Chapter 05. 실행흐름의 컨트롤

05-1. 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 */
}
```

```
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");
    }
}
```

실행결과

```
num은 0보다 크다.
num은 짝수
```

중괄호는 하나의 문장일 때 생략 가능!
if~else는 하나의 문장이다!

■ if~else문의 중첩 그리고 중괄호의 생략

나 정말 JAVA를 공부한 적 없는데

STEP 1.

```
if(num<0)
    System.out.println("0 미만");
else
{
    if(num<100)
        System.out.println("0이상 100 미만");
    else
        System.out.println("100 이상");
}
```

STEP 2.

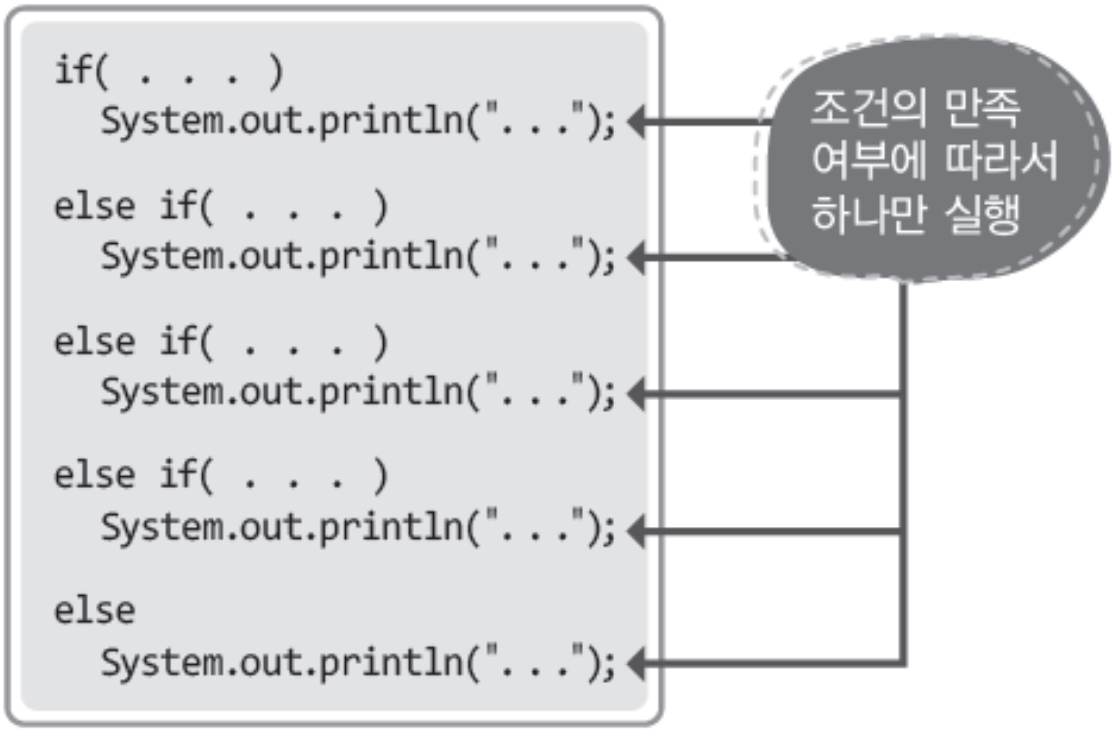
```
if(num<0)
    System.out.println("0 미만");
else
    if(num<100)
        System.out.println("0이상 100 미만");
    else
        System.out.println("100 이상");
```

STEP 3.

```
if(num<0)
    System.out.println("0 미만");
else if(num<100)
    System.out.println("0이상 100 미만");
else
    System.out.println("100 이상");
```

셋 중 하나만 실행이 됨에 주목!

if~else 중첩의 일반화



중간에 else if가 추가되는만큼 if~else가 중첩된 형태이다!

■ if~else와 유사한 성격의 조건 연산자

```
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);
    }
}
```

실행 결과

```
100
50
```

05-2. switch와 break

switch문의 기본 구성


```
switch(n)
{
case 1:
    . . . .
    . . . .
case 2:
    . . . .
    . . . .
case 3:
    . . . .
    . . . .
default:
    . . . .
}
```

n이 10이면 여기서 부터 시작입니다

n이 20이면 여기서 부터 시작하세요

n이 30이면 여기서 부터 시작입니다

이도 저도 아니면 여기서부터 시작!

키워드 case, default를 가리켜 레이블이라 한다.

switch문의 예

```
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?");
    }
}
```

변수 n이 3일 때

```
Fantastic Java
The best programming language
Do you like coffee?
```

변수 n이 5일 때

```
The best programming language
Do you like coffee?
```

■ switch문 + break문: switch문의 일반적 사용 모델


```
switch(n)
{
case 1:
    . . . .
    break;

case 2:
    . . . .
    break;

case 3:
    . . . .
    break;

default:
    . . . .
}
```

case 1 영역

case 2 영역

case 3 영역

default 영역

■ switch문 + break문의 예

```
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?");
    }
}
```

변수 n이 2일 때

```
Funny Java
Do you like coffee?
```

변수 n이 3일 때

```
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;
    . . . . .
}
```

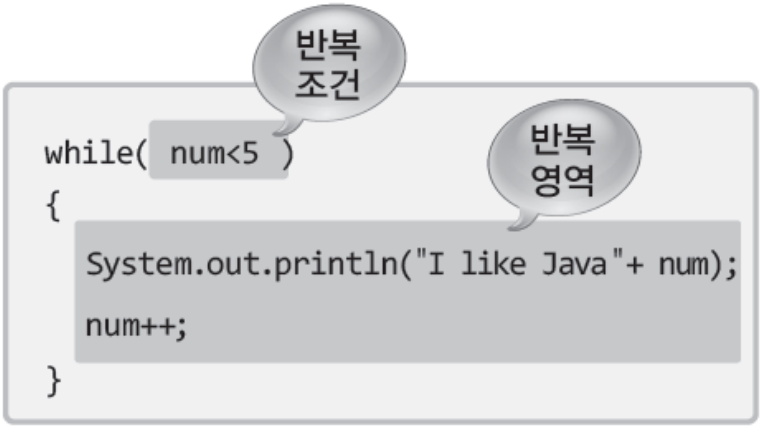
n이 1, 2, 3인 경우를 하나의 부류로 묶는다!

n이 4, 5인 경우를 하나의 부류로 묶는다!

나는 정말 JAVA를 공부한 적이 없다고

05-3. for, while 그리고 do~while

while 반복문



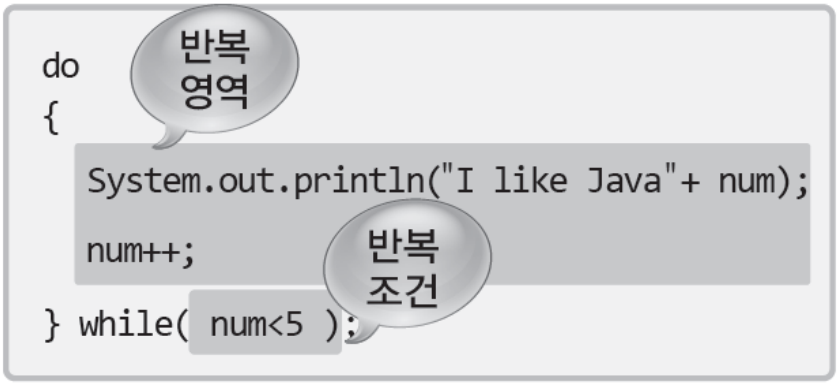
while문은 한번도 실행되지 않을 수 있다!

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

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

실행결과

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

do~while 반복문



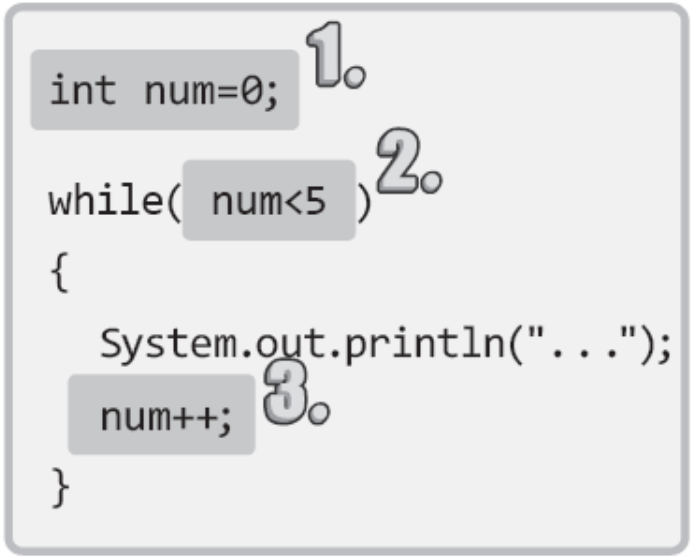
do~while문은 최소 한번은 실행이 된다!

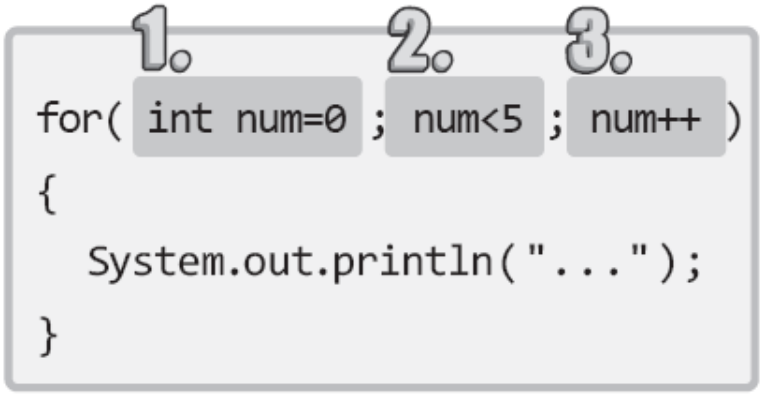
실행결과

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

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

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

■ for 반복문과 while 반복문의 비교





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("...");
}
```

```
class ForBasic
{
    public static void main(String[] args)
    {
        for(int i=0; i<3; i++)
            System.out.println("I love Java " + i);
    }
}
```

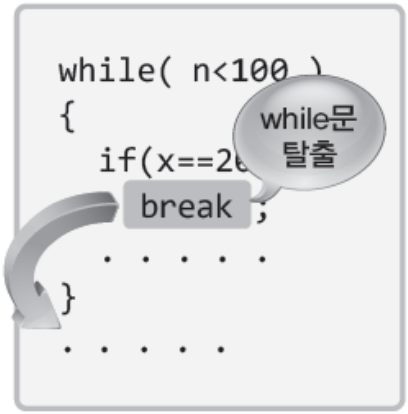
실행결과

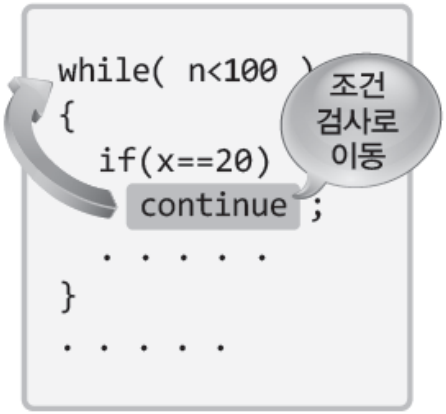
```
I love Java 0
I love Java 1
I love Java 2
```

나는 정말 JAVA를 공부한 적이 없다

05-4. continue & break

continue & break문






```
while(num<100)
{
    if(num%5==0 && num%7==0)
    {
        search=true;
        break;
    }
    num++;
}
```

break문의 예

continue문의 예

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

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

무한루프와 break

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

case 1.

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

case 2.

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

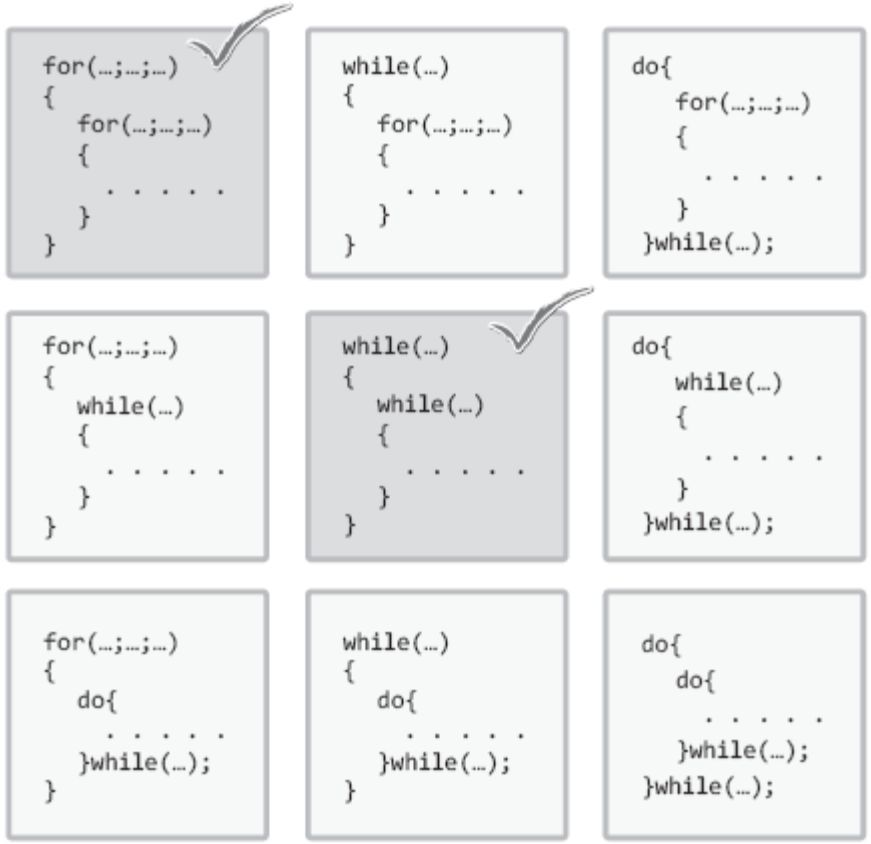
case 3.

```
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!
- 무한루프가 어울리는 사례!

05-5. 반복문의 중첩

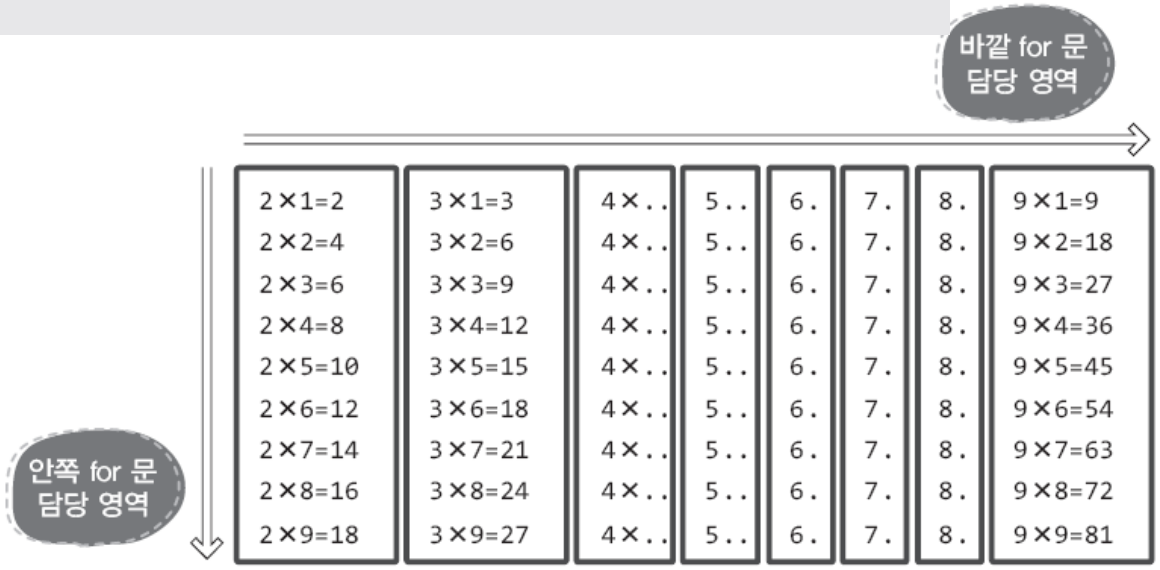
■ 생각해 볼 수 있는 중첩의 종류는?



■ 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
```



while문도 중첩해 봅시다.

```
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++;
        }
    }
}
```

실행 결과

```
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 반복문을 하나밖에 벗어나지 못한다. 때문에 둘 이상의 반복문을 벗어날 때는 break 레이블을 사용을 고려할 수 있다! 하지만 빈번한 사용은 바람직하지 못하다!