

JAVA

- 조건문과 반복문

실행흐름의 컨트롤 (조건문과 반복문)

조건문 – if, switch

- 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");  
}
```



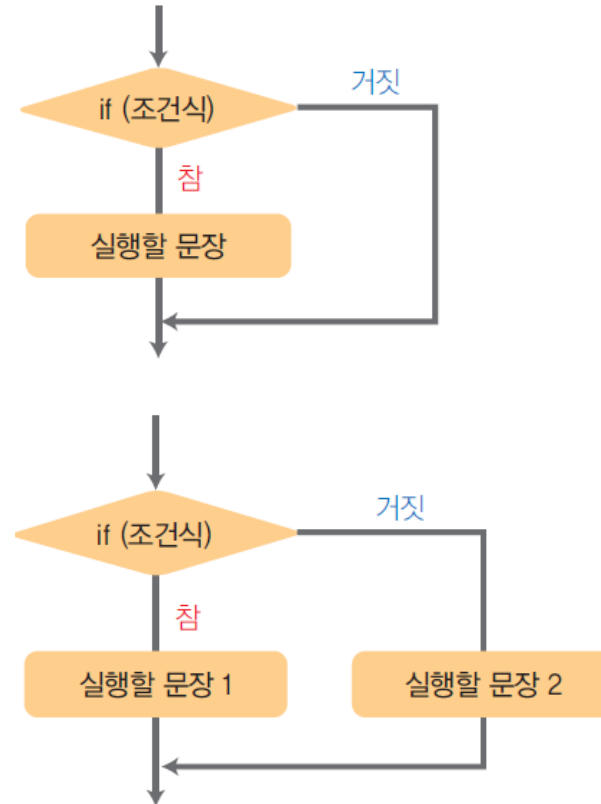
```
switch(num) {  
    case 1:  
        System.out.println("SK");  
        break;  
    case 6:  
        System.out.println("KTF");  
        break;  
    case 9:  
        System.out.println("LG");  
        break;  
    default:  
        System.out.println("UNKNOWN");  
}
```

If 그리고 else

If문과 if~else문

```
if (true or false)
{
    /* true 시 실행되는 영역 */
}
```

```
if (true or false)
{
    /* true 시 실행되는 영역 */
}
else
{
    /* false 시 실행되는 영역 */
}
```



중괄호는 하나의 문장일 때
생략가능!
if~else는 하나의 문장이다!

If문과 if~else문

- 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 ( 조건식 1 ) {  
    // 조건식의 결과가 true일 때 수행될 문장들  
} else if ( 조건식 2 ) {  
    // 조건식의 결과가 false일 때 수행될 문장들  
} else if ( 조건식 3 ) {  
    // 조건식의 결과가 false일 때 수행될 문장들  
}
```

If문과 if~else문

- 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...  
}
```

If문과 if~else문

```
class IEBasic {  
    public static void main(String[] args) {  
  
        if(true) {  
            System.out.println("if & true");  
        }  
        if(false) {  
            System.out.println("if~else & true");  
        } else {  
            System.out.println("if~else & false");  
        }  
    }  
}
```


If문과 if~else문

```
class IEUsage {  
    public static void main(String[] args) {  
        int num=10;  
  
        if(num>0)  
            System.out.println("num은 0보다 크다.");  
  
        if((num%2)==0)  
            System.out.println("num은 짝수");  
        else  
            System.out.println("num은 홀수");  
    }  
}
```

num은 0보다 크다.
num은 짝수

if~else문의 중첩

- 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  
}
```

if~else문의 중첩

```
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";
}
```

if~else 중첩의 일반화

```
if( . . . )  
    System.out.println("...");  
else if( . . . )  
    System.out.println("...");  
else if( . . . )  
    System.out.println("...");  
else if( . . . )  
    System.out.println("...");  
else  
    System.out.println("...");
```

조건이 만족
여부에 따라서
하나만 실행

중간에 else if가
추가 되는 만큼

if~else가
중첩된 형태이다!

문제

문제1.

아래 예제는 두 개의 if문을 사용하고 있다. 한 개의 if 문만 사용하는 방식으로 변경해보자.

```
class IfReit {  
    public static void main(String[] args) {  
        int num=120;  
  
        if(num>0) {  
            if((num%2)==0)  
                System.out.println("양수이면서 짝수");  
        }  
    }  
}
```

문제

문제2.

다음과 같이 출력이 이루어지도록 작성해보자.

$\text{num} < 0$ 이라면 "0 미만" 출력

$0 \leq \text{num} < 100$ 이라면 "0이상 100 미만" 출력

$100 \leq \text{num} < 200$ 이라면 "100이상 200 미만" 출력

$200 \leq \text{num} < 300$ 이라면 "200이상 300 미만" 출력

$300 \leq \text{num}$ 이라면 "300이상 " 출력

if~else와 유사한 성격의 조건 연산자 (삼항연산자)

true or false? 숫자 1 : 숫자 2

```
class CondOp {  
    public static void main(String[] args) {  
        int num1=50, num2=100;  
        int big, diff;  
  
        big = (num1>num2)? num1:num2;  
        System.out.println(big);  
  
        diff = (num1>num2)? num1-num2: num2-num1;  
        System.out.println(diff);  
    }  
}
```

100

50

문제

문제3.

CondOp.java를 조건연산자(3항 연산자)를 사용하지 않고, if~else를 사용하는 형태로 변경해 보자.

Switch와 break

Switch문

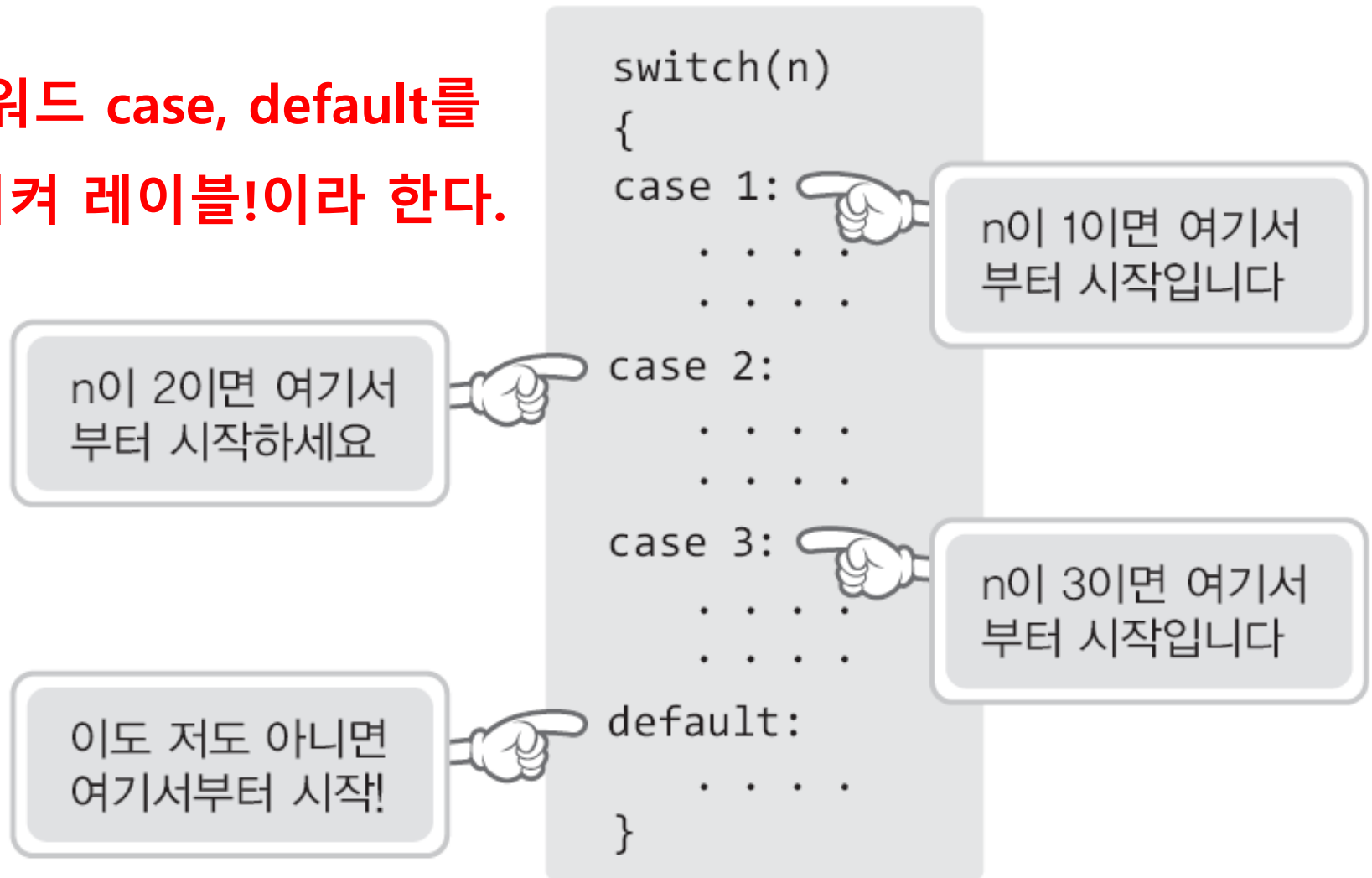
- 조건식의 계산결과와 일치하는 case문으로 이동 후 break문을 만날 때까지 문장들을 수행한다.
(break문이 없으면 switch문의 끝까지 진행한다.)
- 일치하는 case문의 값이 없는 경우 default문으로 이동한다.
(default문 생략가능)
- case문의 값으로 변수를 사용할 수 없다.
(리터럴, 상수만 가능)

Switch문

```
switch (key) { // key --> 조건식 ( int 자료형 이하의 값 )  
    case 1: // 조건식 값이 1과 같다면 수행할 문장들  
        // 조건식 값이 1과 같다면 수행할 문장들  
        break;  
  
    case 2: // 조건식 값이 2와 같다면 수행할 문장들  
        // 조건식 값이 2와 같다면 수행할 문장들  
        break;  
  
    default:  
        // 조건식의 결과와 일치하는 case문이 없을 때 수행할 문장들  
        break;  
}
```

Switch문의 기본 구성

키워드 **case**, **default**를
가리켜 레이블!이라 한다.



예제

```
class SwitchBasic {  
    public static void main(String[] args) {  
        int n=3;  
        switch(n) {  
            case 1:  
                System.out.println("Simple Java");  
            case 2:  
                System.out.println("Funny Java");  
            case 3:  
                System.out.println("Fantastic Java");  
            default:  
                System.out.println("The best programming language");  
        }  
        System.out.println("Do you like coffee?");  
    }  
}
```

Fantastic Java

The best programming language

Do you like coffee?

The best programming language

Do you like coffee?

예제

```
class SwitchBreak {  
    public static void main(String[] args) {  
        int n=3;  
        switch(n) {  
            case 1:  
                System.out.println("Simple Java");  
                break;  
            case 2:  
                System.out.println("Funny Java");  
                break;  
            case 3:  
                System.out.println("Fantastic Java");  
                break;  
            default:  
                System.out.println("The best programming language");  
        }  
        System.out.println("Do you like coffee?");  
    }  
}
```

Funny Java
Do you like coffee?

Fantastic Java
Do you like coffee?

Switch문의 또 다른 구성

```
switch(n)
{
    case 1 : case 2 : case 3 :
        System.out.println("Simple Java");
        break;
    case 4 : case 5 :
        System.out.println("Funny Java");
        break;
    . . . . .
}
```

n이 1, 2, 3인 경우를 하나의 부류로 묶는다!

n이 4, 5인 경우를 하나의 부류로 묶는다!

Switch문

```
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';
}
```


Switch문 중첩

- 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");  
}
```

If문과 Switch문의 비교

- if문이 주로 사용되며, 경우의 수가 많은 경우 switch문을 사용할 것을 고려한다.
- 모든 switch문은 if문으로 변경이 가능하지만, if문은 switch문으로 변경 할 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");  
}
```

문제

문제 4.

SwitchBreak.java를 switch문이 아닌, if~else를 사용하는 형태로 변경해 보자.

문제 5.

문제 2의 답안 코드를 switch 문으로 변경하여 보자.

**반복을 위한
for, while 그리고 do~while**

반복문

- 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++;  
}
```

while 반복문

- while문

- 조건식과 수행할 블록{} 또는 문장으로 구성

```
while (condition) {  
    // 조건식의 연산 결과가 true일 때 수행될 문장을 적는다.  
}
```

while(num<5)

반복
조건

{

```
System.out.println("I like Java"+ num);  
num++;
```

}

반복
영역

while 반복문의 중첩

- 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++;  
}
```

while 반복문

```
class WhileBasic
{
    public static void main(String[] args)
    {
        int num=0;

        while(num<5)
        {
            System.out.println("I like Java " + num);
            num++;
        }
    }
}
```

```
I like Java 0
I like Java 1
I like Java 2
I like Java 3
I like Java 4
```


do~while 반복문

- 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);
```

```
do  
{
```

반복
영역

do~while문은 최소 한번은 실행이 된다!

```
    System.out.println("I like Java"+ num);
```

```
    num++;
```

반복
조건

```
} while( num<5 );
```

do~while 반복문

```
class DoWhileBasic
{
    public static void main(String[] args)
    {
        int num=0;

        do
        {
            System.out.println("I like Java " + num);
            num++;
        }while(num<5);
    }
}
```

```
I like Java 0
I like Java 1
I like Java 2
I like Java 3
I like Java 4
```

문제

문제 6.

while 문을 이용해서 1부터 99까지의 합을 구하는 프로그램을 작성.

문제 7.

1부터 100까지 출력한 후, 다시 거꾸로 100부터 1까지 출력하는 프로그램을 작성. while문과 do~while 문을 각각 한번씩 사용해서 작성

문제 8.

1000 이하의 자연수 중에서 2의 배수 이면서 7의 배수인 숫자를 출력하고, 그 출력된 숫자들의 합을 구하는 프로그램을 while 문을 이용해서 작성

for 반복문, while문과의 비교

- 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.

```
int num=0; 1.  
while( num<5 ) 2.  
{  
    System.out.println("...");  
    num++; 3.  
}
```

```
1. 2. 3.  
for( int num=0 ; num<5 ; num++ )  
{  
    System.out.println("...");  
}
```

1. → 반복의 횟수를 세기 위한 변수

2. → 반복의 조건

3. → 반복의 조건을 무너뜨리기 위한 연산

for 반복문의 실행흐름

첫 번째 루프의 흐름

1 ⇒ 2 ⇒ 3 ⇒ 4 [i=1]

두 번째 루프의 흐름

2 ⇒ 3 ⇒ 4 [i=2]

세 번째 루프의 흐름

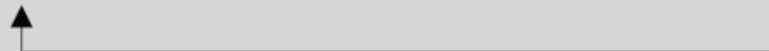
2 ⇒ 3 ⇒ 4 [i=3]

네 번째 루프의 흐름

2 [i=3] 따라서 탈출!

```
for( 1. int i=0 ; 2. i<3 ; 4. i++ )  
{  
    3. System.out.println("...");  
}
```

1. 초기화 → 2. 조건식 → 3. 수행될 문장 → 4. 증감식



for 반복문

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	

for 반복문

```
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);
}
```

for 반복문

for문 작성 예	설 명
<pre>for(;;) { /* 반복해서 수행할 문장들 */ }</pre>	조건식이 없기 때문에 결과가 true로 간주되어 블럭{}안의 문장들을 무한히 반복수행한다.
<pre>for(int i=0;;) { /* 반복해서 수행할 문장들 */ }</pre>	for문에 int형 변수 i를 선언하고 0으로 초기화 했다. 변수 i는 for문 내에 선언되었기 때문에 for문 내에서만 유효하다.
<pre>for(int i=1,j=1;i<10 && i*j<50;i++,j+=2) { /* 반복해서 수행할 문장들 */ }</pre>	쉼표(,)를 이용해서 하나 이상의 변수를 선언하고 초기화 할 수 있다. 단, 같은 타입인 경우만 가능하다. 증감식 역시 쉼표를 이용해서 여러 문장이 수행되게 할 수 있다. 여기서는 매 반복마다 i는 1씩, j는 2씩 증가한다.

```
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

여러발생

문제

문제 9.

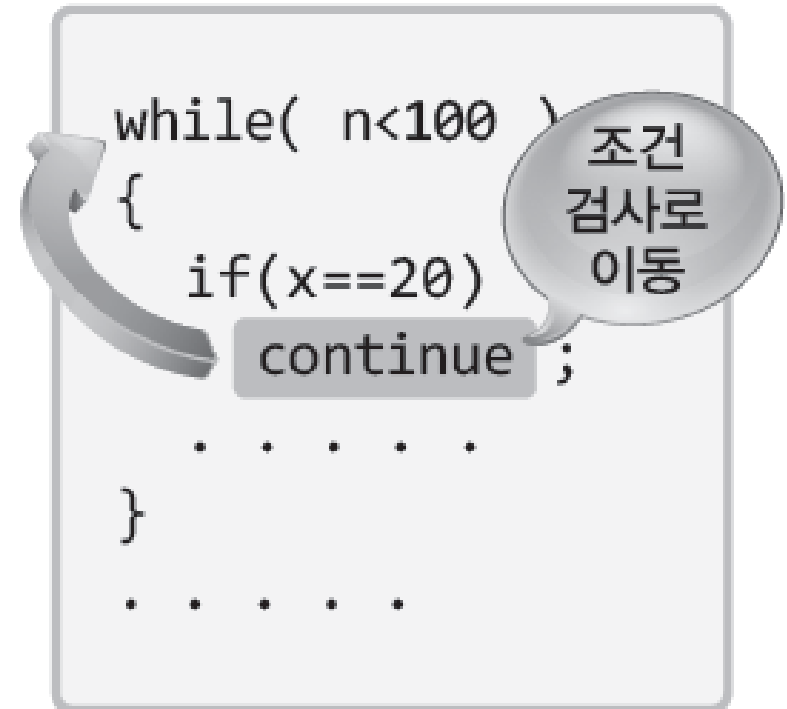
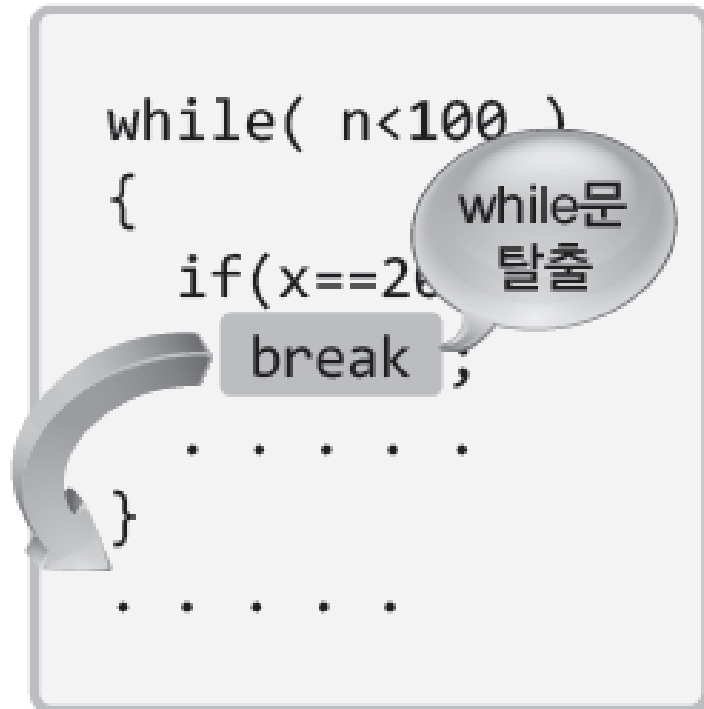
for 문을 이용하여 1부터 10까지를 곱해서 그 결과를 출력하는 프로그램을 작성

문제 10.

for 문을 이용하여 구구단 중 5단을 출력하는 프로그램 작성

continue & break

continue & break문



continue & break문

• 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.

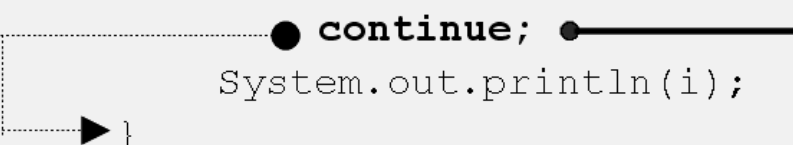
i	sum
0	0
1	1
2	3
3	6
...	...
13	91
14	105

continue & break문

- 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

continue & break문

```
class BreakBasic {  
    public static void main(String[] args) {  
        int num=1;  
        boolean search=false;  
        while(num<100) {  
            if(num%5==0 && num%7==0) {  
                search=true;  
                break;  
            }  
            num++;  
        }  
        if(search)  
            System.out.println("찾는 정수 : " + num);  
        else  
            System.out.println("5의 배수이자 7의 배수를 찾지 못했습  
니다.");  
    }  
}
```

continue & break문

```
class ContinueBasic
{
    public static void main(String[] args)
    {
        int num=0;
        int count=0;

        while((num++)<100)
        {
            if(num%5!=0 || num%7!=0)
                continue;

            count++;
            System.out.println(num);
        }

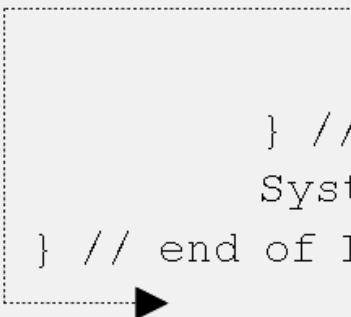
        System.out.println("count: " + count);
    }
}
```

continue & break문

- 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
```


무한 루프와 break

```
while(true)
{
    . . . .
}
```

```
do
{
    . . . .
} while(true);
```

```
for( ; ; )
{
    . . . .
}
```

무한 루프와 break

```
class InfLoop
{
    public static void main(String[] args)
    {
        int num=1;

        while(true)
        {
            if(num%6==0 && num%14==0)
                break;
            num++;
        }

        System.out.println(num);
    }
}
```

**6의 배수이자 14의 배수인
가장 작은 정수 찾기!**

문제

문제 11.

ContinueBasic.java의 내부에 존재하는 while 문을 for 문으로 변경하여 작성

문제 12.

자연수 1부터 시작해서 모든 홀수와 3의 배수인 짝수를 더해 나간다.
그리고 그 합이 언제 (몇을 더했을 때) 1000이 넘어서는지,
그리고 1000이 넘어서선 값은 얼마가 되는지 계산하여 출력하는
프로그램을 작성.

프로그램 내부에 while문을 무한 루프로 구성하여 작성.

반복문의 중첩

생각해볼 수 있는 중첩의 종류는?

```
for(...;...;...)  
{  
    for(...;...;...)  
    {  
        . . . . .  
    }  
}
```

```
while(...)  
{  
    for(...;...;...)  
    {  
        . . . . .  
    }  
}
```

```
do{  
    for(...;...;...)  
    {  
        . . . . .  
    }  
}while(...);
```

```
for(...;...;...)  
{  
    while(...)  
    {  
        . . . . .  
    }  
}
```

```
while(...)  
{  
    while(...)  
    {  
        . . . . .  
    }  
}
```

```
do{  
    while(...)  
    {  
        . . . . .  
    }  
}while(...);
```

```
for(...;...;...)  
{  
    do{  
        . . . . .  
    }while(...);  
}
```

```
while(...)  
{  
    do{  
        . . . . .  
    }while(...);  
}
```

```
do{  
    do{  
        . . . . .  
    }while(...);  
}while(...);
```

생각해볼 수 있는 중첩의 종류는?

```
class DupFor
{
    public static void main(String[] args)
    {
        for(int i=0; i<3; i++)
        {
            System.out.println("변수 i의 값: " + i);
            for(int j=0; j<3; j++)
                System.out.println("***변수 j의 값: " + j);
        }
    }
}
```

가장 많이 등장하는 for문의 중첩

```
class ByTimes
{
    public static void main(String[] args)
    {
        for(int i=2; i<10; i++)
        {
            for(int j=1; j<10; j++)
                System.out.println(i + " x " + j + " = " + i*j);
        }
    }
}
```

2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
/* ~중간생략~ */
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81

안쪽 for 문
담당 영역

2×1=2	3×1=3	4×..	5..	6.	7.	8.	9×1=9
2×2=4	3×2=6	4×..	5..	6.	7.	8.	9×2=18
2×3=6	3×3=9	4×..	5..	6.	7.	8.	9×3=27
2×4=8	3×4=12	4×..	5..	6.	7.	8.	9×4=36
2×5=10	3×5=15	4×..	5..	6.	7.	8.	9×5=45
2×6=12	3×6=18	4×..	5..	6.	7.	8.	9×6=54
2×7=14	3×7=21	4×..	5..	6.	7.	8.	9×7=63
2×8=16	3×8=24	4×..	5..	6.	7.	8.	9×8=72
2×9=18	3×9=27	4×..	5..	6.	7.	8.	9×9=81

바깥 for 문
담당 영역

While문 중첩

```
class ByTimes2
{
    public static void main(String[] args)
    {
        int i=2, j;
        while(i<10)
        {
            j=1;
            while(j<10)
            {
                System.out.println(i + " x " + j + " = " + i*j);
                j++;
            }
            i++;
        }
    }
}
```

2 x 1 = 2

2 x 2 = 4

2 x 3 = 6

/* ~중간생략~ */

9 x 7 = 63

9 x 8 = 72

9 x 9 = 81

break 레이블

```
class LabeledBreak {  
    public static void main(String[] args) {  
  
        outerLoop:  
        for(int i=1; i<10; i++) {  
  
            for(int j=1; j<10; j++) {  
                System.out.println "[" + i + ", " + j + " ]");  
                if(i%2==0 && j%2==0)  
                    break outerLoop;  
            }  
        }  
    }  
}
```

Break문은 자신을 감싸는 반복 문을 하나밖에 벗어나지 못한다.

때문에 둘 이상의 반복 문을 벗어날 때는 break 레이블을 사용을 고려할 수 있다!

하지만 빈번한 사용은 바람직하지 못하다!

문제

문제 13.

구구단의 짝수 단(2,4,8)만 출력하는 프로그램 작성.
단, 2단은 2x2까지, 4단은 4x4까지, 8단은 8x8 까지 출력

문제 14.

다음 식을 만족하는 조합을 찾는 프로그램 작성.

$$\begin{array}{r} A \ B \\ B \ A \\ \hline 9 \ 9 \end{array}$$

문제

문제15.

$n=1$ 일 때 → "현재 인원은 1명 입니다."

$n=2$ 일 때 → "현재 인원은 2명 입니다."

$n=3$ 일 때 → "현재 인원은 3명 입니다."

$n>3$ 일 때 → "현재 많은 사람들이 있습니다."

위의 내용이 출력 되는 프로그램을 작성. 각각 if 문과 switch 문 사용

문제16.

1 부터 99까지의 합을 구하는 프로그램 작성

while문, for문, do while 문을 각각 사용

정리합시다.

- 반복문의 종류
- 반복문의 특징
- 반복의 조건과 탈출의 조건
- 반복이 필요한 이유
- 다음 페이지의 예제들은 코딩 연습을 하면서 문장마다 설명과 주석을 작성하면서 연습합니다.

예제

```
class FlowEx1
{
    public static void main(String[] args)
    {
        int visitCount = 0;
        if (visitCount < 1) {
            System.out.println("처음 오셨군요. 방문해 주셔서 감사합니다.") ;
        }
    }
}
```

예제

```
class FlowEx2
{
    public static void main(String[] args)
    {
        int visitCount = 5;
        if (visitCount < 1) {      // 5 < 1의 연산결과는 false.
            System.out.println("처음 오셨군요. 방문해 주셔서 감사합니다.");
        } else {
            System.out.println("다시 방문해 주셔서 감사합니다.");
        }
        System.out.println("방문횟수는 " + ++visitCount + "번 입니다.");
    }
}
```

예제

```
class FlowEx3 {  
    public static void main(String[] args) {  
  
        int score = 45;  
        char grade = ' '; // 학점을 저장하기 위한 변수. 공백으로 초기화한다.  
  
        if (score >= 90) {           // score가 90점 보다 같거나 크면 A학점  
            grade = 'A';  
        } else if (score >=80) {    // score가 80점 보다 같거나 크면 B학점  
            grade = 'B';  
        } else {                   // 나머지는 C학점  
            grade = 'C';  
        }  
  
        System.out.println("당신의 학점은 " + grade + "입니다.");  
    }  
}
```


예제

```
class FlowEx4
{
    public static void main(String[] args)
    {
        int score = 45;
        char grade = ' ';
        grade = (score >=90) ? 'A' : ((score >=80) ? 'B' : 'C');

        System.out.println("당신의 학점은 " + grade + "입니다.");
    }
}
```

예제

```
class FlowEx5
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        int score = 82;
```

```
        String grade = "";
```

```
        // 두 문자를 저장할 것이므로 변수의 타입을 String으로 함.
```

```
        System.out.println("당신의 점수는 " + score + "입니다.");
```

```
        if (score >= 90) {
```

```
            // score가 90점 보다 같거나 크면 A학점(grade)
```

```
                grade = "A";
```

```
                if ( score >= 98) {
```

```
                    // 90점 이상 중에서도 98점 이상은 A+
```

```
                        grade += "+"; // grade = grade + "+";와 같다.
```

```
                } else if ( score < 94)      {
```

```
                    grade += "-";
```

```
                }
```

예제

```
    } 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";  
    }  
    System.out.println("당신의 학점은 " + grade + "입니다.");  
}  
}
```

예제

```
class FlowEx6 {  
    public static void main(String[] args) {  
        // Math클래스의 random()함수를 이용해서 1~10사이의 값을 얻어낼 수 있다.  
        int score = (int)(Math.random() * 10) + 1;  
        switch(score*100) {  
            case 100 :  
                System.out.println("당신의 점수는 100이고, 상품은 자전거입니다.");  
                break;  
            case 200 :  
                System.out.println("당신의 점수는 200이고, 상품은 TV입니다.");  
                break;  
            case 300 :  
                System.out.println("당신의 점수는 300이고, 상품은 노트북입니다.");  
                break;  
            case 400 :  
                System.out.println("당신의 점수는 400이고, 상품은 자동차입니다.");  
                break;  
            default :  
                System.out.println("죄송하지만 당신의 점수에 해당상품이 없습니다.");  
        }  
    }  
}
```

예제

```
class FlowEx7 {  
    public static void main(String[] args) {  
  
        // 'A', 'B', 'C', 'D' 중의 하나를 얻을 수 있다.  
        char ch = (char)(Math.random() * 4 + 'A');  
        int score = 0;  
  
        switch (ch)  
        {  
            case 'A':  
                score = 90;  
                break;  
            case 'B':  
                score = 80;  
                break;  
            case 'C':  
                score = 70;  
                break;  
        }  
    }  
}
```

예제

```
        case 'D':
            score = 60;
            break;
    }

    System.out.println("당신의 점수는 " + score + "점 이상 입니다.");
}
}
```

예제

```
class FlowEx8
{
    public static void main(String[] args)
    {
        int score = 1;

        switch(score*100) {
            case 100 :
                System.out.println("당신의 점수는 100이고, 상품은 자전거입니다.");
            case 200 :
                System.out.println("당신의 점수는 200이고, 상품은 TV입니다.");
            case 300 :
                System.out.println("당신의 점수는 300이고, 상품은 노트북입니다.");
            case 400 :
                System.out.println("당신의 점수는 400이고, 상품은 자동차입니다.");
            default :
                System.out.println("죄송하지만 당신의 점수에 해당상품이 없습니다.");
        }
    }
}
```

예제

```
class FlowEx9 {  
    public static void main(String[] args) {  
        int score = (int)(Math.random() * 10) + 1;  
        String msg = "";  
        score *= 100;                // score = score * 100;  
        msg = "당신의 점수는 " + score + "이고, 상품은 ";  
  
        switch(score) {  
            case 1000 :  
                msg += "자전거, "; // msg = msg + "자전거, ";  
            case 900 :  
                msg += "TV, ";  
            case 800 :  
                msg += "노트북, ";  
            case 700 :  
                msg += "자전거, ";  
            default :  
                msg += "볼펜";  
        }  
        System.out.println( msg + "입니다.");  
    }  
}
```


예제

```
class FlowEx10 {  
    public static void main(String[] args) {  
        int score = 88;  
        char grade = ' ';  
        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 :
```

예제

```
        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
System.out.println("당신의 학점은 " + grade + "입니다.");

} // end of main
} // end of class
```

예제

```
class FlowEx11 {  
    public static void main(String[] args) {  
        int score = 88;  
        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("당신의 학점은 " + grade + "입니다.");  
    }  
}
```

예제

```
class FlowEx13
{
    public static void main(String[] args)
    {
        int sum =0;                                // 합계를 저장하기 위한 변수.

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

예제

```
class FlowEx14
{
    public static void main(String[] args)
    {
        int sum =0;           // 합계를 저장하기 위한 변수.
        int i;                // 선언부분을 for문 밖으로 옮겼다.
        for(i=1; i <= 10; i++) {
            sum += i;          //      sum = sum + i;
        }
        System.out.println( i-1 + " 까지의 합: " + sum);
    }
}
```

예제

```
class FlowEx15
{
    public static void main(String[] args)
    {
        int sum =0;
        for(int i=0; i <=10; i+=2) {
            sum += i;                // sum = sum + i;
            System.out.println(i + " : " + sum);
        }
    }
}
```

예제

```
class FlowEx16
{
    public static void main(String[] args)
    {
        for(int i=2; i <=9; i++) {
            for(int j=1; j <=9; j++) {
                System.out.println( i + " * " + j + " = " + i*j );
            }
        }
    }
}
```

예제

```
class FlowEx17
{
    public static void main(String[] args)
    {
        for(int i=2; i <=9; i++)
            for(int j=1; j <=9; j++)
                System.out.println( i + " * " + j + " = " + i*j );
    }
}
```


예제

```
class FlowEx18 {  
    public static void main(String[] args) {  
        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);  
    }  
}
```

예제

```
class FlowEx19 {  
    public static void main(String[] args) {  
        long startTime = System.currentTimeMillis();  
        for(int i=0; i < 10000000000; i++) {  
            ;  
        }  
        long endTime = System.currentTimeMillis();  
  
        System.out.println("시작시간 : " + startTime);  
        System.out.println("종료시간 : " + endTime);  
        System.out.println("소요시간 : " + (endTime - startTime));  
    }  
}
```

예제

```
class FlowEx20 {  
    public static void main(String[] args)    {  
        System.out.println("자, 이제 카운트다운을 시작합니다.");  
        for(int i=10; i >= 0; i--) {  
            for(int j = 0; j < 10000000000; j++) {  
                ;  
            }  
            System.out.println(i);  
        }  
        System.out.println("GAME OVER");  
    }  
}
```

예제

```
class FlowEx21 {  
    public static void main(String[] args)  
    {  
        int i=10;  
        while(i >=0) {  
            System.out.println(i--);  
        }  
    }  
}
```

예제

```
class FlowEx22
{
    public static void main(String[] args)
    {
        int i=2;

        while(i<=9) {
            int j=1;

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

            i++;
        } // end of while(i<=9)
    }
}
```

예제

```
class FlowEx23
{
    public static void main(String[] args)
    {
        int sum = 0;
        int i = 0;

        while(sum + i <= 100) {
            sum += ++i;    // sum = sum + ++i;
            System.out.println(i + " - " + sum);
        }
    }
}
```

예제

```
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');  
    }  
}
```

예제

```
class FlowEx25
{
    public static void main(String[] args)
    {
        int sum =0;
        int i = 1;

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

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


예제

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

예제

```
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;  
                // break;  
                // continue Loop1;  
                // continue;  
                System.out.println(i+"*" + j + "=" + i*j);  
            } // end of for i  
            System.out.println();  
        } // end of Loop1  
  
    }  
}
```