```
public class GameOfLife {

    public static void main(String[] args) {
        String fileName = args[0];
        //// Uncomment the test that you want to execute, and re-compile.
        //// (Run one test at a time).
         //test1(fileName);
        ///test2(fileName);
        /// test3(fileName, 3);
        play(fileName);
    }

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

    public static void test2(String fileName) {
        int[][] board = read(fileName);
        int[][] newBoard = new int[board.length][board[1].length];
        print(board);
        for (int i = 1; i < board.length-1; i++) {
            for (int j = 1; j < board[i].length-1; j++) {
        newBoard[i][j]= cellValue(board, i, j);
            }
        }
        System.out.println("The Next Board Will Be- ");
        print(newBoard);
    }

    public 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);
        }
    }
```

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

public static int[][] read(String fileName) {
    In in = new In(fileName); // Constructs an In object for reading the input file
    int rows = Integer.parseInt(in.readLine());
    int cols = Integer.parseInt(in.readLine());
    int[][] board = new int[rows + 2][cols + 2];
    for (int i = 1; i <= rows; i++) {
        String temp = in.readLine();
        if (temp != null){
      for (int j = 0; j < temp.length(); j++) {
        if (temp.charAt(j) == 'x') {
            board[i][j+1] = 1;
        }
      }
    }
}
    //// Replace the following statement with your code.
    return board;
}

public static int[][] evolve(int[][] board) {
    int[][] newBoard = new int[board.length][board[1].length];
    for (int i = 1; i < board.length-1; i++) {
        for (int j = 1; j < board[i].length-1; j++) {
    newBoard[i][j]= cellValue(board, i, j);
        }
    }
    return newBoard;
}
```

```java
public static int cellValue(int[][] board, int i, int j) {
    boolean status = (board[i][j] == 1);
    if (status && count(board, i, j) < 2) {
        return 0;
    }else if(status && (count(board, i, j) == 2 || count(board, i, j) == 3)){
        return 1;
    }else if (status && count(board, i, j) > 3){
        return 0;
    }else if (!status && count(board, i, j) == 3){
        return 1;
    }
    return board[i][j];
}

public static int count(int[][] board, int i, int j) {
    int count = 0;
    for ( int x = i - 1; x <= i + 1; x++){
     for ( int y = j - 1; y <= j + 1; y++){
        if (x != i || y != j){
            if(board[x][y] == 1){
                count++;
            }
        }
     }
    }
    return count;
}

// Prints the board. Alive and dead cells are printed as 1 and 0, respectively.
public static void print(int[][] arr) {
    for (int i = 1; i < arr.length-1; i++) {
        for (int j = 1; j < arr[i].length-1; j++) {
            System.out.printf("%3s", arr[i][j]);
        }
        System.out.println();
    }
}
```

```
    public 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);

        // Enables drawing graphics in memory and showing it on the screen only
when
        // the StdDraw.show function is called.
        StdDraw.enableDoubleBuffering();


        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                int color = 255 * (1 - board[i][j]);
                StdDraw.setPenColor(color, color, color);
                StdDraw.filledRectangle(j + 0.5, rows - i - 0.5, 0.5, 0.5);
            }
        }
        StdDraw.show();
        StdDraw.pause(100);
    }
}
```