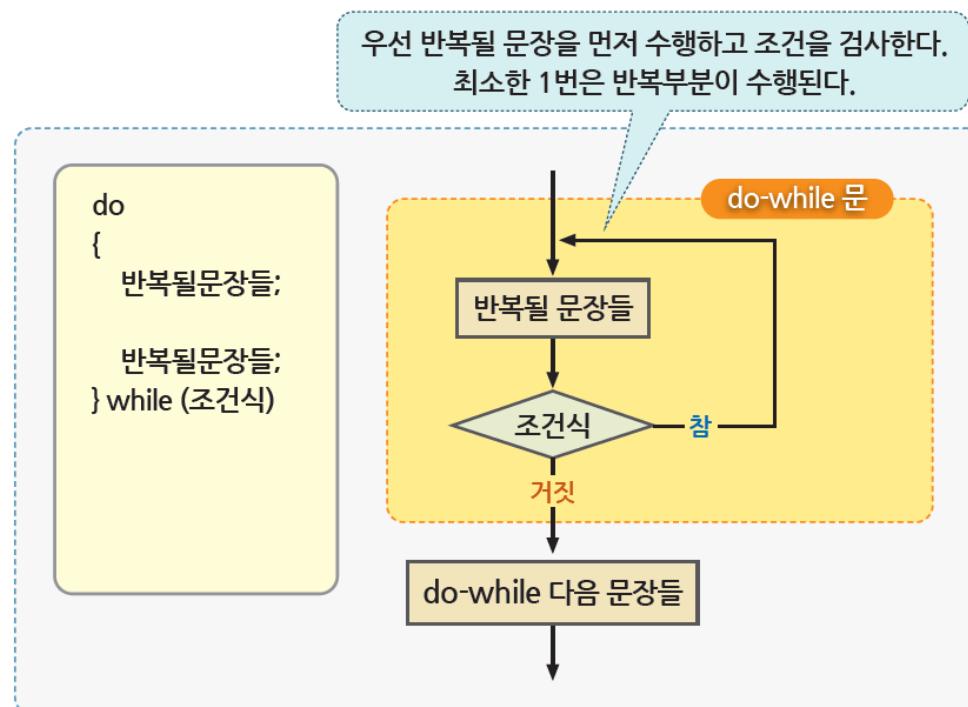


그림 5-2 do-while문



3 do-while문

```
int a = 10, b = 20;  
do  
    System.out.println("do-while문은 반복 부분이 최소한 한 번은 수행된다");  
    while (a > b);
```

do-while문은 반드시 조건 뒤에 세미콜론을 붙여야 한다.

```
do  
{  
    System.out.println("***** 메뉴 *****");  
    System.out.println("1. while 반복문");  
    System.out.println("2. do-while 반복문");  
    System.out.println("3. for 반복문");  
    System.out.println("끝 : 100입력");  
} while(input != 100);
```

do-while문은 아래와 같은 메뉴를 나타낼 때 적합



3 do-while문

예제 5.4

DoWhileTest1.java

```
01: public class DoWhileTest1 {  
02:     public static void main(String args[])  
03:     {  
04:         int hap=0, count=1;  
05:         do  
06:             { ←  
07:                 hap = hap + count; ← 반복 블록  
08:                 count = count + 1; ← 반복을 수행한 후에 조건을 검사하여 참이면 반복 계속.  
09:             } while (count <= 10); ← 끝에 세미콜론을 붙인다  
10:             System.out.println("1부터 10까지의 합은 "+ hap + " 입니다");  
11:     }  
12: }
```

실행 결과

1부터 10까지의 합은 55입니다



3 do-while문

예제 5.5 DoWhileTest2.java

```
01: import java.util.Scanner;
02: public class DoWhileTest2 {
03:     public static void main(String args[])
04:     {
05:         Scanner stdin = new Scanner(System.in);
06:         int month;
07:         do {
08:             System.out.print("월을 입력하세요(끝 : 0) : ");
09:             month = stdin.nextInt();           ← do문 내에서 메뉴를
10:             if (3 <= month && month <= 5 )   ← 나타내고 입력을 받는다.
11:                 System.out.println("봄 입니다");
12:             else if (6 <= month && month <= 8 )
13:                 System.out.println("여름 입니다");
14:             else if (9 <= month && month <= 11 )
15:                 System.out.println("가을 입니다");
16:             else if (1 == month || month == 2 || month == 12 )
17:                 System.out.println("겨울 입니다"); ← 입력된 값에 따라 계절을 처리
18:             else if (month != 0)           ← 계절에 해당이 안 되고 값이 0이 아니면 잘못된 입력
19:                 System.out.println("잘못된 입력 : 해당되는 계절이 없습니다");
20:         } while ( month != 0 );           ← 입력된 값이 0이 아니면 계속 반복
21:         System.out.println("감사합니다");
22:     }
23: }
```

실행 결과

```
월을 입력하세요(끝 : 0) : 3
봄 입니다
월을 입력하세요(끝 : 0) : 33
잘못된 입력 : 해당되는 계절이 없습니다
월을 입력하세요(끝 : 0) : 0
감사합니다
```



3 do-while문

예제 5.6

DoWhileTest3.java

```
01: import java.util.Scanner;
02: public class DoWhileTest3 {
03:     public static void main(String args[]) {
04:         int choice;
05:         Scanner stdin = new Scanner(System.in);
06:         do {
07:             System.out.println("== 반복문 종류 설명 =="); ←
08:             System.out.println(" 1. while 반복문");
09:             System.out.println(" 2. do-while 반복문");
10:             System.out.println(" 3. for 반복문");
11:             System.out.println("끝내시려면 99를 입력하세요");
12:             System.out.print("원하는 번호를 입력하세요 : ");
13:             choice = stdin.nextInt(); ←
```

메뉴를 보여 주고
입력을 받는다.



3 do-while문

```
14:         switch(choice) ←  
15:         {  
16:             case 1 :  
17:                 System.out.println("****while 반복문****");  
18:                 System.out.println("조건을 먼저 검사하고 조건이 참일 경우 반복  
부분을 수행하는 반복문");  
19:                 break;  
20:             case 2 :  
21:                 System.out.println("****do-while 반복문****");  
22:                 System.out.println("반복 부분을 먼저 수행하고 조건을 검사한다.  
최소한 한 번은 수행되는 반복문");  
23:                 break;  
24:             case 3 :  
25:                 System.out.println("****for 반복문****");  
26:                 System.out.println("지정된 횟수만큼 반복 부분을 수행하는  
반복문");  
27:                 break;  
28:             case 99 :  
29:                 System.out.println("사용해 주셔서 감사합니다");  
30:                 break;  
31:             default :  
32:                 System.out.println("숫자를 잘못 입력하셨습니다");  
33:         } ←  
34:         System.out.println();  
35:     } while( choice != 99 ); ←———— 입력된 값이 99가 아니면 다시 메뉴를 나타낸다.  
36: }  
37: }
```

———— switch문에서 입력 값에 따라 처리

———— 입력된 값이 99가 아니면 다시 메뉴를 나타낸다.



3 do-while문

실행 결과

==== 반복문 종류 설명 ====

1. while 반복문
2. do-while 반복문
3. for 반복문

끝내시려면 99를 입력하세요

원하는 번호를 입력하세요 : 1

****while 반복문****

조건을 먼저 검사하고 조건이 참일 경우 반복 부분을 수행하는 반복문

==== 반복문 종류 설명 ====

1. while 반복문
2. do-while 반복문
3. for 반복문

끝내시려면 99를 입력하세요

원하는 번호를 입력하세요 : 22

숫자를 잘못 입력하셨습니다

==== 반복문 종류 설명 ====

1. while 반복문
2. do-while 반복문
3. for 반복문

끝내시려면 99를 입력하세요

원하는 번호를 입력하세요 : 99

사용해 주셔서 감사합니다

Section 4.

for문

4 for문

- 일정한 패턴으로 증가 또는 감소하면서 지정된 횟수만큼 반복 수행

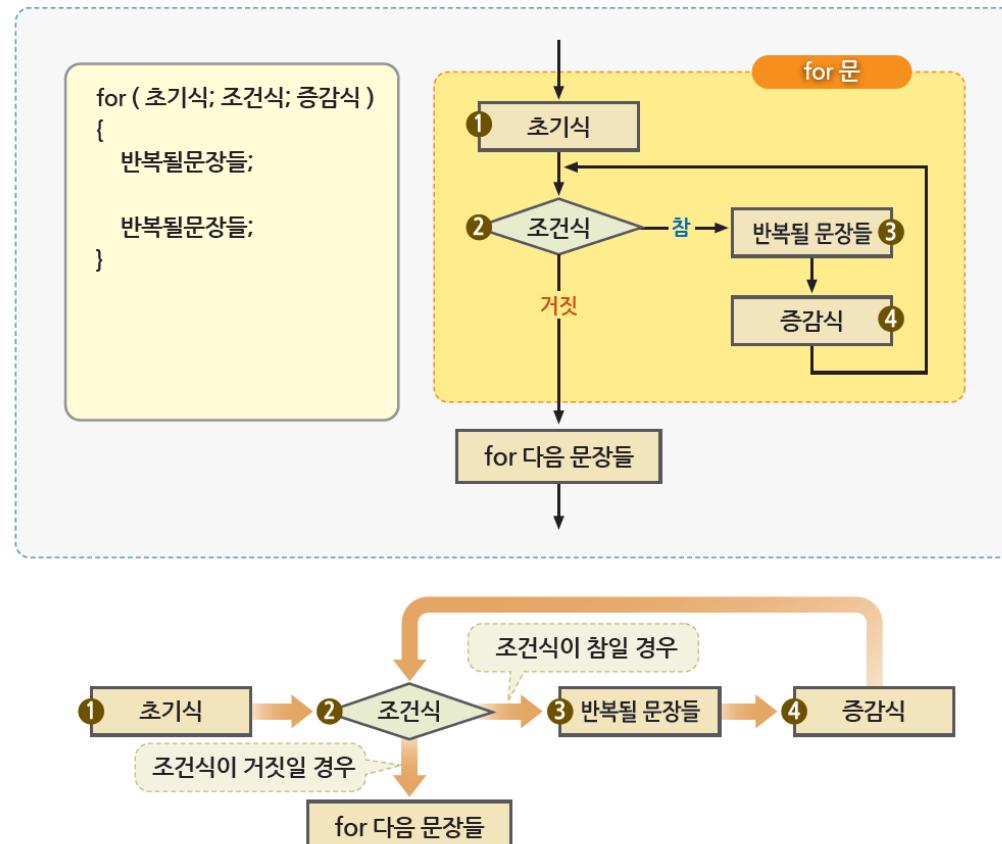


그림 5-3 for문

- 초기식 : 주로 반복 변수의 초기화를 위해 사용. 초기식은 처음 한 번만 수행
- 조건식 : 조건을 나타내며, 조건식이 참일 동안 반복 부분 수행
- 증감식 : 반복 부분을 수행한 후에 반드시 수행되는 문장. 주로 조건식에 변화를 주는 수식으로 구성



4 for문

```
int sum = 0;  
for (int i = 1 ; i <= 10 ; i = i + 1 ) ← 초기식에서 변수 선언 가능  
    sum = sum + i ; ← 초기식에서 선언된 변수는 for문 내에서만 사용 가능  
System.out.println("1부터 " + i + "까지의 합 = " + sum); ← 오류 발생 변수 i를 사용할 수 없다.
```

```
int a, b;  
for ( a=1, b=10 ; a < b ; a++, b-- ) ← 초기식과 증감식에 하나 이상의 문장이 올 수 있다.  
{                                         콤마로 분리하여 사용  
    System.out.println("a = " + a);  
    System.out.println("b = " + b);  
}
```



4 for문

```
boolean flag = false;  
int i = 1;  
for (; ! flag;) // 초기식과 증감식이 생략될 수 있다.  
{  
    System.out.println("i의 값은 " + i + "입니다");  
    if (i == 10) flag = true;  
    i++;  
}
```

```
for (;;) // 모두 생략될 수 있다. 무한 반복된다.  
{  
    .....  
}
```



4 for문

예제 5.7

ForTest1.java

```
01: public class ForTest1 {  
02:     public static void main(String args[])  
03:     {  
04:         int i, sum=0;  
05:         for (i = 1; i <= 10 ; i++) // for문의 초기식, 조건식, 증감식  
06:         {  
07:             sum = sum + i; // 반복 블록  
08:         }  
09:         System.out.println("1부터 10까지의 합은 " + sum + " 입니다 ");  
10:     }  
11: }
```

실행 결과

1부터 10까지의 합은 55 입니다



4 for문

예제 5.8

ForTest2.java

```
01: import java.util.Scanner;
02: public class ForTest2 {
03:     public static void main(String args[])
04:     {
05:         Scanner stdin = new Scanner(System.in); // 입력된 num까지 반복
06:         System.out.print("정수 입력 : ");
07:         int num = stdin.nextInt();
08:         System.out.print(num + "의 약수 : ");
09:         for (int i = 1; i <= num ; i++)
10:         {
11:             if (num % i == 0) //로 나누어 나머지가
12:                 System.out.print(i+" "); // 없으면 약수 출력
13:         }
14:     }
15: }
```

실행 결과

정수 입력 : 1024

1024의 약수 : 1 2 4 8 16 32 64 128 256 512 1024



4 for문

예제 5.9

ForTest3.java

```
01: import java.util.Scanner;
02: public class ForTest3 {
03:     public static void main(String args[])
04:     {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("두 개의 숫자를 입력하세요(공백으로 구분) : ");
07:         int m = stdin.nextInt();
08:         int n = stdin.nextInt();
09:         int i;
10:         for( i=1; i<=m*n ; i++ ) ← m*n보다 작을 때까지 반복
11:         {
12:             if((i%n == 0) && (i%m == 0)) ← 두 수로 모두 나누어지면 만족
13:                 break; ← break를 둘러싸고 있는 반복문을 벗어난다.
14:             }
15:             System.out.print("최소 공배수는 " + i + "입니다. ");
16:             for( i=n; i>=1 ; i-- )
17:             {
18:                 if((m%i == 0) && (n%i == 0)) ← 최대 공약수를 구하기 위한 선택문
19:                     break; ← 반복문을 벗어나 2번 문장 수행
20:                 }
21:                 if(i==1) ← i가 10이면 최대 공약수가 없다.
22:                     System.out.print("최대 공약수가 없습니다.");
23:                 else
24:                     System.out.print("최대 공약수는 " + i + "입니다. ");
25:     }
```

실행 결과

두 번을 실행

두 개의 숫자를 입력하세요(공백으로 구분) : 5 15

최소 공배수는 15입니다. 최대 공약수는 5입니다.

두 개의 숫자를 입력하세요(공백으로 구분) : 3 7

최소 공배수는 21입니다. 최대 공약수가 없습니다.

Section 5.

반복문의 비교와 중첩

5 반복문의 비교와 중첩

5-1 반복문의 비교

- 3개의 반복문이 조금씩 다른 특성을 가짐

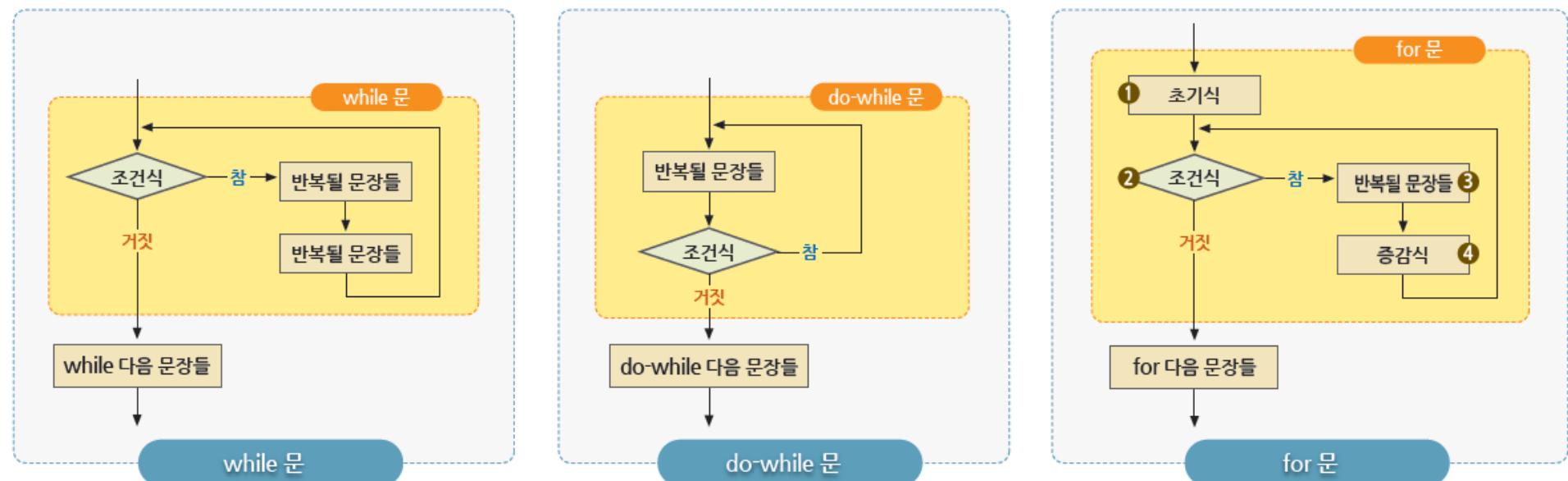


그림 5-4 3개의 반복문 구조



5 반복문의 비교와 중첩

5-1 반복문의 비교

for문의 예

```
for ( j=1 ; j < 0 ; j++ )
```

```
    System.out.println("한 번도 수행 안 됨);
```

조건이 거짓이 되어 수행 안 됨

while문의 예

```
j=1;
```

```
while( j < 0 )
```

```
    System.out.println("한 번도 수행 안 됨);
```

조건이 거짓이 되어 수행 안 됨

do-while문의 예

```
j=1;
```

```
do {
```

```
    System.out.println("적어도 한 번은 수행됨);
```

조건이 나중에 검사되므로 적어도 한 번은 수행됨

```
}while ( j < 0 );
```

5 반복문의 비교와 중첩

5-2 반복의 중첩

- 반복문은 선택문과 마찬가지로 중첩될 수 있습니다

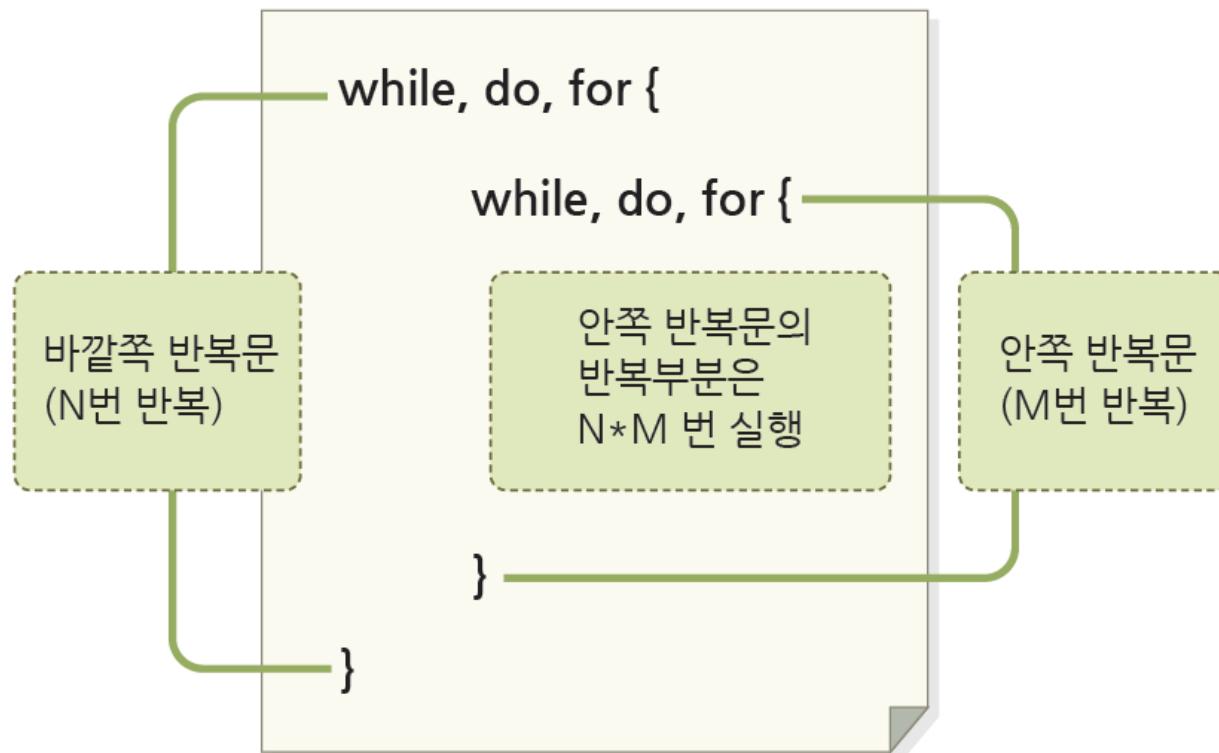


그림 5-5 반복문의 내포된 형태



5 반복문의 비교와 중첩

5-2 반복의 중첩

예제 5.10

NestedLoopTest1.java

```
01: public class NestedLoopTest1{  
02:     public static void main(String args[ ]) {  
03:         {  
04:             int i=2;  
05:             while ( i <= 9 ) // 9단까지 반복  
06:             {  
07:                 System.out.println("*** "+ i + "단 ***"); // 단의 제목을 출력  
08:                 for (int j=1; j<=9 ; j++) // 각 단에 대하여 1부터 9까지의 구구단을 출력  
09:                 {  
10:                     System.out.println((i+" * " + j + " = " + i*j));  
11:                 }  
12:                 System.out.println();  
13:                 i++; // 단을 증가시킨다  
14:             }  
15:         }  
16:     }
```

실행 결과

*** 2단 ***

2 * 1 = 2

2 * 2 = 4

2 * 3 = 6

..... 생략

9 * 7 = 63

9 * 8 = 72

9 * 9 = 81



5 반복문의 비교와 중첩

5-2 반복의 중첩

예제 5.11

NestedLoopTest2.java

```
01: import java.util.Scanner;
02: public class NestedLoopTest2{
03:     public static void main(String args[])
04:     {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("원하는 숫자를 입력하세요 : ");
07:         int num = stdin.nextInt();
08:         int i=1;
09:         while ( i <= num ) ←———— 입력된 숫자만큼 반복
10:         {
11:             int j=1; ←———— 번수 j를 1로 초기화
12:             while ( j <= i ) ←———— 바깥 변수 i 만큼 반복
13:             {
14:                 System.out.print("*");
15:                 j++;
16:             }
17:             System.out.println(); ←———— 라인을 바꾼다.
18:             i++;
19:         }
20:     }
21: }
```

실행 결과

원하는 숫자를 입력하세요 : 9

```
*
```

```
**
```

```
***
```

```
****
```

```
*****
```

```
*****
```

```
*****
```

```
*****
```

```
*****
```

Section 6.

제어의 이동



6 제어의 이동

● 제어를 이동 : 프로그램의 실행 순서를 인위적으로 변경

- 자바 언어는 제어를 이동하기 위해 제한된 형태의 명령어인 break, continue, return 문을 제공
- 프로그램의 제어가 자유롭게 이동되는 것을 허용하면, 프로그램의 구조가 난해해지고 스파게티(spaghetti) 코드가 될 수 있다
- 최근의 프로그래밍 언어에서는 제한된 형태의 제어 이동만 허용한다



6 제어의 이동

6-1 레이블 블록

- 프로그램의 특정 부분을 레이블 블록으로 지정할 수 있다

- break, continue문은 단독으로 사용될 수도 있지만, 레이블 블록과 같이 사용될 수도 있다

[형식] 레이블 블록

레이블 명 : {문장 블록}



6 제어의 이동

6-1 레이블 블록

[예]

```
aa : { ← aa 레이블 블록 선언
        System.out.println("block aa");
    }

bb : { ← bb 레이블 블록 선언. cc 레이블 블록이 내포됨
        System.out.println("block bb");
        cc : { ← cc 레이블 블록 선언
            System.out.println("block cc");
        }
    }

dd : for ( j =1 ; j < 10 ; j++) { ← 반복문을 포함하는 dd 레이블 블록 선언
        sum = sum + j
    }

ee : while ( j < 10 ) { ← 반복문을 포함하는 ee 레이블 블록. ff 블록이 내포됨
        System.out.println(j+"ee 블록");
        ff : while ( k < 10 ) { ← 반복문을 포함하는 ff 레이블 블록 선언
            .....
        }
    }
```



6 제어의 이동

6-2 break문

- 4장의 switch문에서 break 문이 실행되면 프로그램의 실행이 switch문을 벗어난다
- 반복문이나 레이블 블록문 내에서 break문이 사용되면, 역시 반복문과 레이블 블록을 벗어난다



6 제어의 이동

6-2 break문

● 레이블이 없는 break문의 사용

```
while (true) {
    if ( j == 10 ) break; ← while문 밖으로 제어가 이동
    sum = sum + j;
    j++
}

for (j = 1 ; j < 10 ; j++) {
    for (k = 1 ; k < 10 ; k++) {
        if ( k == 5 ) break; ← break를 내포하는 반복문 밖으로 제어가 이동
        .....
    }
    System.out.println(k);
}

aa : {
    .....
    if ( a == 10 ) break; ← 오류 발생. 반복문 내에서만 단독 사용 가능
    .....
}
```

단, 레이블 블록 내에서는 레이블과 같이 사용 가능



6 제어의 이동

6-2 break문

예제 5.12

BreakTest1.java

```
01: import java.util.Scanner;
02: public class BreakTest1 {
03:     public static void main(String args[]) {
04:         Scanner stdin = new Scanner(System.in);
05:         System.out.print("합계를 원하는 정수 입력 : ");
06:         int num = stdin.nextInt();
07:         int sum = 0, i = 1;
08:         while (true) { ← 무한 반복문 사용
09:             sum = sum + i;
10:             if (i == num) break; ← i 값이 사용자가 입력한 값과 같으면 반복을 벗어남
11:             i++;
12:         }
13:         System.out.println(num+"까지의 합계는 = "+ sum);
14:     }
15: }
```

실행 결과

합계를 원하는 정수 입력 : 1000

1000까지의 합계는 = 500500



6 제어의 이동

6-2 break문

예제 5.13

BreakTest2.java

```
01: public class BreakTest2 {  
02:     public static void main(String args[])  
03:     {  
  
04:         int i, j;  
05:         for(i=1 ; i<10 ; i++)  
06:         {  
07:             for(j=1 ; j<i ; j++)  
08:             {  
09:                 if (j > 6) break; ←----- j 값이 6보다 크면 내포된 반복문을 빠져난다.  
10:                 System.out.print(" * ");  
11:             }  
12:             System.out.println(); ←----- 라인을 바꾼다.  
13:         }  
14:     }  
15: }
```

실행 결과

```
*  
* *  
* * *  
* * * *  
* * * * *  
* * * * * *  
* * * * * *
```



6 제어의 이동

6-2 break문

● 레이블이 있는 break문의 사용

```
aa : for (j = 1 ; j < 10 ; j++) {  
    bb : for (k = 1 ; k < 10 ; k++) {  
        if ( j == 5 && k == 5 ) break aa; ← aa 블록 밖으로 제어 이동  
        .....  
    }  
    System.out.println(k);  
}  
  
cc : {  
    .....  
    if ( a == 10 ) break cc; ← cc 블록 밖으로 제어 이동. 레이블 블록에서 사용될 경우에는  
    반드시 레이블명을 지정해야 한다.  
}  
  
dd : {  
    .....  
    ee : {  
        .....  
        if ( a == 5 ) break dd; ← dd 블록 밖으로 제어 이동  
        .....  
    }  
}
```



6 제어의 이동

6-2 break문

- 레이블이 있는 break문의 사용

```
ff : {  
    .....  
}  
  
gg : {  
    .....  
    hh : {  
        .....  
        if ( a == 10 ) break ff;   
    }  
}
```

ff 블록

오류 발생. 내포된 블록만 break 사용 가능



6 제어의 이동

6-2 break문

예제 5.14

BreakLabelTest1.java

```
01: public class BreakLabelTest1 {  
02:     public static void main(String args[])  
03:     {  
04:         boolean t = true;  
05:         First:{ ←  
06:             Second:{ ← 3개의 레이블 블록 선언  
07:             Third:{ ←  
08:                 System.out.println("Third 블록 'break' 문장 전");  
09:                 if(t) break Second; ← 무조건 Second 블록을 벗어남  
10:                 System.out.println("Third 블록 'break' 문장 후");  
11:             }  
12:             System.out.println("Second 블록 문장");  
13:         }  
14:         System.out.println("First 블록 문장");  
15:     }  
16:     System.out.println("main 블록 문장");  
17: }  
18: }
```

실행 결과

Third 블록 'break' 문장 전
First 블록 문장
main 블록 문장



6 제어의 이동

6-2 break문

예제 5.15

BreakLabelTest2.java

```
01: public class BreakLabelTest2{
02:     public static void main(String args[])
03:     {
04:         int i=2,j;
05:         Loop : while(true) <----- 반복문 블록에 레이블 지정
06:         {
07:             j=1; <----- 9단까지만 제목을 출력
08:             if ( i < 10 ) <----- 내포된 반복문 블록에 레이블 지정
09:                 System.out.println("\n==== " + i + "단 ===");
10:             Innerloop : while(true) <----- 내포된 반복문 블록에 레이블 지정
11:             {
12:                 if ( j > 9 ) break; <----- 내포된 반복문만 벗어난다.
13:                 if ( i > 9 ) break Loop; <----- Loop 블록을 벗어난다.
14:                 System.out.println(i+" * "+ j + " = " + i*j);
15:                 j++;
16:             }
17:             System.out.println();
18:             i++;
19:         }
20:     }
21: }
```

실행 결과

```
==== 2단 ===
2 * 1 = 2
2 * 2 = 4
.....생략
9 * 8 = 72
9 * 9 = 81
```



6 제어의 이동

6-3 continue문

- **continue문은 프로그램의 제어를 반복 블록 처음으로 이동시킨다**
- **continue문은 반복문 안에서만 사용**
- **레이블이 없는 continue문의 사용**

```
for (j = 1; j < 10; j++)  
    if ( j % 2 == 0 ) continue; ←----- 제어를 반복문의 처음으로 이동  
    sum = sum + j;  
}  
  
aa : {  
.....  
    if ( j % 2 == 0 ) continue; ←----- 오류 발생. continue는 반복문이 있는 블록에서만 사용 가능  
.....  
}
```



6 제어의 이동

6-3 continue문

예제 5.16

ContinueTest1.java

```
01: import java.util.Scanner;
02: public class ContinueTest1 {
03:     public static void main(String args[])
04:     {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("원하는 정수 입력(짝수의 합) : ");
07:         int num = stdin.nextInt();
08:         int i, sum=0;
09:         for (i=1 ; i <= num ; i++)
10:             if (i % 2 == 1) continue; ←값이 홀수값이면 더하지 않고
11:             sum = sum + i;           반복문의 처음으로 제어 이동
12:         }
13:         System.out.println("1부터 " + num + "까지 짝수의 합 = " + sum);
14:     }
15: }
```

실행 결과

```
원하는 정수 입력(짝수의 합) : 100
1부터 100까지 짝수의 합 = 2550
```



6 제어의 이동

6-3 continue문

● 레이블이 있는 continue문의 사용

- 중첩된 반복문이 레이블로 지정되어 있을 때 사용 가능

```
aa : for (k = 1; j < 20; k++) {  
    .....  
    bb : for (j = 1; j < 20; j++) {  
        .....  
        if ( j % 2 == 0 ) continue; <----- 제어를 내포한 반복문의 처음으로 이동  
        .....  
        if ( k == 10 ) continue aa; <----- 제어를 aa 블록의 처음으로 이동  
        .....  
    }  
}  
  
cc : {  
    .....  
    dd : {  
        .....  
        if ( j % 2 == 0 ) continue cc; <----- 오류 발생. 반복문 블록에서만 사용 가능  
    }  
}
```



6 제어의 이동

6-3 continue문

예제 5.17

ContinueLabelTest1.java

```
01: public class ContinueLabelTest1 {  
02:     public static void main(String args[])  
03:     {  
04:         int i,j;  
05:         Outer : for (i=2; i<=9 ; i=i+1) <----- Outer 반복 블록 선언  
06:         {  
07:             System.out.println("== " + i + "단 ==");  
08:             for(j=1; j<=9; j=j+1)  
09:             {  
10:                 if (j == 3) continue Outer; <----- Outer 블록으로 제어를 이동  
11:                 System.out.println(i+" * " + j + " = " + i*j);  
12:             }  
13:         }  
14:     }  
15: }
```

실행 결과

```
*** 2단 ***  
2 * 1 = 2  
2 * 2 = 4  
*** 3단 ***  
3 * 1 = 3  
.....생략
```

Thank You!