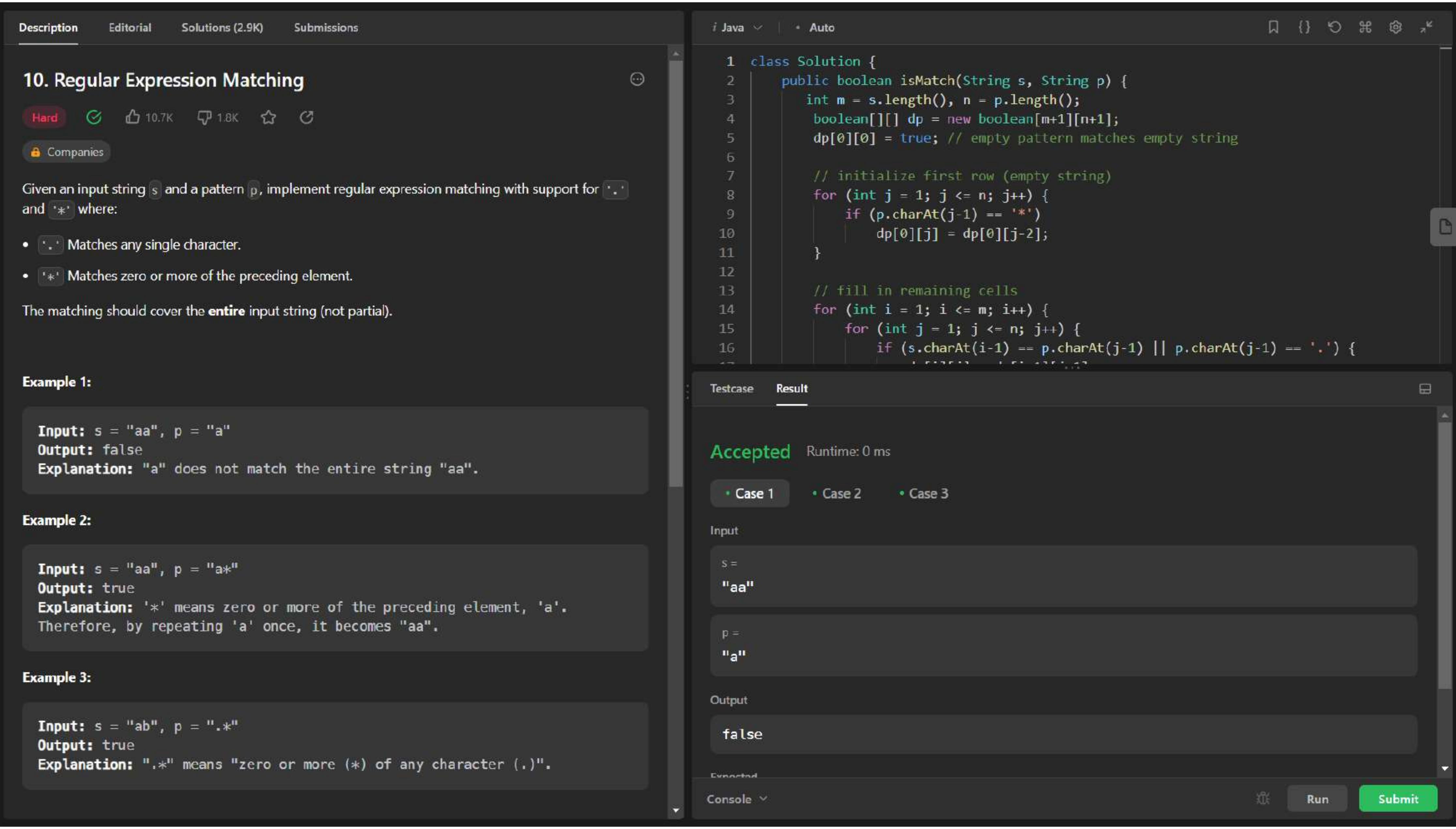


10. Regular Expression Matching

Given an input string `s` and a pattern `p`, implement regular expression matching with support for `'.'` and `'*'` where:

- `'.'` Matches any single character.
- `'*'` Matches zero or more of the preceding element.

The matching should cover the **entire** input string (not partial).

Example 1:

```
Input: s = "aa", p = "a"
Output: false
Explanation: "a" does not match the entire string "aa".
```

Example 2:

```
Input: s = "aa", p = "a*"
Output: true
Explanation: '*' means zero or more of the preceding element, 'a'.
Therefore, by repeating 'a' once, it becomes "aa".
```

Example 3:

```
Input: s = "ab", p = ".*"
Output: true
Explanation: ".*" means "zero or more (*) of any character (.)".
```

```java
class Solution {
    public boolean isMatch(String s, String p) {
      int m = s.length(), n = p.length();
        boolean[][] dp = new boolean[m+1][n+1];
        dp[0][0] = true; // empty pattern matches empty string

        // initialize first row (empty string)
        for (int j = 1; j <= n; j++) {
            if (p.charAt(j-1) == '*')
                dp[0][j] = dp[0][j-2];
        }

        // fill in remaining cells
        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                if (s.charAt(i-1) == p.charAt(j-1) || p.charAt(j-1) == '.') {
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

s =

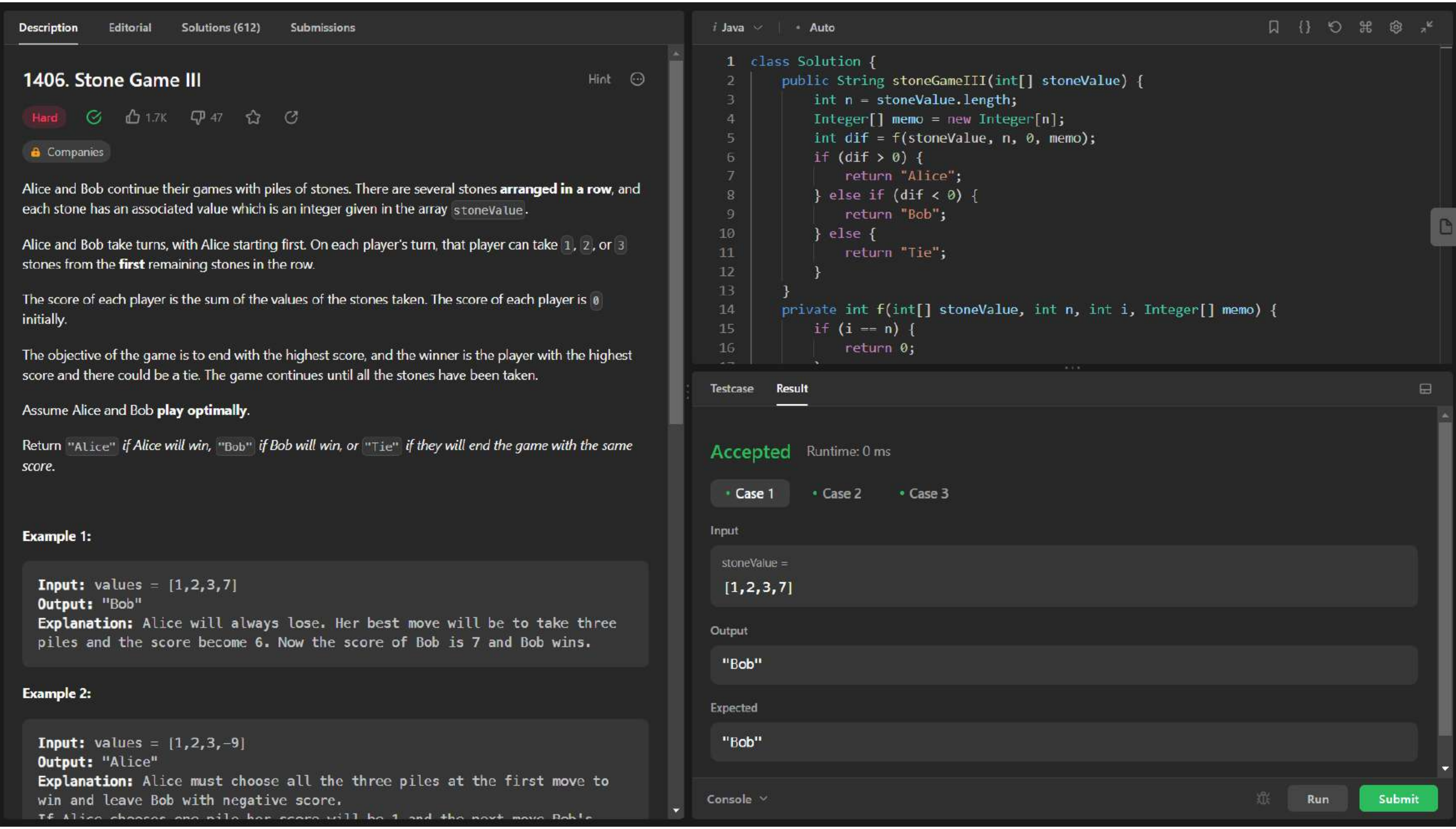
"aa"

p =

"a"

Output

```
false
```



1406. Stone Game III

Alice and Bob continue their games with piles of stones. There are several stones **arranged in a row**, and each stone has an associated value which is an integer given in the array `stoneValue`.

Alice and Bob take turns, with Alice starting first. On each player's turn, that player can take `1`, `2`, or `3` stones from the **first** remaining stones in the row.

The score of each player is the sum of the values of the stones taken. The score of each player is `0` initially.

The objective of the game is to end with the highest score, and the winner is the player with the highest score and there could be a tie. The game continues until all the stones have been taken.

Assume Alice and Bob **play optimally**.

Return `"Alice"` *if Alice will win,* `"Bob"` *if Bob will win, or* `"Tie"` *if they will end the game with the same score.*

Example 1:

```
Input: values = [1,2,3,7]
Output: "Bob"
Explanation: Alice will always lose. Her best move will be to take three
piles and the score become 6. Now the score of Bob is 7 and Bob wins.
```

Example 2:

```
Input: values = [1,2,3,-9]
Output: "Alice"
Explanation: Alice must choose all the three piles at the first move to
win and leave Bob with negative score.
```

```java
class Solution {
    public String stoneGameIII(int[] stoneValue) {
        int n = stoneValue.length;
        Integer[] memo = new Integer[n];
        int dif = f(stoneValue, n, 0, memo);
        if (dif > 0) {
            return "Alice";
        } else if (dif < 0) {
            return "Bob";
        } else {
            return "Tie";
        }
    }
    private int f(int[] stoneValue, int n, int i, Integer[] memo) {
        if (i == n) {
            return 0;
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

stoneValue =

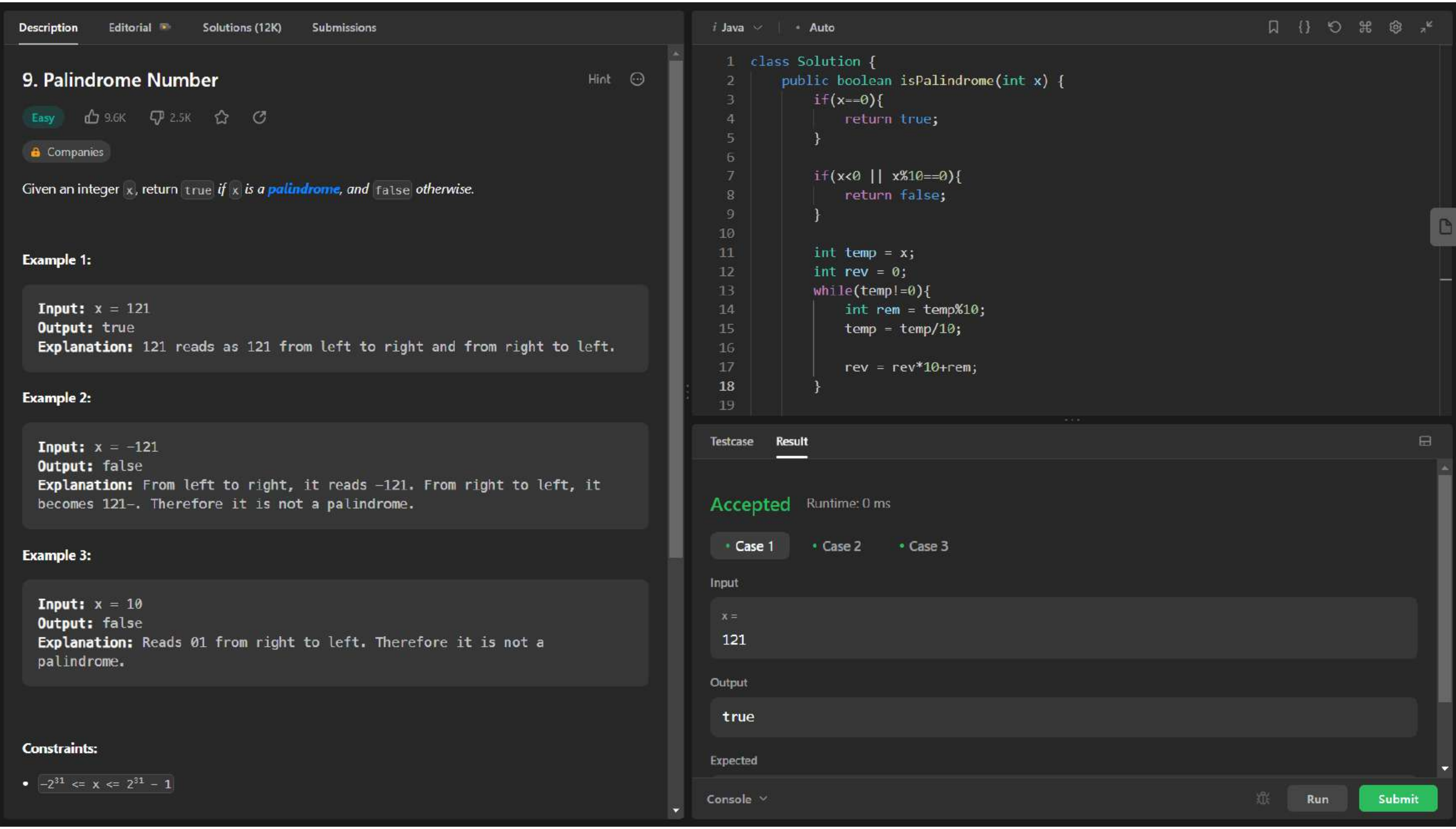
[1,2,3,7]

Output

"Bob"

Expected

"Bob"



9. Palindrome Number

Given an integer `x`, return `true` *if* `x` *is a* ***palindrome****, and* `false` *otherwise.*

Example 1:

```
Input: x = 121
Output: true
Explanation: 121 reads as 121 from left to right and from right to left.
```

Example 2:

```
Input: x = -121
Output: false
Explanation: From left to right, it reads -121. From right to left, it
becomes 121-. Therefore it is not a palindrome.
```

Example 3:

```
Input: x = 10
Output: false
Explanation: Reads 01 from right to left. Therefore it is not a
palindrome.
```

Constraints:

- $-2^{31} <= x <= 2^{31} - 1$

```java
class Solution {
    public boolean isPalindrome(int x) {
        if(x==0){
            return true;
        }

        if(x<0 || x%10==0){
            return false;
        }

        int temp = x;
        int rev = 0;
        while(temp!=0){
            int rem = temp%10;
            temp = temp/10;

            rev = rev*10+rem;
        }

```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

x =
121

Output

true

Expected