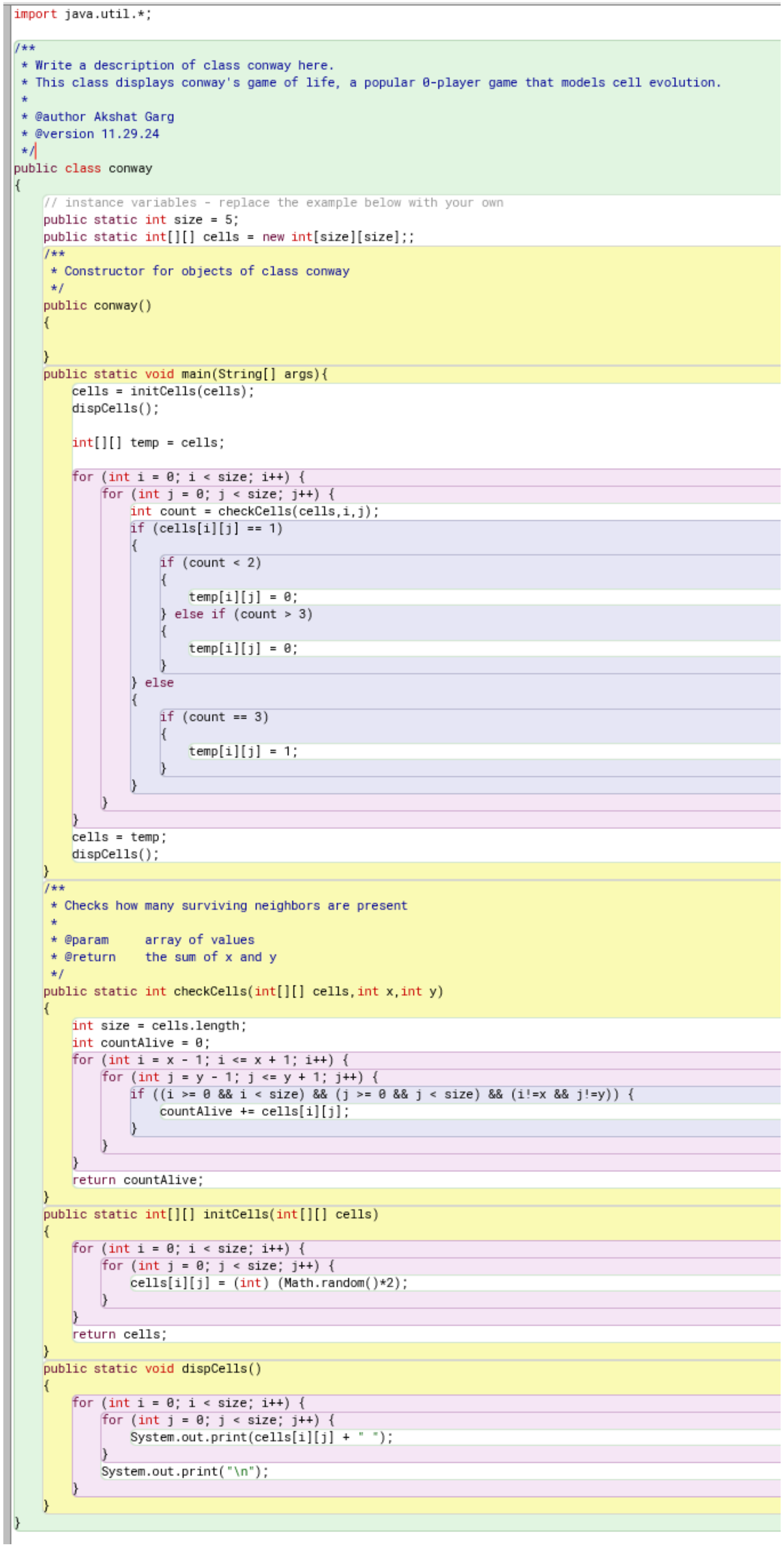


```java
import java.util.*;

/**
 * Write a description of class conway here.
 * This class displays conway's game of life, a popular 0-player game that models cell evolution.
 *
 * @author Akshat Garg
 * @version 11.29.24
 */
public class conway
{
    // instance variables - replace the example below with your own
    public static int size = 5;
    public static int[][] cells = new int[size][size];;
    /**
     * Constructor for objects of class conway
     */
    public conway()
    {

    }
    public static void main(String[] args){
        cells = initCells(cells);
        dispCells();

        int[][] temp = cells;

        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                int count = checkCells(cells,i,j);
                if (cells[i][j] == 1)
                {
                    if (count < 2)
                    {
                        temp[i][j] = 0;
                    } else if (count > 3)
                    {
                        temp[i][j] = 0;
                    }
                } else
                {
                    if (count == 3)
                    {
                        temp[i][j] = 1;
                    }
                }
            }
        }
        cells = temp;
        dispCells();
    }
    /**
     * Checks how many surviving neighbors are present
     *
     * @param     array of values
     * @return    the sum of x and y
     */
    public static int checkCells(int[][] cells,int x,int y)
    {
        int size = cells.length;
        int countAlive = 0;
        for (int i = x - 1; i <= x + 1; i++) {
            for (int j = y - 1; j <= y + 1; j++) {
                if ((i >= 0 && i < size) && (j >= 0 && j < size) && (i!=x && j!=y)) {
                    countAlive += cells[i][j];
                }
            }
        }
        return countAlive;
    }
    public static int[][] initCells(int[][] cells)
    {
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                cells[i][j] = (int) (Math.random()*2);
            }
        }
        return cells;
    }
    public static void dispCells()
    {
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                System.out.print(cells[i][j] + " ");
            }
            System.out.print("\n");
        }
    }
}
```

Check Your Understanding

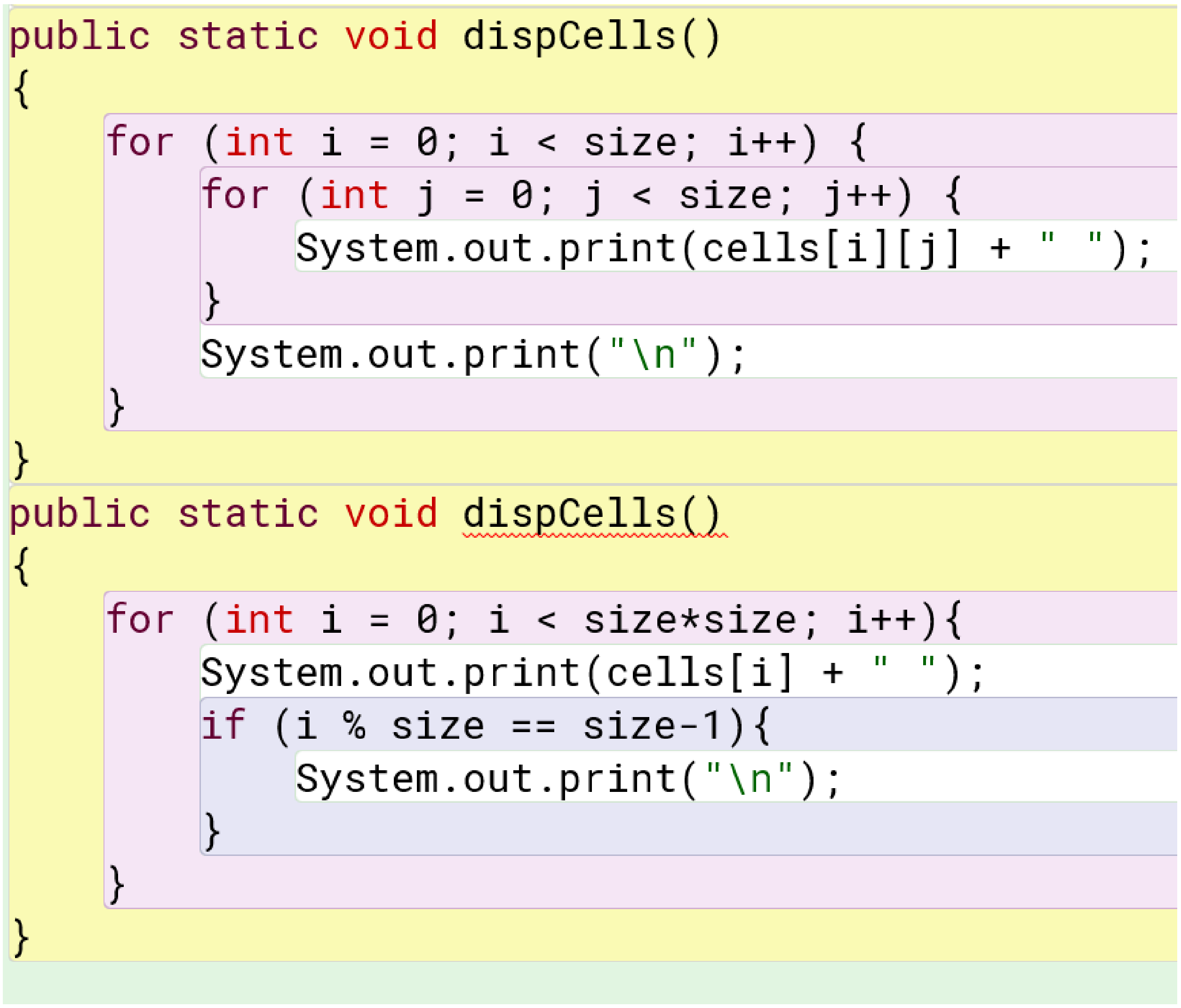
Once your program has been implemented and tested, answer the following questions on your own:

1. Describe the development process used in the completion of the project.

2. Provide the method header for one method that you implemented that takes at least one parameter. Explain why you chose the given parameters, including type, and why you made the method static or non-static. How would your code have been affected if you had made a different decision?

3. Provide a code segment where the elements in a data structure are traversed. Other than specific syntax, explain how using a different data structure would change the complexity of your code. Provide an equivalent code segment to the one included above that uses a different data structure.

1. I started off thinking about how to set up my data structure. I could use a square 2d array, and that would easily hold all my data, and would allow for easy access to that data. I moved onto how I would start off my program, initializing all the cells. I decided on using a random number to do this, choosing from 0 or 1 equally. Then I needed to make the method to check each cells neighbors, and ended up on a handy approach using nested loops to check all 8 neighbors if they exist. From here I had the two methods necessary for the program, but I needed a way to show the user what was happening. I started my last method to display the cells, and this would just print out the array. Moving back to the main method, I completed one iteration of the 2d array, and based on the number of living neighbors and its current status, the cell was modified accordingly. The original and final cell patterns are displayed to the user.
2. public static int checkCells(int[][] cells, int x, int y)
   The parameters, from left to right, cells, x, and y, were all chosen for their specific needs. For example, cells was the 2d array necessary to store my data, and so I used a 2d array for the method, as it needed to use an array input rather than the global variable cells. x and y were both indeces, and must have been integers. I used a static method as to generalize it for any object conway that is to be initialized from the class. My code in this context would not be affected as the variables are all global.
3. I could turn my 2d array into a giant 1d array, this would make it really hard to traverse through rows and columns to get neighbors easily. For my dispCells method, I would instead iterate by a giant 1d array, as shown below.



```java
public static void dispCells()
{
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            System.out.print(cells[i][j] + "  ");
        }
        System.out.print("\n");
    }
}
public static void dispCells()
{
    for (int i = 0; i < size*size; i++){
        System.out.print(cells[i] + "  ");
        if (i % size == size-1){
            System.out.print("\n");
        }
    }
}
```