```java
/**
 *  Game of Life.
 *  Usage: "GameOfLife fileName"
 *  The file represents the initial board.
 *  The file format is described in the HW05 document.
 */

public class GameOfLife {

    public static void main(String[] args) {
        String fileName = args[0];
        // read(fileName);
        //test1(fileName);
        //test2(fileName);
        //test3(fileName, 3);
        play(fileName);
    }

    // Reads the data file and prints the initial board.
    private static void test1(String fileName) {
        int[][] board = read(fileName);
        print(board);
    }

    // Reads the data file, and runs a test that checks
    // the count and cellValue functions.
    private static void test2(String fileName) {
        int[][] board = read(fileName);
        int rows = board.length;
        int cols = board[0].length;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                int newValue = cellValue(board, i, j);
                String message = "the old value was: " + board[i][j] + " and the new is: " + newValue;
                System.out.println(message);
            }
        }
    }

    private static void test3(String fileName, int Ngen) {
        int[][] board = read(fileName);
        for (int gen = 0; gen < Ngen; gen++) {
            System.out.println("Generation " + gen + ":");
            print(board);
            board = evolve(board);
        }
    }

    // Reads the data file and plays the game, for ever.
    private static void play(String fileName) {
        int[][] board = read(fileName);
        while (true) {
            show(board);
            board = evolve(board);
        }
    }

    private static int[][] read(String fileName) {
        StdIn.setInput(fileName);
        int rows = Integer.parseInt(StdIn.readLine());
        int cols = Integer.parseInt(StdIn.readLine());
        int[][] board = new int[rows][cols];
        String structure = "";

        for (int i = 0; i < rows; i++) {
            structure = StdIn.readLine();
            for (int j = 0; j < cols; j++) {
                if (i == 0) {
                    board[i][j] = 0;
                } else if (i == rows - 1) {
                    board[i][j] = 0;
                } else if (structure.length() > 0) {
                    if (j < structure.length() && structure.charAt(j) == 'x') {
                        board[i][j] = 1;
                    } else {
                        board[i][j] = 0;
                    }
                }
            }
        }

        return board;
    }

    private static int[][] evolve(int[][] board){
        int rows = board.length;
        int cols = board.length;
        int[][] newBoard = new int[rows][cols];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                newBoard[i][j] = cellValue(board, i, j);
            }
        }

        return newBoard;
    }

    private static int cellValue(int[][] board, int i, int j) {
        int numOfLiveNeighbers = count(board, i, j);
        String message = " " + numOfLiveNeighbers;
        System.out.println(message);
        int newValue = board[i][j];
        if (board[i][j] == 1) {
            if (numOfLiveNeighbers < 2) {
                newValue = 0;
            }

            if ((numOfLiveNeighbers == 2 || numOfLiveNeighbers == 3)) {
                newValue = 1;
            }

```

```java
            if (numOfLiveNeighbors > 3) {
                newValue = 0;
            }
        } else {
            if (numOfLiveNeighbors == 3) {
                newValue = 1;
            }
        }

        return newValue;
    }

    private static int count(int[][] board, int i, int j) {
        int rows = board.length;
        int cols = board[0].length;
        int numOfLiveNeighbors = 0;
        for (int z = 0; z < rows; z++) {
            // row above
            if (i - 1 >= 0 && z == i - 1) {
                // row above but 1 col before
                if (j - 1 >= 0) {
                    if (board[z][j - 1] == 1) {
                        numOfLiveNeighbors++;
                    }
                }

                // row above but the same col
                if (board[z][j] == 1) {
                    numOfLiveNeighbors++;
                }

                //row above but 1 col to the right
                if (j + 1 < cols - 1 && board[z][j + 1] == 1) {
                    numOfLiveNeighbors++;
                }
            }

            // same row
            if (z == i) {
                if (j - 1 >= 0) {
                    // same row but 1 col before
                    if (board[z][j - 1] == 1) {
                        numOfLiveNeighbors++;
                    }
                }

                // same row but 1 col after
                if (j + 1 < cols - 1) {
                    if (board[z][j + 1] == 1) {
                        numOfLiveNeighbors++;
                    }
                }
            }

            // 1 row after
            if (i + 1 <= rows - 1 && z == i + 1) {
                // 1 row after but 1 col before
                if (j - 1 >= 0) {
                    if (board[z][j - 1] == 1) {
                        numOfLiveNeighbors++;
                    }
                }

                if (board[z][j] == 1) {
                    numOfLiveNeighbors++;
                }

                //1 row after and 1 col after
                if (j + 1 < cols - 1) {
                    if (board[z][j + 1] == 1) {
                        numOfLiveNeighbors++;
                    }
                }
            }
        }
        String message = "the number of live cells for " + i + "," + j + "is: " + numOfLiveNeighbors;
        System.out.println(message);
        return numOfLiveNeighbors;
    }

    // Prints the board. Alive and dead cells are printed as 1 and 0, respectively.
    private static void print(int[][] arr) {
        int rows = arr.length;
        int cols = arr[0].length;
        for (int i = 0; i < rows; i++) {
            String stam = "";
            for (int j = 0; j < cols; j++) {
                stam += arr[i][j] + " ";
            }
            System.out.println(stam);
        }
    }

    private static void show(int[][] board) {
        StdDraw.setCanvasSize(900, 900);
        int rows = board.length;
        int cols = board[0].length;
        StdDraw.setXscale(0, cols);
        StdDraw.setYscale(0, rows);
        StdDraw.show(100); // delay the next display 100 miliseconds
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                int grey = 255 * (1 - board[i][j]);
                StdDraw.setPenColor(grey, grey, grey);
                StdDraw.filledRectangle(j + 0.5, rows - i - 0.5, 0.5, 0.5);
            }
        }
        StdDraw.show();
    }
}
```