

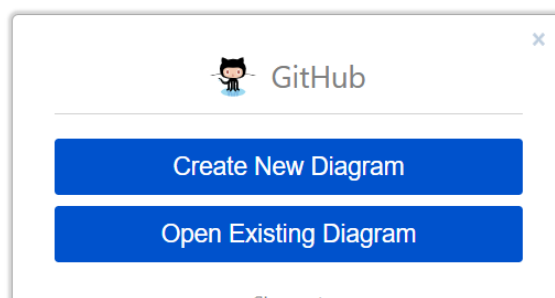
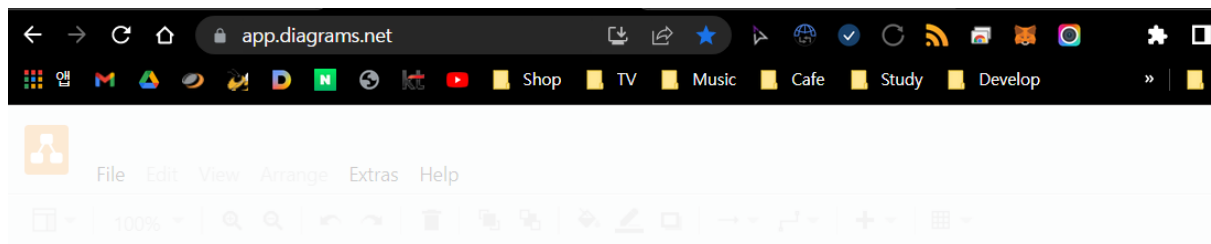
# Class[Algorithm]

## 브라우저로 JAVA 연습하기

### Java Online Compiler & Interpreter

Write and run Java code using our Java online compiler & interpreter. You can build, share, and host applications right from your browser!

 <https://replit.com/languages/java>



## Algorithm

컴퓨터가 아직은 사람의 언어를 완전히 이해하지 못한다.

따라서 알고리즘을 프로그래밍 언어로 기술한 것을 프로그램 이라고한다.

알고리즘은 문제나 과제를 해결하기 위한 처리 절차를 하나하나 구체적인 순서에 따라 표현한 아이디어 또는 생각을 말한다.

알고리즘 컴퓨터에서만 사용되는 것이 아니다. 일상생활에서도 많이 사용된다. 예를들면 요리의 레시피, 음악 악보, 가전제품 설명서 등이 알고리즘의 예이다.

그러나 알고리즘은 아이디어나 생각이기 때문에 표현하기가 어렵다. 그래서 표현하는 도구로

1. psuedo(수도) code
2. **flow chart** 순서도, 흐름도
3. program

요리의 절차 ———> 글로 문장화 ———> 레시피

연주의 절차 ———> 악보로 도면화 ———> 악보

기계 조작 ———> 일러스트 ———> 사용 설명서

## Algorithm & Program

알고리즘을 프로그래밍 언어 C, Java, Python 등으로 기술한 것이 프로그램이다.

알고리즘 자체는 눈으로 볼수 없다. 그래서 이것을 전달하기 위해 문장이나 일러스트 등으로 표현한다.

하지만 컴퓨터에 전달하려면 어떻게 해야 할까...

컴퓨터에 전달할때도 문장이나 글등을 사용하면 컴퓨터는 이해를 하기 어렵다.

사람 ————— 프로그램 —————> 컴퓨터  
알고리즘

프로그래밍의 절차

기획 —> 설계 —> 프로그래밍 —> 디버그 —> 문서화  
(알고리즘)

## 좋은 알고리즘이란?

좋은 프로그램은 알고리즘의 좋고 나쁨에 달렸다. 좋은 알고리즘은 어떻게 판단할까??

### 1. 알기 쉽다.

가능한 알기 쉬우며 이해하기 쉬워야 한다. 특히 여러사람이 작업할때 다른 팀원들도 바로 이해 할수 있어야한다. 복잡하고 난해한 알고리즘은 올바른 결과가 나오는지 검증하기도 어려워 틀린 부분을 찾기도 어렵게 된다.

따라서 알기 쉽고 이해하기 쉽게 작성하자.

### 2. 속도가 빠르다.

속도가 빠르다는 것은 실행하고 그 결과가 나올때까지의 시간이 짧다는 것을 말한다. 결과 출력의 시간이 길다면 좋은 알고리즘이라고 할수 없다.

### 3. 효율적이다. ( 메모리를 적게 차지한다.)

프로그램을 실행 할때 사용되는 메모리의 영역이 작다는 것을 의미한다.

### 4. 재이용이 쉽다.

프로그래밍 시간을 단축하려면 코딩하는 속도 자체를 높이는 것도 하나의 방법이 될수 있지만 이것은 한계가 금방 오게 된다. 과거에 작성한 프로그램을 그대로 또는 부분적으로 이용하는 비율 증가하게 되면 새로운 프로그램을 만드는 시간도 줄어들게 된다.

따라서 프로그램을 작성할때 가능하면 재사용이 쉽고 범용성이 높은 알고리즘을 고려하면 좋은 프로그램을 만들 수 있게 된다.

## 좋은 알고리즘의 2가지 조건

- 정확한 결과를 얻을 수 있어야 한다.
- 반드시 종료되어야 한다.

## 알고리즘을 공부해야 하는 이유

1. 좋은 프로그램을 만들 수 있게 된다.
2. 프로그램의 좋고 나쁨을 판단할 수 있게 된다.
3. 프로그램 작성 과정 전체를 효율화 할 수 있다.
4. 프로그래밍 기술을 향상 시킬 수 있다.

## 3 Basics

1. **순차구조** : 처음부터 순서대로 처리하는 절차.
2. **선택구조(if)** : 조건식으로 판단하여 실행할 처리를 전환하는 절차.
3. **반복구조(while/for)** : 조건을 만족하는 동안 같은 처리를 반복하는 절차.

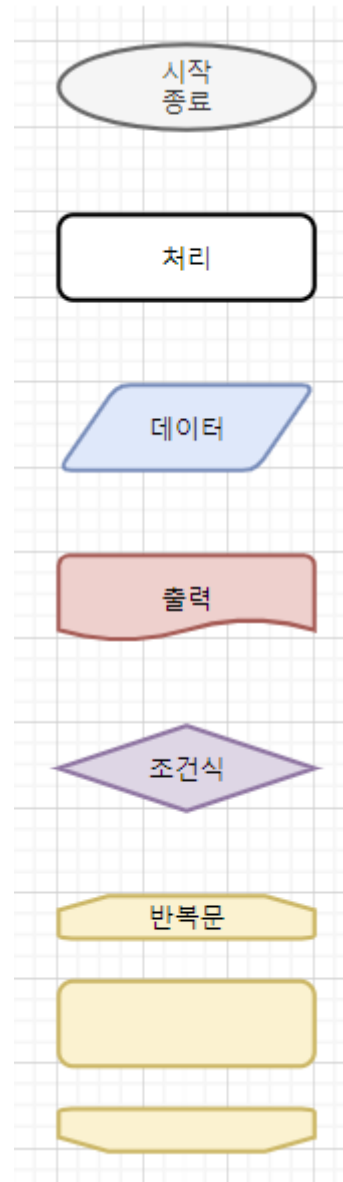
반복구조는 효율적인 알고리즘의 핵심 포인트이다.

모든 알고리즘은 아무리 복잡해 보여도 이 세가지의 절차의 조합으로 표현할 수 있다.

다시 말해 이 세가지만 기억하면 대부분의 알고리즘을 작성할 수 있게 된다.

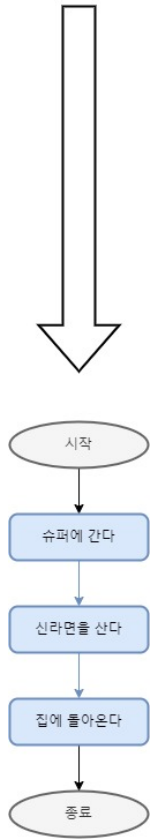
## 알고리즘을 만들기 위한 기본 요소

→

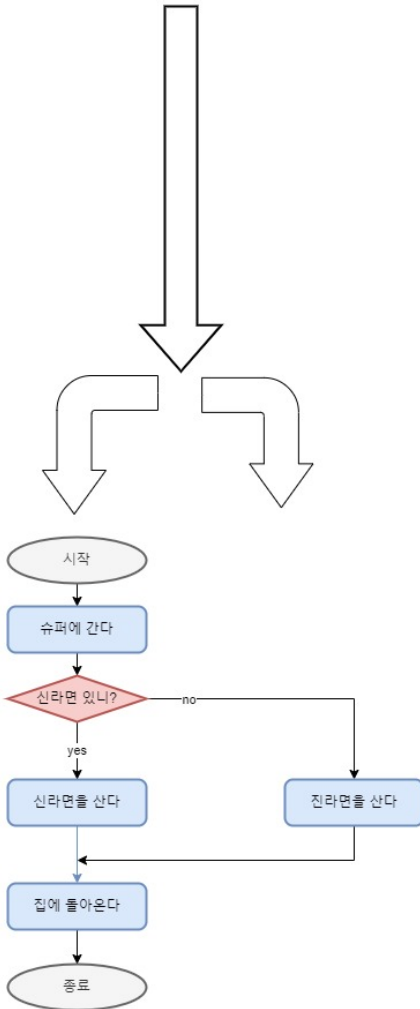


## [3 Basics]의 전체적인 구조

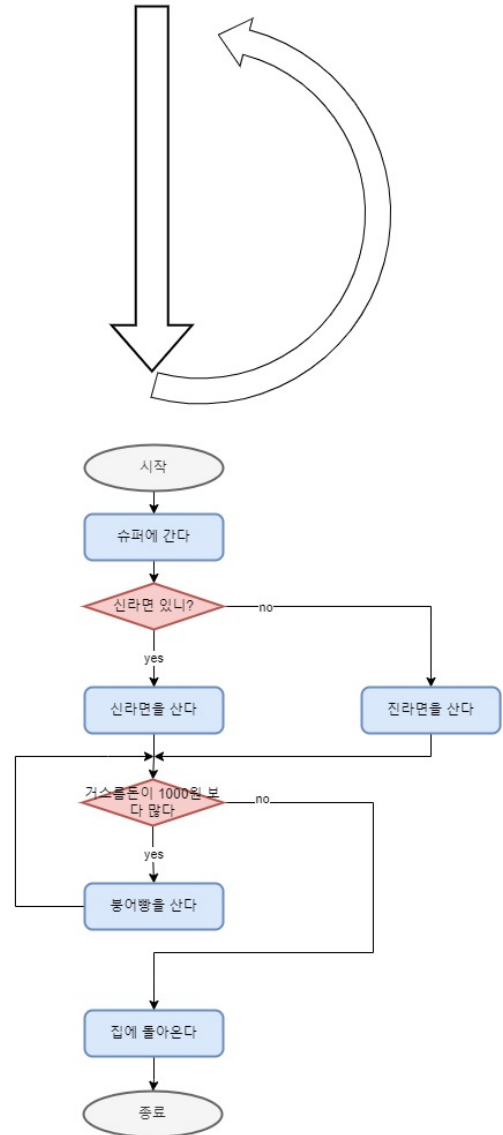
순차 구조



선택 구조

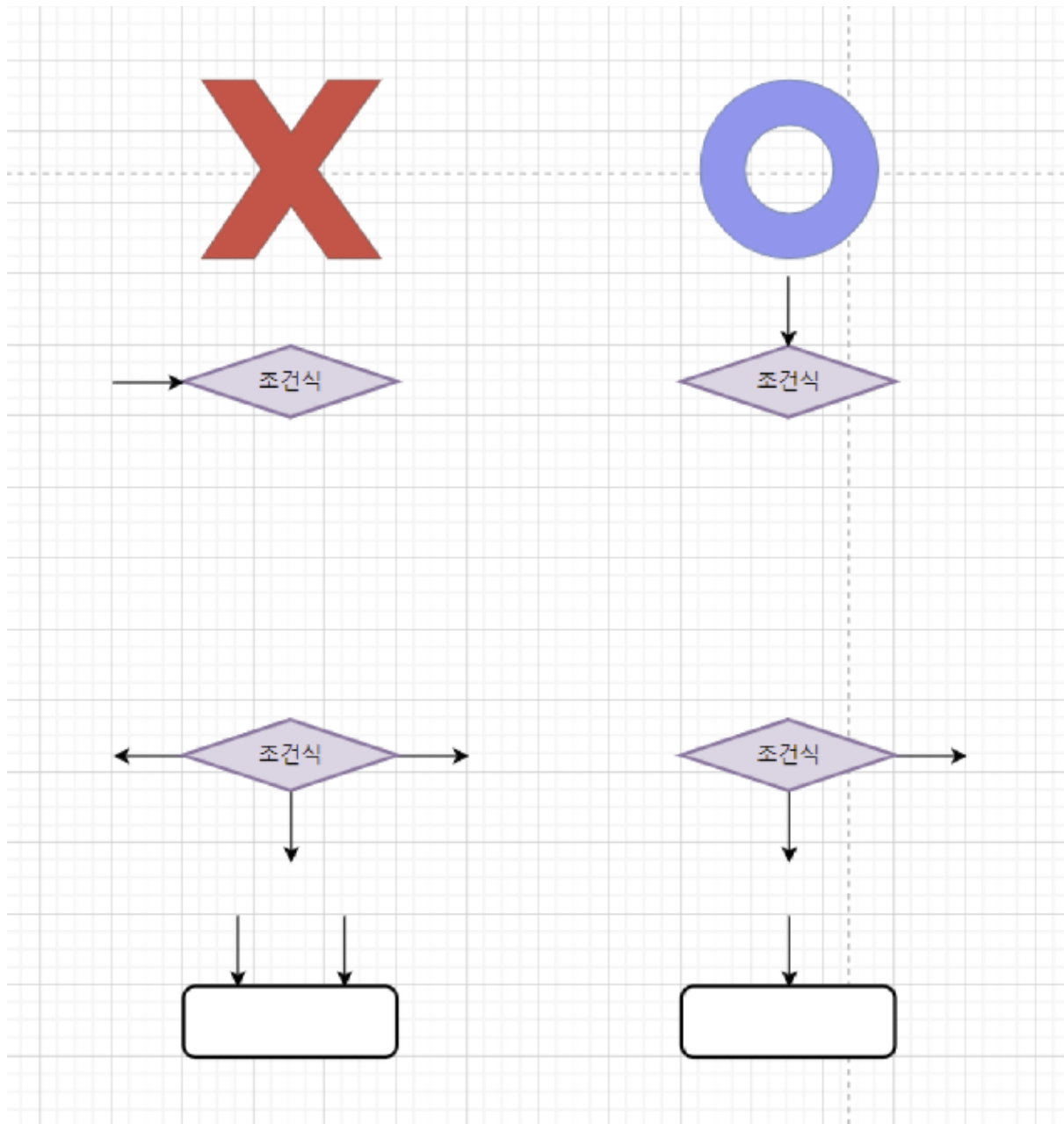


반복 구조



틀린 예시

옳은 예시



## 간단한 알고리즘 예시

### 1. Triangle

삼각형의 면적을 구하는 알고리즘

1. 순차적 분해하여 점차적으로 생각하자

2. 사칙 연산 처리는 산술 연산자를 사용한다.
3. 나눗셈은 / 기호를 사용하자.

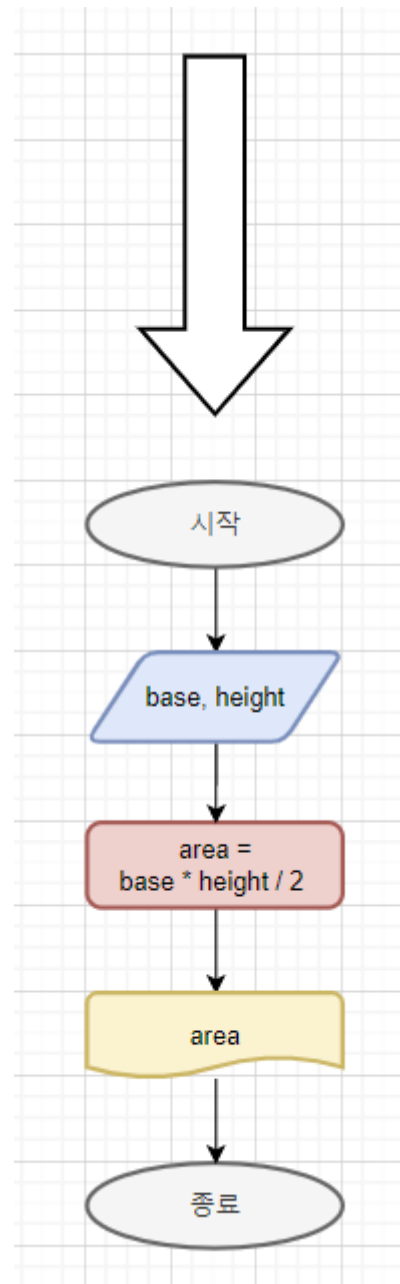
우리가 이미 알고 있는

삼각형의 면적 = 밑변 \* 높이 / 2

변수가 필요하다. 면적은 area 밑변은 base 높이는 height 으로 정하다.

먼저 수도(의사)코드로 생각해 본다.

- base와 height 을 입력
- $base * height / 2 \rightarrow area$
- area 출력



## 2. Bigger

두 수의 대소를 판별

1. 2개 의 데이터를 비교하기 위해서는 선택 구조가 필요하다.



2. 조건식에서는 관계 연산자를 사용한다.

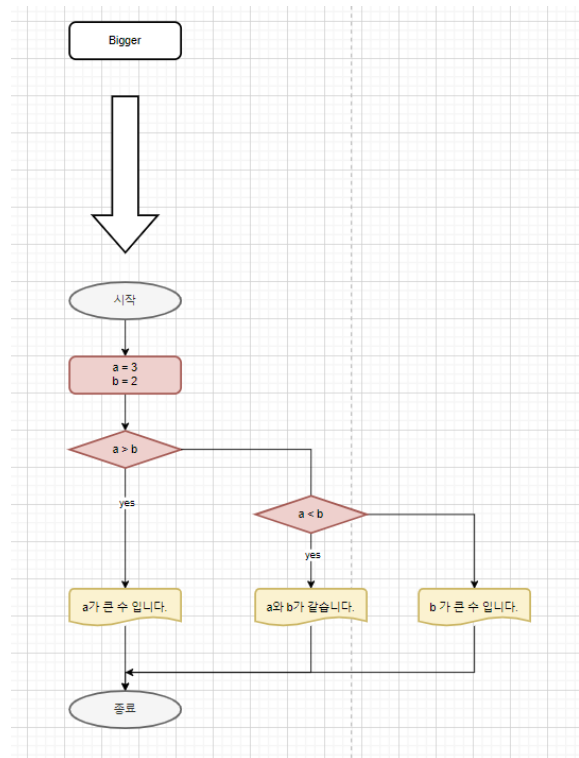
※ 출력이나 조건식 같은 기본요소(도형)를 정확히 구별하자.

데이터를 비교하는 처리를  
고려하기 위해서  
a와 b를 비교하여 a가 크면 a를 출력하  
고  
그렇지 않으면 b를 출력한다.

특별한 계산이 필요하지 않은  
단순한 알고리즘이다.

a=3 , b =2

“a가 큰수 입니다.”



## Bigger 알고리즘 코드

※ 중첩 if보다는 이런 식으로 쉽게 구조가 보이는 알고리즘을 짜라.


```
Main.java x +
Main.java
1 class Main {
2     public static void main(String[] args) {
3
4         double a = 3;
5         double b = 2;
6
7         if(a > b) {
8             System.out.println('a');
9
10        } else if(a < b) {
11            System.out.println('b');
12        } else {
13            System.out.println('a' + "=" + 'b');
14        }
15    }
16 }
```

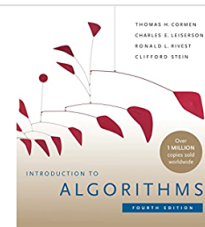
```
- Git x Shell x Console x +
> sh -c javac -classpath ./target/dependency/* -d . $(find .
> f -name '*.java')
> java -classpath ./target/dependency/* Main
a
>
```

## Introduction to Algorithms, fourth edition

Introduction to Algorithms, fourth edition: 9780262046305:

Computer Science Books @ Amazon.com

 [https://www.amazon.com/dp/026204630X/ref=mp\\_s\\_a\\_1\\_3?crid=WDS979R5CAKG&keywords=algorithm&qid=1674170628&srefix=algorithm%2Caps%2C409&sr=8-3](https://www.amazon.com/dp/026204630X/ref=mp_s_a_1_3?crid=WDS979R5CAKG&keywords=algorithm&qid=1674170628&srefix=algorithm%2Caps%2C409&sr=8-3)



★★★★★ 242