JAVA 기초입문과정

CHAPTER04 조건문과 반복문

Contents

조건문

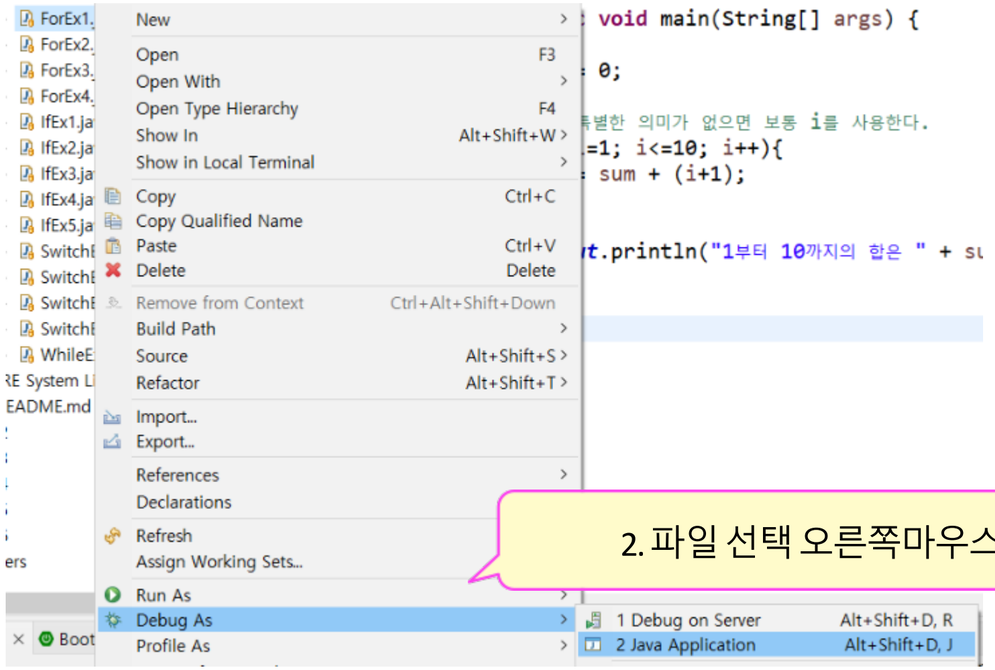
반복문

디버깅 모드로 프로그램 실행하기

* 디버그: 프로그래밍 중에 발생하는 버그를 찾고 수정하기위한 과정이다

```
10          int sum = 0;
11
12
13
14              sum = sum + (i+1);
15          }
16
17          System.out.println("1부터 10까지의 합은 " + sum + "입니다.");
```

1. 원하는 라인에서 더블클릭하여 브레이크 포인트 걸기



또는 이클립스 상단에서 벌레모양 버튼 클릭

2. 파일 선택 오른쪽마우스 클릭, Debug 클릭

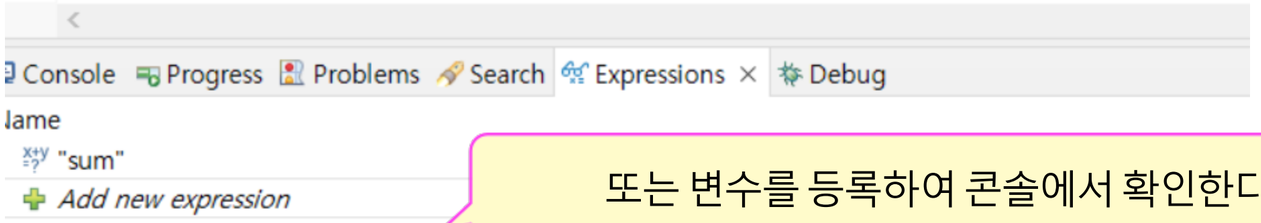
프로그램 실행과정 추적하기

* 진행
F6: 한줄씩 브레이크
F8: 해당라인만 브레이크

```
8    public static void main(String[] args) {
9
10       int sum = 0;
11
12       // 변수에 특별한 의미가 없
13       for(int i=1; i<=10; i++){
14           sum = sum + (i+1);
15       }
16
17       System.out.println("1부터 10까지의 합은 " + sum + "입니다.");
18   }
19 }
20
```

3. 변수에 마우스를 올린다.

* 메뉴추가: windows - show view - others - debug - expression



또는 변수를 등록하여 콘솔에서 확인한다.


PART
01
“
조건문
”

PART 01

조건문

* 조건문이란?

-> 주어진 조건에따라 수행할 문장을 선택하는 것

● 예시

```
만약(나이가 8살 이상이라면){
    "학교를 다닌다" 출력
}
그렇지 않다면{
    "학교를 다니지 않는다" 출력
}
```

블록

*블록의 의미
1. 조건문식 결과에 따라 수행되는 코드의 범위
2. 안에 선언된 변수는 밖에서 사용할수 없다.
   메모리에서 소멸된다.

PART 01

조건문

1. 단순 if문

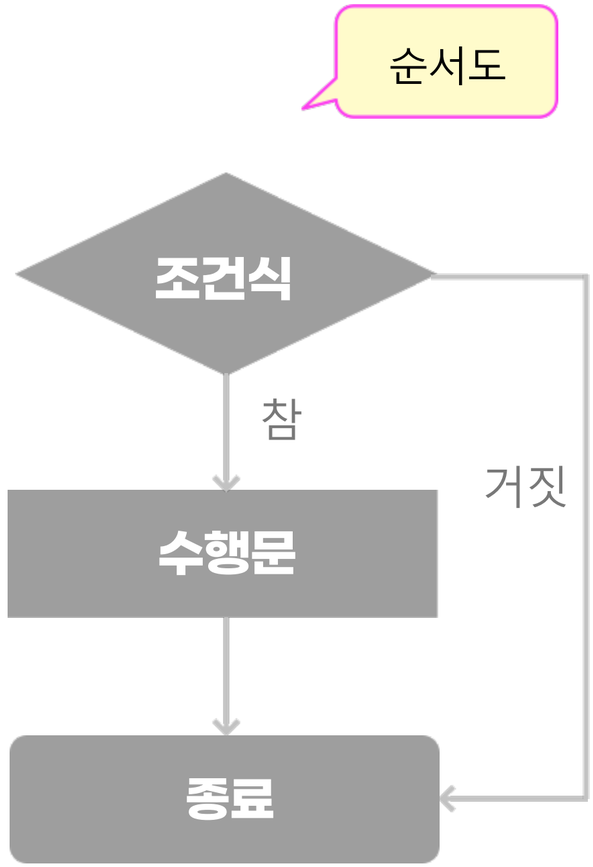
-> if의 괄호 안에 조건식이 ture이면 명령문 실행

```
if (조건식) {
  수행문
}
```

참이면 이문장을 수행함

```
int age = 10;
if ( age >= 8 ){
   System.out.println("학교에 다닙니다.")
}
```

[실행결과]
학교에 다닙니다.



2. if-else문

-> if의 괄호 안에 조건식이 ture이면 명령문 실행

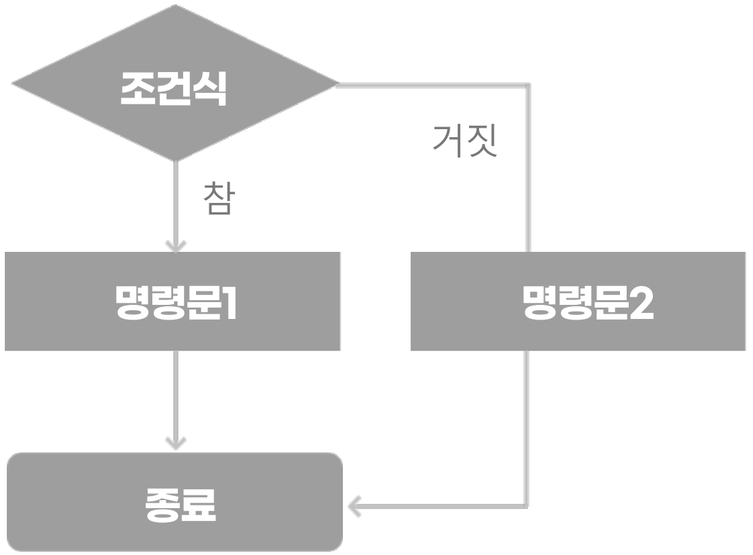
```
if (조건식) {
    수행문1
} else {
    수행문2
}
```

참이면 이문장을 수행함

거짓이면 이문장을 수행함

```
int age = 7;
if ( age >= 8 ){
   System.out.println("학교에 다닙니다.")
} else {
   System.out.println("학교에 다니지 않습니다.")
}
```

[실행결과]
학교에 다니지 않습니다.



3. if~elseif~else 문

-> if의 괄호 안에 조건식이 ture이면 명령문 실행

하나의 상황에 여러 조건을 비교하는 경우에 사용한다
ex. 나이가 15~20 사이인 여성인 경우 -> 조건 3가지 -> 스위치문으로 작성하기 어려움

```
if (조건식1) {
    수행문1
} else if (조건식2){
    수행문2
} else if (조건식3){
    수행문3
} else {
    수행문4
}
```

참이면 이문장을 수행함

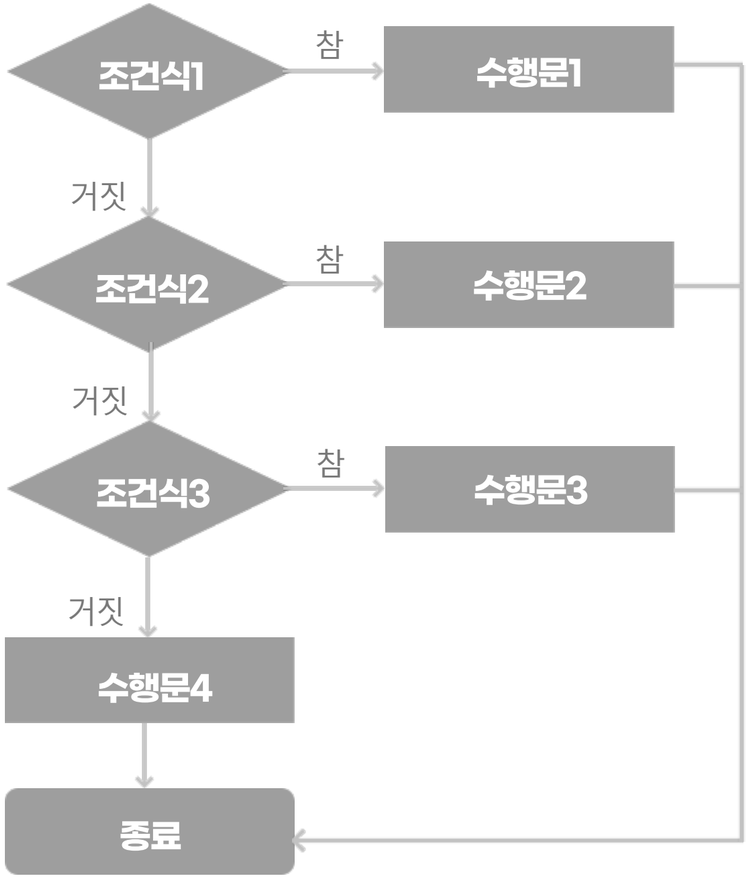
참이면 이문장을 수행함

참이면 이문장을 수행함

모두 거짓이면 이문장을 수행함

```
int n = 9;
if ( n < 8 ){
    System.out.println("아동입니다.")
} else if ( n < 14 ) {
    System.out.println("초등학생입니다.")
} else if ( n < 20 ) {
    System.out.println("청소년입니다.")
} else {
    System.out.println("성인입니다.")
}
```

[실행결과]
초등학생입니다.



* if문을 사용하면, else로 마무리하기 -> 조건에 해당하지 않을 경우 어떻게 처리할지 고민하는 습관을 가진다 -> 프로젝트의 완성도를 높인다

4. switch~case문

- 조건식의 값에따라 여러갈래로 명령이 분기되는 조건문이다.

하나의 값으로 조건을 비교하는 상황에서 사용한다

일반적으로 조건식보다 변수를 사용한다

```
switch (조건식 또는 변수) {
case 1:  수행문1;  break;
case 2:  수행문2;  break;
case 3:  수행문3;  break;
default:  수행문4;
}
```

→ 모두 거짓이면 dafault문을 수행

* 변수타입
o : char, String, 정수형
x : 실수형, boolean

```
int rank = 1;
String medalColor;
switch (rank){
case 1 :
    medalColor = "금색";
    break;
case 2 :
    medalColor = "은색"
    break;
case 3 :
    medalColor = "동색"
    break;
default :
    medalColor = "검정"
System.out.println("메달 색은 medalColor입니다.")
}
```

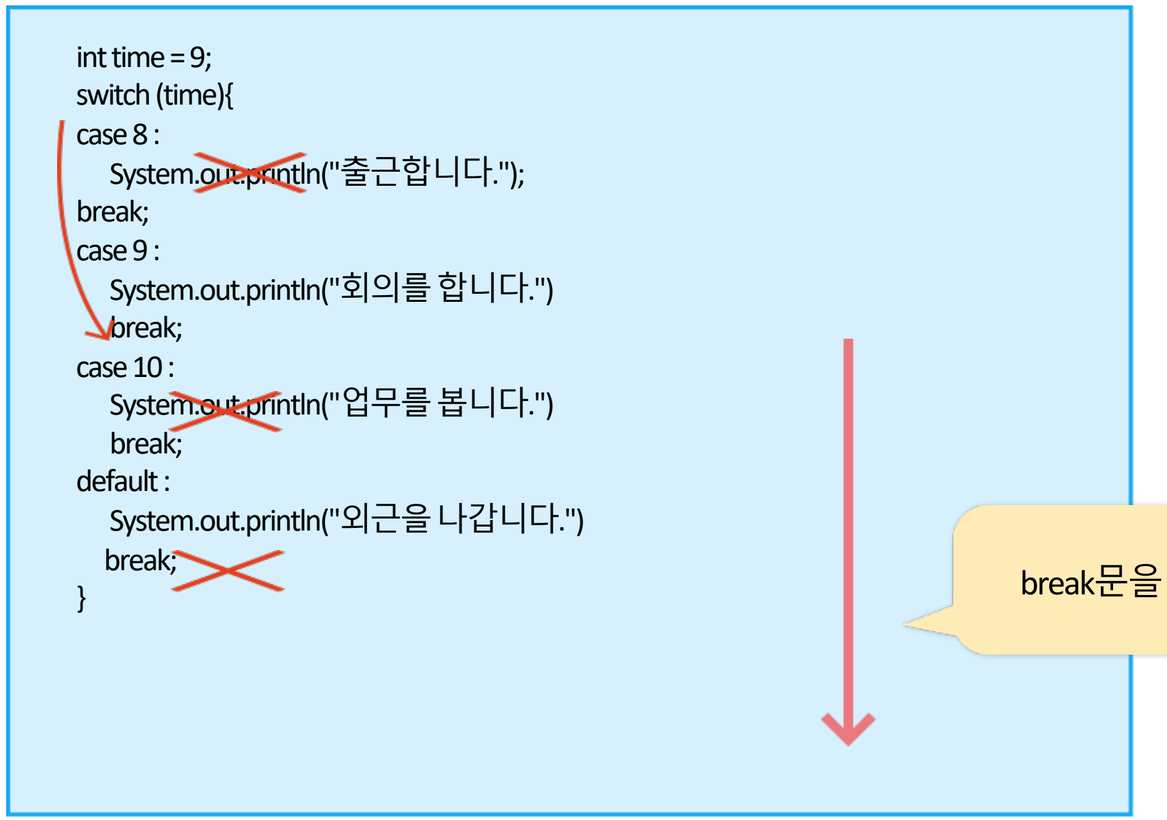
[실행결과]
1등 메달의 색은 금색입니다.

{}를 사용하지 않아서 코드가 깔금하고 가독성이 좋다.

5. switch에서 break의 역할

-> break를 만나면 switch문을 벗어남
-> case문에 break가 없으면, 다음 case문으로 계속 실행 (break를 만날때까지)

```
int time = 9;
switch (time){
case 8 :
    System.out.println("출근합니다.");
break;
case 9 :
    System.out.println("회의를 합니다.")
    break;
case 10 :
    System.out.println("업무를 봅니다.")
    break;
default :
    System.out.println("외근을 나갑니다.")
   break;
}
```



[실행결과]
회의를 합니다.
업무를 봅니다.
외근을 나갑니다.


PART
02
“
반복문
”

PART 02

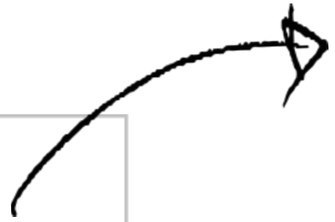
반복문

* 반복문이란?

-> 특정 명령문을 반복하는것

● 예시 "숫자 1 부터 10까지 더한 합을 구하라"

```
int n = 1;
num = num + 2;
num = num + 3;
num = num + 4;
num = num + 5;
num = num + 6;
num = num + 7;
num = num + 8;
num = num + 9;
System.out.println("회의를 합니다.")
```



합을 구하기 위해 코드 10줄을 작성함.
그런데 문제가 1~100까지 합을 구하라는 문제가 나온다면?
-> 코드를 100줄 작성해야함!!

이렇게 반복적인 일을 처리하기 위해 사용하는것이 '반복문'이다.

반복문

* 반복문 종류

1. while문
2. do-while문
3. for문

사용방법의 차이가 있다

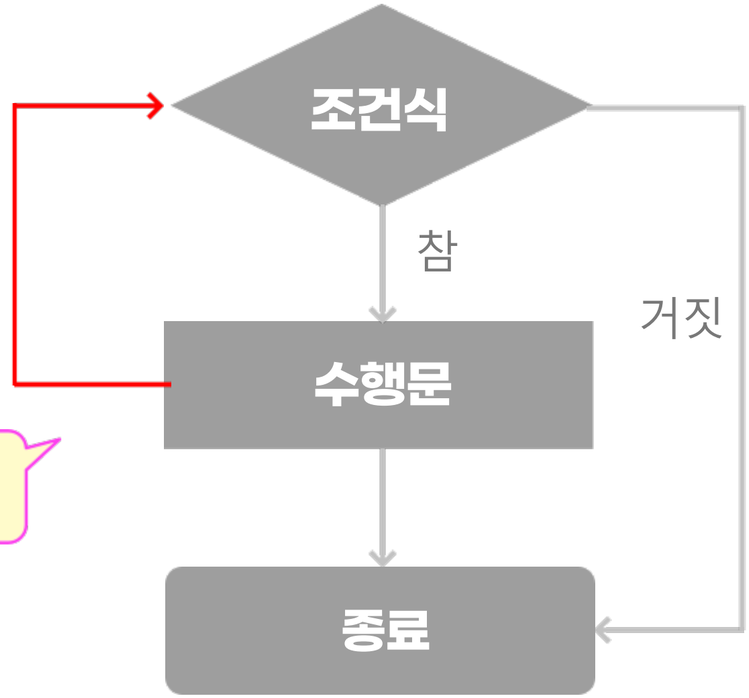
2. while문

-> 조건식이 참인 동안만 수행문을 반복한다.

조건의 참, 거짓에 따라 수행하는 경우에 사용한다

```
while (조건식){
수행문
}
```

→ 조건식이 참인동안 반복수행



되돌아간다!

PART 02

반복문

"숫자 1 부터 10까지 더한 합을 구하라" 문제를 반복문으로 변경한다

- 프로그램 작성시 필요한 요소

1. 조건식
-> 1부터 10까지 커지는 동안

2. 수행문
-> 1씩 더한다

3. 조건식과 수행문에서 사용할 변수
-> num * 조건식과 수행문에서 동시에 사용
-> sum

되돌아간다!

```
int num = 1;
int sum = 0;
 while(num <= 10 ){
    sum = sum + num;
    num++;
}
System.out.println("1부터 10까지의 합은 sum 입니다.")
```

-> 필요한 변수 선언

-> 조건식

-> 수행문

[실행결과]
1부터 10까지의 합은 55입니다.

3. do-while문

수행문을 반드시 한번 수행해야하는 경우에 사용한다

- 조건식이 거짓이더라도 무조건 한번 수행문을 실행한다

```
do {
    수행문
} while (조건식)
```

1

2

```
int num = 1;
int sum = 0;
 do{
   sum = sum + num;
   num++;
} while( num <= 10 )

System.out.println("1부터 10까지의 합은 sum 입니다.")
```

→ 조건이 맞지않더라도 블럭이 한번 실행된다.

[실행결과]
1부터 10까지의 합은 55입니다.

4. for문

- 필요한 요소가 많아서 복잡하다 (초기화식, 조건식, 증감식)
- 조건을 한눈에 볼수 있어서 제일 많이 사용한다

반복횟수가 정해진 경우에 사용한다


```
for ( 초기화식;  조건식;  증감식 ){
... 실행문 ...
}
```

초기화, 조건비교, 증감식을 한줄에 볼수 있어 가독성이 좋다.

반복문

for문의 수행순서를 보자!

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

[실행결과]

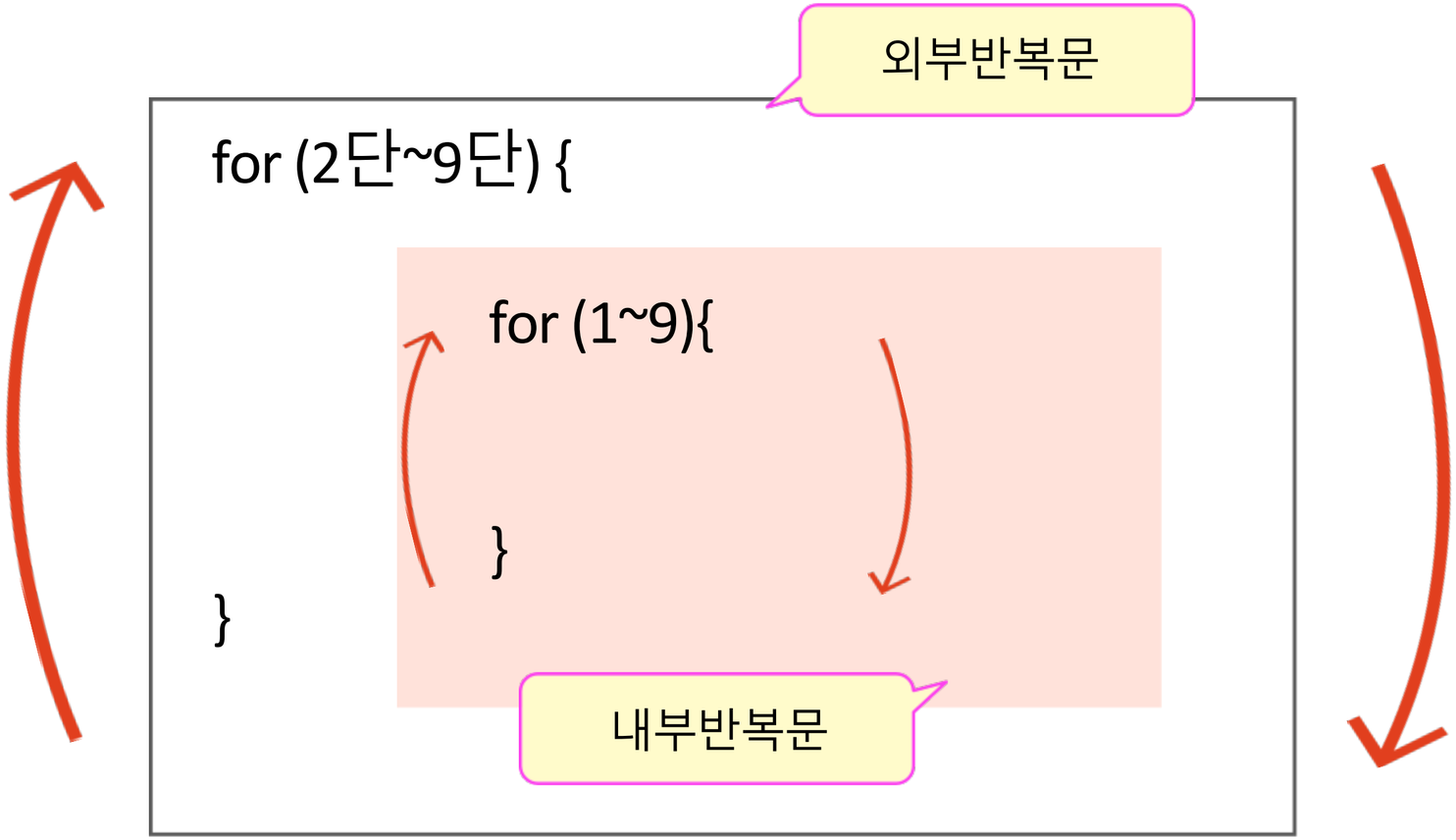
```
1
2
3
4
5
```

*과정 풀이
->교재 113페이지 보기

*for문 요소 생략하기
->교재 116페이지 보기

5. 중첩된 반복문

- 반복문 안에 또 다른 반복문을 중첩해서 사용하는 경우가 있다
- 내부반복문이 끝나면 다시 외부반복문으로 되돌아 간다



6. 반복문에서 continue의 역할

continue;

-> 반복문을 빠져 나가지 않고, 다음 반복으로 제어 변경

특정 조건에서는 수행하지 않고 건너뛰어야 할 때 사용한다.

```
for ( 초기문; 조건식; 증감식 ){
  . . .
  continue;
  . . .
}
```

이후의 문장은 수행하지 않고 for문으로 돌아간다.

7. 반복문에서 break의 역할

break;

-> 반복문을 아예 빠져나온다

```
for ( 초기문; 조건식; 증감식 ){
  . . .
  break;
  . . .
}
```

더이상 진행하지 않 for문 종료

반복문을 끝낸다

* 정리

IF (조건문) : if, if ~ else
SWITCH (선택문) : switch ~ case
LOOP (반복문) : for, while
CONTROL (제어문) : break, continue

Thank You!