

프로그래밍 언어 활용 강의안

(연산자와 제어문)

1. 주석과 실행문

주석문이란?

프로그램 실행과는 상관없이 코드에 설명 붙인 것

컴파일 과정에서 주석은 무시되고 실행문만 바이트 코드로 번역

주석 기호	설명
//	//부터 라인 끝까지 주석으로 처리한다. (행 주석)
/* ~ */	/*와 */ 사이에 있는 모든 범위를 주석으로 처리한다. (범위 주석)

실행문이란?

변수 선언, 값 저장, 메소드 호출에 해당하는 코드

실행문 끝에는 반드시 세미콜론(;)을 붙여 실행문의 끝 표시

```
int x = 1;           //변수 x를 선언하고 1을 저장
int y = 2;           //변수 y를 선언하고 2를 저장
int result = x + y;   //변수 result를 선언하고 변수 x와 y를 더한 값을 저장
System.out.println(result); //콘솔에 출력하는 메소드 호출
```

2. 단항 연산자

단항연산자란?

피연산자가 1개인 연산자

부호 연산자: +, -

boolean 타입과 char 타입을 제외한 기본 타입에 사용 가능

부호 연산자의 산출 타입은 int

증감 연산자: ++, --

변수의 값을 1증가 시키거나 (++) 1 감소 (--) 시키는 연산자

증감 연산자가 변수 뒤에 있으면 다른 연산자 먼저 처리 후 증감 연산자 처리

논리 부정 연산자: !

Boolean type 에만 사용가능

연산식		설명
!	피연산자	피연산자가 true 이면 false 값을 산출 피연산자가 false 이면 true 값을 산출

2. 산술 연산자

산술 연산자

연산식			설명
피연산자	+	피연산자	덧셈 연산
피연산자	-	피연산자	뺄셈 연산
피연산자	*	피연산자	곱셈 연산
피연산자	/	피연산자	좌측 피연산자를 우측 피연산자로 나눴셈 연산
피연산자	%	피연산자	좌측 피연산자를 우측 피연산자로 나눈 나머지를 구하는 연산

`int result = num % 3;`

0, 1, 2 중의 한 값

num 을 3 으로 나눈 나머지

2. 문자열 산술 연산자

문자열 연산자

- 피연산자중 문자열이 있으면 문자열로 결합

```
String str1 = "JDK" + 6.0;
```

```
String str2 = str1 + " 특징";
```

```
System.out.println(str2);
```



```
String str3 = "JDK" + 3 + 3.0;
```

```
String str4 = 3 + 3.0 + "JDK";
```

```
System.out.println(str3);
```

```
System.out.println(str4);
```

【실행 결과】

 Console 

<terminated> Stri

JDK6.0 특징

JDK33.0

6.0JDK

2. 비교 연산자

동등 비교연산자는 모든 타입에 사용 가능

크기 비교 연산자는 boolean 타입 제외한 모든 기본타입에 사용 가능

제어문인 조건문(if), 반복문(for, while)에서 주로 이용

구분	연산식			설명
동등 비교	피연산자	==	피연산자	두 피 연산자의 값이 같은지를 검사
	피연산자	!=	피연산자	두 피 연산자의 값이 다른지를 검사
크기 비교	피연산자	>	피연산자	피 연산자 1 이 큰지를 검사
	피연산자	>=	피연산자	피 연산자 1 이 크거나 같은지를 검사
	피연산자	<	피연산자	피 연산자 1 이 작은지를 검사
	피연산자	<=	피연산자	피 연산자 1 이 작거나 같은지를 검사

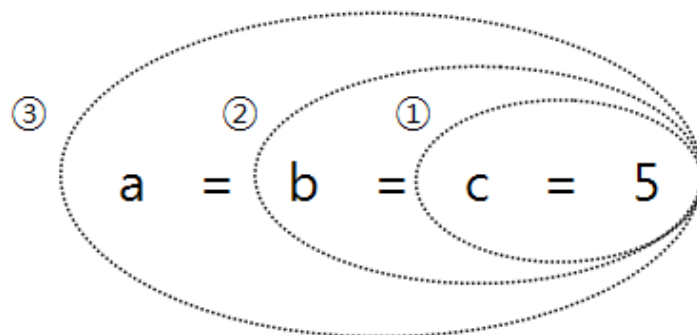
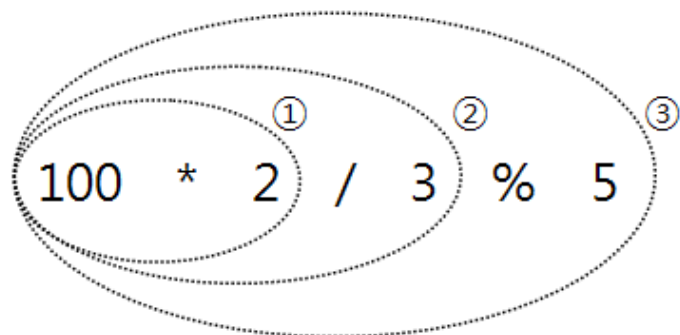
2. 논리 연산자

피연산자는 boolean 타입만 사용가능

구분	연산식			결과	설명
AND (논리곱)	true	&& 또는 &	true	true	피 연산자 모두가 true 일 경우에만 연산 결과는 true
	true		false	false	
	false		true	false	
	false		false	false	
OR (논리합)	true	 또는 	true	true	피 연산자 중 하나만 true 이면 연산 결과는 true
	true		false	true	
	false		true	true	
	false		false	false	
XOR (배타적 논리합)	true	^	true	false	피 연산자가 하나는 true 이고 다른 하나가 false 일 경우에만 연산 결과는 true
	true		false	true	
	false		true	true	
	false		false	false	
NOT (논리부정)		!	true	false	피 연산자의 논리값을 바꿈
			false	true	

3. 연산의 방향과 우선순위

$*$, $/$, $%$ 는 같은 우선 순위를 갖고 있다. 이들 연산자는 연산 방향이 왼쪽에서 오른쪽으로 수행된다. $100 * 2$ 가 제일 먼저 연산되어 200이 산출되고, 그 다음 $200 / 3$ 이 연산되어 66이 산출된다. 그 다음으로 $66 \% 5$ 가 연산되어 1이 나온다.



하지만 단항 연산자($++$, $--$, \sim , $!$), 부호 연산자($+$, $-$), 대입 연산자($=$, $+=$, $-=$, ...)는 오른쪽에서 왼쪽(\leftarrow)으로 연산된다. 예를 들어 다음 연산식을 보자.

3 연산자의 우선순위 참고용

연산자	연산 방향	우선 순위
증감(++ , --), 부호(+, -), 비트(~), 논리(!)	←	<div>높음</div> <div>↑</div> <div>↓</div> <div>낮음</div>
산술(*, /, %)	→	
산술(+, -)	→	
쉬프트(<<, >>, >>>)	→	
비교(<, >, <=, >=, instanceof)	→	
비교(==, !=)	→	
논리(&)	→	
논리(^)	→	
논리()	→	
논리(&&)	→	
논리()	→	
조건(?:)	→	
대입(=, +=, -=, *=, /=, %=, &=, ^=, =, <<=, >>=, >>>=)	←	

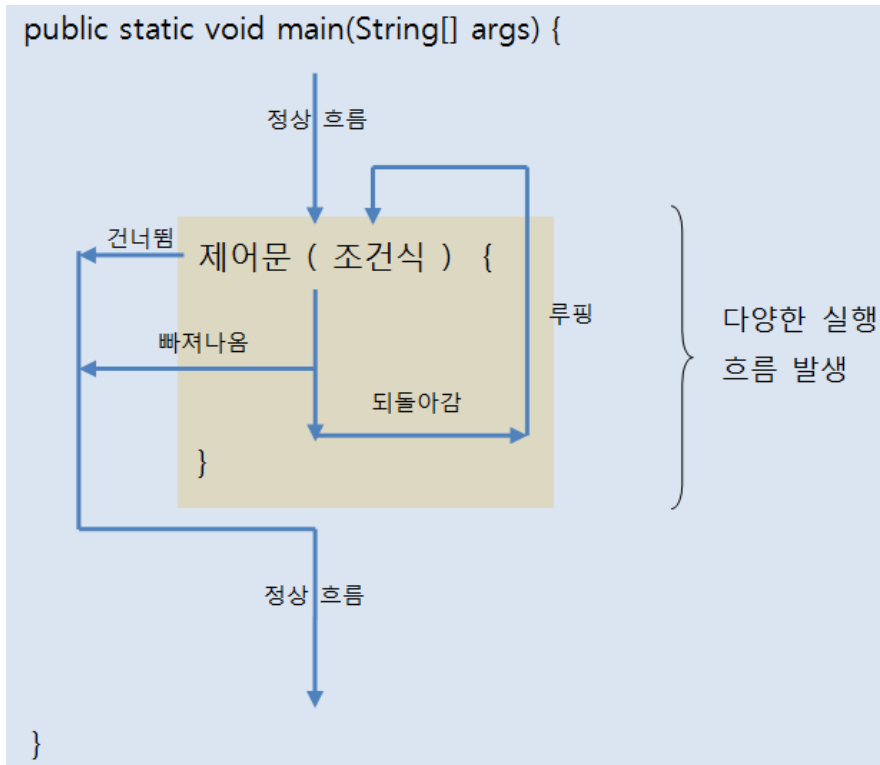
4. 제어문

정상적인 코드 실행 흐름

main() 메소드의 시작인 중괄호 { 에서 끝 중괄호 } 까지 위->아래 방향으로 실행

제어문의 역할

코드 실행 흐름을 개발자가 원하는 방향으로 변경할 수 있도록 도와줌



4. 제어문

제어문의 종류

조건문

if문, switch문

반복문

for문, while문, do-while문

break문, continue문

제어문의 중첩

제어문의 중괄호 내부에 다른 제어문 작성 가능

다양한 흐름 발생 가능

5. 조건문(if문,switch문)

if 문

조건식 결과 따라 중괄호 { } 블록을 실행할지 여부 결정할 때 사용

조건식

true 또는 false값을 산출할 수 있는 연산식

boolean 변수

조건식이 true이면 블록 실행하고 false 이면 블록 실행하지 않음

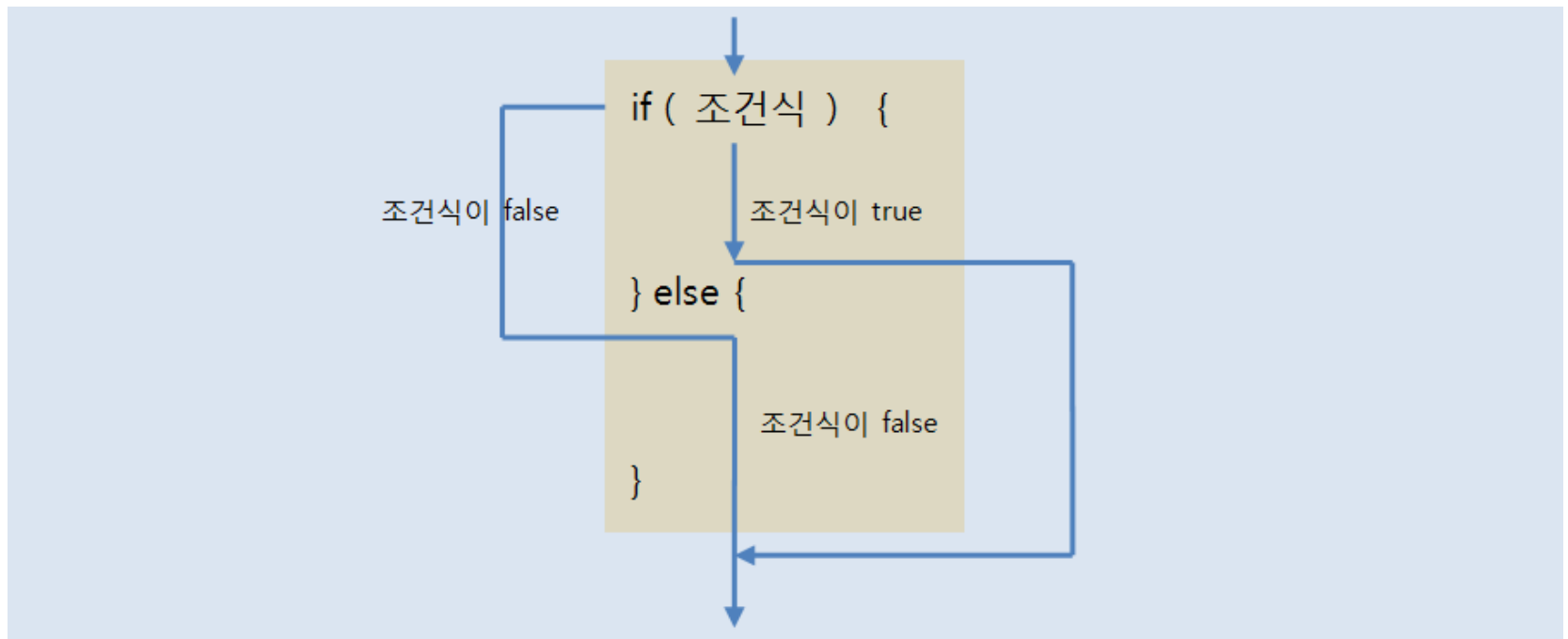
```
if ( 조건식 ) {  
    실행문;  
    실행문;  
    ...  
}
```

```
if ( 조건식 )  
    실행문;
```

5. 조건문(if문,switch문)

if-else 문

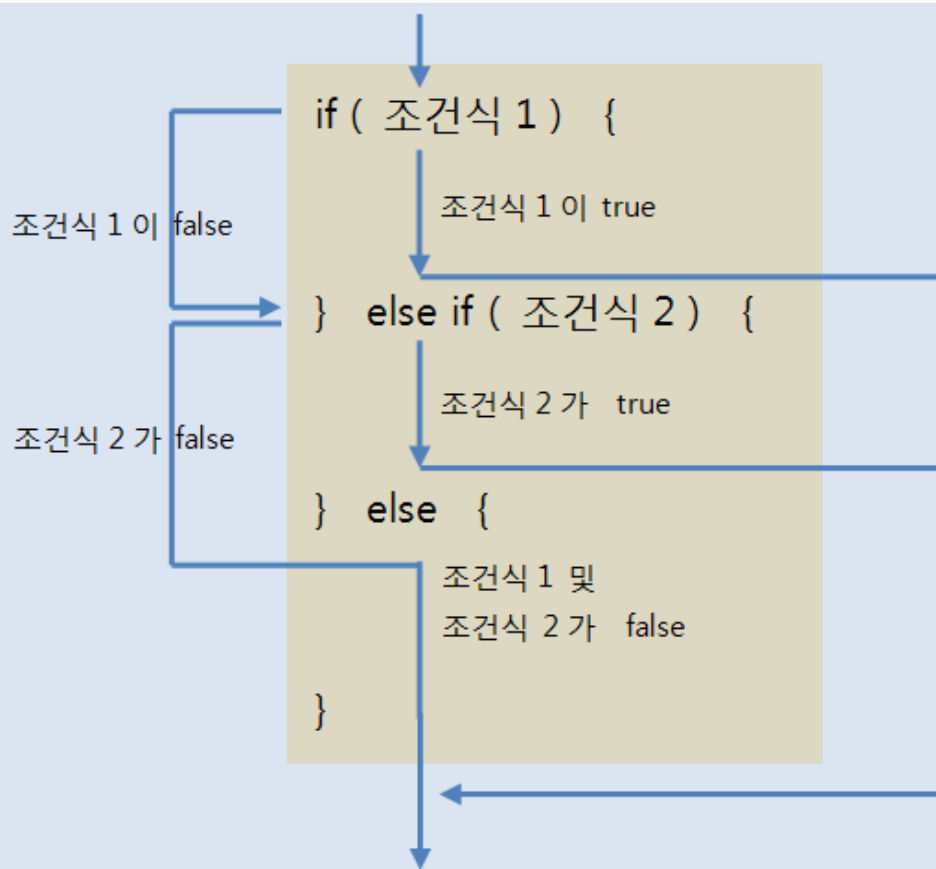
조건식 결과 따라 실행 블록 선택



5. 조건문(if문,switch문)

if-else if-else 문

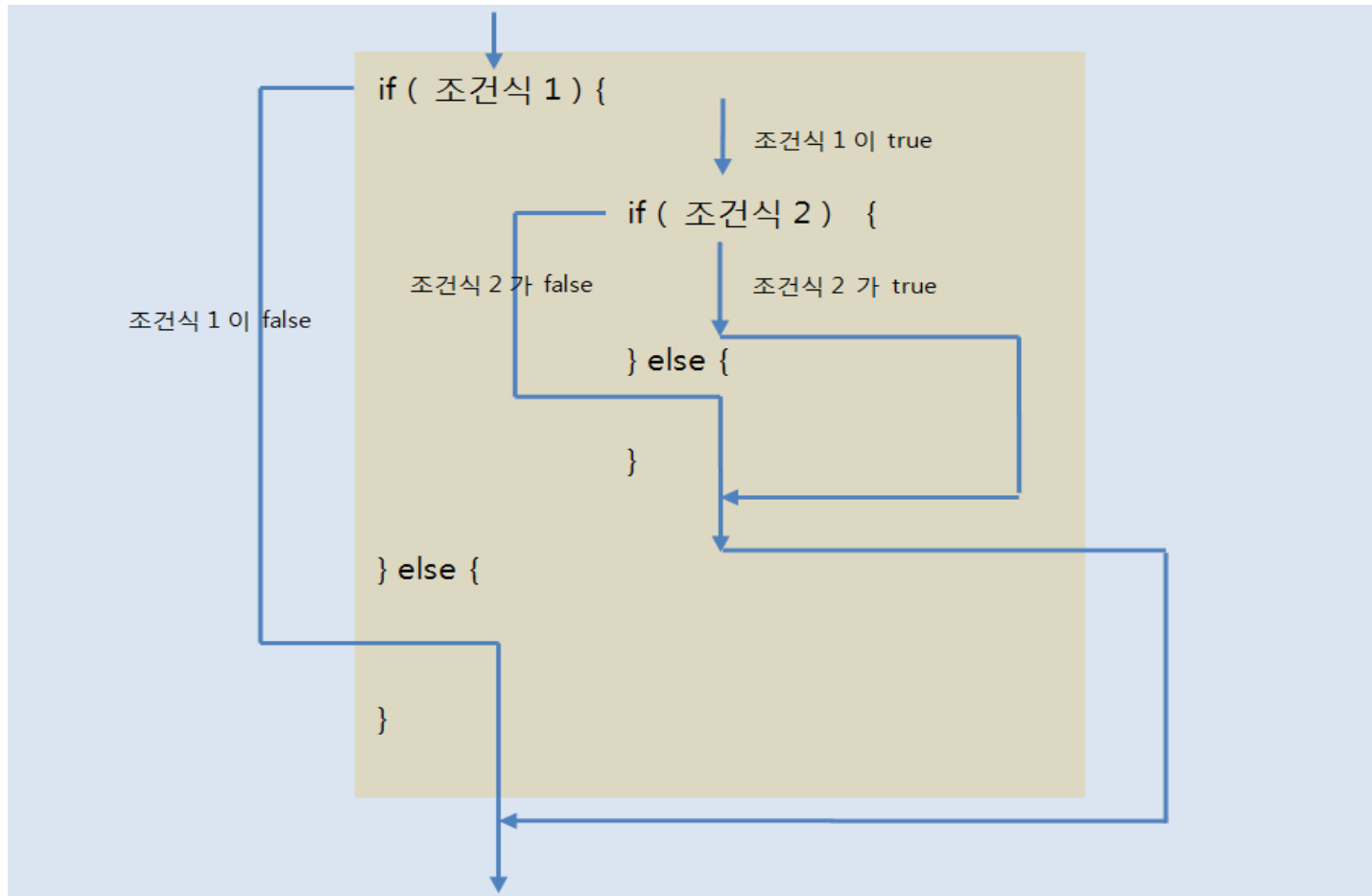
복수의 조건식 두어 조건식을 만족하는 블록만 실행



5. 조건문(if문,switch문)

중첩 if문

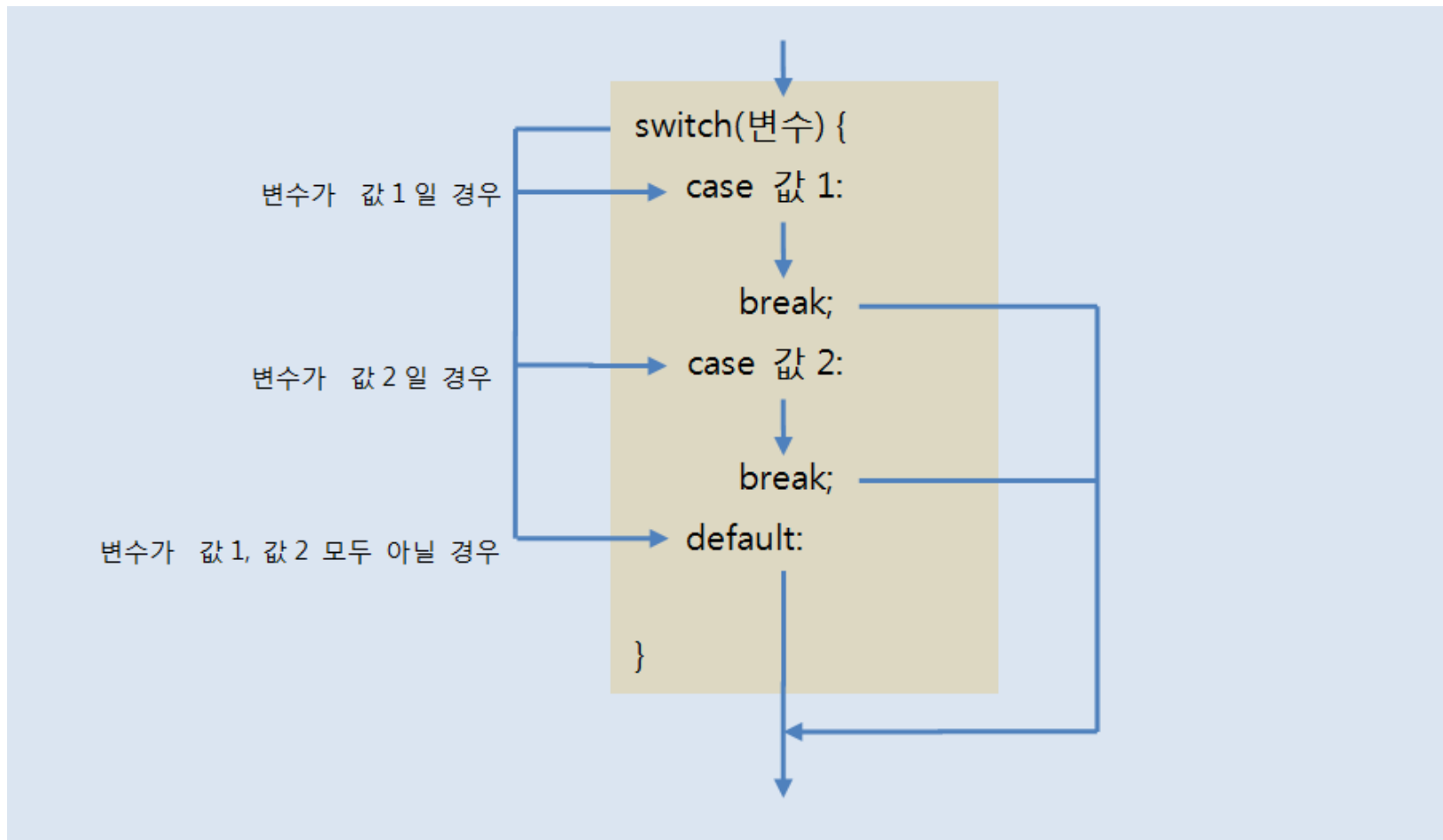
코드 실행 흐름을 이해하는 것이 가장 중요



5. 조건문(if문,switch문)

Switch문

변수나 연산식의 값에 따라 실행문 선택할 때 사용

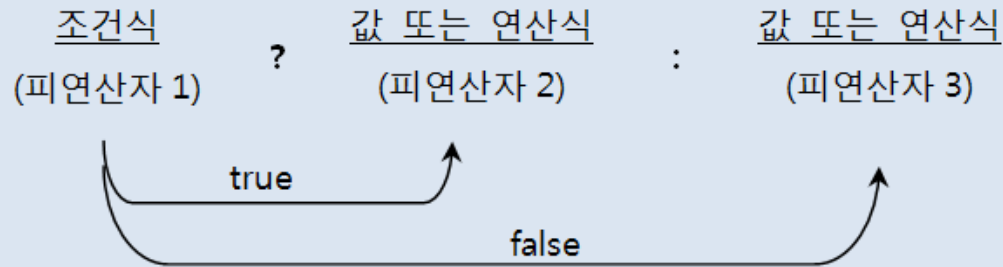


5. 삼항연산자

삼항 연산자란?

세 개의 피연산자를 필요로 하는 연산자

앞의 조건식 결과에 따라 콜론 앞 뒤의 피연산자 선택 -> 조건 연산식



```
int score = 95;  
char grade = (score>90) ? 'A' : 'B'
```

=

```
int score = 95;  
char grade;  
if(score>90) {  
    grade = 'A';  
} else {  
    grade = 'B';  
}
```

6. 반복문(for 문,while문, do-while문)

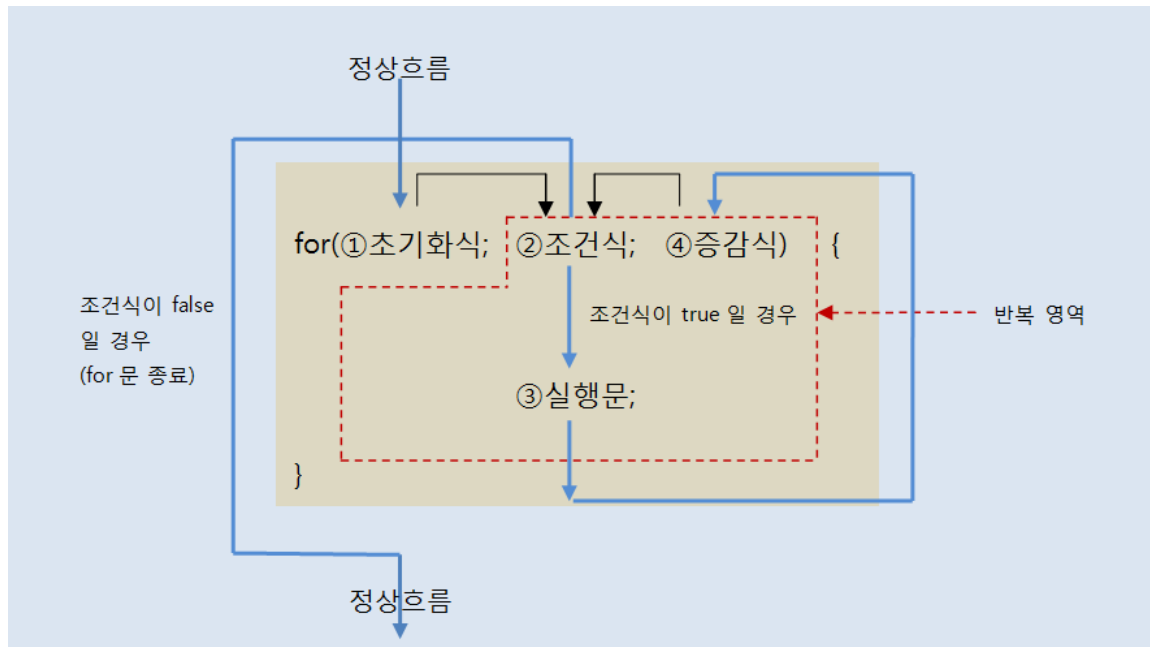
반복문

중괄호 블록 내용을 반복적으로 실행할 때 사용

종류: for문, while문, do-while문

6. 반복문(for 문, while 문, do-while 문)

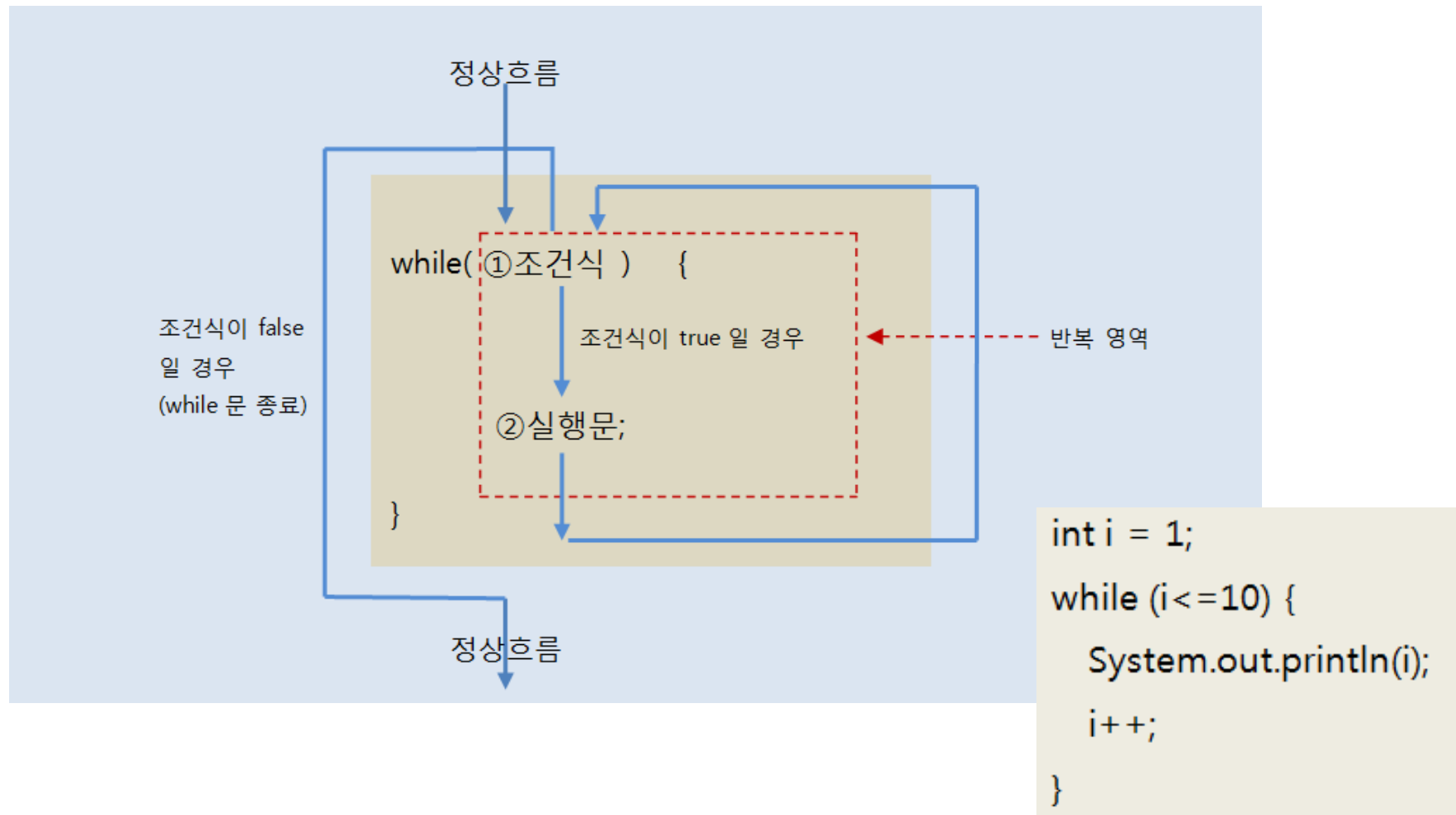
for문: 반복 횟수를 알고 있을 때 주로 사용



```
int sum = 0;
for (int i=1; i<=100; i++) {
    sum = sum + i; ●----- 100 번 반복
}
System.out.println("1~100 까지의 합:" + sum);
```

6. 반복문(for 문,while문, do-while문)

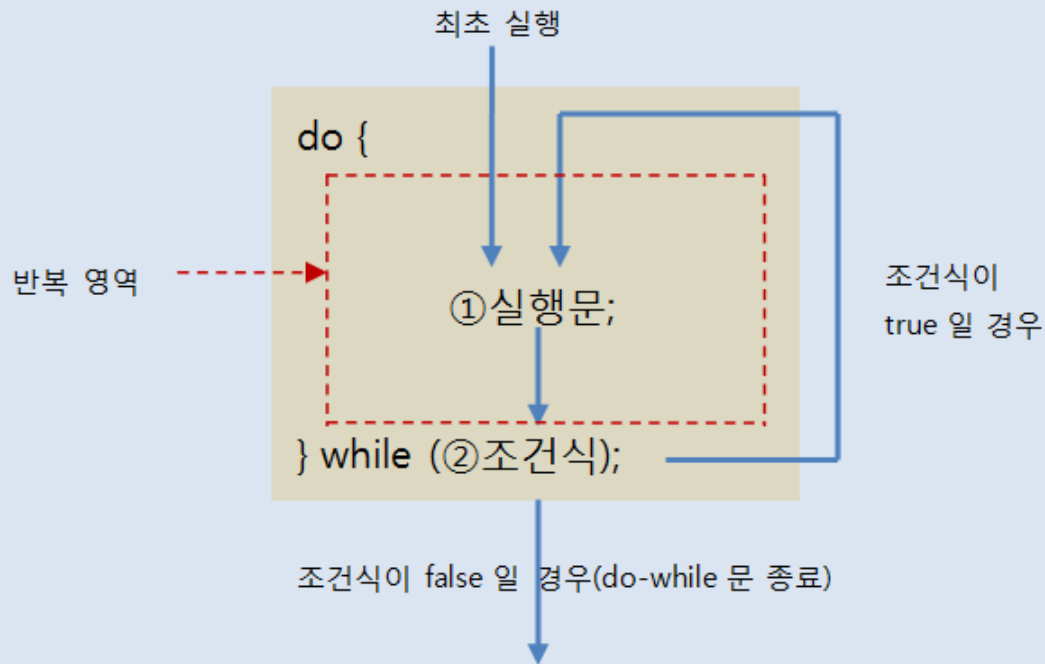
while문: 조건에 따라 반복을 계속할지 결정할 때 사용



6. 반복문(for 문,while문, do-while문)

do-while문

조건 따라 반복 계속할지 결정할 때 사용하는 것은 while문과 동일
무조건 중괄호 { } 블록을 한 번 실행한 후, 조건 검사해 반복 결정



6. 반복문(for 문,while문, do-while문)

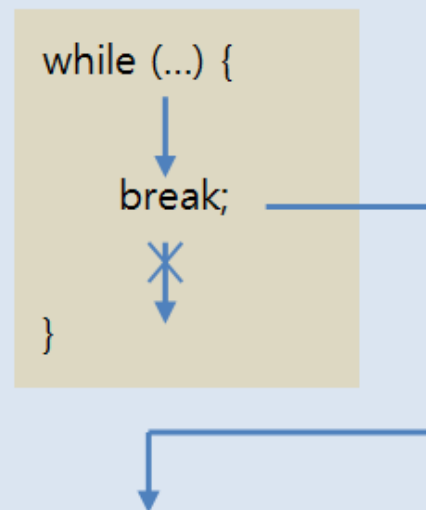
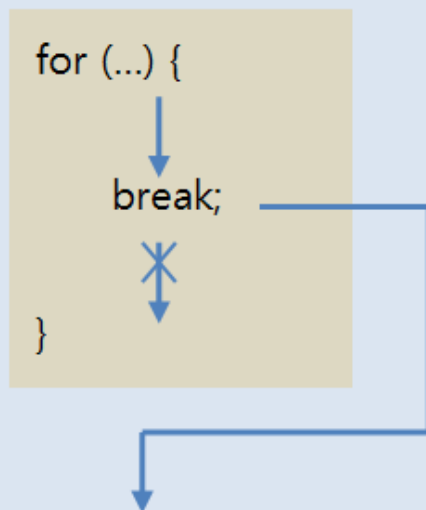
break 문

for문, while문, do-while문 종료 (반복 취소)

Switch문 종료

대개 if문과 같이 사용

if문 조건식에 따라 for문과 while문 종료할 때 사용



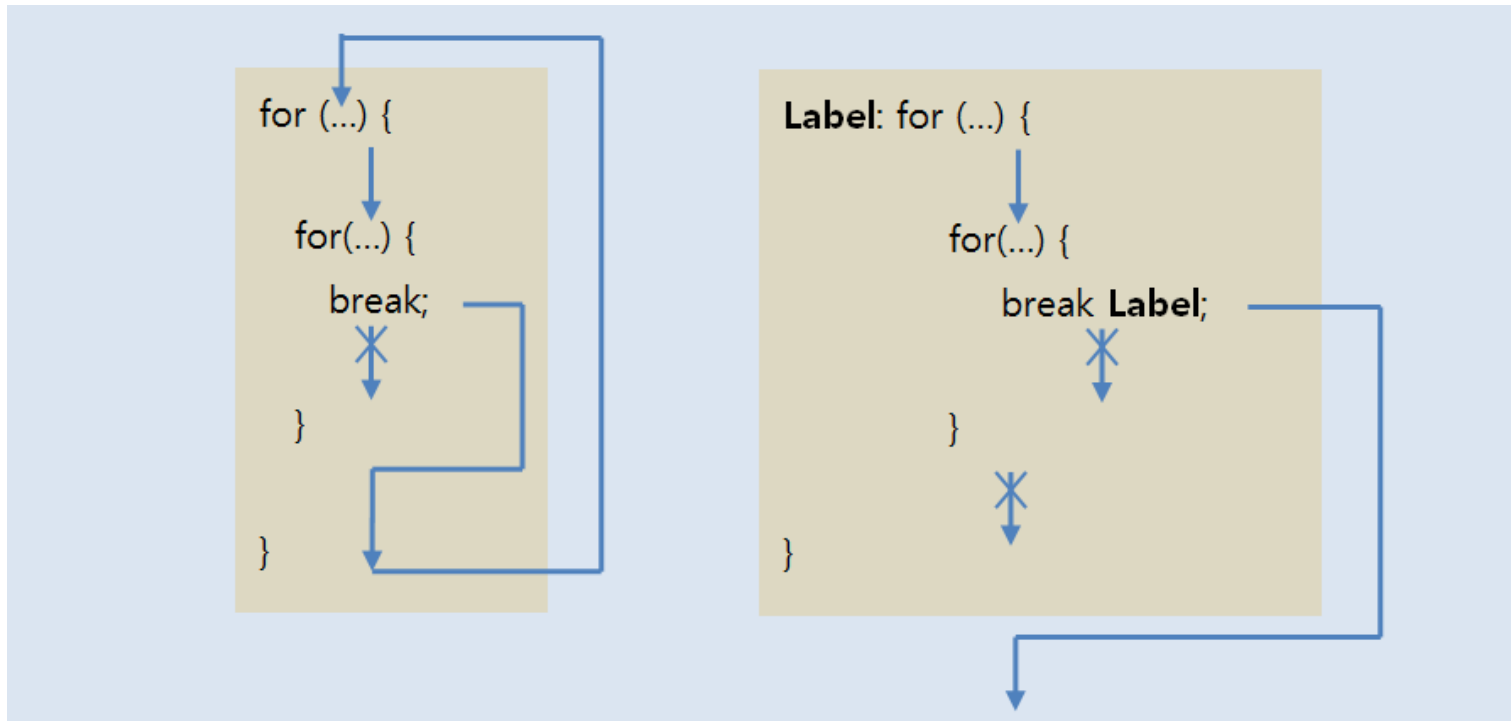
6. 반복문(for 문,while문, do-while문)

break 문

반복문이 중첩된 경우

반복문이 중첩되어 있을 경우 break; 문은 가장 가까운 반복문만 종료

바깥쪽 반복문까지 종료시키려면 반복문에 이름(라벨)을 붙이고, "break 이름;" 사용



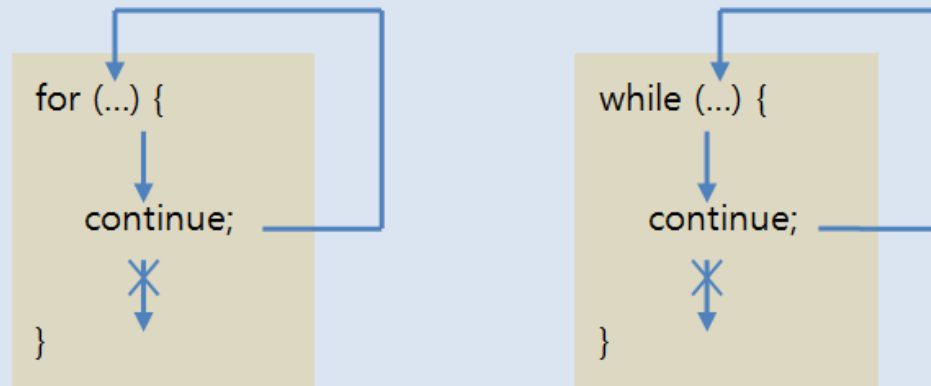
6. 반복문(for 문, while 문, do-while 문)

continue 문

for문, while문, do-while문에서 사용

for문: 증감식으로 이동

while문, do-while문: 조건식으로 이동



```
for(int i=1; i<=10; i++) {
    if(i%2 != 0) { ●----- 2로 나눈 나머지가 0 이 아닐 경우
                        즉 홀수인 경우
        continue;
    }
    System.out.println(i); ●----- 홀수는 실행되지 않는다.
}
```