

JAVA 프로그래밍

조건문 & 반복문

조건문과 반복문

1. 조건문

1.1 조건문(if, switch)

1.2 if문

1.3 중첩 if문

1.4 switch문

1.5 if문과 switch문의 비교

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

조건문 - if, switch

- 조건문은 조건식과 실행될 하나의 문장 또는 블록{ }으로 구성
- Java에서 조건문은 if문과 switch문 두 가지이다.
- if문이 주로 사용되며, 경우의 수가 많은 경우 switch문이 사용된다.
- 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");  
}
```

조건문 - if

- if문은 if, if~else, if~else if의 세 가지 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  
}
```

실습 - if문

```
public class IfTest1 {  
    public static void main(String[] args) {  
        int age = 15;  
        if(age <= 19) // if문  
            System.out.println("미성년자입니다.");  
  
        int score = 80;  
        if(score > 60) // if~else문  
            System.out.println("합격입니다.");  
        else  
            System.out.println("불합격입니다.");  
    }  
}
```

실습 - if문

```
public class IfTest2 {  
    public static void main(String[] args) {  
        int score = 93;  
  
        if(score>=90)  
            System.out.println("A등급");  
        else if(score>=80 && score<90)  
            System.out.println("B등급");  
        else if(score>=70 && score<80)  
            System.out.println("C등급");  
        else  
            System.out.println("F등급");  
    }  
}
```

조건문 - if문의 예시

예1) 변수 i 홀수인지 짝수인지의 조건 : `if(i%2==0) { }`

예2) 변수 i가 3의 배수인지의 조건 : `if(i%3==0) { }`

예3) 문자형 변수 ch가 공백이거나 탭인지 조건 : `if(ch==' ' || ch=='\t') { }`

예4) 문자형 변수 ch에 저장된 문자가 소문자 c 또는 대문자C 인지의 조건 :
`if(ch=='c' || ch=='C') { }`

예5) 문자열 변수 str에 저장된 문자가 소문자 c 또는 대문자C 인지의 조건 :
`if(str.equals("c") || str.equals("C")) { }`

예6) 전원이 ON인지 OFF인지의 조건:

```
boolean powerOn = true;
```

```
if(!powerOn) {
```

```
    // 전원이 꺼져있으면...
```

```
}
```

조건문 - 중첩 if

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

```
if (조건식1) {  
    // 조건식1의 연산결과가 true일 때 수행될 문장들을 적는다.  
    if (조건식2) {  
        // 조건식1과 조건식2가 모두 true일 때 수행될 문장들  
    } else {  
        // 조건식1이 true이고, 조건식2가 false일 때 수행되는 문장들  
    }  
} else {  
    // 조건식1이 false일 때 수행되는 문장들  
}
```


실습 - 중첩 if문

```
public class NestedIf {  
    public static void main(String[] args) {  
        int score = 93;  
        String grade;  
  
        if(score>=90) {  
            grade = "A";  
            if(score>=98) {  
                grade = grade + "+";  
            }  
            else if(score<94) {  
                grade = grade + "-";  
            }  
        } // 바깥쪽 if문의 끝
```

```
else if(score>=80) {  
    grade = "B";  
    if(score>=88) {  
        grade = grade + "+";  
    }  
    else if(score<84) {  
        grade = grade + "-";  
    }  
} // 바깥쪽 else~if문의 끝
```

```
else {  
    grade = "C";  
} // 바깥쪽 else문의 끝
```

```
System.out.printf("grade: %s\n", grade);  
}  
}
```

Quiz if문

< IfExam1.java> if~if(중첩 if)문을 사용하여 정수 값이 양수인지 음수인지 판별하는 문장을 출력하시오.

[변수 선언] 정수형 변수 num 선언과 초기화

[출력 결과]

num의 값이 양수일 경우 → num의 값은 양수입니다.

num의 값이 음수일 경우 → num의 값은 음수입니다.

num의 값이 0일 경우 → num의 값은 0입니다.

Quiz if문

< IfExam2.java> if~else문을 사용하여 나이가 15~100세까지는 회원가입이 가능합니다. 조건에 맞지 않는 나이일 경우 회원가입이 불가능합니다. 라는 문장을 출력하시오. 단, 나이는 정수를 입력 받아서 사용하세요.

[변수 선언] 정수형 변수 age

[출력 결과]

나이를 입력하세요 :

age의 값이 15~100 사이일 경우 → 회원가입이 가능합니다.

age의 값이 다른 나이일 경우 → 회원가입이 불가능합니다.

Quiz if문

< IfExam3.java> 중첩if문을 사용하여 자동차의 성능테스트를 실행해 보세요.

- 성능 : 속도(speed), 디자인(design), 연비(mileage)
- 평균(avg)이 80점 이상이면 “합격”, 70점 이상이면 “정상”, 나머지는 “리콜”
- 속도, 디자인, 연비 모두가 80점 이상인 경우 “베스트”

[변수 선언과 초기화]

```
int speed, design, mileage;  
double avg = (속도+디자인+연비)/3;
```

[출력 결과]

속도 입력 : 80

디자인 입력 : 85

연비 입력 : 88

결과 : 베스트, 합격

조건문 - switch

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

실습 - switch문

```
public class SwitchTest1 {  
    public static void main(String[] args)  
    {  
        int score = 95;  
        char grade;  
  
        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';  
        }  
        System.out.println("grade: "+grade);  
    }  
}
```

실습 - switch문

```
public class SwitchTest2 {  
    public static void main(String[] args)  
    {  
        int num1=10, num2=5, result=0;  
        char op = '*';  
  
        switch(op) {  
            case '+':  
                result = num1+num2;  
                break;  
            case '-':  
                result = num1-num2;  
                break;  
            case '*':  
                result = num1*num2;  
                break;  
            case '/':  
                result = num1/num2;  
                break;  
        }  
        System.out.println("result: " + result);  
    }  
}
```


Quiz switch문

<SwitchExam1.java> switch~case문을 사용하여 스위치 값을 입력 받아 전등을 켜고 싶을 경우 1번을, 전등을 끄고 싶을 경우 2번을, 고장이 났을 경우 3번을 입력 하도록 출력하세요.

[변수 선언] 정수형 변수 num 선언

[출력 결과]

숫자를 입력하세요 :

입력한 숫자가 1일 경우 → 전등 ON

입력한 숫자가 2일 경우 → 전등 OFF

입력한 숫자가 3일 경우 → 전등 고장

다른 숫자일 경우 → 스위치 번호 오류

Quiz switch문

<SwitchExam2.java> switch~case문을 사용하여 "T(t), F(f), S(s)" 중 한 개의 문자의 값에 해당하는 요일 (Thursday, Friday, Saturday)을 출력하세요. 문자열 값을 입력 받아서 실행하세요.

[변수 선언] 문자열 입력 받는 변수 선언

[출력 결과]

문자를 입력하세요 :

입력한 문자가 T 또는 t일 경우 → Thursday

입력한 문자가 F 또는 f일 경우 → Friday

입력한 문자가 S 또는 s일 경우 → Saturday

다른 문자일 경우 → 문자가 잘못 입력되었습니다.

Quiz switch문

<SwitchExam3.java> switch~case문을 사용하여 아래와 같이 출력되도록 하세요.
직급은 입력 받아서 실행하세요.

- 보너스(bonus) = 부장 : 50%, 과장 : 30%, 대리 : 20%, 사원:10%, 나머지: 0
- 월급(salary) = 기본급(pay) + 기본급(pay) * 보너스(bonus)

[변수 선언]

```
int pay = 200;
```

```
double bonus = 0;
```

[출력 결과]

직급을 입력하세요 : 과장

기본급 : 200만원

보너스 : 30%

월급 : 260만원

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 변경 가능하다.
- do-while문 :
while문의 변형으로 조건과 관계없이 최소한 한 번은 블록{}안의 문장이 실행된다.

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

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

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

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

반복문 - for

- 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) 1~10까지의 정수 더하기

```
int sum = 0;  
  
for(int i=1; i<=10; i++) {  
    sum += i; // sum = sum + i;  
}
```

i	sum
1	
2	
3	
4	
...	
10	

실습 - for문

‘반복내용 : 횟수’를 5회 출력하는 반복문

```
public class ForTest1 {  
    public static void main(String[] args) {  
        int num;  
        for(num=0 ; num<5 ; num++) {  
            System.out.println("반복 내용 : " + num);  
        }  
        System.out.printf("반복문을 종료한 후 num : " + num);  
    }  
}
```

실습 - for문

1~10까지의 합계를 출력하는 반복문

```
public class ForTest2 {  
    public static void main(String[] args) {  
        int i, sum=0;  
  
        for(i=1 ; i<=10 ; i++) {  
            sum += i;  
            System.out.println("i = " + i + ", sum = " + sum);  
        }  
  
        System.out.println("-----반복문 종료-----");  
    }  
}
```

Quiz for문

〈ForGuGu1.java〉 입력 받은 숫자의 구구단을 출력하시오.

[변수 선언과 초기화]

num(숫자 변수), i(1~9 곱해지는 수)

[출력 결과]

구구단을 출력할 숫자를 입력하세요 : 5

$$5 * 1 = 5$$

$$5 * 2 = 10$$

$$5 * 3 = 15$$

$$5 * 4 = 20$$

$$5 * 5 = 25$$

$$5 * 6 = 30$$

$$5 * 7 = 35$$

$$5 * 8 = 40$$

$$5 * 9 = 45$$

Quiz for문

<ForGuGu2.java> " ForGuGu1.java" 파일을 수정하여,

입력한 숫자가 2~9 사이인 경우에는 해당 구구단을 출력하고 다른 숫자를 입력한 경우에는 " 2~9 사이의 숫자를 입력하세요. " 를 출력하세요.

[출력 결과]

구구단을 출력할 숫자를 입력하세요 : 10

2~9사이의 숫자를 입력하세요.

반복문 - 중첩 for

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

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

```
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);  
        }  
    }  
}
```

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

실습 - 중첩 for문

```
public class NestedFor {  
    public static void main(String[] args) {  
        int i, j;  
  
        for(i=0; i<2; i++) {  
            System.out.println("큰 반복문 (외부 for)");  
            for(j=0; j<3; j++) {  
                System.out.println("  작은 반복문 (내부 for)");  
            }  
            System.out.println("-----");  
        }  
    }  
}
```

Quiz 중첩 for문

<NestedForGuGu.java> for문을 사용하여 구구단(2~9단)을 출력하시오.

[변수 선언과 초기화]

정수형 변수: i(2~9단 반복), j(1~9 곱해지는 수 반복)

[출력 결과] 2단~9단까지 출력, 단과 단 사이에 “-----” 출력

2 * 1 = 2

2 * 2 = 4

2 * 3 = 6

2 * 4 = 8

2 * 5 = 10

2 * 6 = 12

2 * 7 = 14

2 * 8 = 16

2 * 9 = 18

반복문 - while

- 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--);  
}
```

실습 - while문

‘반복내용: 횟수’를 5회 출력하는 반복문

```
public class WhileTest1 {  
    public static void main(String[] args) {  
        int num=0;  
  
        while(num<5) {  
            System.out.println("반복 내용 : " + num);  
            num++;  
        }  
        System.out.println("반복문을 종료한 후 : " + num);  
    }  
}
```

실습 - while문

1~10까지의 합계를 출력하는 반복문

```
public class WhileTest2 {  
    public static void main(String[] args) {  
        int i=1, sum=0;  
  
        while(i<=10) {  
            sum=sum+i;  
            System.out.println("i = " + i + ", sum = " + sum);  
            i++;  
        }  
        System.out.println("-----반복문 종료-----");  
    }  
}
```

실습 - while문

- while 무한 루프 : 반복문이 종료되지 않고 계속 실행되는 while문

```
public class WhileTest3 {  
    public static void main(String[] args) {  
        int i=0;  
  
        // 무조건 참, 무한 루프, 조건이 참인 경우 무한 반복  
        while(true) {  
            System.out.println("반복 횟수 : " + i);  
            i++;  
        }  
    }  
}
```


실습 - while문

- if~break : while의 무한 루프를 종료하는 조건문

```
public class WhileBreak {  
    public static void main(String[] args) {  
        int i=0;  
  
        while(true) {  
            System.out.println("반복 횟수 : " + i);  
            i++;  
            if(i>10)  
                break; // i값이 10보다 크면 반복문 종료(0~10까지 출력)  
        }  
    }  
}
```

Quiz while문

〈WhileExam1.java〉 입력 받은 숫자의 구구단을 출력하시오.

[변수 선언과 초기화] num(숫자 변수), j(1~9 곱해지는 수)

[출력 결과]

구구단을 출력할 숫자 입력 : 7

$$7 * 1 = 7$$

$$7 * 2 = 14$$

$$7 * 3 = 21$$

$$7 * 4 = 28$$

$$7 * 5 = 35$$

$$7 * 6 = 42$$

$$7 * 7 = 49$$

$$7 * 8 = 56$$

$$7 * 9 = 63$$

반복문 - 중첩 while

- while문 안에 또 다른 while문을 포함시킬 수 있다.
- while문의 중첩 횟수에는 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++;  
}
```

실습 - 중첩 while문

```
public class NestedWhile {  
    public static void main(String[] args) {  
        int i=0, j=0;  
        while(i<2) {  
            System.out.println("큰 반복문 (외부 while) ");  
            while(j<3) {  
                System.out.println(" 작은 반복문 (내부 while) ");  
                j++;  
            }  
            i++;  
            j=0;    // j의 값 초기화  
        }  
    }  
}
```

Quiz 중첩 while문

<NestedWhileExam.java> while문을 사용하여 구구단(2~9단)을 출력하시오.

[변수 선언과 초기화]

정수형 변수: i(2단~9단), j(1~9)

[출력 결과] 2~9단 출력, 단과 단 사이에 "-----" 출력

2 * 1 = 2

2 * 2 = 4

2 * 3 = 6

2 * 4 = 8

2 * 5 = 10

2 * 6 = 12

2 * 7 = 14

2 * 8 = 16

2 * 9 = 18

반복문 - do~while

- while문의 변형, 블록{}을 먼저 수행한 다음에 조건식을 계산한다.
- 조건과 관계없이 블록{}안의 문장이 최소한 1번은 실행된다.

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

```
class DoWhile {  
    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
...	...

실습 - do~while문

```
public class DoWhileTest {  
    public static void main(String[] args) {  
        int num=15;  
  
        do {  
            System.out.println(num);  
            num++;  
        } while(num<10);        // num값이 10보다 작을 때까지 반복  
  
        System.out.println("***while 문을 종료합니다.***");  
    }  
}
```

Quiz do~while문

<DoWhileExam1.java> 1~10까지의 합계를 do~while문으로 출력해보세요.

[출력 결과]

1~10까지의 합계 : 55

반복문 종료 후, i값 : 11

Quiz do~while문

<DoWhileExam2.java> 1~10까지의 범위에서 3의 배수와 4의 배수의 합계를 do~while문으로 출력해보세요.

[출력 결과]

3의 배수와 4의 배수 합계 : 30

반복문 - break

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

```
class WhileBreak
{
    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);
    }
}
```

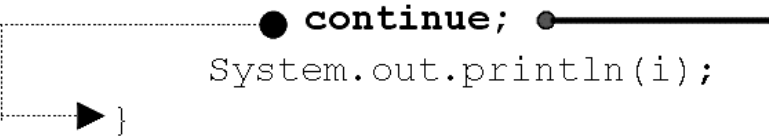
break문이 수행되면 이 부분은 실행되지 않고 while문을 완전히 벗어난다.

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

반복문 - continue

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

```
class Continue
{
    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

Quiz continue문

〈ContinueExam.java〉 1~4까지 그리고 6~10까지의 합계를 for ~ continue문을 사용하여 출력하세요.

[출력 결과]

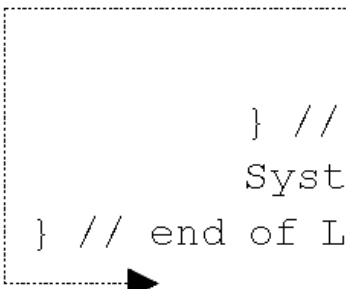
1~4, 6~10까지의 합계 : 50

반복문 종료 후 i의 값 : 11

이름 붙은 반복문과 break, continue

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

```
class NamedLoopTest
{
    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
```