

Java의 정석

제 4 장

조건문과 반복문



1. 조건문

1.1 조건문(if, switch)

1.7 Math.random()

1.2 if문

1.3 중첩 if문

1.4 switch문

1.5 중첩 switch문

1.6 if문과 switch문의 비교

2. 반복문

2.1 반복문(for, while, do-while)

2.2 for문

2.6 do-while문

2.3 중첩 for문

2.7 break문

2.4 while문

2.8 continue문

2.5 중첩 while문

2.9 이름 붙은 반복문과

2.6 do-while문

break, continue

1. 조건문(if, switch)

1.1 조건문 – if, switch

```
if(조건식) { 문장들 }
```

- 조건문은 조건식과 실행될 하나의 문장 또는 블록{}으로 구성
- Java에서 조건문은 if문과 switch문 두 가지 뿐이다.
- if문이 주로 사용되며, 경우의 수가 많은 경우 switch문을 사용할 것을 고려한다.
- 모든 switch문은 if문으로 변경이 가능하지만, if문은 switch문으로 변경할 수 없는 경우가 많다.

```
if(num==1) {  
    System.out.println("SK");  
} else if(num==6) {  
    System.out.println("KTF");  
} else if(num==9) {  
    System.out.println("LG");  
} else {  
    System.out.println("UNKNOWN");  
}
```



```
switch(num) {  
    case 1:  
        System.out.println("SK");  
        break;  
    case 6:  
        System.out.println("KTF");  
        break;  
    case 9:  
        System.out.println("LG");  
        break;  
    default:  
        System.out.println("UNKNOWN");  
}
```

1.2 if문

- if문은 if, if-else, if-else if의 세가지 형태가 있다.
- 조건식의 결과는 반드시 true 또는 false이어야 한다.

```
if(조건식) {  
    // 조건식의 결과가 true일 때 수행될 문장들  
}
```

```
if(조건식) {  
    // 조건식의 결과가 true일 때 수행될 문장들  
} else {  
    // 조건식의 결과가 false일 때 수행될 문장들  
}
```

```
if(조건식1) {  
    // 조건식1의 결과가 true일 때 수행될 문장들  
} else if(조건식2) {  
    // 조건식2의 결과가 true일 때 수행될 문장들  
    // (조건식1의 결과는 false)  
} else if(조건식3) {  
    // 조건식3의 결과가 true일 때 수행될 문장들  
    // (조건식1과 조건식2의 결과는 false)  
} else {  
    // 모든 조건식의 결과가 false일 때 수행될 문장들  
}
```

```
if(score > 60) {  
    System.out.println("합격입니다.");  
}
```

```
if(score > 60)  
    System.out.println("합격입니다.");
```

```
if(score > 60) {  
    System.out.println("합격입니다.");  
} else {  
    System.out.println("불합격입니다.");  
}
```

```
if(num > 0) {  
    System.out.println("양수입니다.");  
} else if(num < 0) {  
    System.out.println("음수입니다.");  
} else {  
    System.out.println("영입니다.");  
}
```

```
if(score >= 90) {  
    System.out.println("A등급");  
} else if(score >= 80 && score < 90) { // 80<=score<90  
    System.out.println("B등급");  
} else if(score >= 70 && score < 80) { // 70<=score<80  
    System.out.println("C등급");  
} else { // score < 70  
    System.out.println("F등급");  
}
```

1.2 if문 - 조건식의 예(example)

```
int i = 0;
if(i%2==0) { }
if(i%3==0) { }

String str = "";
char ch = ' ';
if(ch==' ' || ch=='\t') { }
if(ch=='c' || ch=='C') { }
if(str=="c" || str=="C") { }
if(str.equals("c") || str.equals("C")) { }
if(str.equalsIgnoreCase("c")) { }

if(ch>='0' && ch<='9') { }
if(!(ch>='0' && ch<='9')) { }
if(ch<'0' || ch>'9')) { }
```

```
if(('a'<=ch && ch<='z') ||
   ('A'<=ch && ch<='Z')) { }
```

```
if( i<-1 || i>3 && i<5 ) { }
```

```
str="3"; 문자열 "3" → 문자 '3'
if(str!=null && !str.equals("")) {
    ch = str.charAt(0);
}

boolean powerOn=false;
if(!powerOn) {
    // 전원이 꺼져있으면...
}
```

1.3 중첩 if문

- if문 안에 또 다른 if문을 중첩해서 넣을 수 있다.
- if문의 중첩횟수에는 거의 제한이 없다.

```
if (조건식1) {  
    // 조건식1의  
    if (조건식2)  
        // 조건식2의  
    } else {  
        // 조건식2의  
    }  
} else {  
    // 조건식1의  
}
```

```
if (score >= 90) { // score가 90점 보다 같거나 크면 A학점 (grade)  
    grade = "A";  
  
    if ( score >= 98) { // 90점 이상 중에서도 98점 이상은 A+  
        grade += "+"; // grade = grade + "+";  
    } else if ( score < 94) {  
        grade += "-";  
    }  
} else if (score >= 80) { // score가 80점 보다 같거나 크면 B학점 (grade)  
    grade = "B";  
  
    if ( score >= 88) {  
        grade += "+";  
    } else if ( score < 84) {  
        grade += "-";  
    }  
} else { // 나머지는 C학점 (grade)  
    grade = "C";  
}
```

1.4 switch문

- if문의 조건식과 달리, 조건식의 계산결과가 int범위 이하의 정수만 가능
- 조건식의 계산결과와 일치하는 case문으로 이동 후 break문을 만날 때까지 문장들을 수행한다.(break문이 없으면 switch문의 끝까지 진행한다.)
- 일치하는 case문의 값이 없는 경우 default문으로 이동한다.
(default문 생략가능)
- case문의 값으로 변수를 사용할 수 없다.(리터럴, 상수만 가능)

```
switch (조건식) {  
    case 값1 :  
        // 조건식의 결과가 값1과 같을 경우 수행될 문장들  
        //...  
        break;  
    case 값2 :  
        // 조건식의 결과가 값2와 같을 경우 수행될 문장들  
        //...  
        break;  
    //...  
    default :  
        // 조건식의 결과와 일치하는 case문이 없을 때 수행될 문장들  
        //...  
}
```

```
switch(num) {  
    case 1:  
    case 7:  
        System.out.println("SK");  
        break;  
    case 6:  
    case 8:  
        System.out.println("KTF");  
        break;  
    case 9:  
        System.out.println("LG");  
        break;  
    default:  
        System.out.println("UNKNOWN");  
        break;  
}
```


1.4 switch문 – 사용예(examples)

```
int level = 3;

switch(level) {
    case 3 :
        grantDelete(); // 삭제권한을 준다.
    case 2 :
        grantWrite(); // 쓰기권한을 준다.
    case 1 :
        grantRead(); // 읽기권한을 준다.
}
```

```
switch(score) {
    case 100: case 99: case 98: case 97: case 96:
    case 95: case 94: case 93: case 92: case 91:
    case 90 :
        grade = 'A';
        break;
    case 89: case 88: case 87: case 86:
    case 85: case 84: case 83: case 82: case 81:
    case 80 :
        grade = 'B';
        break;
    case 79: case 78: case 77: case 76:
    case 75: case 74: case 73: case 72: case 71:
    case 70 :
        grade = 'C';
        break;
    case 69: case 68: case 67: case 66:
    case 65: case 64: case 63: case 62: case 61:
    case 60 :
        grade = 'D';
        break;
    default :
        grade = 'F';
} // end of switch
```

```
char op = '+';


switch(op) {
    case '+':
        result = num1 + num2;
        break;
    case '-':
        result = num1 - num2;
        break;
    case '*':
        result = num1 * num2;
        break;
    case '/':
        result = num1 / num2;
        break;
}
```

```
switch(score/10) {
    case 10:
    case 9 :
        grade = 'A';
        break;
    case 8 :
        grade = 'B';
        break;
    case 7 :
        grade = 'C';
        break;
    case 6 :
        grade = 'D';
        break;
    default :
        grade = 'F';
}
```

1.5 중첩 switch문

- switch문 안에 또 다른 switch문을 중첩해서 넣을 수 있다.
- switch문의 중첩횟수에는 거의 제한이 없다.

```
switch(num) {  
    case 1:  
    case 7:  
        System.out.println("SK");  
        switch(num) {  
            case 1:  
                System.out.println("1");  
                break;  
            case 7:  
                System.out.println("7");  
                break;  
        }  
        break;  
    case 6:  
        System.out.println("KTF");  
        break;  
    case 9:  
        System.out.println("LG");  
        break;  
    default:  
        System.out.println("UNKNOWN");  
}
```



```
switch(num) {  
    case 1:  
    case 7:  
        System.out.println("SK");  
        if(num==1) {  
            System.out.println("1");  
        } else if(num==7) {  
            System.out.println("7");  
        }  
        break;  
    case 6:  
        System.out.println("KTF");  
        break;  
    case 9:  
        System.out.println("LG");  
        break;  
    default:  
        System.out.println("UNKNOWN");  
}
```

1.6 if문과 switch문의 비교

- if문이 주로 사용되며, 경우의 수가 많은 경우 switch문을 사용할 것을 고려한다.
- 모든 switch문은 if문으로 변경이 가능하지만, if문은 switch문으로 변경할 수 없는 경우가 많다.
- if문 보다 switch문이 더 간결하고 효율적이다.

```
if(num==1) {  
    System.out.println("SK");  
} else if(num==6) {  
    System.out.println("KTF");  
} else if(num==9) {  
    System.out.println("LG");  
} else {  
    System.out.println("UNKNOWN");  
}
```



```
switch(num) {  
    case 1:  
        System.out.println("SK");  
        break;  
    case 6:  
        System.out.println("KTF");  
        break;  
    case 9:  
        System.out.println("LG");  
        break;  
    default:  
        System.out.println("UNKNOWN");  
}
```

1.7 Math.random()

- Math클래스에 정의된 난수(亂數) 발생함수
- 0.0과 1.0 사이의 double값을 반환한다. ($0.0 \leq \text{Math.random()} < 1.0$)

예) 1~10범위의 임의의 정수를 얻는 식 만들기

1. 각 변수에 10을 곱한다.

```
0.0 * 10 <= Math.random() * 10 < 1.0 * 10  
0.0 <= Math.random() * 10 < 10.0
```

2. 각 변수를 int형으로 변환한다.

```
(int)0.0 <= (int)(Math.random() * 10) < (int)10.0  
0 <= (int)(Math.random() * 10) < 10
```

3. 각 변수에 1을 더한다.

```
int score = (int)(Math.random() * 10) + 1;
```

```
0 + 1 <= (int)(Math.random() * 10) + 1 < 10 + 1  
1 <= (int)(Math.random() * 10) + 1 < 11
```

2. 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
<input type="text"/>
sum
<input type="text"/>

i	sum
1	
2	
3	
4	
...	
10	

2.2 for문 – 작성 예(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>for(int i=1,j=1;i<10 & { /* 반복해서 수행할 */ }</pre>	

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

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

    System.out.println(i-1 + "까지의 합: " + sum);
}
```

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

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

2.5 중첩 while문

- 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. 블럭{}을 먼저 수행한 다음에 조건식을 계산한다.
- 블럭{}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.

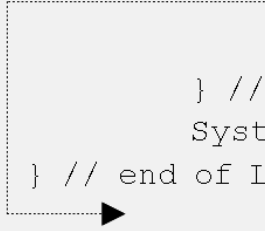
[실행결과]

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
```