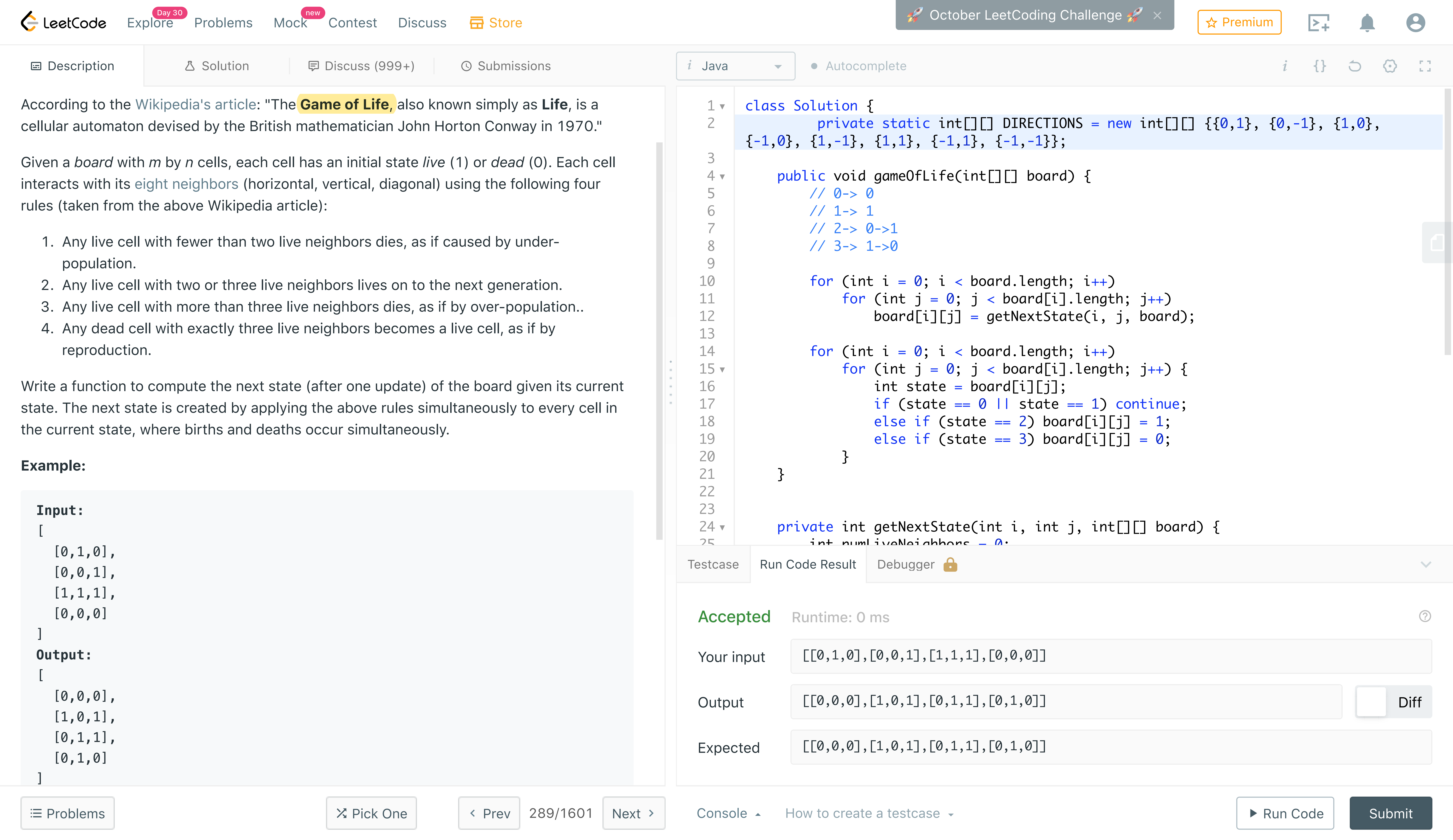

According to the Wikipedia's article: "The **Game of Life**, also known simply as **Life**, is a cellular automaton devised by the British mathematician John Horton Conway in 1970."

Given a *board* with *m* by *n* cells, each cell has an initial state *live* (1) or *dead* (0). Each cell interacts with its eight neighbors (horizontal, vertical, diagonal) using the following four rules (taken from the above Wikipedia article):

1. Any live cell with fewer than two live neighbors dies, as if caused by under-population.
2. Any live cell with two or three live neighbors lives on to the next generation.
3. Any live cell with more than three live neighbors dies, as if by over-population..
4. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

Write a function to compute the next state (after one update) of the board given its current state. The next state is created by applying the above rules simultaneously to every cell in the current state, where births and deaths occur simultaneously.

Example:

```
Input:
[
  [0,1,0],
  [0,0,1],
  [1,1,1],
  [0,0,0]
]
Output:
[
  [0,0,0],
  [1,0,1],
  [0,1,1],
  [0,1,0]
]
```

```java
class Solution {
        private static int[][] DIRECTIONS = new int[][] {{0,1}, {0,-1}, {1,0},
{-1,0}, {1,-1}, {1,1}, {-1,1}, {-1,-1}};

    public void gameOfLife(int[][] board) {
        // 0-> 0
        // 1-> 1
        // 2-> 0->1
        // 3-> 1->0

        for (int i = 0; i < board.length; i++)
            for (int j = 0; j < board[i].length; j++)
                board[i][j] = getNextState(i, j, board);

        for (int i = 0; i < board.length; i++)
            for (int j = 0; j < board[i].length; j++) {
                int state = board[i][j];
                if (state == 0 || state == 1) continue;
                else if (state == 2) board[i][j] = 1;
                else if (state == 3) board[i][j] = 0;
            }
    }


    private int getNextState(int i, int j, int[][] board) {
        int numLiveNeighbors = 0;
```

Accepted   Runtime: 0 ms

Your input `[[0,1,0],[0,0,1],[1,1,1],[0,0,0]]`

Output `[[0,0,0],[1,0,1],[0,1,1],[0,1,0]]`

Expected `[[0,0,0],[1,0,1],[0,1,1],[0,1,0]]`