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

public class GameOfLife {

	public static void main(String[] args) {
		String fileName = args[0];
		play(fileName);
	}

	public static void test1(String fileName) {
		int[][] board = read(fileName);
		print(board);
	}

	public static void test2(String fileName) {
		int[][] board = read(fileName);
		print(board);
		System.out.println(" count:" + count(board, 3, 3));
		System.out.println();
		for (int i=1; i<board.length-1; i++) {
			for (int j=1; j<board[0].length-1; j++) {
				board[i][j] = cellValue(board, i, j);
			}
		}
		print(board);
	}

	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);
		}
	}

	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);
        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 line=in.readLine();
                if (line!=null) {
                        for (int j=1; j<=line.length(); j++) {
                                board[i][j] = (line.charAt(j-1)=='x') ? 1 : 0;
                        }
                }
        }

        return board;
}

public static int[][] evolve(int[][] board) {
        int row = board.length;
        int col = board[0].length;
        int[][] next = new int[row][col];
        for (int i=1; i<row-1; i++) {
                for (int j=1; j<col-1; j++) {
                        next[i][j] = cellValue(board, i, j);
                }
        }
        return next;
}

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

```
	public static int count(int[][] board, int i, int j) {
		int alive=0;
		for (int p=Math.max(i-1,0); p<=Math.min(i+1,board.length-1); p++) {
			for (int q=Math.max(j-1,0); q<=Math.min(j+1,board[0].length-1); q++) {
				if (board[p][q]==1) alive++;
			}
		}
		return alive-board[i][j];
	}


    public static void print(int[][] arr) {
		for (int i=1; i<arr.length-1; i++) {
			for (int j=1; j<arr[0].length-1; j++) {
				System.out.printf("%3d", 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);

		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);
	}
}
```