Day16

Algorithm

```
문제를 해결하기 위한 절차

컴퓨터 뿐만 아니라 음악의 악보, 요리의 레시피, 제품 사용설명서 등
상세하게 절차가 적혀있는 모든 것이 해당
```

프로그램 : 알고리즘을 프로그래밍 언어로 작성한 것
(상세하게 분해하여 구체적인 방법은 컴퓨터에게 지시)

1. 기획
2. 설계 ------- 알고리즘
3. 프로그래밍
4. 디버깅
5. 문서화

좋은 알고리즘

* 알기 쉽다
* 속도가 빠르다
* 효율적이다
* 재사용이 쉽다

알고리즘 기본형

아무리 복잡해 보이는 알고리즘도 이 기본형 3가지의 조합으로 구성됨

1) 순차구조 : 일반적인 흐름
2) 선택구조 : if문

3) 반복구조 : while문 for문

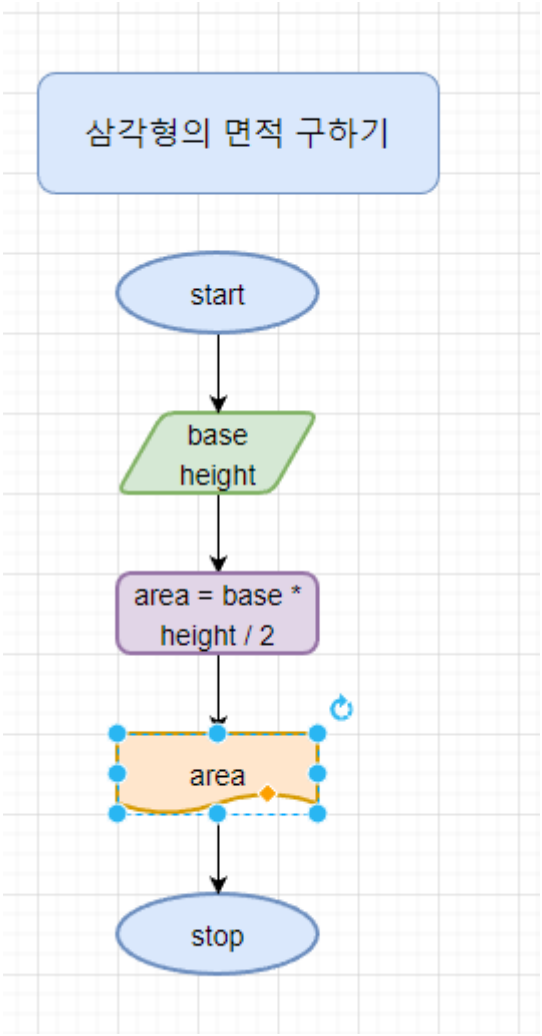
알고리즘 표현방법

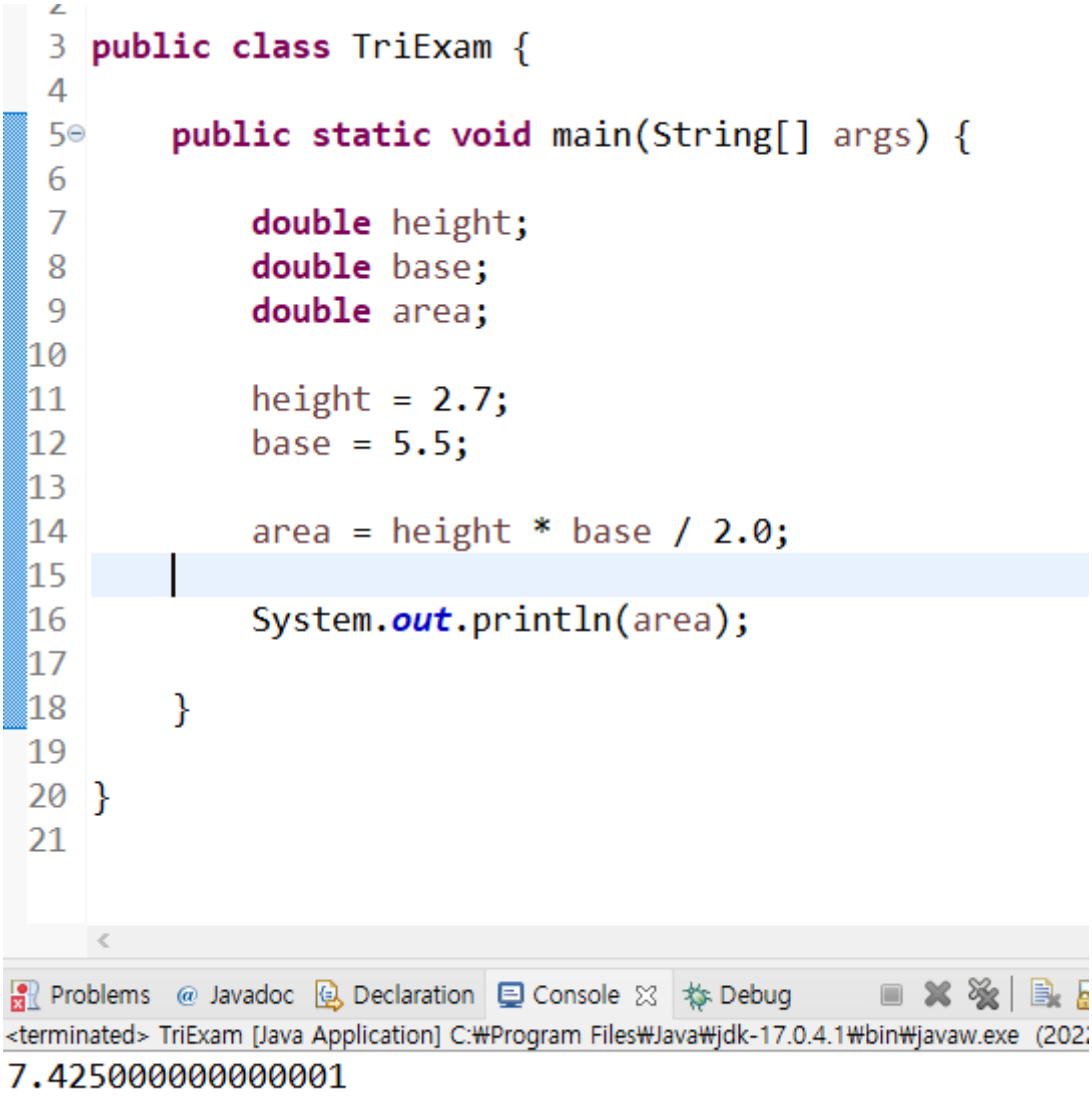
- 수도코드 pseudo code : 컴퓨터는 이해하지 못하는 컴퓨터 언어 ; 의사코드
- 흐름도 Flow Chart : 그림으로 표현하는 알고리즘

삼각형의 면적을 계산하는 알고리즘

```
삼각형의 면적 = 밑변 * 높이 * 1/2

면적 : area
밑변 : base
높이 : height

area = base * height * 1/2
```

1. base와 height를 입력한다
2. base와 height 곱한 값을 2로 나누어 area 변수에 대입
3. area 출력


삼각형의 면적 구하기
start
base
height
area = base *
height / 2
area
stop



두 데이터의 대소 판별

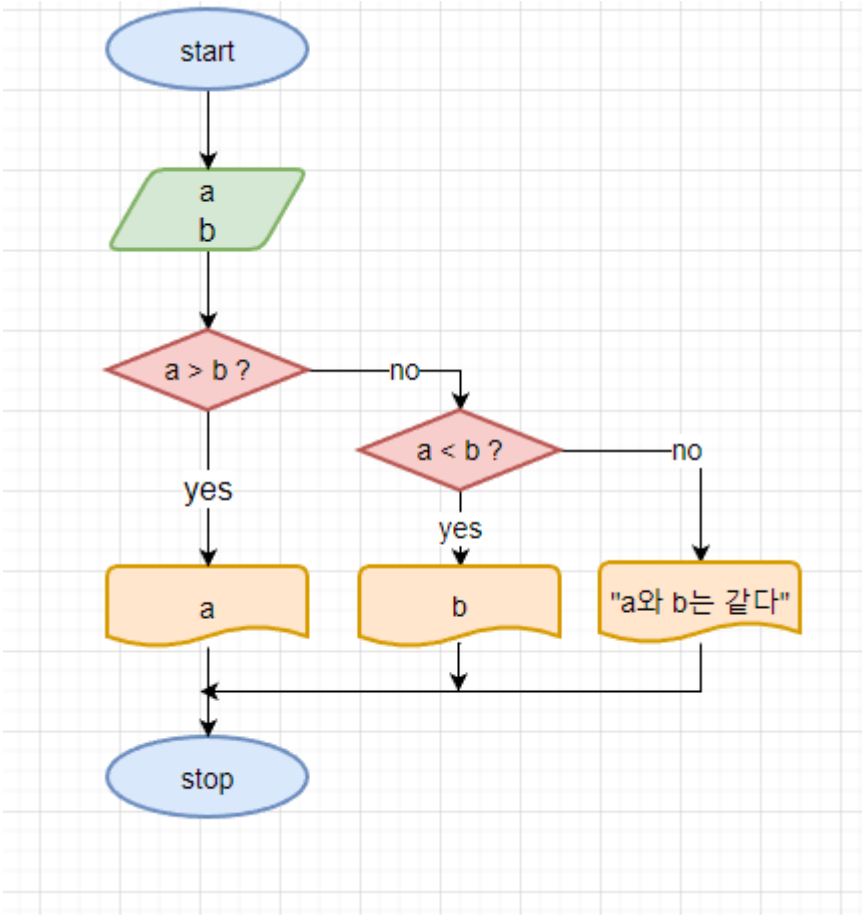
```
변수 a와 b의 데이터를 비교하여 a가 크면 a를 출력 b가 크면 b를 출력
```

1. a, b 입력
2. a>b 비교

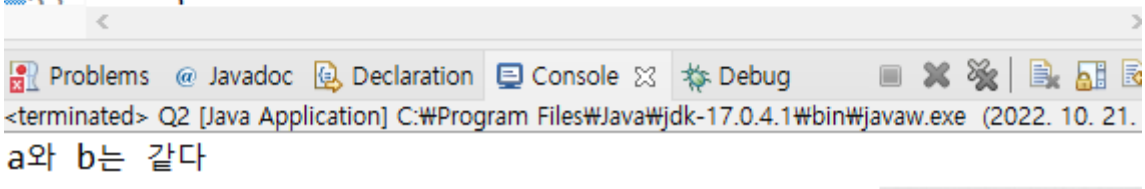
3-1. a가 큰 경우 a를 출력

3-2. b가 큰 경우 b를 출력

3-3 a와 b가 같은 경우 “a와 b는 같다” 출력


start
a
b
a > b ?
no
a < b ?
no
yes
yes
a
b
"a와 b는 같다"
stop

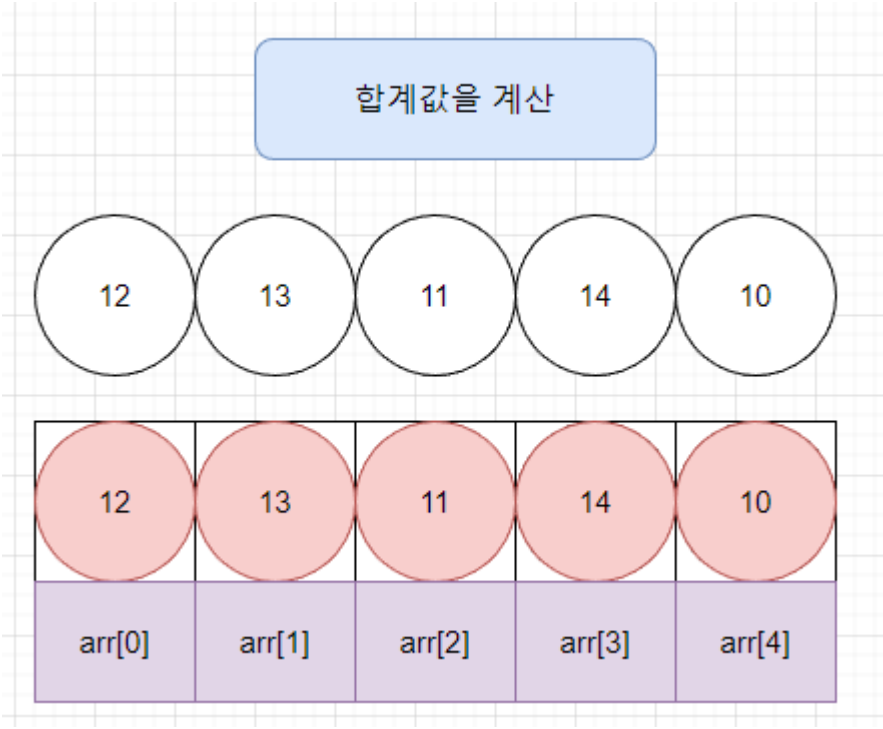
```java
public class Q2 {

    public static void main(String[] args) {

        int a = 5;
        int b = 5;

        if (a > b) {

            System.out.println(a);

        }else if (a < b) {

            System.out.println(b);

        }else {

        System.out.println("a와 b는 같다");

        }
```

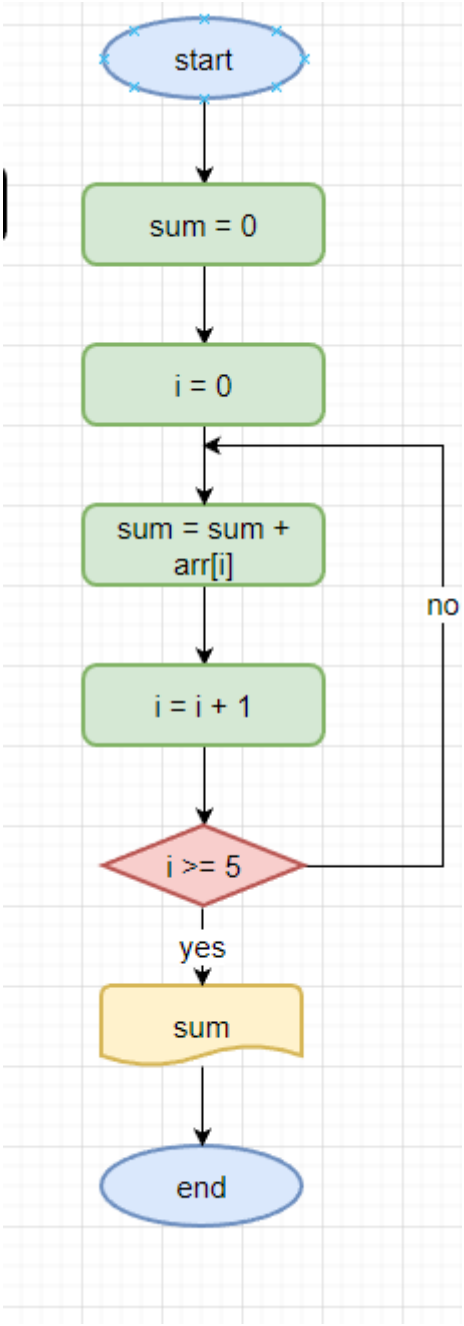


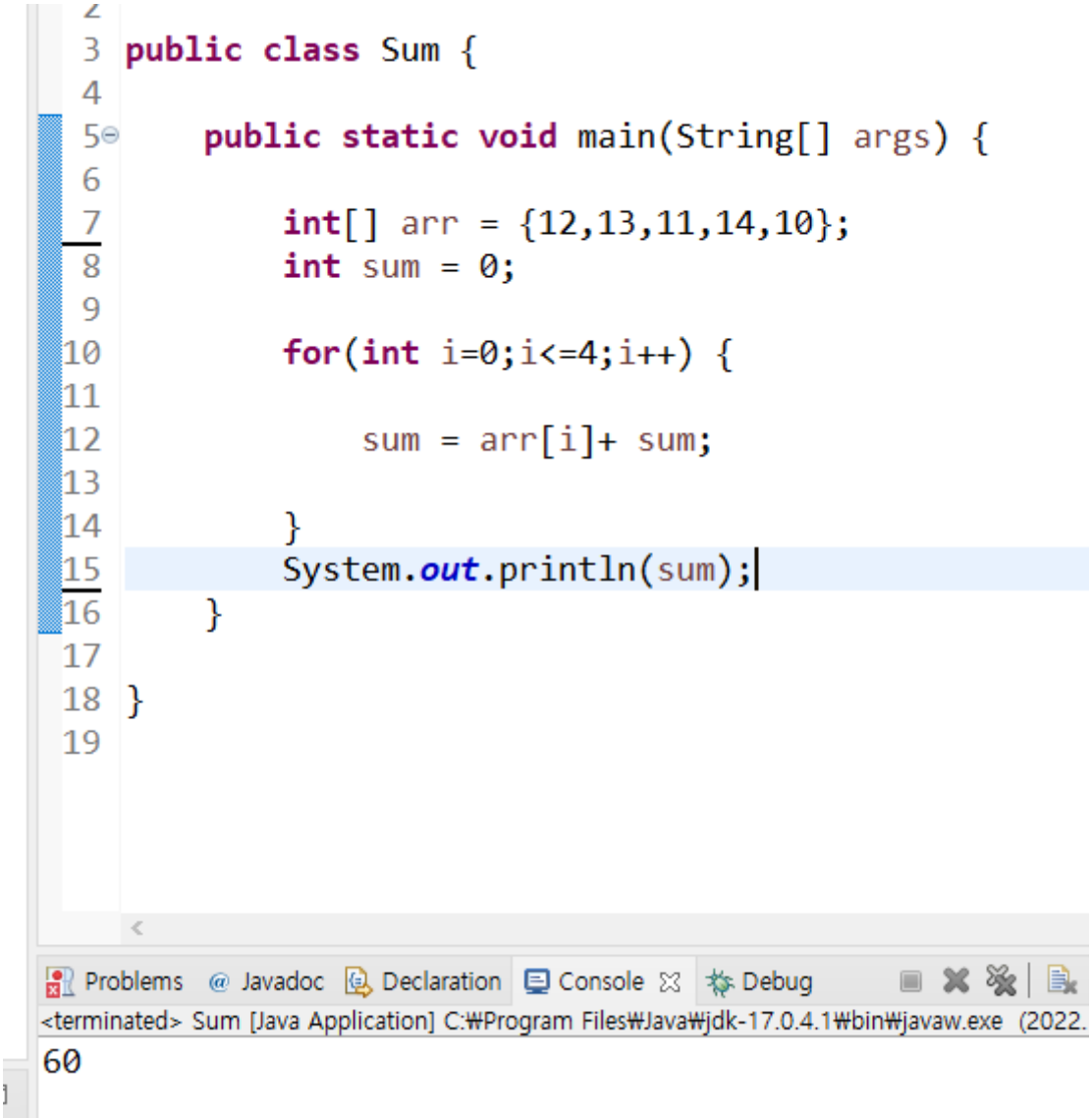

```
a와 b는 같다
```

합계값을 계산

```
12,13,11,14,10 의 합을 구하기
```

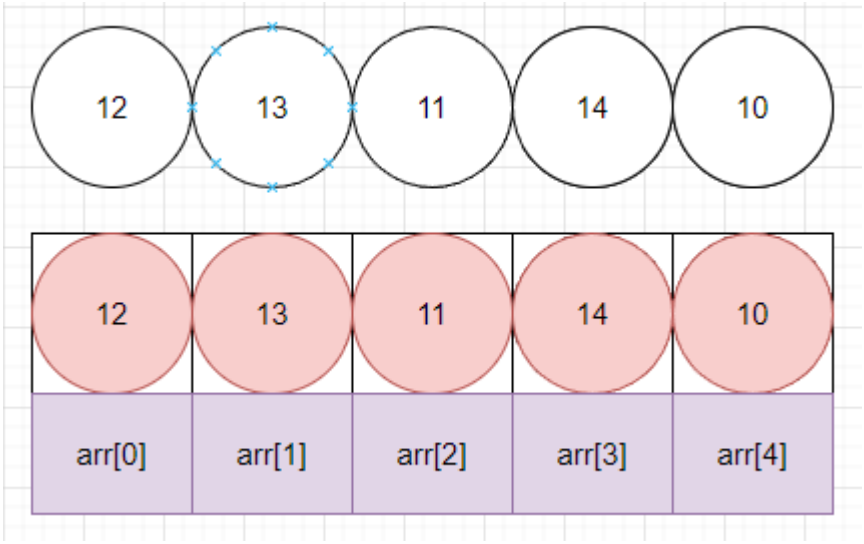

합계값을 계산
12
13
11
14
10
12
13
11
14
10
arr[0]
arr[1]
arr[2]
arr[3]
arr[4]

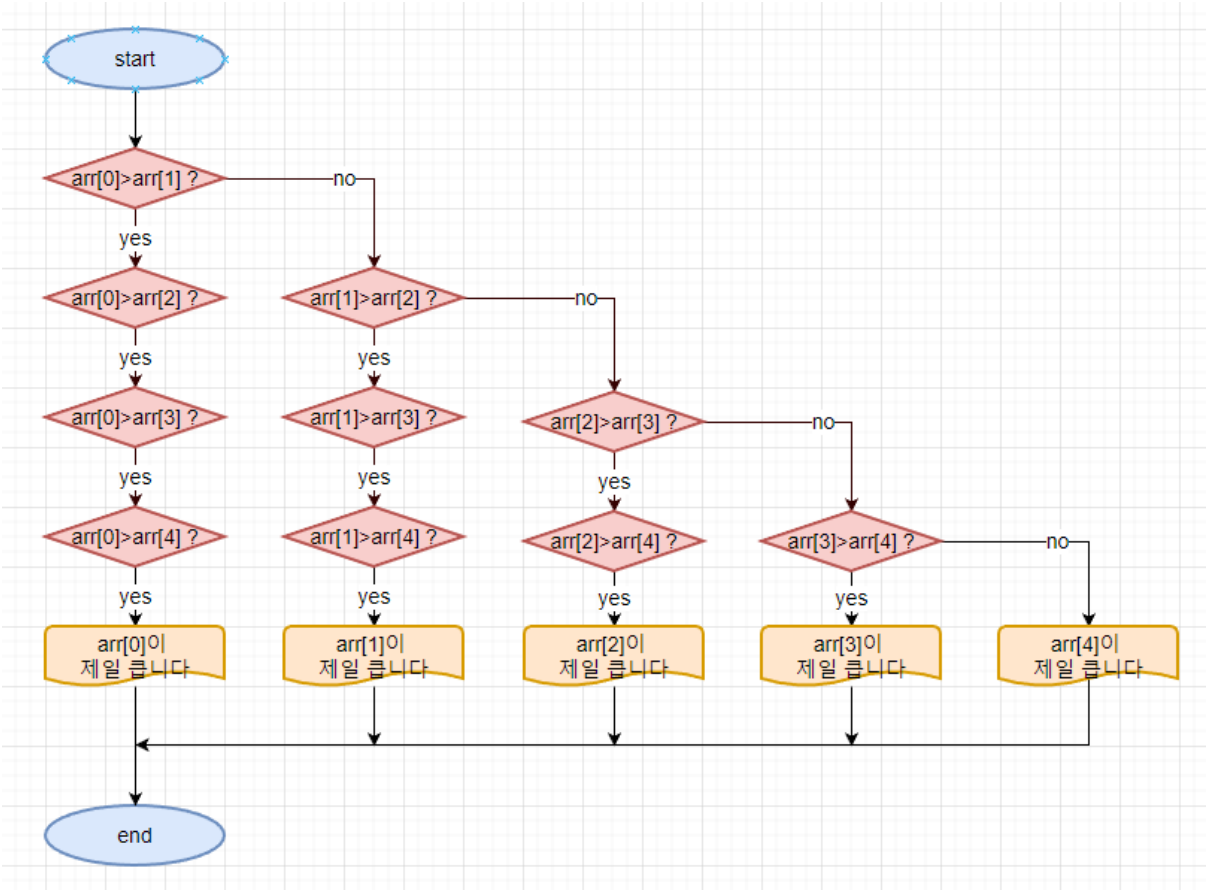

start
sum = 0
i = 0
sum = sum + arr[i]
i = i + 1
i >= 5
no
yes
sum
end

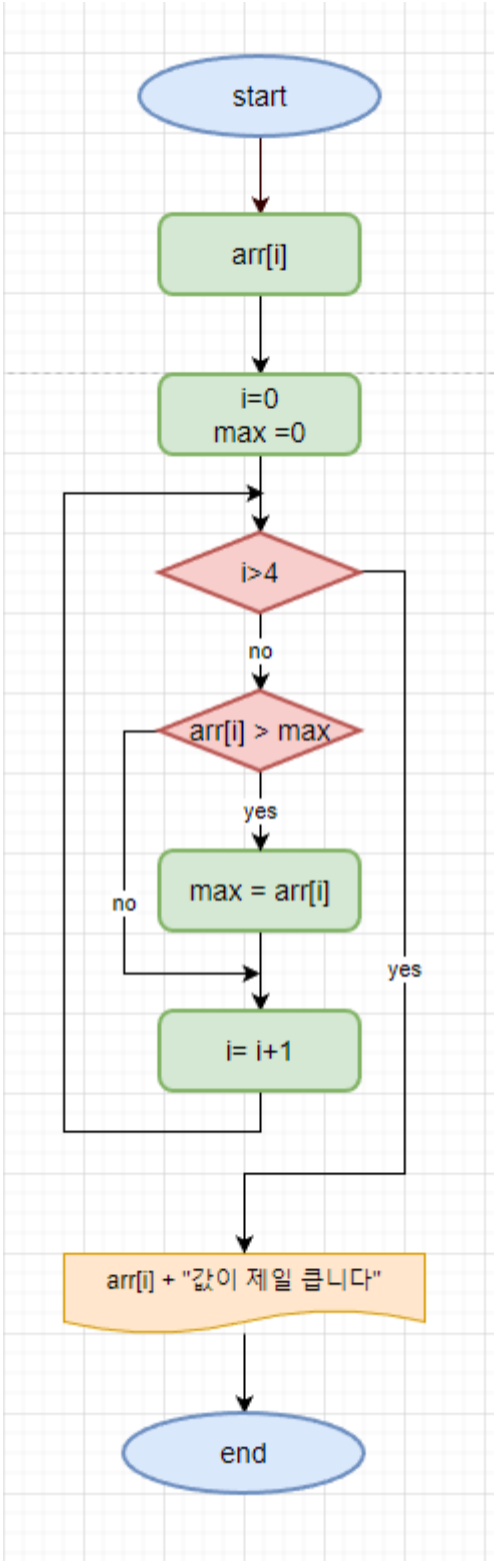


최대값을 찾는 알고리즘

```
12,13,11,14,10 중 최대값 찾기
```


12
13
11
14
10
12
13
11
14
10
arr[0]
arr[1]
arr[2]
arr[3]
arr[4]


start
arr[0]>arr[1] ?
no
yes
arr[0]>arr[2] ?
arr[1]>arr[2] ?
no
yes
yes
arr[0]>arr[3] ?
arr[1]>arr[3] ?
arr[2]>arr[3] ?
no
yes
yes
yes
arr[0]>arr[4] ?
arr[1]>arr[4] ?
arr[2]>arr[4] ?
arr[3]>arr[4] ?
no
yes
yes
yes
yes
arr[0]이
제일 큽니다
arr[1]이
제일 큽니다
arr[2]이
제일 큽니다
arr[3]이
제일 큽니다
arr[4]이
제일 큽니다
end


start
arr[i]
i=0
max =0
i>4
no
arr[i] > max
yes
max = arr[i]
no
yes
i= i+1
arr[i] + "값이 제일 큽니다"
end

알고리즘의 유형

탐색알고리즘 : 선형탐색, 이진탐색, 해시탐색

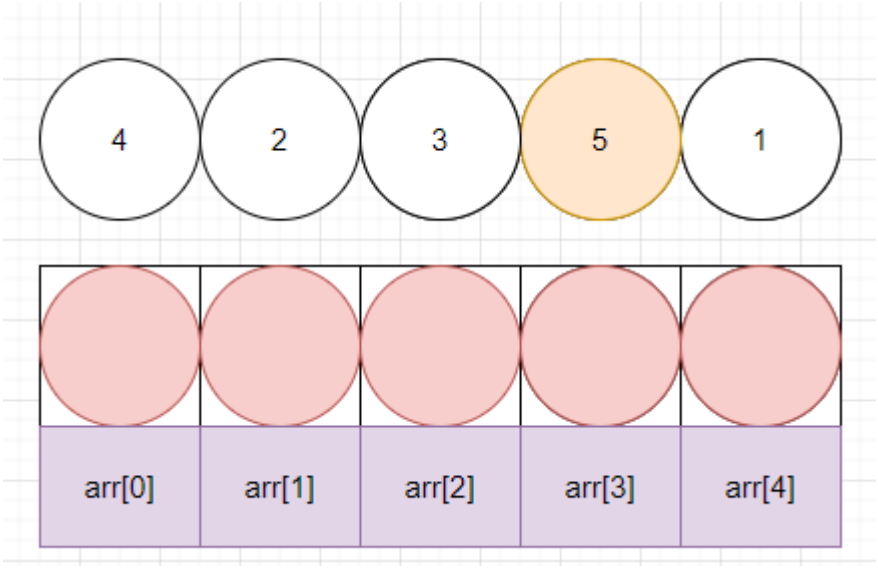
정렬 알고리즘 : 단순 정렬, 단순 교환, 단순 삽입, 퀵 정렬

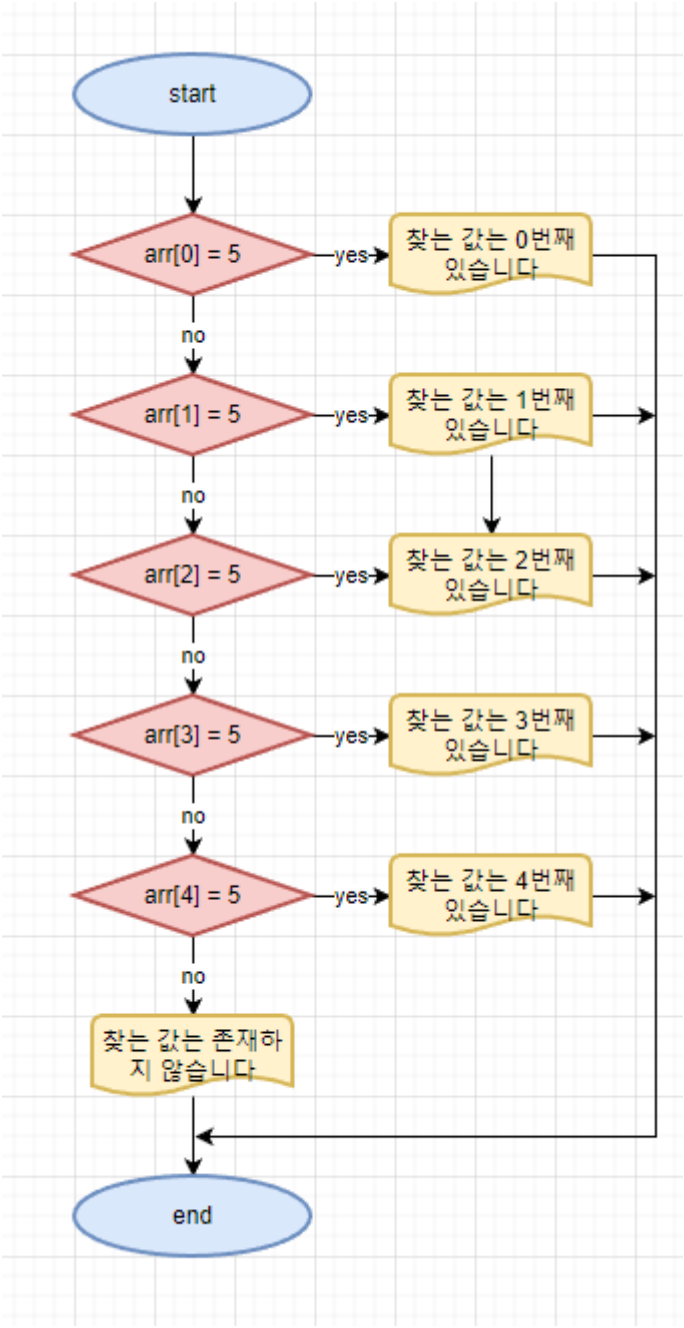
에라토스테네스의 알고리즘 : 소수 판별 알고리즘

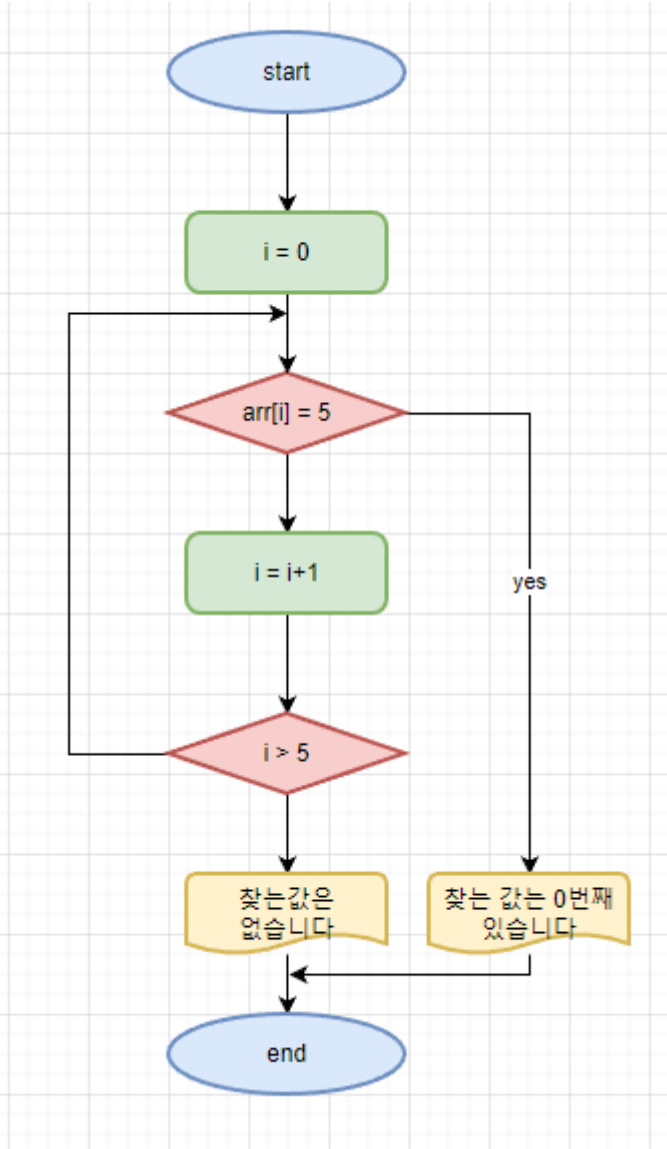
유클리드 알고리즘 : 최대 공약수 알고리즘

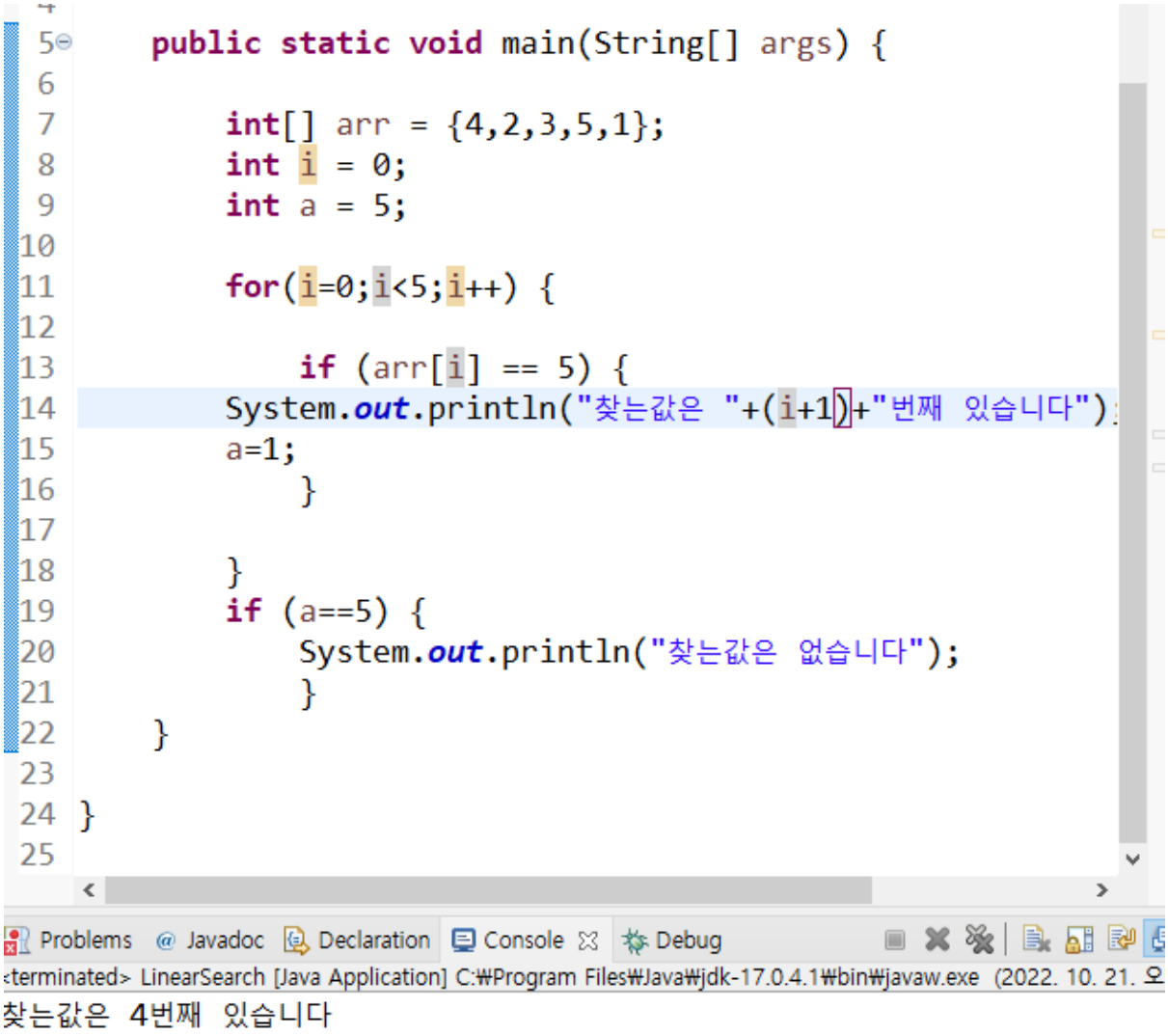
선형탐색법 (Linear Search)

```
맨 앞부터 차례로 원하는 값을 탐색 (알고리즘 자체가 단순해서 이해하기 쉬움)
```




start
arr[0] = 5
yes
찾는 값는 0번째 있습니다
no
arr[1] = 5
yes
찾는 값는 1번째 있습니다
no
arr[2] = 5
yes
찾는 값는 2번째 있습니다
no
arr[3] = 5
yes
찾는 값는 3번째 있습니다
no
arr[4] = 5
yes
찾는 값는 4번째 있습니다
no
찾는 값는 존재하지 않습니다
end


start
i = 0
arr[i] = 5
i = i+1
yes
i > 5
찾는값은
없습니다
찾는 값는 0번째
있습니다
end



public class LinearSearch {

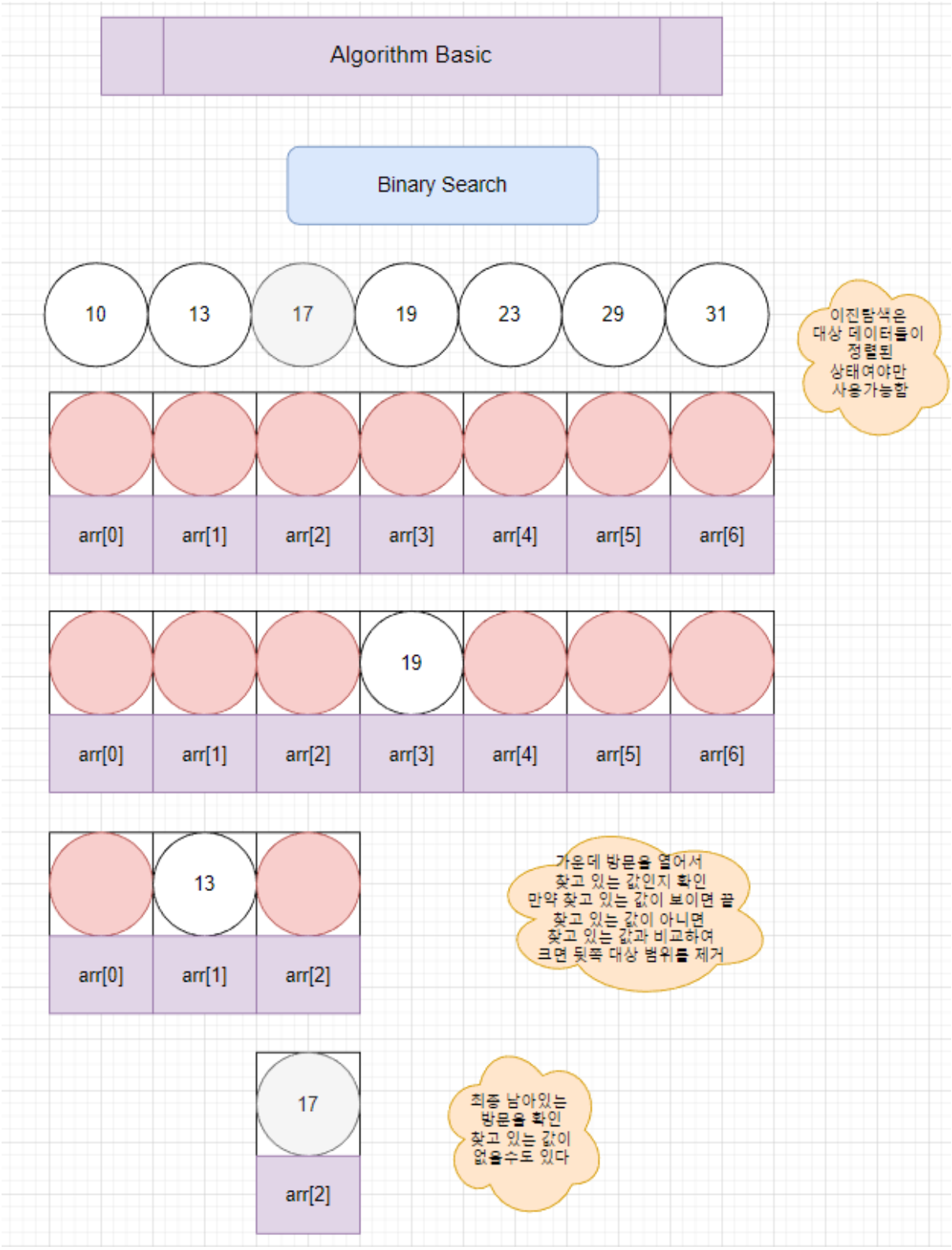
```
public static void main(String[] args) {

  int[] arr = {4,2,3,5,1};
  int i = 0;
  int a = 5;

  for(i=0;i<5;i++) {

    if (arr[i] == 5) {
  System.out.println("찾는값은 "+(i+1)+"번째 있습니다");
  a=1;
    }

  }
  if (a==5) {
    System.out.println("찾는값은 없습니다");
```

```
    }
}
```

}

이진 탐색법(Binary Search)

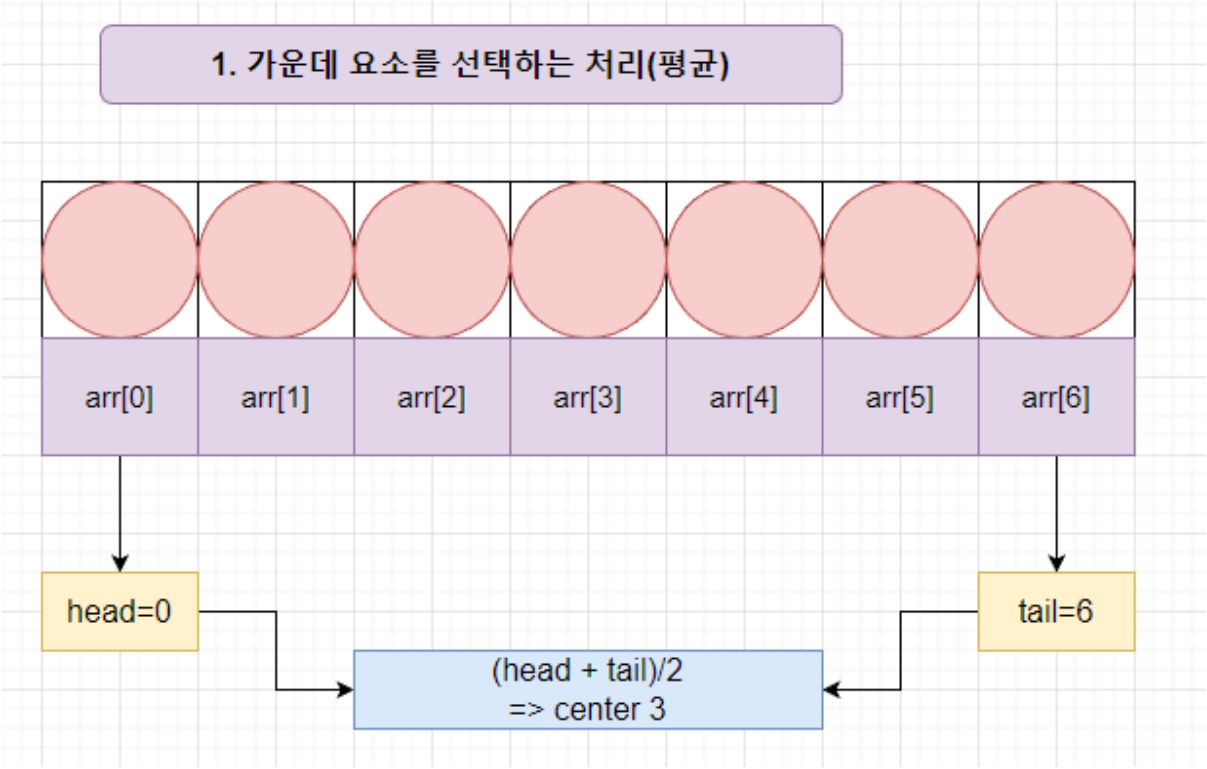
```
탐색의 범위중 중간값을 보고 해당하지 않는쪽의 절반을 줄여가며 탐색
탐색의 범위의 값들이 반드시 정렬된 상태여야만 가능
```


Algorithm Basic
Binary Search
10
13
17
19
23
29
31
이진탐색은
대상 데이터들이
정렬된
상태여야만
사용가능함
arr[0]
arr[1]
arr[2]
arr[3]
arr[4]
arr[5]
arr[6]
19
가운데 방문을 열어서
찾고 있는 값인지 확인
만약 찾고 있는 값이 보이면 끝
찾고 있는 값이 아니면
찾고 있는 값과 비교하여
크면 뒷쪽 대상 범위를 제거
13
최종 남아있는
방문을 확인
찾고 있는 값이
없을수도 있다

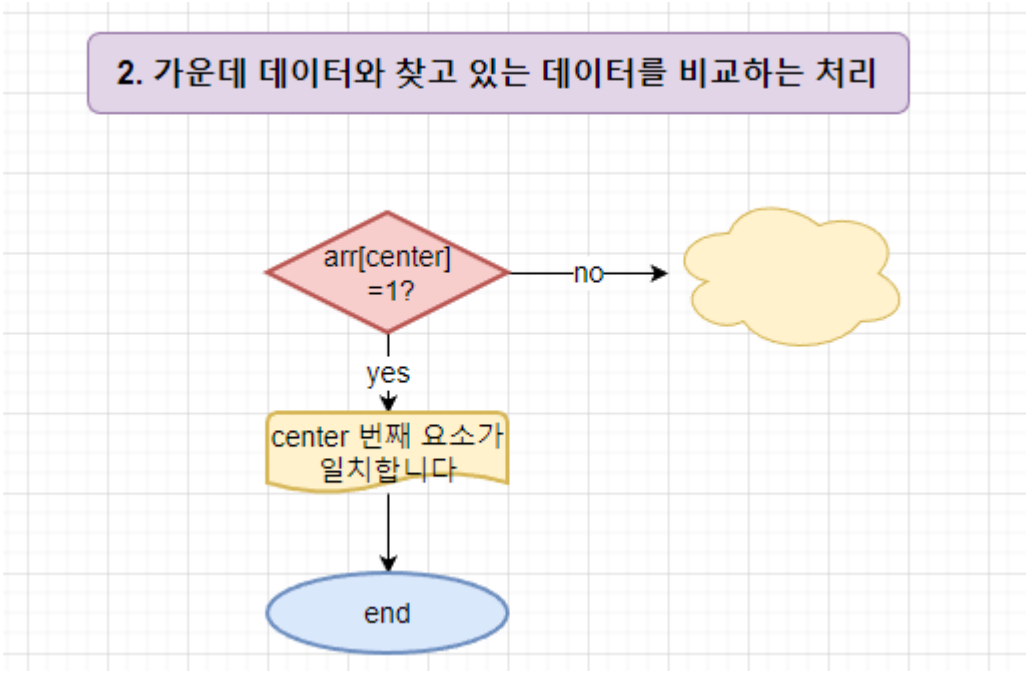
로직

1. 가운데 요소를 선택하는 처리 (평균)
2. 가운데 데이터와 찾고 있는 데이터를 비교하는 처리
3. 대상 데이터를 절반으로 줄이는 처리

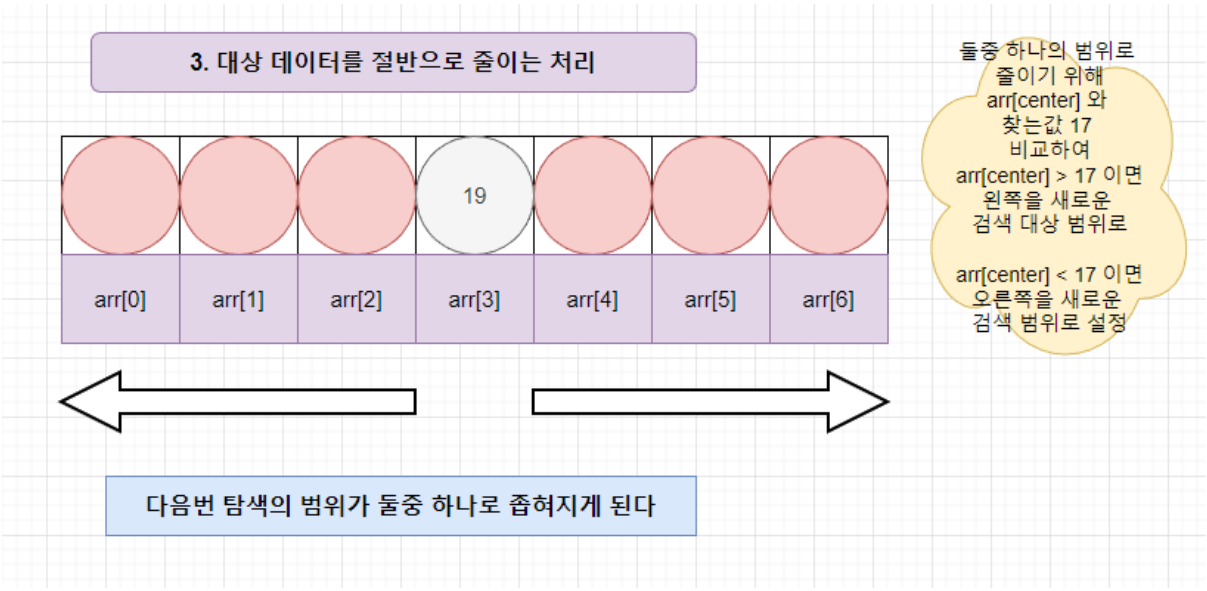
1 평균 처리



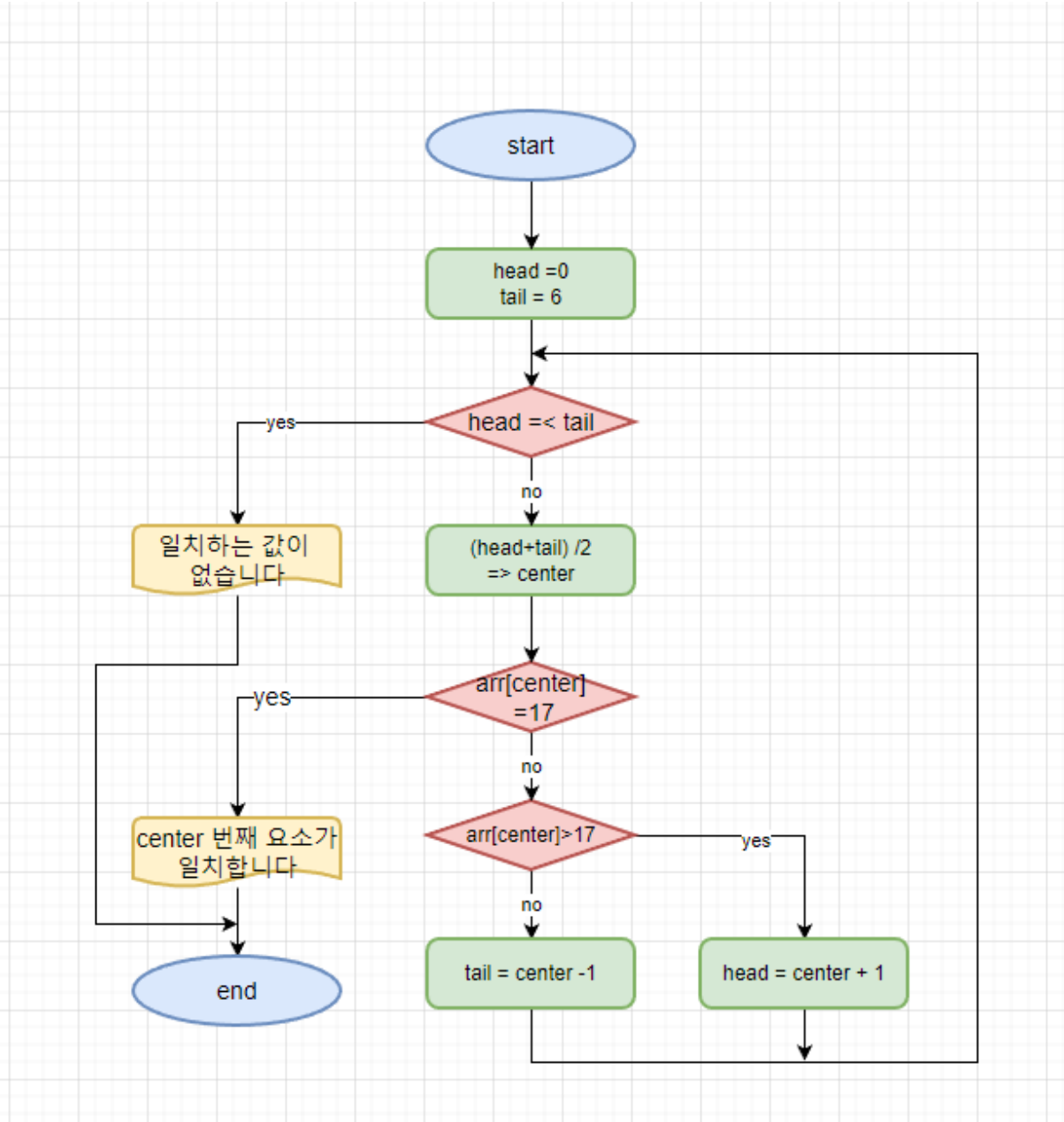
2 비교


2. 가운데 데이터와 찾고 있는 데이터를 비교하는 처리
arr[center]
=1?
no
yes
center 번째 요소가
일치합니다
end

3 절반 줄이기


3. 대상 데이터를 절반으로 줄이는 처리
19
arr[0]
arr[1]
arr[2]
arr[3]
arr[4]
arr[5]
arr[6]
둘중 하나의 범위로
줄이기 위해
arr[center] 와
찾는값 17
비교하여
arr[center] > 17 이면
왼쪽을 새로운
검색 대상 범위로
arr[center] < 17 이면
오른쪽을 새로운
검색 범위로 설정
다음번 탐색의 범위가 둘중 하나로 좁혀지게 된다

흐름도



public class BinarySearchFor {

```
public static void main(String[] args) {
  int[] arr = {11,13,17,19,23,29,31};
  int head = 0;
  int tail = 6;
  int center = 0;
  int f = 23;

  for (int i=0; i<10; i++){
  if (head>tail) {
```

```
  System.out.println("일치하는 값이 없습니다");
  break;
  }else {}

  center = (int)((head+tail)/2);

  if (arr[center] == f) {
    System.out.println((center+1)+"번째 요소가 일치합니다 "+arr[center]);
    break;}

  if (arr[center]<f) {
    head = center + 1;

  }else if (arr[center]>f) {
    tail = center -1;
  }
  }
}
```

}