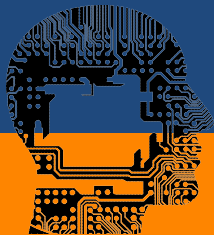




한국IT진흥부설

정보보호교육학원

아이섹



# 자바 프로그래밍 기초

## 제어문

By SoonGu Hong(Kokono)

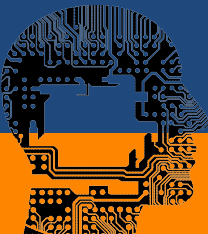


한국IT진흥부설

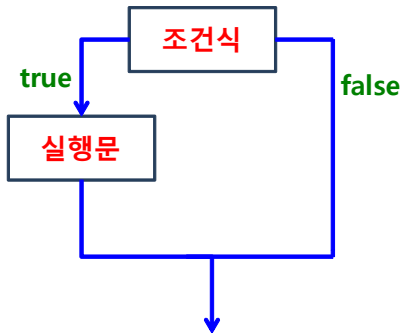
정보보호교육원 아이섹

# 자바 프로그래밍 기초

## 1. 조건문



## 1-1-1. 조건문 if



- 조건문은 프로그램에서 조건식의 참, 거짓에 따라 코드를 다르게 실행하게 하는 **분기점을 만드는 제어문**입니다.

- if는 조건식의 논리결과가 **참일 경우** 블록 내의 코드를 **실행**하며 거짓일 경우 코드를 실행하지 않습니다.

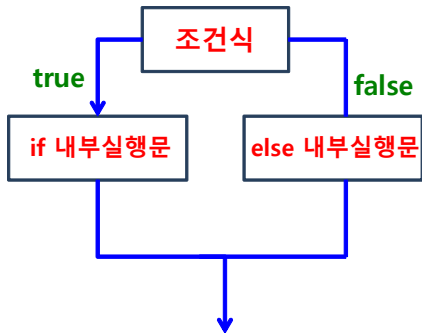
- if블록 내부의 코드가 **단 한 줄**일 경우 블록(중괄호)을 **생략**할 수 있습니다.

## 1-1-2. if문 사용법

```
//0~100까지의 정수 난수값을 발생시켜 point변수에 저장.  
int point = (int)(Math.random() * 101);  
System.out.println("점수: " + point);  
  
if(point >= 60) {  
    System.out.println("60점 이상입니다.");  
    System.out.println("합격이야~~");  
}
```

1. 키워드 if를 사용하여 ()안에 논리형 값이 도출되는 조건식이나 함수를 넣어줍니다.
2. 블록을 만들어 준 뒤 블록 내부에 조건이 참일 경우 실행할 코드를 적습니다. (단 한문장일 경우 블록 생략 가능)

## 1-2-1. 조건문 if ~ else



- else 키워드는 if문과 반드시 **함께** 사용해야 하며 단독으로 사용할 수 없습니다.

- if가 가지고 있는 **조건식**의 결과가 **거짓**일 경우 자동으로 **else**가 가진 코드가 **실행**되는 구조입니다.

- else 블록에도 실행문이 단 한문장이라면 블록을 생략할 수 있습니다.

## 1-2-2. if ~ else문 사용법

```
int point = (int)(Math.random() * 101);
System.out.println("점수: " + point);

if(point >= 60) {
    System.out.println("60점 이상입니다.");
    System.out.println("합격이야~~");
} else {
    System.out.println("60점 미만입니다.");
    System.out.println("응~ 불합격~");
}
System.out.println("수고용~~");
```

1. 기본적인 if문을 만들고 추가로 else키워드를 사용하여 새로운 블록을 만듭니다.
2. else블록 내부에 조건식이 거짓일 경우 실행할 코드를 적습니다.

## 1-2-3. Quiz

신장: 145

나이: 11

놀이기구에 탑승할 수 있습니다.

오늘 하루 즐거운 시간되세요~!

신장: 138

나이: 11

놀이기구에 탑승할 수 없습니다.

오늘 하루 즐거운 시간되세요~!

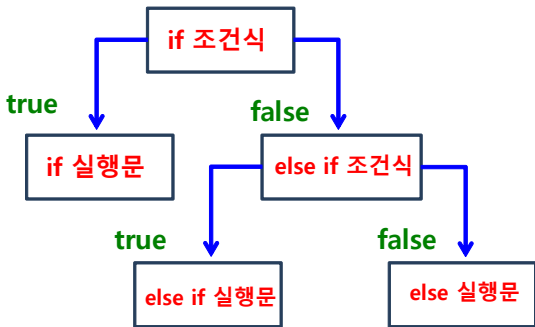
문제) 위와 같은 결과가 나오는 프로그램을 코딩하세요.

- 요구사항

1. 사용자는 신장과 나이의 정보를 숫자로 입력할 수 있어야 함.
2. 프로그램은 2가지 정보를 입력하고 엔터를 누르면 조건에 따라 다른 결과를 출력해야 함.
3. 키가 140이상이고 나이가 8세이상인 2개의 조건을 모두 만족할 경우 "놀이기구에 탑승할 수 있습니다."를 출력할 것.
4. 두 개의 조건 중 하나라도 만족하지 않을 시 "놀이기구에 탑승할 수 없습니다"를 출력할 것.
5. 조건과는 관계없이 "오늘 하루 즐거운 시간되세요!"를 출력할 것!

<힌트> 논리 연산자를 사용해 볼 것!

### 1-3-1. 다중분기 조건문 if ~ else if



- if ~ else if문은 좀 더 많은 분기를 만들고 싶을 때 사용하는 조건문입니다.
- 다만 위에서부터 조건을 검색하면서 내려오기 때문에 범위조건일 경우 상위 조건이 하위조건을 포괄적으로 포함하지 않도록 주의해야 합니다.



## 1-3-2. if ~ else if문 사용법

```
Scanner sc = new Scanner(System.in);
System.out.print("나이: ");
int age = sc.nextInt();
sc.close();

if(age >= 20) {
    System.out.println("성인입니다.");
} else if(age >= 17) {
    System.out.println("고딩입니다.");
} else if(age >= 14) {
    System.out.println("중딩입니다.");
} else if(age >= 8) {
    System.out.println("초딩입니다.");
} else {
    System.out.println("아동입니다.");
}
```

1. 기본적인 if문을 만들고 추가로 else if키워드를 사용하여 새로운 블록을 만듭니다.

2. else if블록 내부에 추가 조건식을 설정합니다.

3. else if블록을 여러 개 만들 수 있습니다.

4. 모든 조건이 거짓일 경우 실행할 else문은 선택적으로 만듭니다.

## 1-3-3. Quiz

정수: 84

입력하신 숫자는 7의 배수입니다.

정수: -72

입력하신 숫자는 7의 배수가 아닙니다.

정수: 0

입력하신 숫자는 0입니다.

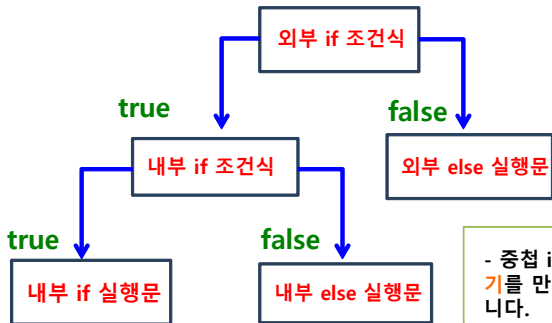
문제) 위와 같은 결과가 나오는 프로그램을 코딩하세요.

- 요구사항

1. 사용자는 정수를 입력할 수 있어야 함.
2. 프로그램은 정수를 입력하고 엔터를 누르면 조건에 따라 다른 결과를 출력해야 함.
3. 입력한 정수가 7의 배수라면 "입력하신 숫자는 7의 배수입니다."를 출력할 것.
4. 입력한 정수가 7의 배수가 아니라면 "입력하신 숫자는 7의 배수가 아닙니다."를 출력할 것.
5. 입력한 정수가 0이라면 "입력한 숫자는 0입니다."를 출력할 것.

<힌트> 다중 분기 조건문의 순서에 주의할 것!

## 1-4-1. 중첩 if문



- 중첩 if문은 단계적 조건 분기를 만들고 싶을 때 사용합니다.

- 외부 1차 if의 조건식을 만족하면 2차적으로 내부 if의 조건을 판단하고 참, 거짓에 따라 실행 코드를 결정합니다.

## 1-4-2. 중첩 if문 사용법

```
Scanner sc = new Scanner(System.in);
System.out.print("신장: ");
double height = sc.nextDouble();
sc.close();

if(height >= 140) {
    System.out.print("나이: ");
    int age = sc.nextInt();

    if(age >= 8) {
        System.out.println("놀이기구 타러가세요~~");
    } else if(age >= 6) {
        System.out.println("보호자 동반시 탑승 가능!");
    } else {
        System.out.println("나이 때문에 못타요~~");
    }
} else {
    System.out.println("키 때문에 못타요~~");
}
```

1. 기본적인 if문을 만들고 해당 if블록 내부에 새로운 if블록을 생성합니다.

2. 전 단계에서 배운 다중 분기 if ~ else if문도 사용가능합니다.

3. 중첩의 단계가 많아질수록 블록에 주의해서 프로그래밍해야 합니다.

## 1-5-1. 다중분기 조건문 switch ~ case

```
switch (enum or int형 or String형 표현식) {  
    case 값1:  
        실행문 1;  
        break;  
    case 값2:  
        실행문 2;  
        break;  
        .....  
    case 값N:  
        실행문 N;  
        break;  
    default:  
        기본 실행문;  
        break;  
}
```

- 다중 분기 구현은 if ~ else if문으로도 구현이 가능하지만 **분기의 개수가 많아질 수록 프로그램의 효율이 감소**하는 단점이 있습니다.

- 이런 단점을 극복하기 위해 사용하는 조건 분기문이 switch문입니다.

- switch문은 if ~ else if문보다 **코드가 간결하며 효율적**입니다.

- default는 else와 같은 효과를 가집니다.

## 1-5-2. switch문 사용법

```
Scanner sc = new Scanner(System.in);
System.out.println("### 여행지 추천 프로그램 ###");
System.out.println("[여행을 원하는 계절을 적어주세요!]");
System.out.print(">> ");
String season = sc.next();
sc.close();

switch(season) {

case "봄":
    System.out.println("봄에는 상하이로 가보세요~~");
    break; //switch블록을 나가라~
case "여름":
    System.out.println("여름에는 뽕으로 가보세요~~");
    break;
case "가을":
    System.out.println("가을에는 방콕으로 가보세요~~");
    break;
case "겨울":
    System.out.println("겨울에는 삿포로로 가보세요~~");
    break;
default:
    System.out.println("[봄, 여름, 가을, 겨울 중에 선택하세요!]");
} //end switch
```

1. switch 키워드를 쓰고 괄호안에 조건식이 아닌 **문자열, 정수타입의 변수**를 씁니다.

2. 블록을 열고 변수에 저장될 경우의 수 값들을 case와 함께 **상수나 리터럴형태**로 적습니다.

3. case가 흘러내려가지 않도록 **break** 키워드로 멈추게 합니다.

4. default는 어떤 case에도 걸리지 않을 때 실행할 코드를 씁니다.

## 1-5-3. Quiz

직급을 입력하세요.

[사원, 대리, 과장, 차장, 부장]

> 대리

대리의 급여는 300만원입니다.

직급을 입력하세요.

[사원, 대리, 과장, 차장, 부장]

> 부장새끼

부장새끼은(는) 없는 직급입니다.

직급을 다시 입력해주세요.

문제) 위와 같은 결과가 나오는 프로그램을 코딩하세요.

- 요구사항

1. 사용자는 직급을 문자열로 입력할 수 있어야 함.
2. 프로그램은 직급을 입력하고 엔터를 누르면 직급에 따라 급여정보를 출력해야 함.
3. 입력한 직급이 [사원, 대리, 과장, 차장, 부장] 중 하나라면 각각 "[해당직급]의 급여는 [200, 300, 400, 500, 600]만원입니다." 라고 출력할 것.
4. 입력한 직급이 위 예시와 다르다면 "[입력한 값]은(는) 없는 직급입니다."를 출력할 것.

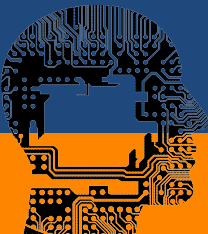


한국IT진흥부설

정보보호교육학원 아이섹

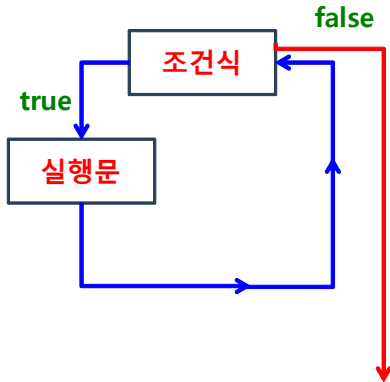
# 자바 프로그래밍 기초

## 2. 반복문





## 2-1-1. 반복문 while



- while문은 조건식을 검사하여 참일 경우 블록 내부의 코드를 실행하며 실행이 끝날 때마다 반복적으로 조건식을 검사하여 false가 나올 때까지 반복합니다.

- while문도 if와 마찬가지로 논리값이 도출되는 조건식이나 함수를 사용합니다.

- while문은 반복 횟수를 제어하기 위한 증감식이나 탈출문이 필요합니다.

## 2-1-2. while문 사용법

```
int total = 0; //총합을 저장할 변수
int n = 1; //1. 제어변수의 선언(begin)
while(n <= 10) { //2. 논리형 조건식: 반복문의 끝 지점을 결정(end)
    total += n; //반복해야할 코드
    n++; //3. 제어변수의 증감식: 반복문의 종료를 위해 제어변수의 값을 조정(step)
}
```

1. 반복문의 시작점이 되는 값을 저장하는 제어변수를 선언합니다.
2. while문의 조건식 자리에 반복문이 끝나는 시점을 조건식이나 함수로 표현합니다.
3. 반복실행할 코드를 적고 1번에서 만든 제어변수의 증감식을 적어 반복문이 언젠가 false가 되어 종료될 수 있게 합니다.

## 2-1-3. Quiz

```
정수 1: 5  
정수 2: 8  
5~8까지의 총합: 26
```

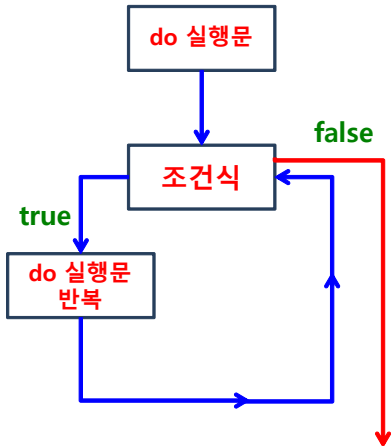
문제) 위와 같은 결과가 나오는 프로그램을 코딩하세요.

- 요구사항

1. 사용자는 정수 2개를 각각 입력할 수 있어야 함.
2. 프로그램은 2번째 정수를 입력하고 엔터를 누르면 1번째 정수부터 2번째 정수까지의 총합(2번째 정수포함)을 출력해야 함.
3. 반복문 while을 사용할 것!

## 2-2-1. 반복문 do ~ while

<< 최초 1회 강제 실행 >>



- **while**문은 **조건식을 먼저 검사**하고 실행문이 반복되기 때문에 처음 실행 조건이 **false**라면 실행문이 단 한번도 실행되지 않습니다.

- **do ~ while**문은 **do** 이하의 구문이 먼저 한 번 실행된 뒤에 조건식을 검사하므로 결과가 **true**이든 **false**이든 **무조건 한번은 실행**이 됩니다.

- **do ~ while**문은 조건식의 결과에 상관없이 **루프를 반드시 한번 이상 실행**시키도록 할 때 사용합니다.

## 2-2-2. do~ while VS while

```
Scanner sc = new Scanner(System.in);

int number = 0;
int total = 0;

while(number != 0) {
    System.out.print("정수(0입력시 종료): ");
    number = sc.nextInt();
    total += number;
}

System.out.println("입력값 누적합: " + total);
sc.close();
```

**while**

```
Scanner sc = new Scanner(System.in);

int number = 0;
int total = 0;

do {
    System.out.print("정수(0입력시 종료): ");
    number = sc.nextInt();
    total += number;
} while(number != 0);

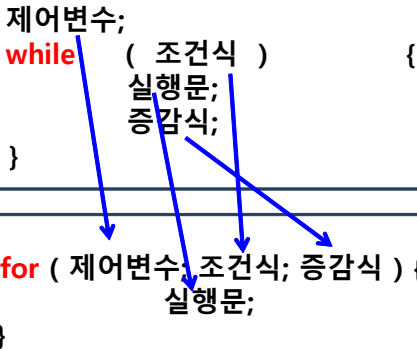
System.out.println("입력값 누적합: " + total);
sc.close();
```

**do ~ while**

### < while문과 for문 비교 >

```
제어변수;  
while ( 조건식 ) {  
    실행문;  
    증감식;  
}
```

```
for ( 제어변수; 조건식; 증감식 ) {  
    실행문;  
}
```



- for문은 **반복제어조건을 한꺼번에 지정**한다는 점이 다른 반복문과는 다릅니다.

- 따라서 **정확한 반복 횟수를 알고 있을 때는** for문이 while문보다 효율적입니다.

## 2-3-2. for문 사용법

```
int total = 0;
for(int n = 1; n <= 10; n++) {
    total += n;
}
System.out.println("1~10까지의 총합: " + total);
```

1. for 키워드와 함께 시작값을 저장하는 제어변수선언, 끝지점을 체크할 조건식, 제어변수를 조작할 증감문을 소괄호 안에 **순서대로 세미콜론과 함께 배치**합니다.
2. 블록을 열고 반복실행할 문장들을 적습니다.
3. 실행순서는 제어변수 선언 -> 조건식 판단 -> 실행문 -> 증감식 -> 조건식 판단 -> 실행문 -> 증감식 순서로 진행되므로 **순서에 주의**하세요!

## 2-3-3. Quiz

랜덤 구구단 4단

-----

$$4 \times 1 = 4$$

$$4 \times 2 = 8$$

$$4 \times 3 = 12$$

$$4 \times 4 = 16$$

$$4 \times 5 = 20$$

$$4 \times 6 = 24$$

$$4 \times 7 = 28$$

$$4 \times 8 = 32$$

$$4 \times 9 = 36$$

문제) 왼쪽과 같은 결과가 나오는 프로그램을 코딩하세요.

- 요구사항

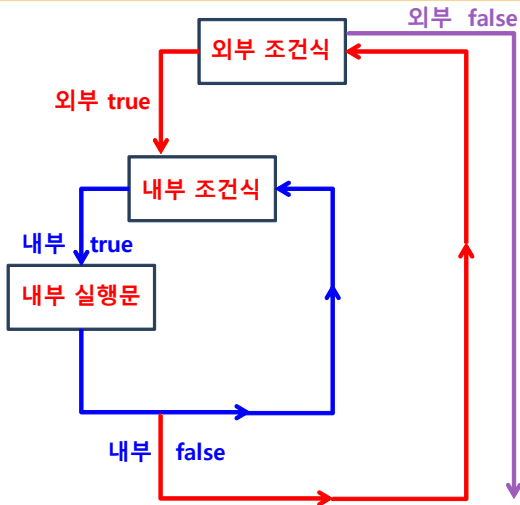
1. 프로그램 실행 시 2~9단 중 무작위로 구구단이 등장하게 하세요.

2. for문과 while문으로 각각 구현하세요.

힌트) 먼저 2단을 구현해 본 뒤, 성공하면 랜덤 구구단으로 바꿔보기



## 2-4-1. 중첩 반복문



## 2-4-2. 중첩 반복문 예시

```
for(int dan=2; dan<=9; dan++) {  
    for(int hang=1; hang<=9; hang++) {  
        System.out.printf("%d x %d = %d\n"  
            , dan, hang, dan * hang);  
    }  
    System.out.println("-----");  
}
```

```
2 x 1 = 2  
2 x 2 = 4  
2 x 3 = 6  
2 x 4 = 8  
2 x 5 = 10  
2 x 6 = 12  
2 x 7 = 14  
2 x 8 = 16  
2 x 9 = 18  
-----
```

...

```
9 x 1 = 9  
9 x 2 = 18  
9 x 3 = 27  
9 x 4 = 36  
9 x 5 = 45  
9 x 6 = 54  
9 x 7 = 63  
9 x 8 = 72  
9 x 9 = 81  
-----
```

## 2-4-3. Quiz

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

문제) 위와 같은 결과가 나오는 프로그램을 코딩하세요.

- 요구사항

1. \* 을 한개 출력하는 출력문을 반복문을 사용하여 순회시킬 것.
2. 공백을 한개 출력하는 출력문을 반복문을 사용하여 순회시킬 것.
3. 예시는 5층짜리 피라미드지만 층을 제어하는 변수값 조정을 통해 간단히 10층으로도 변화 가능하게 프로그래밍할 것.

## 2-4-4. Quiz

문제) 중첩 for문을 이용하여 방정식  $4x + 5y = 60$  의 모든 해를 구해서 ( x, y ) 형태로 출력하세요.

- 요구사항

1. x와 y는 10이하의 자연수입니다.

```
<terminated> LoopNestingE
```

```
( 5, 8 )
```

```
( 10, 4 )
```

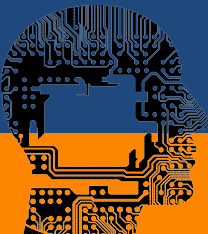


한국IT진흥부설

정보보호교육학원 아이섹

# 자바 프로그래밍 기초

## 3. 탈출문



### 3-1-1. 탈출문 break

```
for ( ... ) {
```

```
break;
```



```
}
```

- break문은 for문이나 while문에서 사용되며 반복문이 실행 중 break를 만나는 순간 **반복문을 해당 시점에서 종료**합니다.

- break를 만나는 순간 이하의 **반복문의 남은 코드는 모두 실행되지 않으며** 블록을 탈출합니다.

- 대개 if문과 함께 사용되며 조건에 따라 반복문을 종료할 때 사용합니다.

## 3-1-2. break문 사용 예시

```
for(int i=1; i<11; i++) {  
    if(i == 5) break;  
    System.out.print(i + " ");  
}//end for  
System.out.println("\n반복문 종료!");
```



```
1 2 3 4  
반복문 종료!
```

```
for (int i = 0; i < 3; i++) {  
    for (int j = 0; j < 2; j++) {  
        if(i==j) break;  
        System.out.println(i+" "+j);  
    }//end inner for  
}//end outer for
```



```
1 0  
2 0  
2 1
```

```
while ( true ) {  
  
    if( ... ) {  
  
        break;  
  
    }  
  
}
```

- 무한 루프는 반복문의 **반복 횟수**를 개발자가 **사전에** 정확하게 **인지하지 못하는 상황**에서 사용하며 특정 조건 하에서 반복문을 강제로 종료하는 형태로 구성합니다.

- 프로그램이 **중단되지 않게 유지**할 때도 무한루프를 사용합니다.

- 무한 루프는 일반적으로 while문을 사용하며 while의 조건식 자리에 논리값 true를 적으면 무한 루프를 구성할 수 있습니다.



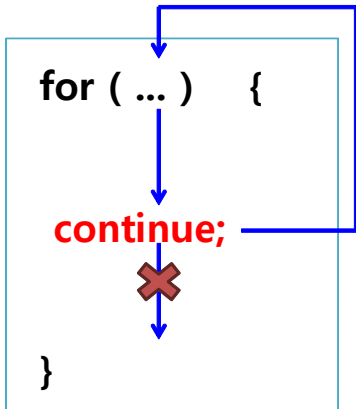
### 3-1-4. 무한루프 사용 예시

```
while(true) {  
  
    int num = (int) (Math.random() * 6) + 1;  
    System.out.println(num);  
    if(num == 6) break;  
  
}  
System.out.println("프로그램 종료!");
```



```
4  
5  
3  
6  
프로그램 종료!
```

### 3-2-1. 탈출문 continue



- **continue**문은 for문이나 while문에서 사용되며 반복문이 실행 중 **continue**를 만나는 순간 **for문의 경우 증감식**, **while문의 경우 조건식**으로 이동합니다.

- **continue**는 **break**와 달리 반복문을 종료하지 않고 계속 수행합니다.

- 조건부로 특정 반복회차를 건너뛸 때 사용합니다.

## 3-2-2. continue문 사용 예시

```
for (int i = 1; i < 11; i++) {  
    if(i%2 == 0) continue;  
    System.out.print(i + " ");  
}
```



```
1 3 5 7 9
```

```
Scanner sc = new Scanner(System.in);  
  
while(true) {  
    System.out.print("정수: ");  
    int n = sc.nextInt();  
  
    if(n == 0) break;  
    else if(n == 1) continue;  
    System.out.println("메롱메롱~~");  
}
```



```
정수: 3  
메롱메롱~~  
정수: 1  
정수: 0  
프로그램 종료!
```

### 3-3. Quiz

-----  
5 + 80 = ?

> 85

정답입니다!!

-----  
51 + 47 = ?

> 22

틀렸는데요??

-----  
11 - 63 = ?

> 0

입력 종료!

=====

정답횟수: 1회

오답횟수: 1회

문제) 왼쪽와 같은 결과가 나오는 프로그램을 코딩하세요.

- 요구사항

1. 프로그램 실행 시 0~100사이의 무작위 두 수의 합을 물어보는 문제가 지속적으로 출제되게 하세요.

2. 올바른 답을 입력할 시 "정답입니다"를 틀린 답을 입력할 시 "오답입니다"를 출력하세요.

3. 사용자가 0을 입력하면 문제 출제를 중단하고 누적된 정답 횟수와 오답횟수를 출력하세요.

**감사합니다**  
**THANK YOU**