

```
1 /**
2  * Game of Life.
3  * Usage: "GameOfLife fileName"
4  * The file represents the initial board.
5  * The file format is described in the HW05 document.
6  */
7
8 public class GameOfLife {
9
10     public static void main(String[] args) {
11         String fileName = args[0];
12         // read(fileName);
13         //test1(fileName);
14         //test2(fileName);
15         //test3(fileName, 3);
16         play(fileName);
17     }
18
19     // Reads the data file and prints the initial board.
20     private static void test1(String fileName) {
21         int[][] board = read(fileName);
22         print(board);
23     }
24
25     // Reads the data file, and runs a test that checks
26     // the count and cellValue functions.
27     private static void test2(String fileName) {
28         int[][] board = read(fileName);
29         int rows = board.length;
30         int cols = board[0].length;
31         for (int i = 0; i < rows; i++) {
32             for (int j = 0; j < cols; j++) {
33                 int newValue = cellValue(board, i, j);
34                 String message = "the old value was: " + board[i][j] + " and the new is: " + newValue;
35                 System.out.println(message);
36             }
37         }
38     }
39
40     private static void test3(String fileName, int Ngen) {
41         int[][] board = read(fileName);
42         for (int gen = 0; gen < Ngen; gen++) {
43             System.out.println("Generation " + gen + ":");
44             print(board);
45             board = evolve(board);
46         }
47     }
48
49     // Reads the data file and plays the game, for ever.
50     private static void play(String fileName) {
51         int[][] board = read(fileName);
52         while (true) {
53             show(board);
54             board = evolve(board);
55         }
56     }
57
58     private static int[][] read(String fileName) {
59         StdIn.setInput(fileName);
60         int rows = Integer.parseInt(StdIn.readLine());
61         int cols = Integer.parseInt(StdIn.readLine());
62         int[][] board = new int[rows][cols];
63         String structure = "";
64
65         for (int i = 0; i < rows; i++) {
66             structure = StdIn.readLine();
67             for (int j = 0; j < cols; j++) {
68                 if (i == 0) {
69                     board[i][j] = 0;
70                 } else if (i == rows - 1) {
71                     board[i][j] = 0;
72                 } else if (structure.length() > 0) {
73                     if (j < structure.length() && structure.charAt(j) == 'x') {
74                         board[i][j] = 1;
75                     } else {
76                         board[i][j] = 0;
77                     }
78                 }
79             }
80         }
81
82         return board;
83     }
84
85     private static int[][] evolve(int[][] board){
86         int rows = board.length;
87         int cols = board[0].length;
88         int[][] newBoard = new int[rows][cols];
89         for (int i = 0; i < rows; i++) {
90             for (int j = 0; j < cols; j++) {
91                 newBoard[i][j] = cellValue(board, i, j);
92             }
93         }
94
95         return newBoard;
96     }
97
98     private static int cellValue(int[][] board, int i, int j) {
99         int numOfLiveNeighbors = count(board, i, j);
100         String message = " " + numOfLiveNeighbors;
101         System.out.println(message);
102         int newValue = board[i][j];
103         if (board[i][j] == 1) {
104             if (numOfLiveNeighbors < 2) {
105                 newValue = 0;
106             }
107
108             if ((numOfLiveNeighbors == 2 || numOfLiveNeighbors == 3)) {
109                 newValue = 1;
110             }
111         }
```

```

112         if (numOfLiveNeighbors > 3) {
113             newValue = 0;
114         }
115     } else {
116         if (numOfLiveNeighbors == 3) {
117             newValue = 1;
118         }
119     }
120
121     return newValue;
122 }
123
124 private static int count(int[][] board, int i, int j) {
125     int rows = board.length;
126     int cols = board[0].length;
127     int numOfLiveNeighbors = 0;
128     for (int z = 0; z < rows; z++) {
129         // row above
130         if (i - 1 >= 0 && z == i - 1) {
131             // row above but 1 col before
132             if (j - 1 >= 0) {
133                 if (board[z][j - 1] == 1) {
134                     numOfLiveNeighbors++;
135                 }
136             }
137
138             // row above but the same col
139             if (board[z][j] == 1) {
140                 numOfLiveNeighbors++;
141             }
142
143             //row above but 1 col to the right
144             if (j + 1 < cols - 1 && board[z][j + 1] == 1) {
145                 numOfLiveNeighbors++;
146             }
147         }
148
149         // same row
150         if (z == i) {
151             if (j - 1 >= 0) {
152                 // same row but 1 col before
153                 if (board[z][j - 1] == 1) {
154                     numOfLiveNeighbors++;
155                 }
156             }
157
158             // same row but 1 col after
159             if (j + 1 < cols - 1) {
160                 if (board[z][j + 1] == 1) {
161                     numOfLiveNeighbors++;
162                 }
163             }
164         }
165
166         // 1 row after
167         if (i + 1 <= rows - 1 && z == i + 1) {
168             // 1 row after but 1 col before
169             if (j - 1 >= 0) {
170                 if (board[z][j - 1] == 1) {
171                     numOfLiveNeighbors++;
172                 }
173             }
174
175             if (board[z][j] == 1) {
176                 numOfLiveNeighbors++;
177             }
178
179             //1 row after and 1 col after
180             if (j + 1 < cols - 1) {
181                 if (board[z][j + 1] == 1) {
182                     numOfLiveNeighbors++;
183                 }
184             }
185         }
186     }
187     String message = "the number of live cells for " + i + ", " + j + " is: " + numOfLiveNeighbors;
188     System.out.println(message);
189     return numOfLiveNeighbors;
190 }
191
192 // Prints the board. Alive and dead cells are printed as 1 and 0, respectively.
193 private static void print(int[][] arr) {
194     int rows = arr.length;
195     int cols = arr[0].length;
196     for (int i = 0; i < rows; i++) {
197         String stam = "";
198         for (int j = 0; j < cols; j++) {
199             stam += arr[i][j] + " ";
200         }
201         System.out.println(stam);
202     }
203 }
204
205 private static void show(int[][] board) {
206     StdDraw.setCanvasSize(900, 900);
207     int rows = board.length;
208     int cols = board[0].length;
209     StdDraw.setXscale(0, cols);
210     StdDraw.setYscale(0, rows);
211     StdDraw.show(100); // delay the next display 100 milliseconds
212     for (int i = 0; i < rows; i++) {
213         for (int j = 0; j < cols; j++) {
214             int grey = 255 * (1 - board[i][j]);
215             StdDraw.setPenColor(grey, grey, grey);
216             StdDraw.filledRectangle(j + 0.5, rows - i - 0.5, 0.5, 0.5);
217         }
218     }
219     StdDraw.show();
220 }
221 }

```