

# Java

## 제 5 강 반복문

### 2. 반복문

2.1 반복문(for, while, do-while)

2.2 for문

2.3 중첩 for문

2.4 while문

2.5 중첩 while문

2.6 do-while문

2.6 do-while문

2.7 break문

2.8 continue문

2.9 이름 붙은 반복문과  
break, continue

# 1. 반복문(for, while, do-while)

### 2.1 반복문 – for, while, do-while

- 문장 또는 문장들을 반복해서 수행할 때 사용
- 조건식과 수행할 블록{} 또는 문장으로 구성
- 반복회수가 중요한 경우에 for문을 그 외에는 while문을 사용한다.
- for문과 while문은 서로 변경가능하다.
- do-while문은 while문의 변형으로 블록{}이 최소한 한번은 수행될 것을 보장한다.

```
System.out.println(1);  
System.out.println(2);  
System.out.println(3);  
System.out.println(4);  
System.out.println(5);
```

```
for(int i=1;i<=5;i++) {  
    System.out.println(i);  
}
```

```
int i=0;  
  
do {  
    i++;  
    System.out.println(i);  
} while(i<=5);
```

```
int i=1;  
  
while(i<=5) {  
    System.out.println(i);  
    i++;  
}
```

## 2.2 for문

- 초기화, 조건식, 증감식 그리고 수행할 블록{} 또는 문장으로 구성

```
for (초기화;조건식;증감식) {
    // 조건식이 true일 때 수행될 문장들을 적는다.
}
```

[참고] 반복하려는 문장이 단 하나일 때는 중괄호{}를 생략할 수 있다.



예) 1부터 10까지의 정수를 더하기

```
int sum = 0;

for(int i=1; i<=10; i++) {
    sum += i; // sum = sum + i;
}
```

i

sum

i	sum
1	
2	
3	
4	
...	
10	

## 2.2 for문 – 작성예(examples)

for문 작성 예	설 명
<pre>for(;;) {     /* 반복해서 수행할 </pre>	조건식이 없기 때문에 결과가 true로 간주되
<pre>for(int i=0;;) {     /* 반복해서 수행할 </pre>	<pre>public static void main(String[] args) {     int sum = 0;     int i;      for(i=1; i&lt;=10;i++) {         sum += i; // sum = sum + i;     }      System.out.println(i-1 + "까지의 합: " + sum); }</pre>
<pre>for(int i=1,j=1;i&lt;10 &amp; {     /* 반복해서 수행할 </pre>	
<pre>for(int i=1,j=1;i&lt;10 &amp; {     /* 반복해서 수행할 </pre>	

```
public static void main(String[] args)
{
    int sum = 0

    for(int i=1; i <= 10; i++) {
        sum += i ;          // sum = sum + i;
    }
    System.out.println( i-1 + " 까지의 합: " + sum);
}
```

변수 sum 이  
유효한 범위

변수 i가  
유효한 범위

←에러발생

## 2.3 중첩for문

- for문 안에 또 다른 for문을 포함시킬 수 있다.
- for문의 중첩횟수에는 거의 제한이 없다.

```
for(int i=2; i<=9; i++) {
    for(int j=1; j<=9; j++) {
        System.out.println(i+" * "+j+" = "+i*j);
    }
}
```

↓

```
for(int i=2; i<=9; i++)
    for(int j=1; j<=9; j++)
        System.out.println(i+" * "+j+" = "+i*j);
```

i \* j = i\*j

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
...
2 * 9 = 18
3 * 1 = 3
3 * 2 = 6
...
9 * 8 = 72
9 * 9 = 81
```

ijk

```
111
112
113
121
122
123
...
331
332
333
```

```
for(int i=1; i<=3; i++) {
    for(int j=1; j<=3; j++) {
        for(int k=1; k<=3; k++) {
            System.out.println(" "+i+j+k);
        }
    }
}
```

→

```
for(int i=1; i<=3; i++)
    for(int j=1; j<=3; j++)
        for(int k=1; k<=3; k++)
            System.out.println(" "+i+j+k);
```

### 2.4 while문

- 조건식과 수행할 블록{} 또는 문장으로 구성

```
while (조건식) {  
    // 조건식의 연산결과가 true일 때 수행될 문장들을 적는다.  
}
```

```
int i=10;  
  
while(i >= 0) {  
    System.out.println(i--);  
}
```



```
for(int i=10;i>=0;i--) {  
    System.out.println(i);  
}
```

```
int i=0;  
  
while(i <= 10) {  
    i=10;  
    System.out.println(i--);  
}
```

```
int i=10;  
  
while(i > 0) {  
    System.out.println(i--);  
}
```



### 2.5 중첩while문

- while문 안에 또 다른 while문을 포함시킬 수 있다.
- while문의 중첩횟수에는 거의 제한이 없다.

```
for(int i=2; i<=9; i++) {  
    for(int j=1; j<=9; j++) {  
        System.out.println(i+" * "+j+" = "+i*j);  
    }  
}
```



```
int i=2;  
while(i <= 9) {  
    int j=1;  
    while(j <= 9) {  
        System.out.println(i+" * "+j+" = "+i*j);  
        j++;  
    }  
    i++;  
}
```

## 2.6 do-while문

- while문의 변형. 블록{}을 먼저 수행한 다음에 조건식을 계산한다.
- 블록{}이 최소한 1번 이상 수행될 것을 보장한다.

```
do {
    // 조건식의 연산결과가 true일 때 수행될 문장들을 적는다.
} while (조건식);
```

```
class FlowEx24 {
    public static void main(String[] args) throws java.io.IOException {
        int input=0;

        System.out.println("문장을 입력하세요.");
        System.out.println("입력을 마치려면 x를 입력하세요.");
        do {
            input = System.in.read();
            System.out.print((char)input);
        } while(input!=-1 && input !='x');
    }
}
```

문자	코드
...	...
A	65
B	66
C	67
...	...
a	97
b	98
c	99
...	...
x	120
...	...

### 2.7 break문

- 자신이 포함된 하나의 반복문 또는 switch문을 빠져 나온다.
- 주로 if문과 함께 사용해서 특정 조건을 만족하면 반복문을 벗어나게 한다.

```
class FlowEx25
{
    public static void main(String[] args)
    {
        int sum = 0;
        int i = 0;

        while(true) {
            if(sum > 100)
                ● break;
            i++;
            sum += i;
        } // end of while

        System.out.println("i=" + i);
        System.out.println("sum=" + sum);
    }
}
```

break문이 수행되면 이 부분은 실행되지 않고 while문을 완전히 벗어난다.

i	sum
0	0
1	1
2	3
3	6
...	...
13	91
14	105

### 2.8 continue문

- 자신이 포함된 반복문의 끝으로 이동한다.(다음 반복으로 넘어간다.)
- continue문 이후의 문장들은 수행되지 않는다.

```
class FlowEx26
{
    public static void main(String[] args)
    {
        for(int i=0;i <= 10;i++) {
            if (i%3==0)
                ● continue; ●
            System.out.println(i);
        }
    }
}
```

조건식이 true가 되어 continue문이 수행되면 반복문의 끝으로 이동한다.  
break문과 달리 반복문 전체를 벗어나지 않는다.

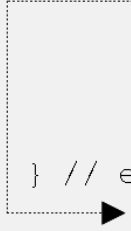
#### [실행결과]

1  
2  
4  
5  
7  
8  
10

### 2.9 이름 붙은 반복문과 break, continue

- 반복문 앞에 이름을 붙이고, 그이름을 break, continue와 같이 사용함으로써 둘 이상의 반복문을 벗어나거나 반복을 건너뛰는 것이 가능하다.

```
class FlowEx27
{
    public static void main(String[] args)
    {
        // for문에 Loop1이라는 이름을 붙였다.
        Loop1 : for(int i=2; i <=9; i++) {
            for(int j=1; j <=9; j++) {
                if(j==5)
                ● break Loop1;
                System.out.println(i+"*"+ j +"="+ i*j);
            } // end of for i
            System.out.println();
        } // end of Loop1
    }
}
```



#### [실행결과]

```
2*1=2
2*2=4
2*3=6
2*4=8
```

## 3.1 키보드입력을 받는 방법

```
1 import java.util.Scanner;
2
3 public class KeyInput {
4
5     public static void main(String[] args) {
6
7         Scanner scan = new Scanner(System.in);
8         System.out.print("이름을 입력하세요: ");
9         String name = scan.nextLine();
10
11         System.out.print("정수를 입력하세요?");
12         int number = scan.nextInt();
13
14         System.out.print("실수를 입력하세요?");
15         double real = scan.nextDouble();
16
17         System.out.println("이름은 " + name);
18         System.out.println("정수는 " + number);
19         System.out.println("실수는 " + real);
20
21     }
22
23 }
```

## 3.2 사용자 입력받기 – 입력창(InputDialog)

- ▶ Swing패키지의 JOptionPane.showInputDialog()를 사용

**[예제5-16]**/ch5/ArrayEx16.java

```
import javax.swing.*; // JOptionPane클래스를 사

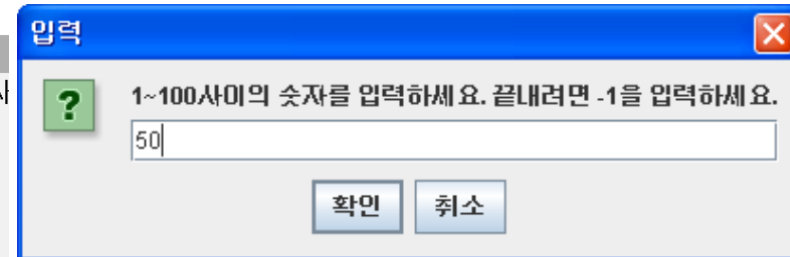
class ArrayEx16 {
    public static void main(String[] args)
    {
        // 1~100사이의 임의의값을 얻어서 answer에 저장한다.
        int answer = (int) (Math.random() * 100) + 1;
        int input = 0; // 사용자입력을 저장할 공간
        String temp = ""; // 사용자입력을 저장할 임시공간
        int count = 0; // 시도횟수를 세기위한 변수

        do {
            count++;
            temp = JOptionPane.showInputDialog("1~100사이의 숫자를 입력하세요."
                                                + " 끝내려면 -1을 입력하세요.");

            // 사용자가 취소버튼을 누르거나 -1을 입력하면 do-while문을 벗어난다.
            if(temp==null || temp.equals("-1")) break;

            System.out.println("입력값 : "+temp);

            // 사용자입력을 문자열로 받아오기 때문에 int로 변환해 주어야한다.
            input = Integer.parseInt(temp);
        } while (count < 10);
    }
}
```



【그림5-4】 JOptionPane.showInputDialog()에 의해서 생성된 입력창

감사합니다.