```
 1
 2 package GameOfLife;
 3
 4 import java.awt.*;
 5 import javax.swing.*;
 6 import java.util.Random;
 7
 8
 9 /**
10  * Object to represent the grid running the game of life
11  *
12  * @author Sharif Shaker
13  * @version 4/6/2017
14  */
15 public class Grid {
16
17     private final int  numRows;
18     private final int numCols;
19     private final JPanel grid;
20     private final Cell[][] cellWindows;
21
22     /**
23      * takes the number of rows and columns of the grid as a parameter
24      * @param rows number of rows in the grid
25      * @param cols number of columns in the grid
26      * @param deadColor color of a dead cell
27      * @param aliveColor color of a live cell
28      */
29     public Grid(int rows, int cols, Color deadColor, Color aliveColor) {
30         numRows = rows;
31         numCols = cols;
32         /* initializes a JPanel set up as a grid layout of the
33         specified number of rows and columns*/
34         grid = new JPanel(new GridLayout(rows, cols, 1, 1));
35         cellWindows = new Cell[rows][cols]; // represents a table of cells
36
37         /*
38         for each location in the grid layout create and add a cell
39         then add that cell to the specified location of the cells table
40         */
41         for (int i = 0; i < rows; i++) {
42             for (int j = 0; j < cols; j++) {
43                 // create new cell with given colors
44                 Cell cell = new Cell(deadColor, aliveColor);
45                 cell.setPreferredSize(new Dimension(10, 10));
46                 grid.add(cell);
47                 cellWindows[i][j] = cell;
48             }
49         }
50
51     }
52
53     /**
54      *
55      * @return JPanel representation of the grid
56      */
57     public JPanel getGridPanel() {
58         return grid;
59     }
60
61     /**
62      *runs one generation -> checks what is dead or alive
63      */
64     public void runGeneration() {
65
66         int aliveNeighbours;
67         boolean[][] livingTable = new boolean[numRows][numCols];
68         int top;
69         int bot;
70         int right;
71         int left;
72         /*
73         for each cell, checks whether the neighbours are living, increment aliveNeighbours
74         */
75         for (int i = 0; i < numRows; i++) {
76             for (int j = 0; j < numCols; j++) {
77                 top = (j > 0 ? j - 1 : numCols - 1);
78                 bot = (j < numCols - 1 ? j + 1 : 0);
79                 right = (i < numRows - 1 ? i + 1 : 0);
80                 left = (i > 0 ? i - 1 : numRows - 1);
```

```
                aliveNeighbours = 0;
                aliveNeighbours = getAliveNeighbours(aliveNeighbours, top, bot, right, i, j);
                aliveNeighbours = getAliveNeighbours(aliveNeighbours, bot, top, left, i, j);

                /*
                using the number of aliveNeighbours, is every cell in the grid alive or dead?
                create a copy of the gird
                */
                livingTable[i][j] = cellWindows[i][j].isCellAlive(aliveNeighbours);

            }
        }

        /*
        each cell is set alive -> refering to the copy in living table
        */
        for (int i = 0; i < numRows; i++) {
            for (int j = 0; j < numCols; j++) {
                cellWindows[i][j].setAlive(livingTable[i][j]);
            }
        }

        grid.repaint();
    }

    private int getAliveNeighbours(int aliveNeighbours, int top, int bot, int right, int i, int j) {
        if (cellWindows[i][top].isLiving()) {
            aliveNeighbours++;
        }
        if (cellWindows[right][top].isLiving()) {
            aliveNeighbours++;
        }
        if (cellWindows[right][j].isLiving()) {
            aliveNeighbours++;
        }
        if (cellWindows[right][bot].isLiving()) {
            aliveNeighbours++;
        }
        return aliveNeighbours;
    }

    public void clear(){
        for (int i = 0; i < numRows; i++) {
            for (int j = 0; j < numCols; j++) {
                cellWindows[i][j].setAlive(false);
            }
        }

        grid.repaint();
    }

    public void random(){
        Random random = new Random();
        for (int i = 0; i < numRows; i++) {
            for (int j = 0; j < numCols; j++) {
                cellWindows[i][j].setAlive(random.nextBoolean());
            }
        }
        grid.repaint();
    }

}

```