

제어문과 메서드

예제 소스 코드는 파일과 연결되어 있습니다.
editplus(유료), notepad++(무료)와 같은 편집 도구를
미리 설치하여 PPT를 슬라이드 쇼로 진행할 때 소스
파일과 연결하여 보면 강의하실 때 편리합니다.

제어문

■ 필요성



제어문

- 제어문은 실행문의 수행 순서를 변경

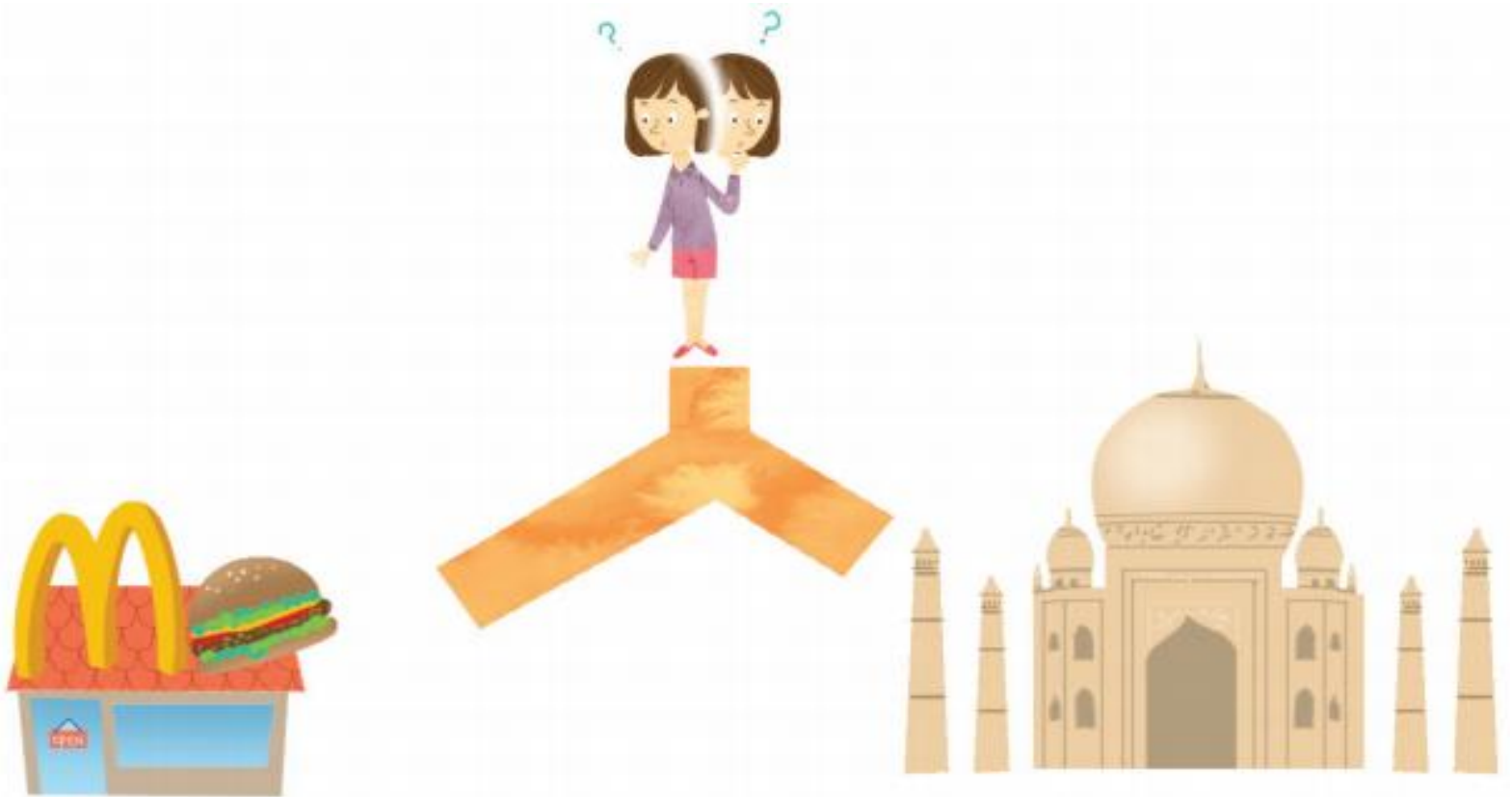


- 종류

- 조건문, 반복문, 분기문

조건문

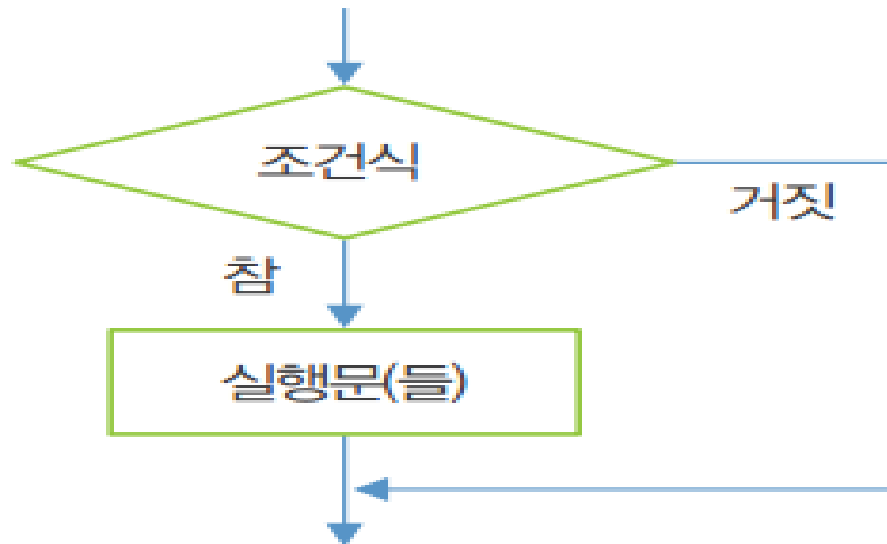
■ 조건에 따라 실행문을 선택을 할 때 사용



조건문

■ 단순 if 문

```
if (조건식) {  
    실행문(들);  
}
```



- 예제 : [sec02/SimpleIfDemo](#)

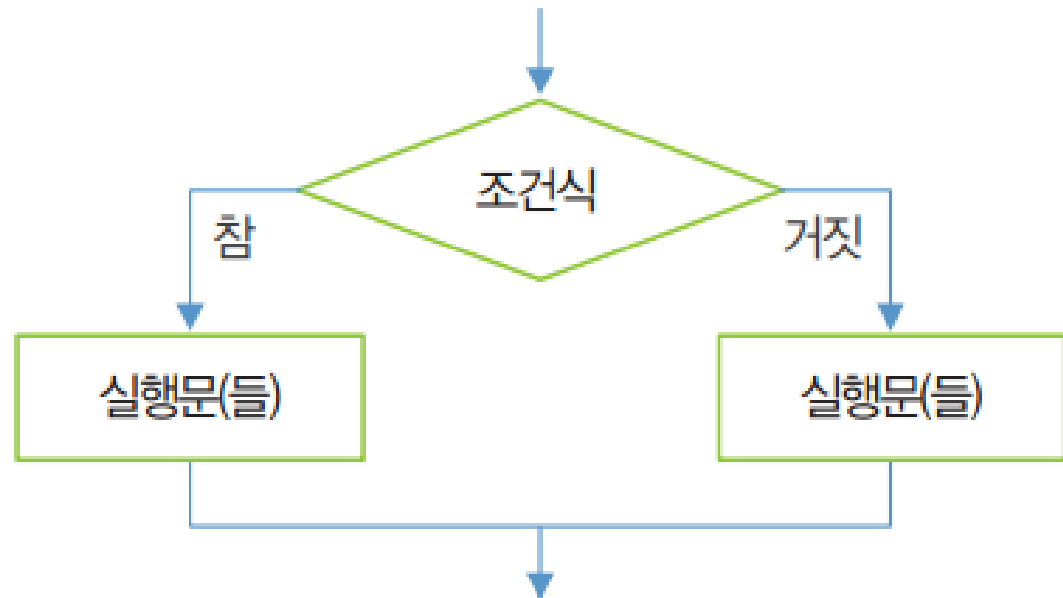
```
숫자를 입력하세요 : 2  
짝수!  
종료
```

```
숫자를 입력하세요 : 3  
홀수!  
종료
```

조건문

■ if~else 문

```
if (조건식) {  
    실행문(들);  
} else {  
    실행문(들);  
}
```



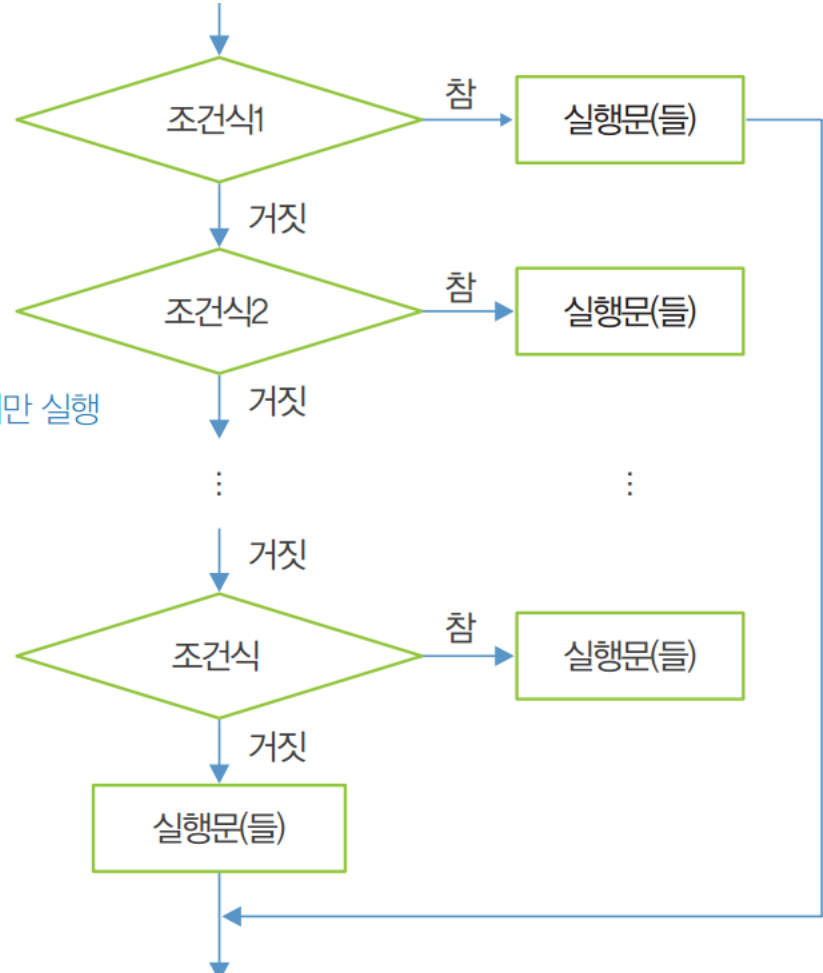
- 예제 : [sec02/IfElseDemo](#)

조건문

■ 다중 if 문

```
if (조건식1) {  
    실행문(들); 조건식1이 참일 때만 실행  
}  
else if (조건식2) {  
    실행문(들); 조건식1이 거짓이며 조건식2가 참일 때만 실행  
}  
else if (조건식3) {  
    ...  
}  
else {  
    실행문(들); 모든 조건을 만족하지 않을 때만 실행  
}
```

- 예제 : [sec02/MultifDemo](#)



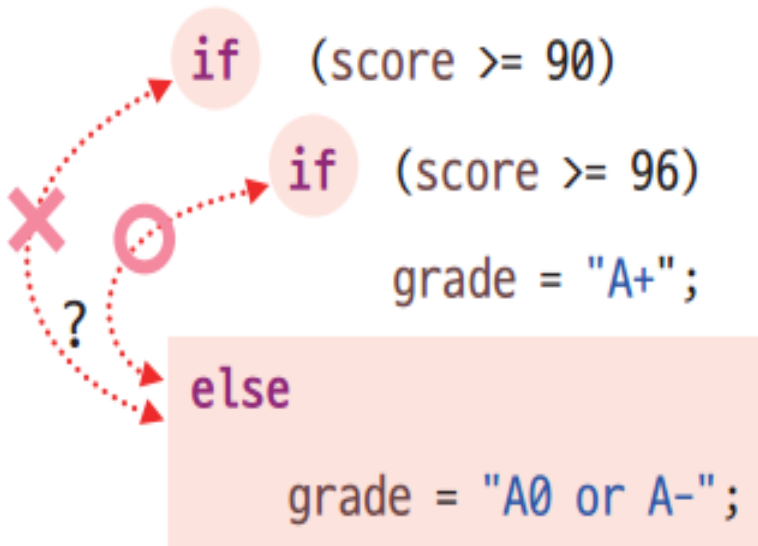
점수를 입력하세요 : 95
당신의 학점은 A

점수를 입력하세요 : 87
당신의 학점은 B

조건문

■ 중첩 if 문

- if 문에 다른 if 문이 포함되는 것을 중첩 if 문이라고 한다
- 주의 사항



```
if (score >= 90) {  
    if (score >= 96)  
        grade = "A+";  
    else  
        grade = "A0 or A-";  
}
```

- 예제 : [sec02/NestedIfDemo](#)

반복문

■ 조건에 따라 같은 처리를 반복



while 문, do~while 문

종이 한 쪽을 다 채울 때까지 반복해 쓰기
반복할 조건을 안다.



for 문

100번 반복해 쓰기
반복 횟수를 안다.

반복문

■ while 문

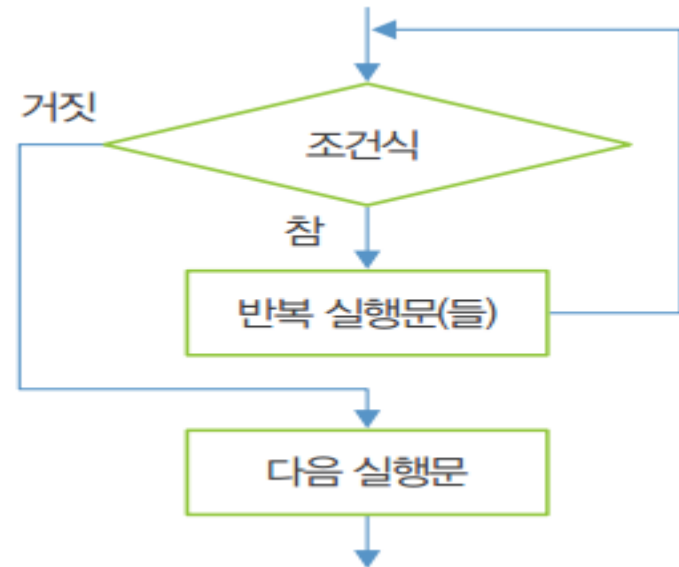
조건식이 거짓이면 본체를
한 번도 실행하지 않는다.

```
while (조건식) {  
    반복 실행문(들);  
}
```

본체를 탈출할
실행문이 필요하다.

```
while (true) {  
    반복 실행문(들);  
}
```

(a) 오류 미발생



```
while (false) {  
    반복 실행문(들);  
}
```

도달하지 않는 코드라는
오류를 발생시킨다.

(b) 오류 발생

반복문

■ while 문

- 예제 : [sec03/While1Demo](#)



1234

- 예제 : [sec03/While2Demo](#)

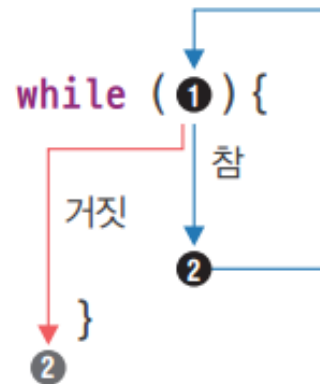
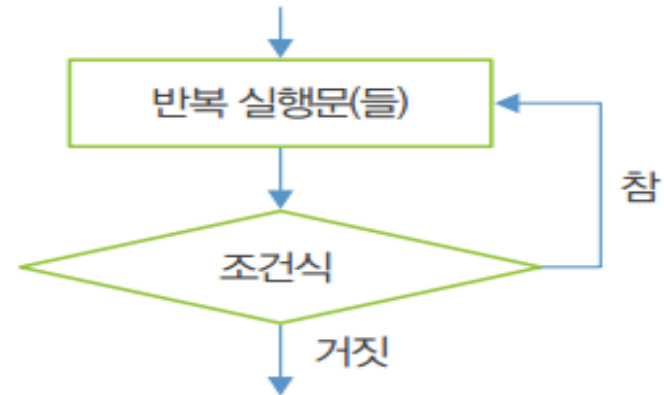
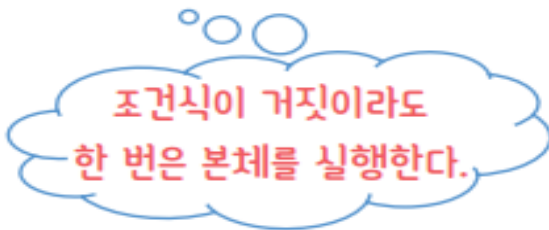


2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

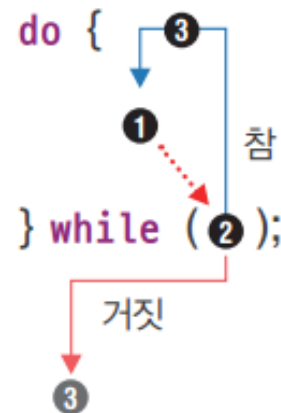
반복문

■ do~while 문

```
do {  
    반복 실행문(들); 본체  
} while (조건식);
```



(a) while 문



(b) do~while 문

반복문

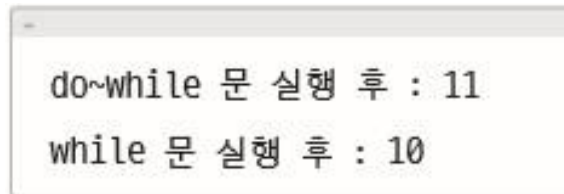
■ do~while 문

- 예제 : [sec03/DoWhile1Demo](#)



```
1234
```

- 예제 : [sec03/DoWhile2Demo](#)



```
do~while 문 실행 후 : 11  
while 문 실행 후 : 10
```

- 예제 : [sec03/DoWhile3Demo](#)



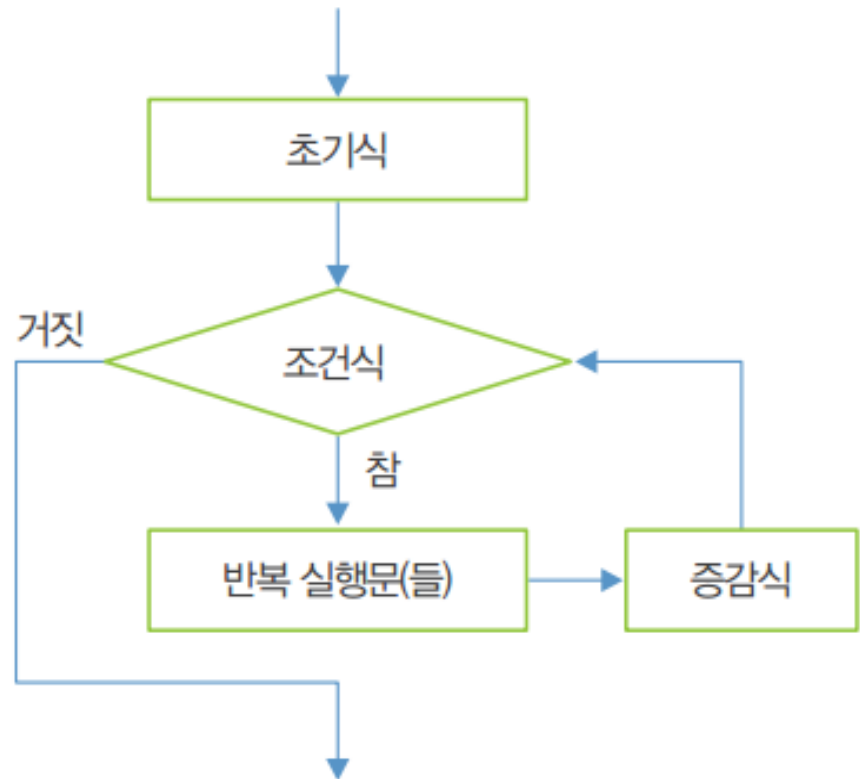
```
2  4  6  8  10 12 14 16 18  
3  6  9  12 15 18 21 24 27  
4  8  12 16 20 24 28 32 36  
5  10 15 20 25 30 35 40 45  
6  12 18 24 30 36 42 48 54  
7  14 21 28 35 42 49 56 63  
8  16 24 32 40 48 56 64 72  
9  18 27 36 45 54 63 72 81
```

반복문

■ for 문

조건식이 거짓이면 본체와 증감식은
한 번도 실행되지 않는다.

```
for (초기식; 조건식; 증감식) {  
    반복 실행문(들); 본체  
}
```



반복문

■ for 문

첫 번째 반복

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

두 번째 반복

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

세 번째 반복

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

현재는 i가 2므로 조건이
거짓이 되어 반복 종료

반복문

■ for 문

```
for ( ; ; )    // 무한 반복문  
;
```

```
          초기식      조건식      증감식  
for (int i = 0, j = 10; i < j; i++, j--) {  
    ... 본체  
}
```

- 예제 : [sec03/For1Demo](#)



1234

- 예제 : [sec03/For2Demo](#)

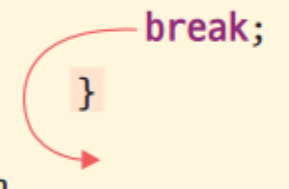


2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

분기문

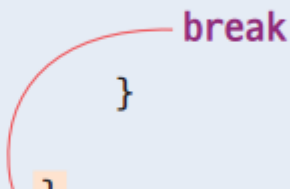
■ break 문

```
while () {  
    while () {  
        break;  
    }  
}
```



(a) break를 포함한 맨 안쪽 반복문 종료

```
out: while () {  
    while () {  
        break out;  
    }  
}
```



(b) 레이블이 표시된 반복문 종료

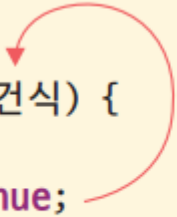
- 예제 : [sec04/BreakDemo](#)



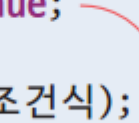
분기문

■ continue 문


```
while (조건식) {  
    continue;  
}
```



```
do {  
    continue;  
} while (조건식);
```



```
for (초기식; 조건식; 증감식) {  
    continue;  
}
```



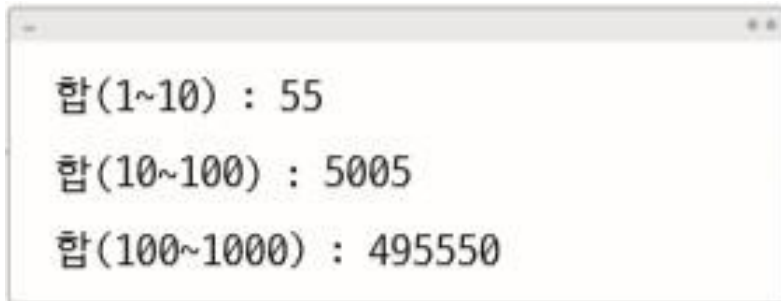
- 예제 : [sec04/ContinueDemo](#)



메서드

■ 필요성

- 메서드를 이용하지 않은 예제 : [sec06/Method1Demo](#)
- 메서드를 이용한 예제 : [sec06/Method2Demo](#)



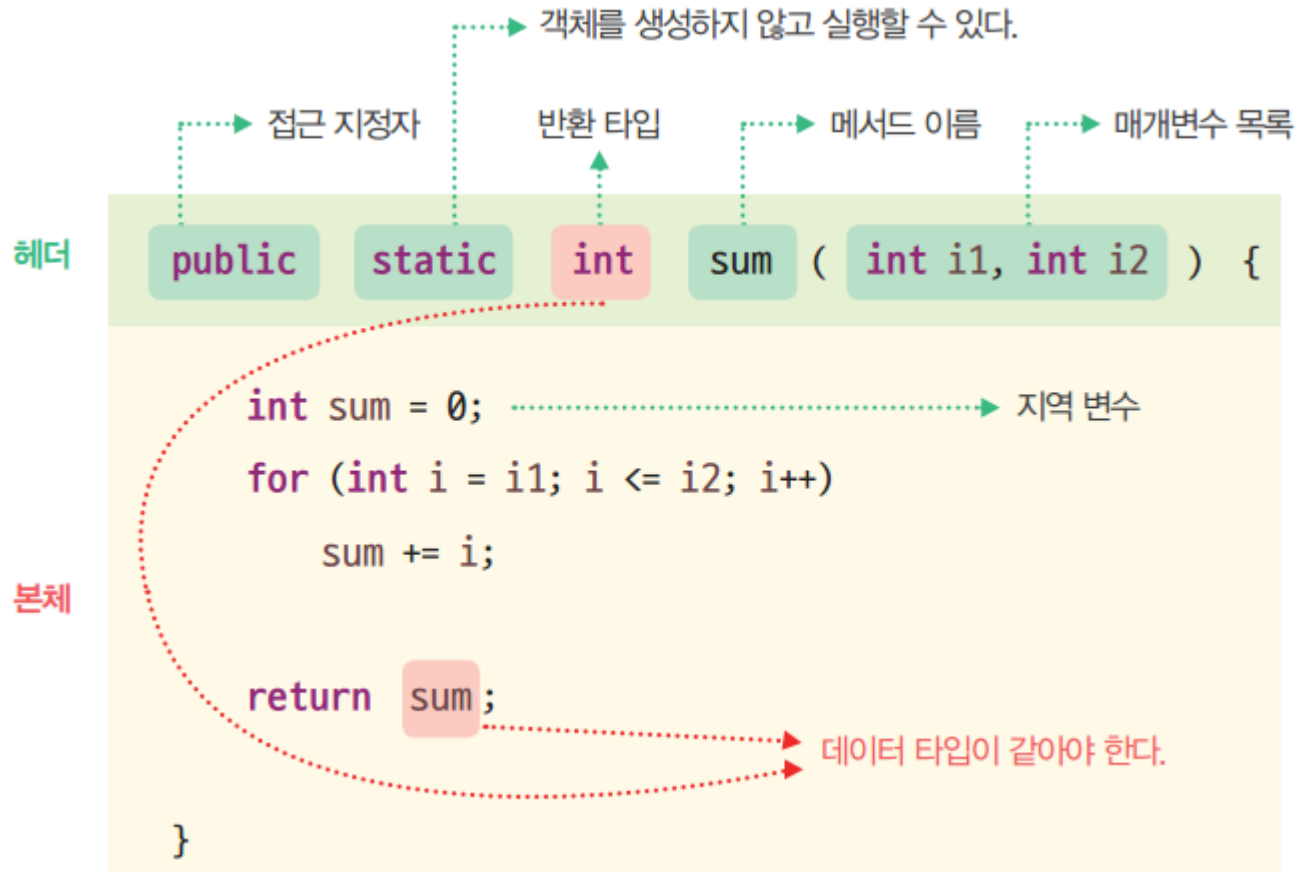
```
합(1~10) : 55  
합(10~100) : 5005  
합(100~1000) : 495550
```

■ 메서드를 이용하면 얻을 수 있는 장점

- 중복 코드를 줄이고 코드를 재사용할 수 있다.
- 코드를 모듈화해 가독성을 높이므로 프로그램의 품질을 향상시킨다.

메서드

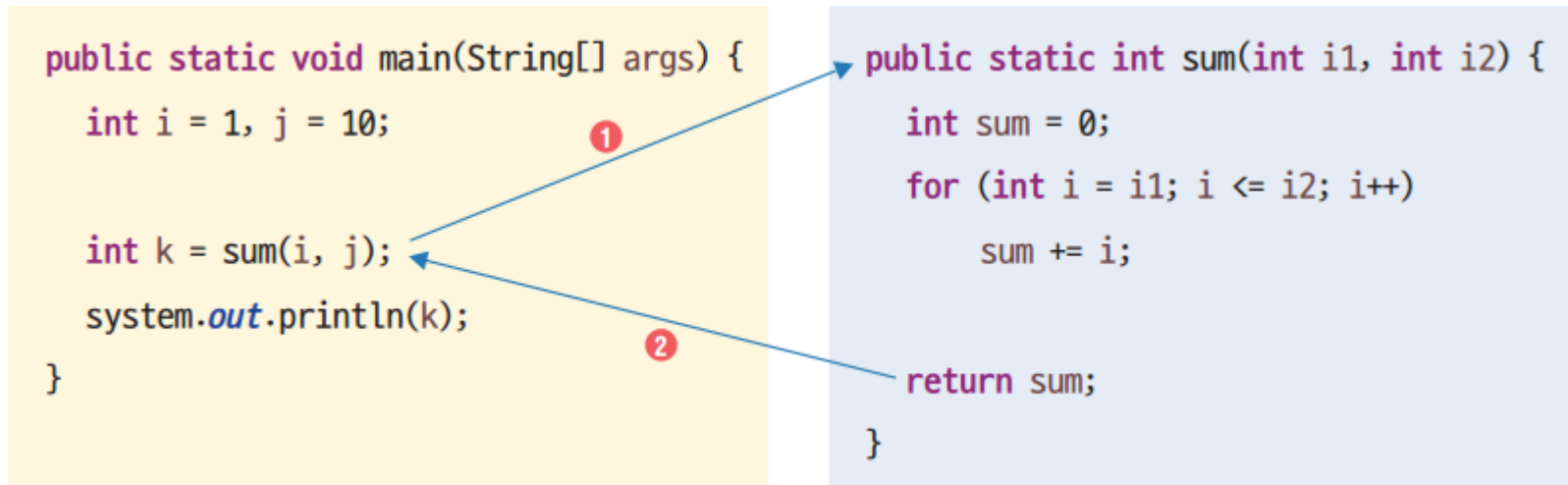
■ 메서드의 구조



메서드

■ 메서드의 호출과 반환

- 메서드를 호출하면 제어가 호출된 메서드(callee)로 넘어갔다가 호출된 메서드의 실행을 마친 후 호출한 메서드(caller)로 다시 돌아온다. 단, return 문을 사용하면 다음과 같이 메서드의 실행 도중에도 호출한 메서드로 제어를 넘길 수 있다.



- 예제 : [sec06/ReturnDemo](#)



메서드

■ 메서드의 매개변수

- 예제 : [sec06/EchoDemo](#)



메서드

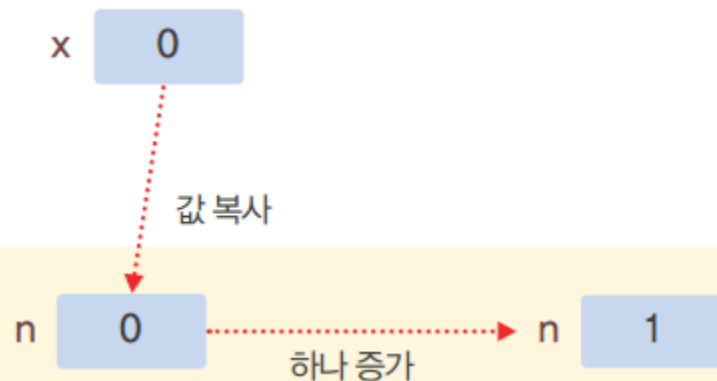
■ 값 전달(call by value)

- 예제 : [sec06/IncrementDemo](#)

```
increment() 메서드를 호출하기 전의 x는 0  
increment() 메서드를 시작할 때의 n은 0  
increment() 메서드가 끝날 때의 n은 1  
increment() 메서드를 호출한 후의 x는 0
```

```
int x = 0;  
increment(x);  
// x는 여전히 0
```

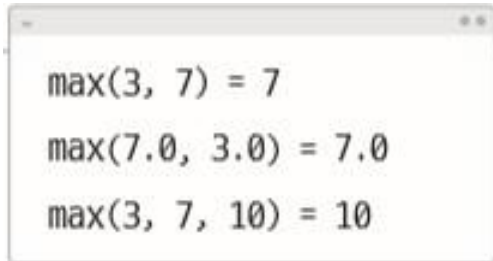
```
increment(int n)  
{  
    n++;  
}
```



메서드

■ 메서드 오버로딩

- 메서드 시그너처(Method Signature) : 메서드 이름, 매개변수의 개수, 매개변수의 타입과 순서를 의미
- 메서드 이름은 같지만 메서드 시그니처가 다른 메서드를 정의하는 것을 메서드 오버로딩(Method Overloading)이라고 한다.
- 예제 : [sec06/OverloadDemo](#)



```
max(3, 7) = 7  
max(7.0, 3.0) = 7.0  
max(3, 7, 10) = 10
```


Switch 문

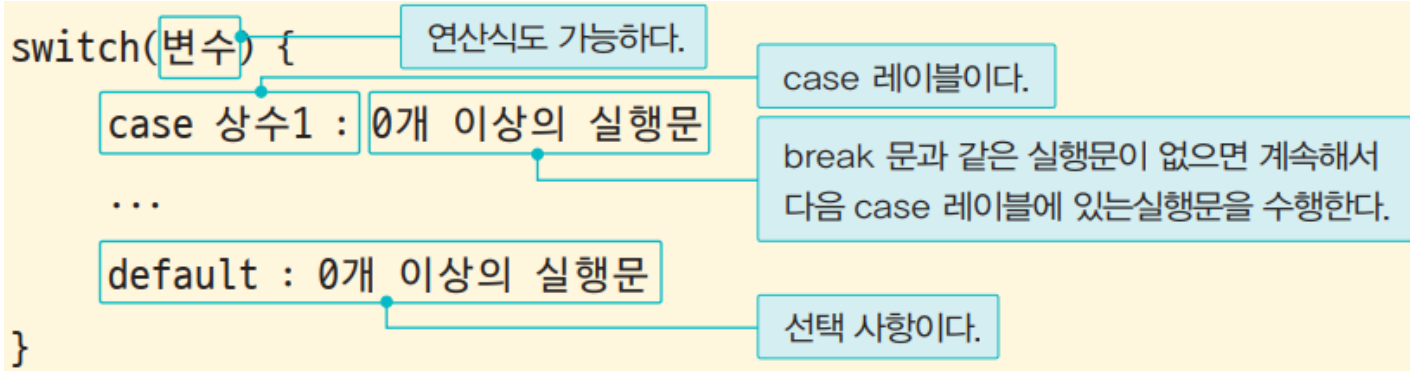
■ 기초



- switch 문은 if 문과 마찬가지로 조건문의 일종
- 여러 경로 중 하나를 선택할 때 사용
- 기존 switch 문은 낙하 방식으로 콜론 case 레이블 이용
- 자바 14부터는 비낙하 방식의 화살표 case 레이블 도입, switch 연산식 가능

Switch 문

■ 콜론 레이블을 사용하는 기존 switch 문



- 0개 이상의 case 절과 0이나 1개의 default 절로 구성
- Switch 변수로 정수 타입만 사용할 수 있었지만, 자바 7부터는 문자열과 열거 타입도 사용 가능
- 예제 : [sec05/Switch1Demo](#), [sec05/Switch2Demo](#)



호랑이는 포유류이다.
참새는 조류이다.
고등어는 어류이다.
어이쿠! 곰팡이는 ...이다.

Switch 문

■ 개선된 switch 문

- 필요성 : 깔끔하지 못하고 가독성도 떨어지며, break문의 누락으로 인한 오류 가능성도 크다
- 자바 14부터 다음과 같은 변화를 도입
 - 화살표 case 레이블(case상수->) : 콜론 case 레이블과 달리 일치하는 case레이블의 실행문만 수행하고 다음 case 레이블로 이동하지 않음. case 레이블의 실행문은 실행문 블록도 가능함.
 - switch 연산식 : 기존 switch 문과 달리 값을 반환할 수 있는 연산식으로 사용 가능.
 - 다중 case 레이블 : 콤마로 연결된 다수의 상수를 case 레이블로 사용 가능
 - Yield 예약어 : yield는 값을 반환하면서 switch 연산식을 종료하므로 yield 문 뒤에 break 문이 필요없음. case 혹은 default 레이블의 실행문이 블록일 경우에만 사용할 수 있음.

Switch 문

■ 개선된 switch 문

- 예제 : [sec05/Switch3Demo](#)(switch 문), [sec02/Switch4Demo](#)(switch 연산식)

호랑이는 포유류이다.

참새는 조류이다.

고등어는 어류이다.

어이쿠! 곰팡이는 ...이다.

Switch 문

■ 개선된 switch 문

- 자바 14부터는 기존 switch 문도 연산식, 다중 case 레이블, yield 예약어를 허용

```
String kind = switch (bio) {  
    case "호랑이", "사자":  
        yield "포유류";  
    case "독수리", "참새":  
        yield "조류";  
    case "고등어", "연어":  
        yield "어류";  
    default:  
        System.out.print("아이쿠! ");  
        yield "...";  
};
```

기존 switch 문에서는 블록이 아니더라도 yield 예약어를 사용할 수 있다.

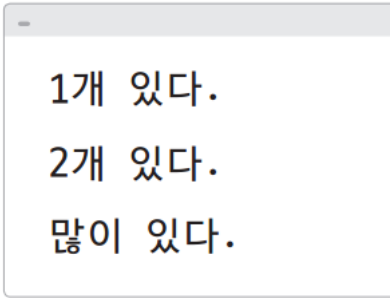
Switch 문

■ Switch 연산식의 주의 사항

- 가능한 모든 값에 대하여 일치하는 case 레이블이 없으면 오류가 발생
- 다음 코드에서 변수 n의 모든 가능한 값은 정수이므로 오류 발생

```
static String howMany(int n){  
    return switch(n){  
        case 1 -> "1개";  
        case 2 -> "2개";  
    }; // default 문은 선택 사항  
}
```

- 예제 : [sec05/Switch5Demo](#)



1개 있다.
2개 있다.
많이 있다.