|  |  |
| --- | --- |
| 1. The LightBoard class models a two-dimensional display of lights, where each light is either on or off, as represented by a Boolean value. You will implement a constructor to initialize the display and a method to evaluate a light.  public class LightBoard  {  /\*\* The lights on the board, where true represents on and false  \* represents off.  \*/  private boolean[][] lights;  /\*\* Constructs a LightBoard object having numRows rows and numCols columns  \* **Precondition**: numRows > 0, numCols > 0  \* **Postcondition**: each light has a 60% probability of being set to on  \*/  public LightBoard(int numRows, int numCols)  { /\* To be implemented in part (a) \*/ }  /\*\* Evaluates a light in row index row and column index col  \* and returns a status as described in part (b).  \* Precondition: row and col are valid indexes in lights.  \*/  public boolean evaluateLight(int row, int col)  { /\* to be implemented in part (b) \*/ }  // There may be additional instance variables, constructors, and methods not shown.  } | |
| (a) Write the constructor for the LightBoard class, which initializes lights so that each light is set to on with a 60% probability. The notation lights[r][c] represents the array element at row r and column c.  Complete the LightBoard constructor below.  /\*\* Constructs a LightBoard object having numRows rows and numCols columns.  \* Precondition: numRows > 0, numCols > 0  \* Postcondition: each light has a 60% probability of being set to on.  \*/  public LightBoard(int numRows, int numCols) {    lights = new boolean[numRows][numCols];          for(int row = 0; row < lights.length; row++){              for(int col = 0; col < lights[row].length; col++){                  lights[row][col] = (Math.random() < .6);              }          }  } | |
|  | /4 |
| (b) Write the method evaluateLight, which computes and returns the status of a light at a given row and column based on the following rules.  1. If the light is off, return false if the number of lights in its column that are on is odd, including the current light.  2. If the light is on, return true if the number of lights in its column that are on is divisible by three.  3. Otherwise, return the light’s current status.   |  | | --- | | Class information for this question  public class LightBoard  private boolean[][] lights  public LightBoard(int numRows, int numCols)  public boolean evaluateLight(int row, int col) |   Complete the evaluateLight method below.  /\*\* Evaluates a light in row index row and column index col and returns a status  \* as described in part (b).  \* Precondition: row and col are valid indexes in lights.  \*/  public boolean evaluateLight(int row, int col) {    int count = 0;          for(int r = 0; r < lights.length; r++){              if(lights[r][col]){                  count++;              }          }          if(!lights[row][col] && count%2!=0){              return false;          }          if(lights[row][col] && count%3 == 0){              return true;          }          return lights[row][col];  } | |
|  | /5 |

|  |
| --- |
| 2. |
| A picture containing text  Description automatically generated |
| int[] temp = new int[arr2D.length];  for(int row = 0; row < arr2D.length; row++){       temp = arr2D[row][c];  }  return temp; |
|  |
| Text  Description automatically generated with medium confidence  int[] firstRow = square[0];  if(containsDuplicates(firstRow)){      return false;  }  for(int r = 1; r < square.length; r++){       if(!hasAllValues(firstRow, square[0])){              return false;        }  }  for(int r = 0; r < square.length; r++){       if(!hasAllValues(firstRow, getColumn(square, r))){               return false;       }  }  return true; |