제어문

자바 프로그램이 원하는 결과를 얻기 위해서는 프로그램의 순차적인 흐름을 제어해야만 할 경우가 생깁니다.

이때 사용하는 명령문을 제어문이라고 하며, 이러한 제어문에는 조건문, 반복문 등이 있습니다.

이러한 제어문에 속하는 명령문들은 중괄호({})로 둘러싸여 있으며, 이러한 중괄호 영역을 블록(block)이라고 합니다.

조건문

조건문은 주어진 조건식의 결과에 따라 별도의 명령을 수행하도록 제어하는 명령문입니다.
조건문 중에서도 가장 기본이 되는 명령문은 바로 if 문입니다.

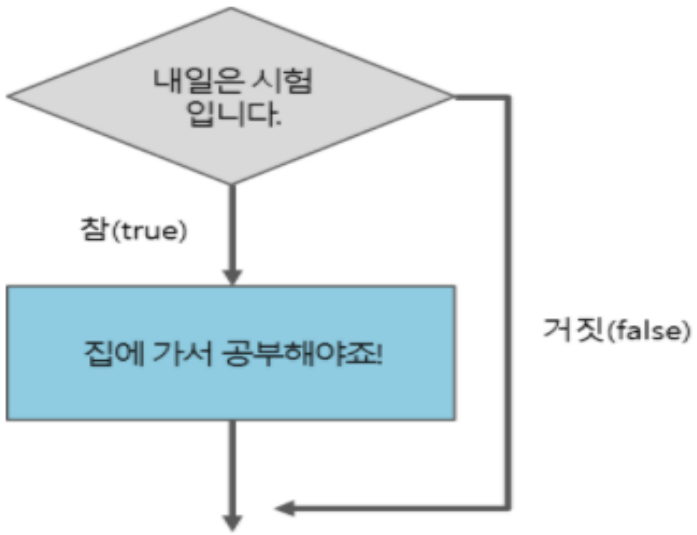
자바에서 사용하는 대표적인 조건문의 형태는 다음과 같습니다.

1. if 문
2. if / else 문
3. if / else if / else 문
4. switch 문

if 문

if 문은 조건식의 결과가 참(true)이면 주어진 명령문을 실행하며, 거짓(false)이면 아무것도 실행하지 않습니다.

if 문을 순서도로 표현하면 다음 그림과 같습니다.



자바에서 if 문의 문법은 다음과 같습니다.

문법

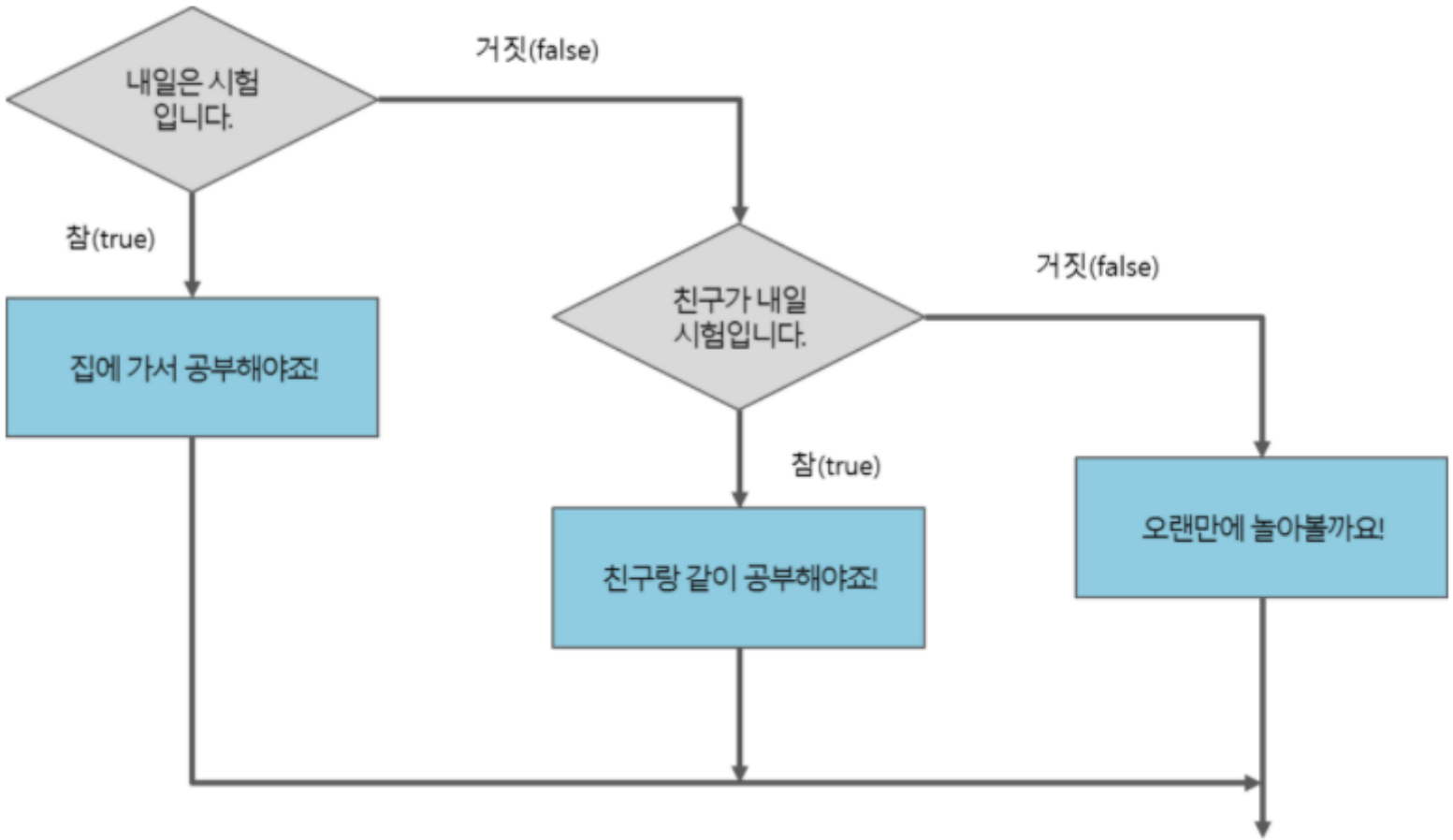
```
if (조건식) {
    조건식의 결과가 참일 때 실행하고자 하는 명령문;
}
```

위의 코드에서 블록에 속한 명령문은 중괄호({})를 기준으로 오른쪽으로 들여쓰기가 되어 있는 것을 볼 수 있습니다.

if / else if / else 문

if / else if / else 문은 마치 새로운 구문처럼 보이지만, 사실은 두 개의 if / else 문이 연달아 나온 것뿐입니다.
이러한 if / else if / else 문은 조건식을 여러 개 명시할 수 있으므로 중첩된 if 문을 좀 더 간결하게 표현할 수 있습니다.

if / else if / else 문을 순서도로 표현하면 다음 그림과 같습니다.



자바에서 if / else if / else 문의 문법은 다음과 같습니다.

문법

```
if (조건식1) {
    조건식1의 결과가 참일 때 실행하고자 하는 명령문;
} else if (조건식2) {
    조건식2의 결과가 참일 때 실행하고자 하는 명령문;
} else {
    조건식1의 결과도 거짓이고, 조건식2의 결과도 거짓일 때 실행하고자 하는 명령문;
}
```

이때 else if 문은 여러 번 나와도 상관없지만, if 문과 else 문은 단 한 번만 나올 수 있습니다.

switch 문

switch 문은 if / else 문과 마찬가지로 주어진 조건 값의 결과에 따라 프로그램이 다른 명령을 수행하도록 하는 조건문입니다.
이러한 switch 문은 if / else 문보다 가독성이 더 좋으며, 컴파일러가 최적화를 쉽게 할 수 있어 속도 또한 빠른 편입니다.

하지만 switch 문의 조건 값으로는 int형으로 승격할 수 있는(integer promotion) 값만이 사용될 수 있습니다.

즉, 자바에서는 swich 문의 조건 값으로 byte형, short형, char형, int형의 변수나 리터럴을 사용할 수 있습니다.
또한, 이러한 기본 타입에 해당하는 데이터를 객체로 포장해 주는 래퍼 클래스(Wrapper class) 중에서 위에 해당하는 Byte, Short, Character, Integer 클래스의 객체도 사용할 수 있습니다.
그리고 enum 키워드를 사용한 열거체(enumeration type)와 String 클래스의 객체도 사용할 수 있습니다.
따라서 switch 문은 if / else 문보다는 사용할 수 있는 상황이 적습니다.

자바에서 switch 문의 문법은 다음과 같습니다.

문법

```
switch (조건 값) {
    case 값1:
        조건 값이 값1일 때 실행하고자 하는 명령문;
        break;
    case 값2:
        조건 값이 값2일 때 실행하고자 하는 명령문;
        break;
    ...
    default:
        조건 값이 어떠한 case 절에도 해당하지 않을 때 실행하고자 하는 명령문;
        break;
}
```

default 절은 조건 값이 위에 나열된 어떠한 case 절에도 해당하지 않을 때만 실행됩니다.
이 절은 반드시 존재해야 하는 것은 아니며 필요할 때만 선언할 수 있습니다.

반복문

반복문이란 프로그램 내에서 똑같은 명령을 일정 횟수만큼 반복하여 수행하도록 제어하는 명령문입니다.
프로그램이 처리하는 대부분의 코드는 반복적인 형태가 많으므로, 가장 많이 사용되는 제어문 중 하나입니다.

자바에서 사용되는 대표적인 반복문의 형태는 다음과 같습니다.

1. while 문
2. do / while 문
3. for 문
4. Enhanced for 문

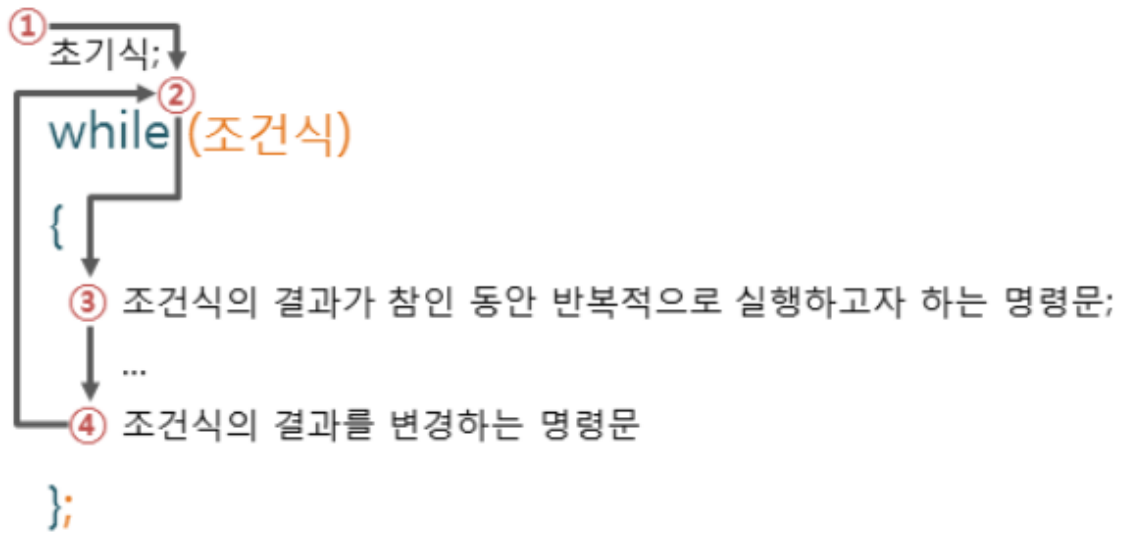
while 문

while 문은 특정 조건을 만족할 때까지 계속해서 주어진 명령문을 반복 실행합니다.

자바에서 while 문의 문법은 다음과 같습니다.

문법

```
while (조건식) {
    조건식의 결과가 참인 동안 반복적으로 실행하고자 하는 명령문;
}
```



while 문은 우선 조건식이 참(true)인지를 판단하여, 참이면 내부의 명령문을 실행합니다.
내부의 명령문을 전부 실행하고 나면, 다시 조건식으로 돌아와 또 한 번 참인지를 판단하게 됩니다.
이렇게 조건식의 검사를 통해 반복해서 실행되는 반복문을 루프(loop)라고 합니다.

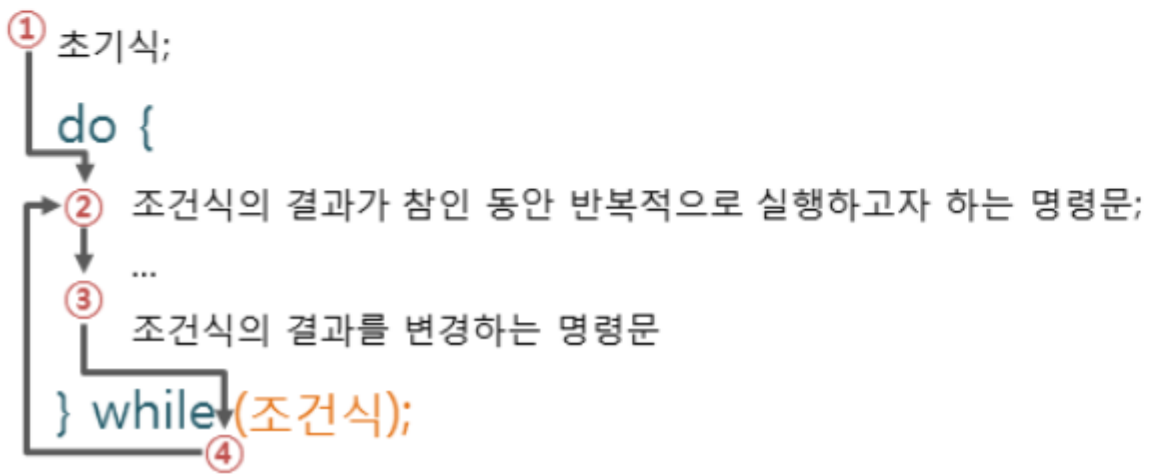
do / while 문

while 문은 루프에 진입하기 전에 먼저 조건식부터 검사합니다.
하지만 do / while 문은 먼저 루프를 한 번 실행한 후에 조건식을 검사합니다.
즉, do / while 문은 조건식의 결과와 상관없이 무조건 한 번은 루프를 실행합니다.

자바에서 do / while 문의 문법은 다음과 같습니다.

문법

```
do {
    조건식의 결과가 참인 동안 반복적으로 실행하고자 하는 명령문;
} while (조건식);
```



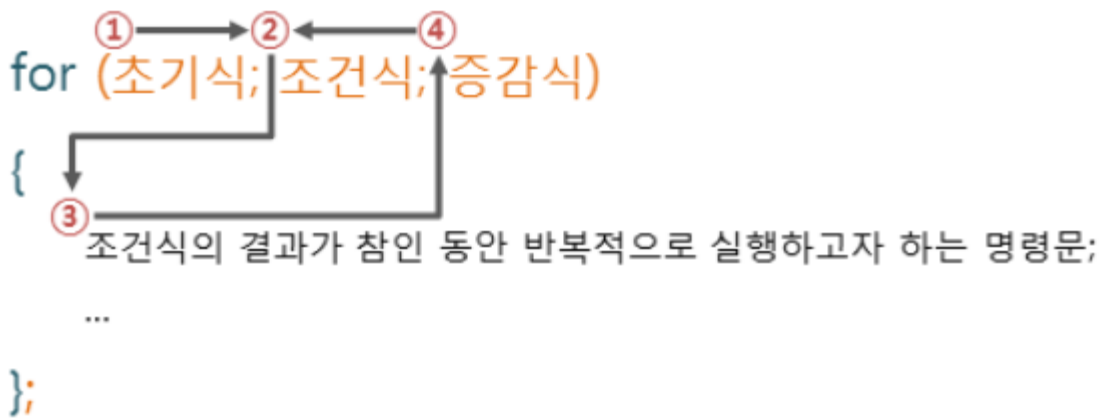
for 문

for 문은 while 문과는 달리 자체적으로 초기식, 조건식, 증감식을 모두 포함하고 있는 반복문입니다.
따라서 while 문보다는 좀 더 간결하게 반복문을 표현할 수 있습니다.

자바에서 for 문의 문법은 다음과 같습니다.

문법

```
for (초기식; 조건식; 증감식) {
    조건식의 결과가 참인 동안 반복적으로 실행하고자 하는 명령문;
}
```



이때 for 문을 구성하는 초기식, 조건식, 증감식은 각각 생략할 수 있습니다.

기타 제어문

루프의 제어

일반적으로 조건식의 검사를 통해 루프로 진입하면, 다음 조건식을 검사하기 전까지 루프 안에 있는 모든 명령문을 실행합니다.
하지만 continue 문과 break 문은 이러한 일반적인 루프의 흐름을 사용자가 직접 제어할 수 있도록 도와줍니다.

continue 문

continue 문은 루프 내에서 사용하여 해당 루프의 나머지 부분을 건너뛰고, 바로 다음 조건식의 판단으로 넘어가게 해줍니다.
보통 반복문 내에서 특정 조건에 대한 예외 처리를 하고자 할 때 자주 사용됩니다.

```

//다음 예제는 1부터 100까지의 정수 중에서 5의 배수와 7의 배수를 모두 출력하는 예제입니다.

public class prog {
    public static void main(String[] args) {
        for (int i = 1; i <= 100; i++) {
            if (i % 5 == 0 || i % 7 == 0) {
                System.out.println(i);
            } else {
                continue;
            }
        }
    }
}

/*

실행 결과
5

7

10

14

15

...

95

98

100


*/
```

break 문

break 문은 루프 내에서 사용하여 해당 반복문을 완전히 종료시킨 뒤, 반복문 바로 다음에 위치한 명령문을 실행합니다.
즉 루프 내에서 조건식의 판단 결과와 상관없이 반복문을 완전히 빠져나가고 싶을 때 사용합니다.

다음 예제는 1부터 100까지의 합을 무한 루프를 통해 구하는 예제입니다.

예제

```
int num = 1, sum = 0;

while (true) { // 무한 루프
    sum += num;
    if (num == 100) {
        break;
    }
    num++;
}
System.out.println(sum);
```

실행 결과

```
5050
```

이름을 가지는 반복문(break with label)

일반적인 break 문은 단 하나의 반복문만을 빠져나가게 해줍니다.
따라서 여러 반복문이 중첩된 상황에서 한 번에 모든 반복문을 빠져나가거나, 특정 반복문까지만 빠져나가고 싶을 때는 다른 방법을 사용해야 합니다.

이때 사용할 수 있는 방법이 바로 반복문에 이름(label)을 설정하는 것입니다.
가장 바깥쪽 반복문이나 빠져나가고 싶은 특정 반복문에 이름을 설정한 후, break 키워드 다음에 해당 이름을 명시하면 됩니다.
그러면 해당 break 키워드는 현재 반복문이 아닌 해당 이름의 반복문 바로 다음으로 프로그램의 실행을 옮겨줍니다.

단, 이때 이름(label)은 가리키고자 하는 반복문의 키워드 바로 앞에 위치해야 합니다.
이름과 반복문의 키워드 사이에 명령문이 존재하면, 자바 컴파일러는 오류를 발생시킬 것입니다.

C언어나 C++과는 달리 자바에는 goto 문이 없습니다.
따라서 이렇게 반복문을 가리키는 이름(label)은 break 문이나 continue 문에만 사용될 수 있습니다.

다음 예제는 구구단 중에서 2단부터 4단까지를 출력하는 예제입니다.

예제

```
allLoop :
for (int i = 2; i < 10; i++) {
    for (int j = 2; j < 10; j++) {
        if (i == 5) {
            break allLoop;
        }
        System.out.println(i + " * " + j + " = " + (i * j));
    }
}
```

실행 결과

```
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
...
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
```

위의 예제에서 변수 i의 값이 5가 되는 순간, 해당 프로그램의 제어는 두 개의 for 문을 모두 빠져나와 종료됩니다.