PART 7 연구과제

369게임 (Level.0)

https://school.programmers.co.kr/learn/courses/30/lessons/120891?language=java

📌 문제 해결 개요

머쓱이가 369게임을 진행하며 특정 숫자가 포함된 경우 박수를 쳐야 하는 횟수를 구하는 문제이다. 주어진 숫자의 각 자리에서 3, 6, 9가 몇 번 등장하는지를 세면 된다.

🔷 핵심 개념

- 숫자의 각 자리수를 분석하기 위해 문자열로 변환
- 특정 숫자(3, 6, 9)가 등장하는 횟수 세기
- 최종적으로 3, 6, 9의 등장 횟수를 반환

🚀 해결 방법

1. 입력된 숫자를 문자열로 변환한다.
2. 문자열을 순회하면서 각 자리의 숫자를 확인한다.
3. 현재 숫자가 '3', '6', '9' 중 하나라면 카운트를 증가시킨다.
4. 최종적으로 3, 6, 9의 등장 횟수를 반환한다.

```
public class Solution {
    public int solution(int order) {
        int answer = 0;  // 박수를 칠 횟수를 저장할 변수

        // 숫자를 문자열로 변환하여 각 자리수를 분석
        String orderStr = String.valueOf(order);

        // 문자열을 순회하면서 각 자리의 숫자를 확인
        for (char c : orderStr.toCharArray()) {
            // '3', '6', '9' 중 하나라면 카운트 증가
            if (c == '3' || c == '6' || c == '9') {
                answer++;
            }
        }

        return answer;  // 최종적으로 박수를 칠 횟수 반환
    }

    // 테스트 코드
    public static void main(String[] args) {
        Solution sol = new Solution();
        System.out.println(sol.solution(3));      // 1 (3이 포함됨)
        System.out.println(sol.solution(29423));  // 2 (9와 3이 포함됨)
```

```
    }
}
```

7의 개수 (Level.0)

https://school.programmers.co.kr/learn/courses/30/lessons/120912?language=java

📌 문제 해결 개요

주어진 정수 배열에서 숫자 7이 등장하는 횟수를 세는 문제이다. 각 숫자를 문자열로 변환한 후 '7'이 포함된 개수를 누적해서 계산하면 된다.

🔷 핵심 개념

- 배열 내 모든 숫자를 개별적으로 확인
- 숫자를 문자열로 변환하여 특정 문자('7')를 찾기
- 등장 횟수를 누적하여 반환

🚀 해결 방법

1. 배열의 각 원소를 확인한다.
2. 각 숫자를 문자열로 변환한다.
3. 변환된 문자열을 순회하면서 '7'의 개수를 센다.
4. 최종적으로 7이 등장한 횟수를 반환한다.

```
import java.util.*;

public class Solution {
    public int solution(int[] array) {
        int count = 0;  // 숫자 '7'의 개수를 저장할 변수

        // 배열의 모든 요소를 확인
        for (int num : array) {
            // 숫자를 문자열로 변환하여 '7'의 개수를 세기
            String numStr = String.valueOf(num);

            // 문자열을 순회하면서 '7'이 몇 번 등장하는지 확인
            for (char c : numStr.toCharArray()) {
                // 문자가 '7'이면 카운트 증가
                if (c == '7') {
                    count++;
                }
            }
        }

        return count;  // 최종적으로 7이 등장한 횟수 반환
    }

    // 테스트 코드
```

```
    public static void main(String[] args) {
        Solution sol = new Solution();
        System.out.println(sol.solution(new int[]{7, 77, 17}));  // 4 ('7'이 4
번 등장)
        System.out.println(sol.solution(new int[]{10, 29}));     // 0 ('7'이 없
음)
    }
}
```

대문자와 소문자 (Level.0)

https://school.programmers.co.kr/learn/courses/30/lessons/120893?language=java

📌 문제 해결 개요

주어진 문자열에서 대문자는 소문자로, 소문자는 대문자로 변환하는 문제이다. 이를 위해 각 문자를 하나씩 확인하여 대소문자를 변환하면 된다.

🔷 핵심 개념

- 문자열을 순회하면서 각 문자의 대소문자를 판별
- 대문자는 소문자로 변환, 소문자는 대문자로 변환
- 변환된 문자들을 문자열에 추가한 후 최종 문자열 반환

🚀 해결 방법

1. 변환된 문자열을 저장할 빈 문자열 `result`를 선언한다.
2. 문자열의 각 문자를 확인하며, 대문자인 경우 소문자로, 소문자인 경우 대문자로 변환한다.
3. 변환된 문자를 기존 문자열에 추가한다.
4. 최종적으로 변환된 문자열을 반환한다.

```
public class Solution {
    public String solution(String myString) {
        String result = "";  // 변환된 문자열을 저장할 변수

        // 문자열의 각 문자를 순회하며 변환
        for (char c : myString.toCharArray()) {
            if (Character.isUpperCase(c)) {
                // 현재 문자가 대문자인 경우 소문자로 변환하여 추가
                result += Character.toLowerCase(c);
            } else {
                // 현재 문자가 소문자인 경우 대문자로 변환하여 추가
                result += Character.toUpperCase(c);
            }
        }

        return result;  // 변환된 문자열 반환
    }
```

```
    // 테스트 코드
    public static void main(String[] args) {
        Solution sol = new Solution();
        System.out.println(sol.solution("cccCCC"));       // CCCccc
        System.out.println(sol.solution("abCdEfghIJ"));   // ABcDeFGHij
    }
}
```

문자 반복 출력하기 (Level.0)

https://school.programmers.co.kr/learn/courses/30/lessons/120825?language=java

📌 문제 해결 개요

주어진 문자열에서 각 문자를 n번 반복하여 새로운 문자열을 만드는 문제이다.

🔷 핵심 개념

- `substring()`을 사용하여 문자열에서 한 글자씩 추출
- 각 문자를 n번 반복하여 새로운 문자열을 생성

🚀 해결 방법

1. 변환된 문자열을 저장할 빈 문자열 `result`를 선언한다.
2. `for` 문을 이용해 문자열의 길이만큼 반복하며, `substring()`을 이용해 한 글자씩 추출한다.
3. 내부 루프를 사용하여 현재 문자를 n번 반복하여 `result`에 추가한다.
4. 최종적으로 변환된 문자열을 반환한다.

```
public class Solution {
    public String solution(String myString, int n) {
        String result = "";  // 결과를 저장할 문자열 초기화

        // myString의 각 문자를 순회
        for (int i = 0; i < myString.length(); i++) {
            // 현재 문자 추출
            String currentChar = myString.substring(i, i + 1);

            // 현재 문자를 n번 반복하여 추가
            for (int j = 0; j < n; j++) {
                result += currentChar;
            }
        }

        return result;  // 최종 변환된 문자열 반환
    }

    // 테스트 코드
    public static void main(String[] args) {
        Solution sol = new Solution();
```

```
        System.out.println(sol.solution("hello", 3));  // "hhheeellllllooo"
    }
}
```

문자열 계산하기 (Level.0)

https://school.programmers.co.kr/learn/courses/30/lessons/120902?language=java

📌 문제 해결 개요

문자열 형태의 수식을 계산하는 문제이다. 수식은 공백을 기준으로 연산자와 숫자가 구분되므로, 이를 파싱하여 연산을 수행하면 된다.

🔷 핵심 개념

- `split(" ")`을 사용하여 문자열을 공백 기준으로 나누기
- 첫 번째 숫자를 초기값으로 설정
- 연산자와 숫자를 순차적으로 탐색하며 계산 수행

🚀 해결 방법

1. `split(" ")`을 사용하여 문자열을 공백 기준으로 나눈다.
2. 첫 번째 숫자를 정수로 변환하여 결과 변수(`answer`)에 저장한다.
3. `for` 문을 이용해 리스트를 순회하면서 연산자와 숫자를 차례대로 처리한다.
4. "+" 연산자면 더하고, "-" 연산자면 뺀다.
5. 최종적으로 계산된 결과를 반환한다.

```
public class Solution {
    public int solution(String myString) {
        // 문자열을 공백 기준으로 분리하여 배열로 변환
        String[] myList = myString.split(" ");

        // 첫 번째 숫자를 정수로 변환하여 초기값으로 설정
        int answer = Integer.parseInt(myList[0]);

        // 1부터 2씩 증가하면서 연산자와 숫자를 순차적으로 처리
        for (int i = 1; i < myList.length - 1; i += 2) {
            // 연산자 추출
            String operator = myList[i];

            // 숫자 추출
            int number = Integer.parseInt(myList[i + 1]);

            if (operator.equals("+")) {
                // "+" 연산자이면 다음 숫자를 더함
                answer += number;
            } else {
                // "-" 연산자이면 다음 숫자를 뺌
                answer -= number;
```

```
            }
        }

        return answer;  // 계산된 최종 결과 반환
    }

    // 테스트 코드
    public static void main(String[] args) {
        Solution sol = new Solution();
        System.out.println(sol.solution("3 + 4"));           // 7
        System.out.println(sol.solution("5 + 7 - 2 - 5 + 10"));  // 15
    }
}
```

문자열 뒤집기 (Level.0)

https://school.programmers.co.kr/learn/courses/30/lessons/120822?language=java

📌 문제 해결 개요

이 문제는 문자열을 거꾸로 뒤집는 기능을 구현하는 문제입니다. 문자열을 리스트로 변환한 후, 양 끝에서 문자를 교환하면서 반전된 문자열을 생성합니다.

🔷 핵심 개념

- 문자열을 리스트로 변환하여 문자 단위로 조작 가능하게 만들기
- 리스트의 앞과 뒤 요소를 교환하는 방식으로 문자열 뒤집기
- 리스트를 다시 문자열로 변환하여 반환하기

🚀 해결 방법

1. 문자열을 리스트로 변환하여 각 문자를 개별적으로 조작할 수 있도록 합니다.
2. 리스트의 길이를 구하고, 절반 길이만큼 반복하여 양 끝의 문자를 교환합니다.
3. 변경된 리스트를 다시 문자열로 변환하여 반환합니다.

```
public class Solution {
    public String solution(String myString) {
        // 문자열을 문자 배열로 변환하여 저장
        char[] answer = myString.toCharArray();

        // 문자열의 길이를 구함
        int n = answer.length;

        // 문자열의 절반 길이만큼 반복하면서 앞뒤 문자를 교환
        for (int i = 0; i < n / 2; i++) {
            int p = n - i - 1;      // 뒤에서부터의 인덱스를 계산
            char temp = answer[i];  // 현재 문자를 임시 저장
            answer[i] = answer[p];  // 뒤쪽 문자를 앞쪽으로 이동
            answer[p] = temp;       // 임시 저장한 문자를 뒤쪽으로 이동
```

```
        }

        // 문자 배열을 다시 문자열로 변환하여 반환
        return new String(answer);
    }

    // 테스트 예제 실행
    public static void main(String[] args) {
        Solution sol = new Solution();
        System.out.println(sol.solution("jaron")); // "noraj"
        System.out.println(sol.solution("bread")); // "daerb"
    }
}
```

문자열 밀기 (Level.0)

https://school.programmers.co.kr/learn/courses/30/lessons/120921?language=java

📌 문제 해결 개요

이 문제는 문자열을 오른쪽으로 회전시켜 목표 문자열과 일치하는지 확인하는 문제입니다. 원본 문자열을 두 번 이어붙여 검색하는 방법을 사용하여 최소 이동 횟수를 구합니다.

🔷 핵심 개념

- 문자열을 오른쪽으로 회전하는 방법 이해하기
- 원본 문자열을 두 번 이어붙여 포함 여부를 확인하는 기법 활용
- `indexOf()`를 사용하여 문자열의 첫 등장 위치 찾기

🚀 해결 방법

1. 문자열 B를 두 번 이어붙여 새로운 문자열을 생성합니다.
2. 이 새로운 문자열에서 A가 포함되어 있는지 확인합니다.
3. 포함되어 있다면 A가 처음 등장하는 위치를 반환합니다.
4. 포함되어 있지 않다면 -1을 반환합니다.

```
public class Solution {
    public int solution(String A, String B) {
        // B 문자열을 두 번 이어붙여 새로운 문자열 생성
        String doubledB = B + B;

        // A가 처음 등장하는 위치 찾기
        int index = doubledB.indexOf(A);
        if (index != -1) {
            // A가 포함되어 있으면 해당 위치 반환
            return index;
        } else {
            // A가 포함되지 않으면 -1 반환
            return -1;
```

```
        }
    }

    // 테스트 예제 실행
    public static void main(String[] args) {
        Solution sol = new Solution();
        System.out.println(sol.solution("hello", "ohell")); // 1
        System.out.println(sol.solution("apple", "elppa")); // -1
        System.out.println(sol.solution("atat", "tata")); // 1
        System.out.println(sol.solution("abc", "abc")); // 0
    }
}
```

문자열안에 문자열 (Level.0)

https://school.programmers.co.kr/learn/courses/30/lessons/120908?language=java

📌 문제 해결 개요

이 문제는 문자열 내에서 특정 문자열이 포함되어 있는지를 확인하는 문제입니다. `indexOf()` 메서드를 사용하여 str2가 str1에 포함되는지를 판별합니다.

🔷 핵심 개념

- 문자열 검색 (`indexOf()` 메서드 활용)
- `indexOf()`가 -1을 반환하는 경우를 이용한 조건 판별

🚀 해결 방법

1. `indexOf()` 메서드를 사용하여 str1에서 str2의 위치를 찾습니다.
2. `indexOf()`의 결과가 -1보다 크면 1을 반환하고, -1이면 2를 반환합니다.

```
public class Solution {
    public int solution(String str1, String str2) {
        // str1에서 str2를 찾았을 때
        if (str1.indexOf(str2) != -1) {
            return 1; // 포함되면 1 반환
        } else {
            return 2; // 포함되지 않으면 2 반환
        }
    }

    // 테스트 예제 실행
    public static void main(String[] args) {
        Solution sol = new Solution();
        System.out.println(sol.solution("ab6CDE443fgh22iJKlmn1o", "6CD")); // 1
        System.out.println(sol.solution("ppprrrogrammers", "pppp")); // 2
        System.out.println(sol.solution("AbcAbcA", "AAA")); // 2
```

```
    }
}
```

숨어있는 숫자의 덧셈 (1) (Level.0)

https://school.programmers.co.kr/learn/courses/30/lessons/120851?language=java

📌 문제 해결 개요

이 문제는 문자열에서 숫자만 추출하여 합을 구하는 문제입니다. 각 문자를 확인하여 숫자인 경우 더하는 방식으로 해결합니다.

🔷 핵심 개념

- 문자열 순회하며 숫자인지 판별 (`Character.isDigit()` 활용)
- 숫자인 경우 정수로 변환 후 누적 합산
- 최종적으로 합산된 값을 반환

🚀 해결 방법

1. 문자열을 순회하며 각 문자가 숫자인지 확인합니다.
2. 숫자인 경우 정수로 변환하여 누적 합산합니다.
3. 최종 합산된 값을 반환합니다.

```
public class Solution {
    public int solution(String myString) {
        int answer = 0; // 숫자의 합을 저장할 변수

        // 문자열의 각 문자를 순회
        for (char m : myString.toCharArray()) {
            // 숫자인 경우
            if (Character.isDigit(m)) {
                // 정수로 변환하여 더함
                answer += Character.getNumericValue(m);
            }
        }

        return answer; // 최종 결과 반환
    }

    // 테스트 예제 실행
    public static void main(String[] args) {
        Solution sol = new Solution();
        System.out.println(sol.solution("aAb1B2cC34oOp")); // 10
        System.out.println(sol.solution("1a2b3c4d123")); // 16
    }
}
```

숫자 찾기 (Level.0)

https://school.programmers.co.kr/learn/courses/30/lessons/120904?language=java

📌 문제 해결 개요

이 문제는 정수를 문자열로 변환한 후 특정 숫자가 포함된 위치를 찾아 반환하는 문제입니다. `indexOf()` 메서드를 사용하여 가장 먼저 등장하는 위치를 찾습니다.

🔷 핵심 개념

- 정수를 문자열로 변환하여 개별 숫자 접근
- `indexOf()`를 사용해 특정 숫자의 위치 찾기
- 0-based index를 1-based로 변환하여 반환

🚀 해결 방법

1. `num`을 문자열로 변환합니다.
2. `indexOf()`를 사용하여 `k`가 등장하는 첫 번째 위치를 찾습니다.
3. `indexOf()` 결과가 -1보다 크다면 1을 더해 위치를 반환합니다.
4. `indexOf()` 결과가 -1이면 -1을 반환합니다.

```
public class Solution {
    public int solution(int num, int k) {
        // num을 문자열로 변환
        String numStr = String.valueOf(num);

        // k의 위치 찾기
        int index = numStr.indexOf(String.valueOf(k));

        // k가 포함된 경우 1-based index로 변환하여 반환
        if (index != -1) {
            return index + 1;
        } else {
            return -1; // k가 포함되지 않으면 -1 반환
        }
    }

    public static void main(String[] args) {
        Solution sol = new Solution();

        // 테스트 예제 실행
        System.out.println(sol.solution(29183, 1)); // 3
        System.out.println(sol.solution(232443, 4)); // 4
        System.out.println(sol.solution(123456, 7)); // -1
    }
}
```

암호 해독 (Level.0)

https://school.programmers.co.kr/learn/courses/30/lessons/120892?language=java

📌 문제 해결 개요

이 문제는 문자열에서 특정 간격(`code`의 배수 번째 위치)으로 문자를 추출하여 새로운 문자열을 만드는 문제입니다.

🔷 핵심 개념

- 문자열 인덱스를 활용한 문자 선택
- `code`의 배수 번째 문자만 선택하여 누적
- `String`의 `+=` 연산을 사용하여 문자열 생성

🚀 해결 방법

1. 결과를 저장할 문자열 변수를 초기화합니다.
2. 반복문을 사용하여 `code`의 배수 번째 인덱스에 해당하는 문자를 선택합니다.
3. 선택된 문자를 누적하여 새로운 문자열을 생성합니다.
4. 최종 결과 문자열을 반환합니다.

```
public class Solution {
    public String solution(String cipher, int code) {
        String result = ""; // 결과를 저장할 문자열 초기화

        // code의 배수 번째 위치에 있는 문자만 추가
        for (int i = code - 1; i < cipher.length(); i += code) {
            result += cipher.charAt(i); // 해당 위치의 문자 추가
        }

        return result; // 최종 해독된 문자열 반환
    }

    public static void main(String[] args) {
        Solution sol = new Solution();

        // 테스트 예제 실행
        System.out.println(sol.solution("dfjardstddetckdaccccdegk", 4)); //
"attack"
        System.out.println(sol.solution("pfqallllabwaoclk", 2)); // "fallback"
    }
}
```

인덱스 바꾸기 (Level.0)

https://school.programmers.co.kr/learn/courses/30/lessons/120895?language=java

📌 문제 해결 개요

이 문제는 문자열에서 두 위치의 문자를 교환하는 문제입니다. 문자열을 리스트로 변환하여 문자 교환 후 다시 문자열로 변환하는 방식으로 해결합니다.

◈ 핵심 개념

- 문자열을 리스트로 변환하여 문자 교환 가능하도록 변경
- 리스트에서 두 인덱스의 값을 교환
- 리스트를 다시 문자열로 변환하여 반환

🚀 해결 방법

1. 문자열을 리스트로 변환합니다.
2. 리스트에서 num1과 num2 인덱스의 문자를 교환합니다.
3. 변경된 리스트를 다시 문자열로 변환하여 반환합니다.

```
public class Solution {
    public String solution(String myString, int num1, int num2) {
        // 문자열을 문자 배열로 변환
        char[] charArray = myString.toCharArray();

        // num1과 num2의 위치에 있는 문자 교환
        char temp = charArray[num1];
        charArray[num1] = charArray[num2];
        charArray[num2] = temp;

        // 문자 배열을 다시 문자열로 변환하여 반환
        return new String(charArray);
    }

    public static void main(String[] args) {
        Solution sol = new Solution();

        // 테스트 예제 실행
        System.out.println(sol.solution("hello", 1, 2)); // "hlelo"
        System.out.println(sol.solution("I love you", 3, 6)); // "I l veoyou"
    }
}
```

중복된 문자 제거 (Level.0)

https://school.programmers.co.kr/learn/courses/30/lessons/120888?language=java

📌 문제 해결 개요

이 문제는 문자열에서 중복된 문자를 제거하고 최초 등장한 문자만 남기는 문제입니다.

`indexOf()` 메서드를 활용하여 중복 여부를 검사하는 방식으로 해결합니다.

◈ 핵심 개념

- `indexOf()`를 사용하여 현재 문자가 이미 결과 문자열에 포함되어 있는지 확인
- 포함되지 않은 경우에만 결과 문자열에 추가
- 문자열을 순회하면서 중복 문자를 제거

🚀 해결 방법

1. 결과를 저장할 문자열 변수를 초기화합니다.
2. 문자열을 순회하며 현재 문자가 결과 문자열에 이미 존재하는지 `indexOf()`를 통해 확인합니다.
3. 존재하지 않으면 결과 문자열에 추가합니다.
4. 최종 변환된 문자열을 반환합니다.

```
public class Solution {
    public String solution(String myString) {
        String result = ""; // 결과 문자열 초기화

        // 문자열 순회하며 중복되지 않은 문자만 추가
        for (char c : myString.toCharArray()) {
            if (result.indexOf(c) == -1) { // 현재 문자가 결과 문자열에 없는 경우
                result += c; // 결과 문자열에 추가
            }
        }

        return result;
    }

    public static void main(String[] args) {
        Solution sol = new Solution();

        // 테스트 예제 실행
        System.out.println(sol.solution("people")); // "peol"
        System.out.println(sol.solution("We are the world")); // "We arthwold"
    }
}
```

특정 문자 제거하기 (Level.0)

https://school.programmers.co.kr/learn/courses/30/lessons/120826?language=java

📌 문제 해결 개요

이 문제는 주어진 문자열에서 특정 문자를 제거하는 문제입니다.

`replace()` 메서드를 사용하는 방법과 문자열을 순회하면서 letter와 다른 문자만 추가하는 두 가지 방법으로 접근할 수 있습니다.

🔷 핵심 개념 (`replace()` 메서드를 사용하지 않는 경우)

- 문자열 순회하여 특정 문자 제거

- `+=` 연산을 이용하여 새로운 문자열 생성
- 대소문자를 구분하여 정확한 비교 수행

🚀 해결 방법

- 풀이(1)
    1. `replace()` 메서드를 사용
- 풀이(2)
    1. 결과를 저장할 문자열 변수를 초기화합니다.
    2. 반복문을 사용하여 `my_string`을 순회하면서 `letter`가 아닌 문자만 결과 문자열에 추가합니다.
    3. 최종 변환된 문자열을 반환합니다.

```
public class Solution {
    public String solution(String myString, String letter) {
        // [풀이1]
        // return myString.replace(letter, "");

        // [풀이2]
        String result = ""; // 결과 문자열 초기화

        // 문자열 순회하며 letter와 다른 문자만 추가
        for (char c : myString.toCharArray()) {
            if (c != letter.charAt(0)) { // letter와 다른 문자만 추가
                result += c;
            }
        }

        return result; // 최종 변환된 문자열 반환
    }

    public static void main(String[] args) {
        Solution sol = new Solution();

        // 테스트 예제 실행
        System.out.println(sol.solution("abcdef", "f")); // "abcde"
        System.out.println(sol.solution("BCBdbe", "B")); // "Cdbe"
    }
}
```

편지 (Level.0)

https://school.programmers.co.kr/learn/courses/30/lessons/120898?language=java

📌 문제 해결 개요

이 문제는 주어진 문자열의 길이를 계산하고, 각 문자가 2cm의 공간을 차지한다는 조건을 활용하여 필요한 가로 길이를 구하는 문제입니다.

🔷 핵심 개념

- 문자열의 길이를 구하여 각 문자당 2cm를 곱하는 방식
- 공백도 하나의 문자로 취급하여 계산
- 최종적으로 계산된 값을 반환

🚀 해결 방법

1. `message`의 길이를 구합니다.
2. 각 문자당 2cm씩 차지하므로 길이에 2를 곱합니다.
3. 계산된 가로 길이를 반환합니다.

```
public class Solution {
    public int solution(String message) {
        // 문자열의 길이를 구한 후 각 문자당 2cm를 곱하여 반환
        return message.length() * 2;
    }

    public static void main(String[] args) {
        Solution sol = new Solution();

        // 테스트 예제 실행
        System.out.println(sol.solution("happy birthday!")); // 30
        System.out.println(sol.solution("I love you~")); // 22
    }
}
```

k의 개수 (Level.0)

https://school.programmers.co.kr/learn/courses/30/lessons/120887?language=java

📌 문제 해결 개요

이 문제는 주어진 범위에서 특정 숫자가 등장하는 횟수를 계산하는 문제입니다. 각 숫자를 문자열로 변환한 후 `count()` 메서드를 사용하여 특정 숫자의 등장 횟수를 누적하는 방식으로 해결합니다.

🔷 핵심 개념

- 숫자를 문자열로 변환하여 특정 숫자(`k`)의 등장 횟수 찾기
- `count()` 메서드를 사용하여 해당 숫자의 빈도 누적
- `for` 문을 사용하여 i부터 j까지 모든 숫자를 검사

🚀 해결 방법

1. `k`를 문자열로 변환합니다.
2. `i`부터 `j`까지 반복하며 각 숫자를 문자열로 변환하고 `count()` 메서드를 사용하여 `k`의 개수를 누적합니다.
3. 최종 누적된 개수를 반환합니다.

```java
public class Solution {
    public int solution(int i, int j, int k) {
        int count = 0; // 결과를 저장할 변수 초기화
        String kStr = String.valueOf(k); // k를 문자열로 변환

        // i부터 j까지 반복하며 k의 등장 횟수 계산
        for (int n = i; n <= j; n++) {
            String numStr = String.valueOf(n); // 현재 숫자를 문자열로 변환
            count += numStr.length() - numStr.replace(kStr, "").length(); // k
의 개수를 누적
        }

        return count; // 최종 등장 횟수 반환
    }

    public static void main(String[] args) {
        Solution sol = new Solution();

        // 테스트 예제 실행
        System.out.println(sol.solution(1, 13, 1)); // 6
        System.out.println(sol.solution(10, 50, 5)); // 5
        System.out.println(sol.solution(3, 10, 2)); // 0
    }
}
```