객체지향 프로그래밍

Object-Oriented Programming

Chapter 02. 자바 프로그램 구조와 기초 문법

1. 자바 프로그램 기본 구조
2. 식별자
3. 변수
4. 타입 변환
5. 기본 입출력
6. 연산자

연산자

비교 연산자

- 값의 일치 및 대소 비교
- **boolean** 타입의 결괏값을 갖는다 (true, false)
- 대소 비교는 정수, 실수형 타입에만 사용 가능

연산자	사용 예	설명
==	x == y	x와 y는 같은가?
!=	x != y	x와 y가 다른가?
〉	x 〉y	x는 y보다 큰가?
〉=	x 〉= y	x는 y보다 크거나 같은가?
〈	x 〈y	x는 y보다 작은가?
〈=	x 〈= y	x는 y보다 작거나 같은가?

연산자

논리 연산자

- 두 개의 boolean 피연산자의 값을 결합해 boolean을 반환하는 연산자
- NOT, AND, OR, XOR

a	b	!a	a && b	a || b	a ^ b
false	false	true	false	false	false
false	true	true	false	true	true
true	false	false	false	true	true
true	true	false	true	true	false

연산자

논리 연산자 – 쇼트서킷(short circuit)

조건식1 && 조건식2	조건식 1이 false이면 조건식 2의 진릿값과 상관없이 결과가 무조건 false가 된다. 따라서 조건식 2의 진릿값을 조사할 필요가 없다.
조건식1 || 조건식2	조건식 1이 true이면 조건식 2의 진릿값과 상관없이 결과가 무조건 true가 된다. 따라서 조건식 2의 진릿값을 조사할 필요가 없다.

연산자

Practice

- **예제 2-9**: 비교·논리 연산자 응용 (p.65)
  - package: chap02
  - class: CompLogicDemo

연산자

비트 연산자

연산자	설명
&	두 비트가 모두 1일 때만 1이며, 나머지는 모두 0이다.
|	두 비트가 모두 0일 때만 0이며, 나머지는 모두 1이다.
^	두 비트가 서로 다를 때는 1, 동일할 때는 0이다.
~	1을 0으로, 0을 1로 바꾼다.

```
  0 1 0 1          0 1 0 1          0 1 0 1
& 0 0 1 1        | 0 0 1 1        ^ 0 0 1 1        ~ 0 0 1 1
---------        ---------        ---------        ---------
  0 0 0 1          0 1 1 1          0 1 1 0          1 1 0 0
```

연산자

시프트 연산자

연산자	a 연산자 b일 경우 설명(예를 들어, a 《 b)
《	a의 모든 비트를 왼쪽으로 b비트만큼 이동하며, 이동할 때마다 최하위 비트를 0으로 채운다. 곱셈 효과가 나타나기 때문에 산술적 왼쪽 시프트(Arithmetic Left Shift)라고 한다.
》	a의 모든 비트를 오른쪽으로 b비트만큼 이동하며, 이동할 때마다 최상위 비트와 동일한 비트로 채운다. 나눗셈 효과가 나타나기 때문에 산술적 오른쪽 시프트(Arithmetic Right Shift)라고 한다.
》》	a의 모든 비트를 오른쪽으로 b비트만큼 이동하며, 이동할 때마다 최상위 비트를 0으로 채운다. 산술적 효과가 없기 때문에 논리적 오른쪽 시프트(Logical Right Shift)라고 한다.

0b00000110 》2

오른쪽으로 2비트씩 이동
왼쪽 빈 2비트 공간을 00으로 채움

0b00000110 《2

왼쪽으로 2비트씩 이동
오른쪽 빈 2비트 공간을 00으로 채움

연산자

Practice

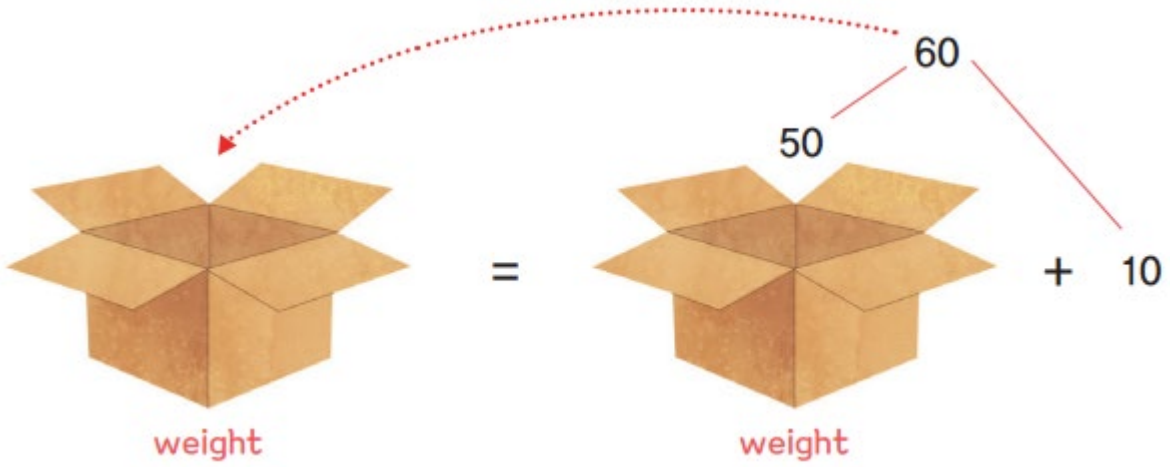
- **예제 2-10**: 비트·시프트 연산자 응용 (p.67)
  - package: chap02
  - class: BitOperatorDemo

연산자

대입 연산자

- 오른쪽에 있는 연산식의 결과 값을 왼쪽에 있는 변수에 대입하는 연산자

```
int weight = 50;
weight = weight + 10;
```



연산자

Practice

- **예제 2-11**: 대입 연산자 응용 (p.69)
  - package: chap02
  - class: AssignmentDemo

연산자

부호·증감 연산자

- 부호 연산자: 피연산자의 부호를 반전
- 증감 연산자: 변수에 1을 더하거나 뺌

연산자	설명
+	부호 유지
−	부호 반전

연산자	설명	
++	++x	연산 전 x 값 증가(전위 증가)
	x++	연산 후 x 값 증가(후위 증가)
−−	−−x	연산 전 x 값 감소(전위 감소)
	x−−	연산 후 x 값 감소(후위 감소)

연산자

Practice

- **예제 2-11**: 대입 연산자 응용 (p.69)
  - package: chap02
  - class: SignIncrementDemo

연산자

조건 연산자(삼항 연산자)

- 하나의 조건식과 두 연산식으로 구성
- 조건식이 true이면 결과 값은 연산식1의 값을 반환
- 조건식이 false이면 결과 값은 연산식2의 값을 반환

```
조건식 ? 연산식1 : 연산식2
```

- 쇼트서킷 로직을 이용하기 때문에 조건식에 따라 연산식1과 연산식2 중 하나만 실행된다

연산자

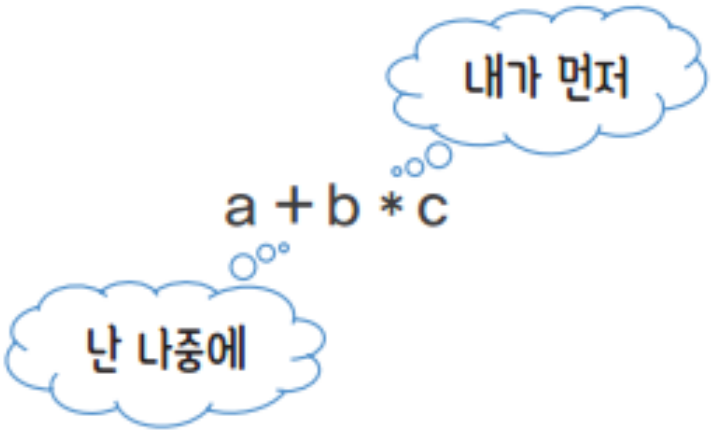
Practice

- **예제 2-11**: 삼항 연산 예제 (p.71)
  - package: chap02
  - class: TernaryOperatorDemo

연산자

연산자 우선순위

모호하게 해석가능한 수식에서 어느 연산을 먼저 계산할 것인가를 결정하는 규칙

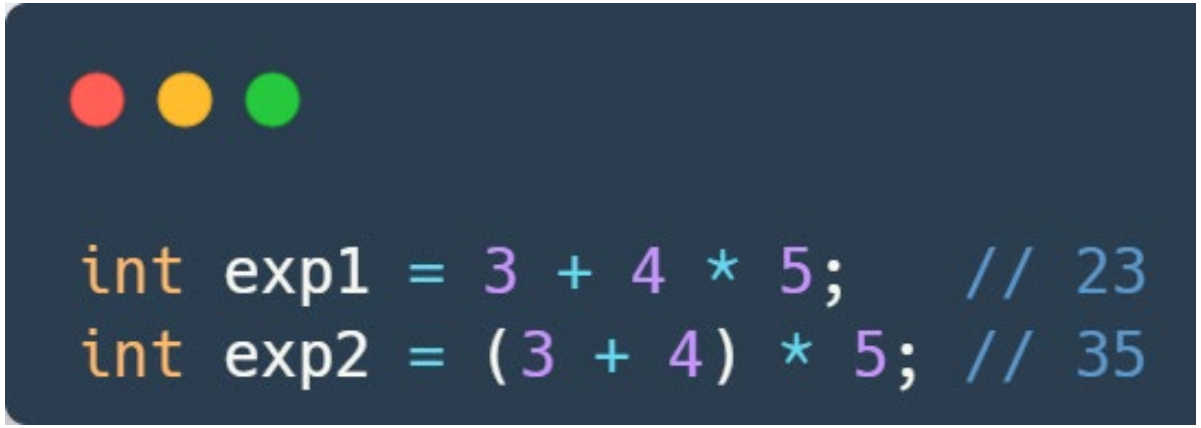


연산자

연산자 우선순위

모호하게 해석가능한 수식에서 어느 연산을 먼저 계산할 것인가를 결정하는 규칙

괄호를 통한 연산 순서 설정



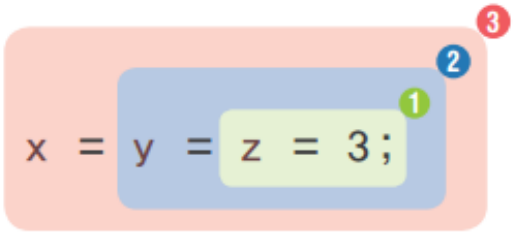
연산자

연산자 우선순위

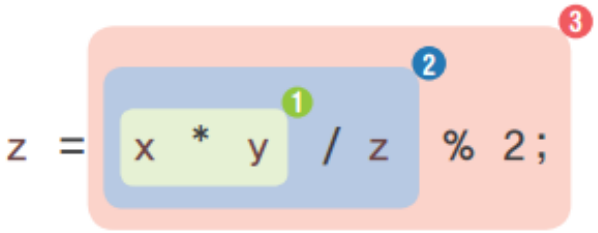
연산자	설명
[ ], ., ( ), ++, −−	배열 접근, 객체 접근, 메서드 호출, 후위 증가, 후위 감소
+x, −x, ++x, −−x, ~(비트), !(논리)	부호 +/−, 선위 증가, 선위 감소, 비트 부정, 논리 부정
( ), new	타입 변환, 객체 생성
*, /, %	곱셈, 나눗셈, 모듈로
+, −	덧셈, 뺄셈
〉〉, 〈〈, 〈〈〈	시프트
〉, 〈, 〉=, 〈=, instanceof	비교
==, !=	동등 여부
&	비트 AND
^	비트 XOR
|	비트 OR
&&	조건 AND
||	조건 OR
? :	조건 연산
=, +=, −=, *=, /=, %=, &=, ^=, !=, 〈〈=, 〉〉=, 〉〉〉=	대입

연산자

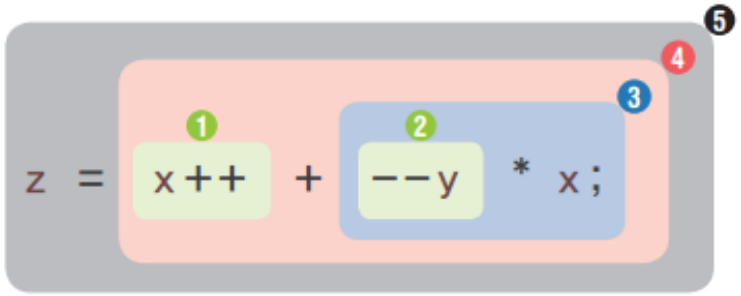
결합 규칙



3을 z, y, x 순(오른쪽에서 왼쪽 순)으로 대입한다.



*, /, % 연산자는 우선순위가 모두 같으므로
왼쪽에서 오른쪽으로 순서대로 연산한다.
3*3/3%2를 연산하면 z에 1을 대입한다.



연산자의 우선순위에 따라 연산하면 ❶은 3,
❷는 2, ❸은 2*4이므로 8, ❹는 3+8이므로 11이다.
따라서 z에 11을 대입한다.

연산자

Challenge

- **도전과제 2**: 직사각형 넓이 계산 (p. 76)
  - package: chap02
  - class: Challenge02

정수를 입력 받아 짝수인지 홀수인지 판별하는 프로그램 작성

Hint

- 나머지 연산자
- 삼항 연산자 `조건식 ? 연산식1 : 연산식2`

Chapter 03. 제어문과 메서드

1. 제어문
    1. 조건문
    2. 반복문
    3. 분기문
2. 메서드

제어문

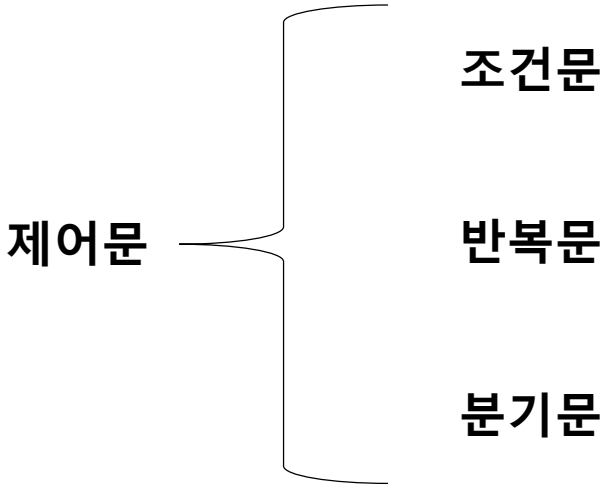
제어문의 필요성

- 일반적으로 프로그램에 포함된 실행문은 순차적으로 수행된다
- 하지만, 순차적으로만 실행될 경우:
  - 프로그램이 매우 길어짐
  - 선택적 실행이 불가

제어문

제어문

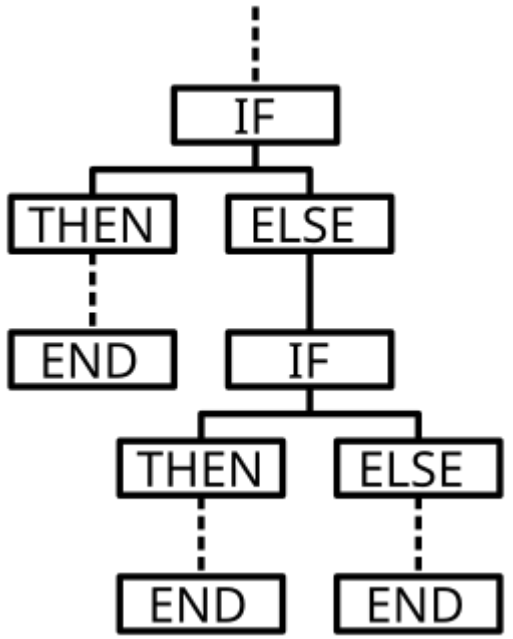
- **실행문의 수행 순서를 변경하는 구문**
- 제어문을 사용해 실행문을 비순차적으로 수행할 수 있다



조건문

조건문

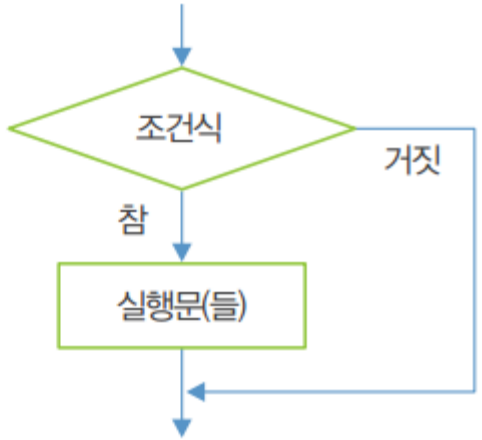
- 조건에 따라 여러 실행 경로 중 하나를 선택하는 제어문



조건문

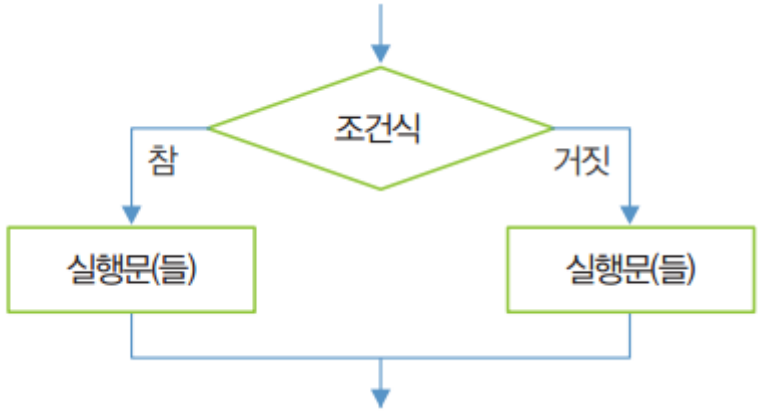
if 문

```
if (조건식) {
    실행문(들);
}
```



if ~ else 문

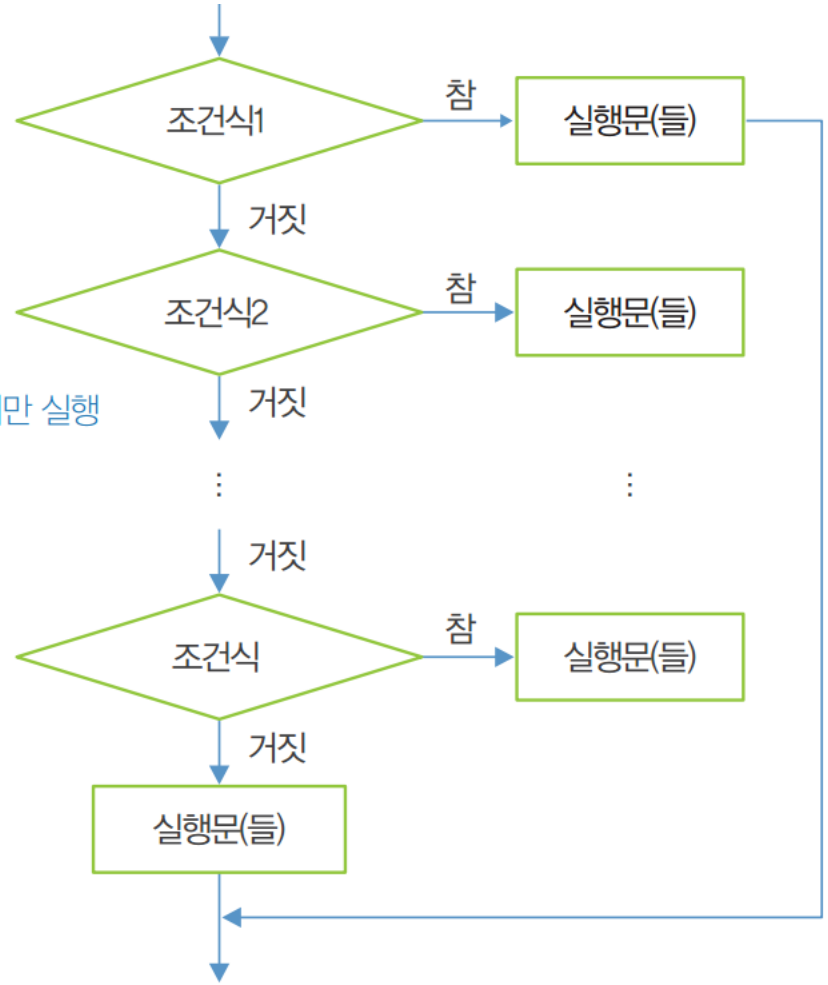
```
if (조건식) {
    실행문(들);
} else {
    실행문(들);
}
```



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
}
```



조건문

중첩 if 문

- if 문에 다른 if 문이 포함되는 경우

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

조건문

Practice

- **예제 3-1**: if문을 이용한 홀짝 조사 (p.86)
  - package: chap03
  - class: SimpleIfDemo

- **예제 3-4**: 중첩 if문 (p.89)
  - package: chap03
  - class: NestedIfDemo

반복문

반복문

- **조건에 따라 같은 처리를 반복하는 제어문**
- while, do-while, for 문

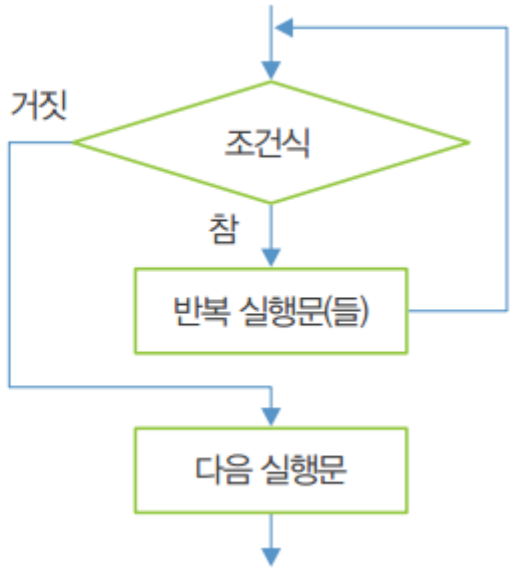
반복문

while 문

- 조건식이 true인 동안 반목 실행문을 계속해서 실행

조건식이 거짓이면 본체를
한 번도 실행하지 않는다.

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



반복문

Practice

- **예제 3-5**: while문 이용 연속 숫자 출력 (p.92)
  - package: chap03
  - class: WhileDemo

- **예제 3-6**: while문 이용 구구단 출력 (p.93)
  - package: chap03
  - class: While2Demo

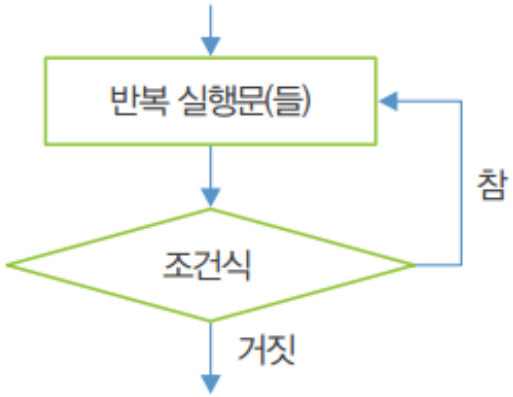
반복문

do ~ while 문

- 반복 실행문을 실행하고, 조건식이 거짓이라면 반복을 종료

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

조건식이 거짓이라도
한 번은 본체를 실행한다.



반복문

Practice

- **예제 3-8**: while과 do-while 비교(p.95)
  - package: chap03
  - class: DoWhile2Demo

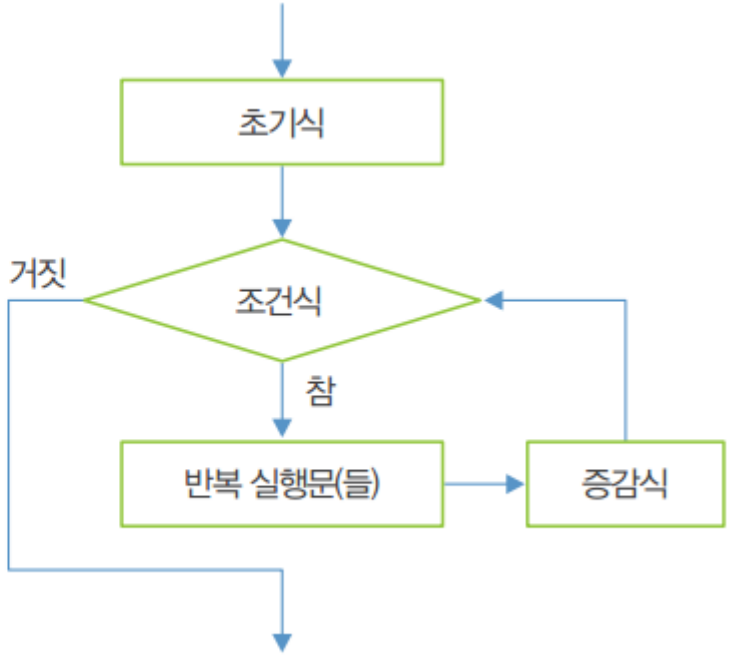
반복문

for 문

- 초기식과 증감식이 도입된 반복문

조건식이 거짓이면 본체와 증감식은
한 번도 실행되지 않는다.

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



반복문

Practice

- **예제 3-11**: for문 이용한 구구단 출력 (p.98)
  - package: chap03
  - class: For2Demo

분기문

분기문

- **반복문의 실험 흐름을 변경하는 제어문**
- break
- continue

분기문

break 문

- **반복문을 종료시키는 제어문**
- break 문이 실행되면, 해당 break문과 가장 인접한 반복문이 종료됨
- 레이블을 표시해 특정 반복문을 지정해서 종료할 수 있다

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

(a) break를 포함한 맨 안쪽 반복문 종료

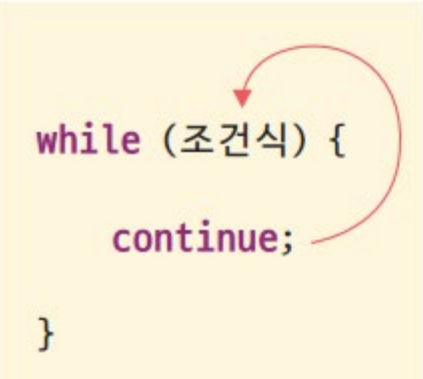
```
out: while () {
    while () {
        break out;
    }
}
```

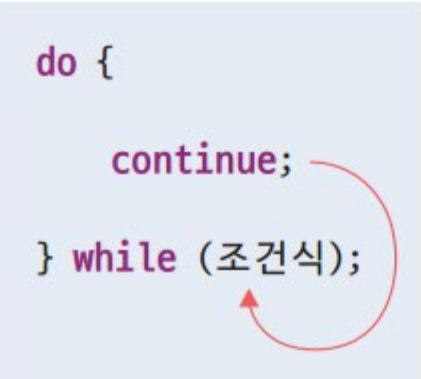
(b) 레이블이 표시된 반복문 종료

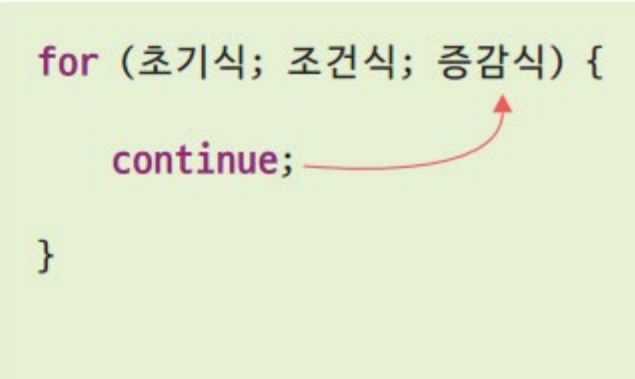
분기문

continue 문

- 현재 반복을 건너뛰도록 하는 제어문







분기문

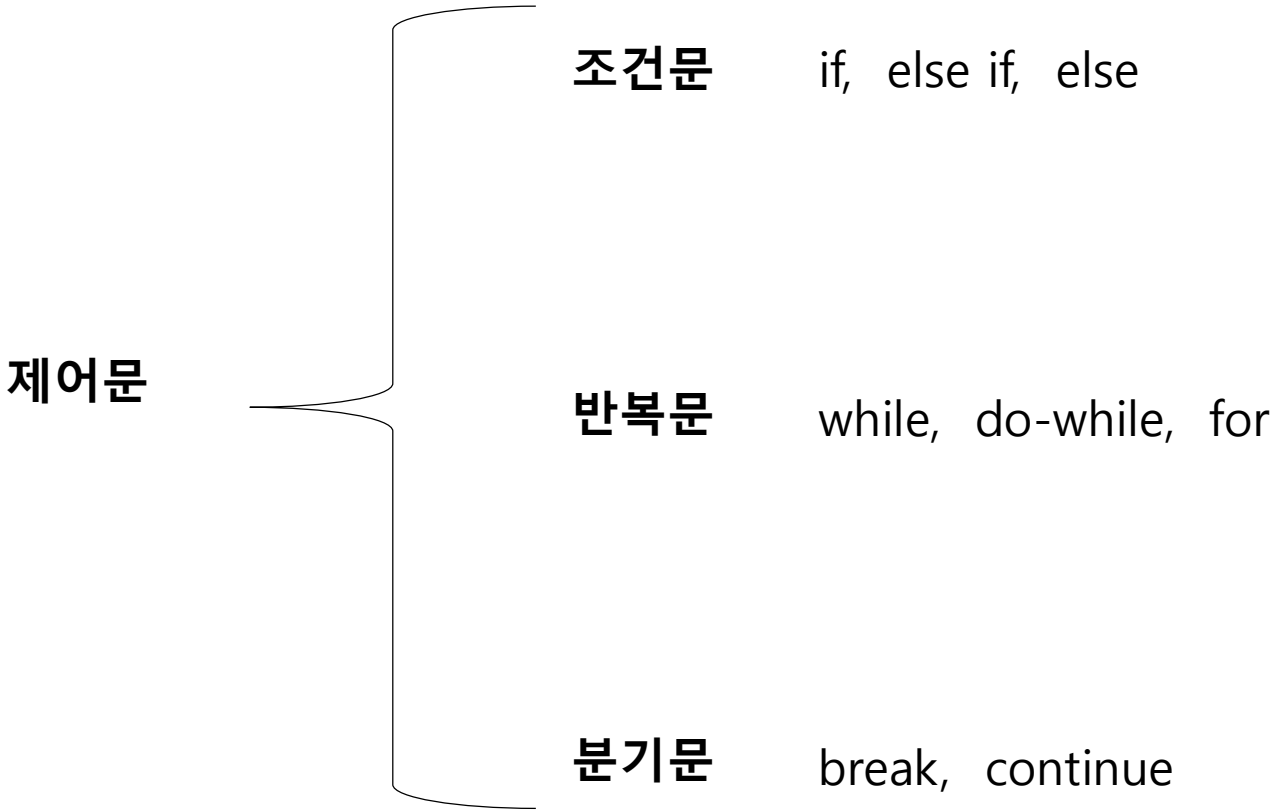
Practice

- **예제 3-12**: break문 예제 (p.98)
  - package: chap03
  - class: BreakDemo

- **예제 3-13**: continue문 예제 (p.100)
  - package: chap03
  - class: ContinueDemo

Recap

제어문 - Recap



제어문

Challenge

- **도전과제 1**: Factorial (p. 117)
  - **Goal**: 키보드로 입력한 정수의 팩토리얼 값을 구하는 프로그램을 작성
  - **package**: chap03, **class**: Challenge01

제어문

Practice

정수 자릿수 나열하기

- 키보드로부터 정수를 하나 입력 받는다.
- 입력 받은 정수의 모든 자릿수를 공백(space)으로 구분해 출력한다.

예제 입력
20001234

예제 출력
2 0 0 0 1 2 3 4

제어문

Practice

분해합

- 키보드로부터 자연수 S를 입력 받는다. → *int s*
- 100,000 이하의 자연수 중 각 자릿수의 합이 S인 자연수를 모두 출력한다. (줄바꿈을 통해 구분)

예제 입력
44

예제 출력
89999
98999
99899
99989
99998