

??

- [6. \[Java\] 문자열 배열 - list.toArray\(new String\[0\]\);](#)
- [7. \[Java\] int 배열의 길이 - return new int\[\]{max, idx};](#)
- [8. \[Java\] 문자열이 숫자인지 - Character.isDigit\(c\)](#)
- [9. \[Java\] 문자열이 Set에 있는지 - set.contains\(str\)](#)
- [11. \[Java\] 문자열의 길이 \(2\): case2 문자열이 더 길거나 짧거나 ?](#)

6. [Java] ?????? - list.toArray(new String[0]);

<https://school.programmers.co.kr/learn/courses/30/lessons/120897>



```
import java.util.*;

class Solution {
    public int[] solution(int n) {
        ArrayList<Integer> list = new ArrayList<>();
        for (int i = 0; i <= n; i++) {
            if(n%i==0) list.append(i);
        }
        int[] answer = list.toArray(new String[0]);
        return answer;
    }
}
```

실행 결과

```
/Solution.java:7: error: cannot find symbol
    if(n%i==0) list.append(i);
                    ^
  symbol:   method append(int)
  location: variable list of type ArrayList<Integer>
/Solution.java:9: error: incompatible types: inference variable T has incompatible bounds
    int[] answer = list.toArray(new String[0]);
                               ^
  lower bounds: int,Object
  lower bounds: String
  where T is a type-variable:
    T extends Object declared in method <T>toArray(T[])
2 errors
```

테스트 결과 (~~~)~

2개 중 0개 성공

? for (int i = 0; i <= n; i++)

- i = 0 일 때 n % 0 일 경우 ArithmeticException (0으로 나눌 수 없음) 발생
- i = 1 일 때 n % 1 일 경우 항상 0이므로 append() 호출되지 않음

? list.add(i)

- ArrayList의 add() 메서드는 append()와 동일함
- list.append(i) → list.add(i)

? int[] answer = list.toArray(new String[0]);

- List → Array 변환을 위해 toArray() 메서드를 사용함. 이때 String[]을 전달하면 String[]을 반환함
- String[]을 int[]로 변환하기 위해 for문을 사용함

import java.util.*;

```
import java.util.*;

class Solution {
    public int[] solution(int n) {
```

```

    ArrayList<Integer> list = new ArrayList<>();
    for (int i = 1; i <= n; i++) {
        if(n%i==0) list.add(i);
    }
    int[] answer = new int[list.size()];
    for(int i = 0; i < list.size(); i++) {
        answer[i] = list.get(i);
    }
    return answer;
}
}

```

Stream ?? ??

```

import java.util.*;
import java.util.stream.*;

class Solution {
    public int[] solution(int n) {
        return IntStream.rangeClosed(1, n) // 1~n 까지 모든 수
            .filter(i -> n % i == 0)      // n의 약수
            .toArray();                    // int[]로 변환
    }
}

```

List ? Array ???

List	Array
List<String>	list.toArray(new String[0])
List<Integer>	list.toArray(new Integer[0])
List<Integer> → int[]	list.stream().mapToInt().toArray()

? 1. List ? Array (???, ?: Integer, String ?)

- toArray() 返回一个 Object[] 类型的数组。

```
List<String> list = Arrays.asList("a", "b", "c");
String[] arr = list.toArray(new String[0]);
```

- list.toArray(new String[0]): 返回一个 Object[] 类型的数组。
- new String[0] 返回一个长度为 0 的 String[] 类型的数组。

```
List<Integer> list = Arrays.asList(1, 2, 3);
Integer[] arr = list.toArray(new Integer[0]);
```

? 2. List ? Array (???, ?: int)

- Java 中 List<Integer> 不能直接转换为 int[] 类型的数组。
- 需要先将 List 转换为 Object[] 类型的数组。

```
List<Integer> list = Arrays.asList(1, 2, 3);
int[] arr = new int[list.size()];

for (int i = 0; i < list.size(); i++) {
    arr[i] = list.get(i); // 将 List 中的元素复制到数组中
}
```

Array ? List ???

	返回的 List 类型
<code>String[]</code>	<code>Arrays.asList(arr)</code>
<code>Integer[]</code>	<code>Arrays.asList(arr)</code>
<code>int[]</code>	<code>Arrays.stream(arr).boxed().collect(Collectors.toList())</code>

? 1. ??? ?? ? List (?: String[], Integer[] ?)

7. [Java] ?? ? ? ??- return new int[]{max, idx};

<https://school.programmers.co.kr/learn/courses/30/lessons/120899>

□ □

```
import java.util.*;

class Solution {
    public int[] solution(int[] array) {
        Arrays.sort(array);
        int[] answer = new int[2];
        answer[0] = array[array.length-1];
        answer[1] = array.length-1;
        return answer;
    }
}
```

실행 결과

테스트 1

입력값 > [1, 8, 3]
기댓값 > [8, 1]
실행 결과 > 실행한 결과값 [8,2]이 기댓값 [8,1]과 다릅니다.

테스트 2

입력값 > [9, 10, 11, 8]
기댓값 > [11, 2]
실행 결과 > 실행한 결과값 [11,3]이 기댓값 [11,2]과 다릅니다.

테스트 결과 (~▽~)~

2개 중 0개 성공

? index ?? ???

- array[array.length-1] 是数组的最后一个元素，array.length-1 是它的索引。
- 我们遍历数组，找到最大值和它的索引。
- 我们初始化 max 为 array[0]，idx 为 0。然后遍历数组，如果当前元素大于 max，我们就更新 max 和 idx。

Java 实现

```
import java.util.*;

class Solution {
    public int[] solution(int[] array) {
        int max = array[0];
        int idx = 0;

        for(int i = 1; i < array.length; i++) {
            if (array[i] > max) {
                max = array[i];
                idx = i;
            }
        }
        return new int[]{max, idx};
    }
}
```

Python 实现

int 是 Python 的整数类型，array 是 Python 的数组类型。

```
return new int[]{max, idx};
```

我们初始化 max 为 array[0]，idx 为 0。然后遍历数组，如果当前元素大于 max，我们就更新 max 和 idx。

```
int[] result = new int[]{max, idx};
return result;
```


Java에서 return은 메소드가 호출된 후, 호출한 곳으로 데이터를 반환하는 키워드입니다. 예를 들어, int[] result = ...과 같이 선언된 배열을 반환할 수 있습니다.

다음은 배열을 반환하는 메소드의 예시입니다.

```
int[] a;  
a = {1, 2}; // 배열 초기화
```

→ 배열을 반환하는 메소드.

```
int[] a = new int[]{1, 2};
```

8. [Java] ??? ????? - Character.isDigit(c)

<https://school.programmers.co.kr/learn/courses/30/lessons/120902>

?? ?? (1? ??)

```
class Solution {
    public int solution(String my_string) {
        char[] arr = my_string.toCharArray();
        int sum = 0;
        for (char c : arr ) {
            if (c != '+' && c != '-' && c != ' ') {
                sum+= Integer.parseInt(c);
            }
        }
        return sum;
    }
}
```

실행 결과

```
/Solution.java:7: error: incompatible types: char cannot be converted to String
    sum+= Integer.parseInt(c);
                        ^
```

Note: Some messages have been simplified; recompile with -Xdiags:verbose to get full output
1 error

테스트 결과 (~~~)~

1개 중 0개 성공

? Integer.parseInt()

- String → int → ??? → ??? → ?? → ??? → ?? .
- char → ??? → ??? → ?? Integer.parseInt() → ?? char → ?? → - '0' → ?? . (ASCII ??)

String → int VS char → int

1. String → int Integer.parseInt()

```
String str = "123";

// String → int
int num = Integer.parseInt(str);    // 123
```

2. char → int char - '0'

```
char c = '7';

// char → int (int)
int digit = c - '0';    // '7' - '0' = 7
```

2. 2 (2 2)

```
class Solution {
    public int solution(String my_string) {
        char[] arr = my_string.toCharArray();
        int sum = 0;
        for (char c : arr ) {
            if (c != '+' && c != '-' && c != ' ') {
                sum+= c - '0';
            }
        }
        return sum;
    }
}
```

실행 결과

채점을 시작합니다.

정확성 테스트

```
테스트 1 > 실패 (0.03ms, 98.6MB)
테스트 2 > 실패 (0.03ms, 86.9MB)
테스트 3 > 실패 (0.04ms, 72.1MB)
테스트 4 > 실패 (0.02ms, 83.6MB)
테스트 5 > 실패 (0.03ms, 75MB)
테스트 6 > 실패 (0.04ms, 77.7MB)
테스트 7 > 실패 (0.03ms, 75.8MB)
테스트 8 > 통과 (0.03ms, 72.3MB)
테스트 9 > 실패 (0.02ms, 89.5MB)
테스트 10 > 실패 (0.02ms, 82.1MB)
```

채점 결과

정확성: 10.0

합계: 10.0 / 100.0

문자열을 배열로 변환하여, "12"를 배열로 변환하여 1 + 2 = 3을 출력합니다.

```
String my_string = "a12b3";
→ arr = ['a', '1', '2', 'b', '3']
→ sum: 1 + 2 + 3 = 6 (문자열 12 + 3을 합산)
```

문자열 "12 + 3" → 15를 출력합니다.

String num을 배열로 변환하여, Character.isDigit()을 사용하여 c가 0~9인지 확인하고 num을 출력합니다.

문자열 (3을 출력)

```
class Solution {
    public int solution(String my_string) {
        char[] arr = my_string.toCharArray();
        int sum = 0;
        String num = "";
        for (char c : arr) {
```

```

        if (Character.isDigit(c)) {
            num += c;
        } else {
            if (!num.isEmpty()) {
                sum += Integer.parseInt(num);
                num = "";
            }
        }
    }

    if (!num.isEmpty()) {
        sum += Integer.parseInt(num);
    }

    return sum;
}

```

$\square\square\square \quad \square\square\square \quad \square\square\square \quad \square\square\square \quad \square\square\square \quad . \quad \square\square\square \quad (+, -) \quad \square \quad \square\square \quad \square\square\square \quad \square\square \quad \square\square\square\square \quad \square\square\square \quad \square\square \quad \square\square \quad .$

4. 문자열이 "+", "-"로 구성된 수식 문자열이 주어질 때, 수식을 계산하여 결과를 반환하는 프로그램을 작성하시오.

문자열이 "+", "-"로 구성된 수식 문자열이 주어질 때, 수식을 계산하여 결과를 반환하는 프로그램을 작성하시오. (4점)

```
class Solution {
    public int solution(String my_string) {
        String[] splited = my_string.split(" ");
        int answer = Integer.parseInt(splited[0]);

        for(int i = 1; i < splited.length; i += 2) {
            String op = splited[i];
            int num = Integer.parseInt(splited[i + 1]);

            if (op == '+') {
                answer += num;
            } else if (op == '-') {
                answer -= num;
            }
        }
        return answer;
    }
}
```

? if(op == '+')

- op 문자열이 "+"일 때, answer에 num을 더한다.
- op 문자열이 "-"일 때, answer에 num을 빼준다.
- Java에서 String을 비교할 때는 op.equals(" ")를 사용한다. op == " "은 String과 String을 비교하는 것이 아니라, op의 주소값과 " "의 주소값을 비교하는 것이다. 따라서 op.equals(" ")를 사용한다.
- op 문자열이 "+"일 때, answer에 num을 더한다.

문자열이 "+", "-"로 구성된 수식 문자열이 주어질 때, 수식을 계산하여 결과를 반환하는 프로그램을 작성하시오. (4점)

```
class Solution {
    public int solution(String my_string) {
        String[] splited = my_string.split(" ");
        int answer = Integer.parseInt(splited[0]);
```

```
for(int i = 1; i < splited.length; i += 2) {  
    String op = splited[i];  
    int num = Integer.parseInt(splited[i + 1]);  
  
    if (op.equals("+")) {  
        answer += num;  
    } else if (op.equals("-")) {  
        answer -= num;  
    }  
}  
return answer;  
}  
}
```

9. [Java] ??? ???(??, Set) - set.contains(str)

<https://school.programmers.co.kr/learn/courses/30/lessons/120903>

Set을 이용하여 풀이

```
import java.util.*;

class Solution {
    public int solution(String[] s1, String[] s2) {
        Set<String> set = new HashSet<>(Arrays.asList(s1));
        int cnt = 0;
        for (String str : s2) {
            if (set.contains(str)) cnt++;
        }
        return cnt;
    }
}
```

? Set<String> set = new HashSet<>(Arrays.asList(s1));

- Set은 중복을 허용하지 않는 집합, 즉 같은 원소가 여러 개 들어갈 수 없습니다. s1에 Set을 생성하고 s2에 대해 s1에 있는지 확인합니다.

for문 (중첩)을 이용하여 풀이

```
class Solution {
    public int solution(String[] s1, String[] s2) {
        int cnt = 0;
        for (int i = 0; i < s1.length; i++) {
            for (int j = 0; j < s2.length; j++) {
```



```

        if (s1[i].equals(s2[j])) {
            cnt++;
            break;
        }
    }
}
return cnt;
}
}

```

- 在 循环 中 使用 break; 来 提前 结束 循环。
- 在 循环 中 使用 s1[i].equals(s2[j]) 来 判断 两个 字符串 是否 相等 (== 是 比较 地址)

11. [Java] ?????(2): case2? ? longer - shorter???

1. ?? ??

- <https://school.programmers.co.kr/learn/courses/30/lessons/120868>
- [longer] [shorter] [x] (2)
- [a], [b] [x] [a+b] [a-b] [a-b-1] [a+b-1]
- [a] [b] [x] : [a] [b] < [a] [b]

2. ?? ?? ??

- $x \in (\text{longer} - \text{shorter}, \text{longer} + \text{shorter})$
- [a], [b] [x] [a+b]
- $\text{longer} - \text{shorter} + 1 \leq x \leq \text{longer} + \text{shorter} - 1$

??

```
import java.util.*;

class Solution {
    public int solution(int[] sides) {
        Arrays.sort(sides); // [shorter] [longer]
        int shorter = sides[0];
        int longer = sides[1];

        // case1: x [shorter] [longer] -> x < a + b
        int case1 = longer + shorter - 1;

        // case2: x [longer] [shorter] -> x > max - min
        int case2 = longer - shorter;

        return case1 - case2;
    }
}
```

```
}
```

3. ? case2 = longer - shorter???

- case2 是 1 的 数量
- 1 的 数量 longer - shorter + 1
- 1 的 数量 1 的 数量 - 1 的 数量 1 的 数量
- case2 = longer - shorter 1 的 数量 1 的 数量

4. ??

- sides = [3, 6]
- x: 4, 5, 6, 7, 8 → 5
- case1 = 3 + 6 - 1 = 8
- case2 = 6 - 3 = 3
- answer = 8 - 3 = 5

??

- case2 是 1 的 数量
- 1 的 数量 1 的 数量 + 1 的 数量 - 1 的 数量
- (shorter + longer - 1) - (longer - shorter)