CS2014 - HUGH GIBBONS
PROGRAMMING TECHNIQUES
ASSIGNMENT: THE GAME OF LIFE
TADHG RIORDAN 12309240

My program starts by defining a number of constant integer types which make the program a lot easier to understand. However, I did not make the puzzle width, puzzle height, generation and variable game dimension final integers as these are variable as to the choice of the user and would be finalized later.

In the main, I begin by defining a number of variables unique to the main execution of the program. The program then adds all of the dimensions of the game to a string array. This array is then used when the user gets a choice as to the size of the game. A scanner reads the first number selected by the user and creates a 2-dimensional game based on this dimension. The string array 'gridList' is then given a size; this stores the number of grids based on the dimensions chosen by the user. It is then filled with a blank white square at each position.

The next block of code (75-79) creates a JList for the program. I decided to use a JList as it allows my Graphical User Interface to handle multiple selections from the user and for the GUI to output correctly(setVisableRowCount). I then create a JScrollPane from this.

The user is now presented with the game, and is given the chance to select multiple cells for the game simulation. An array is created which takes the indices of each of the cells selected, and one by one, passes it to the **setUpPeople** function. When finished, the board now has it's first generation and is ready for simulation. The **printBoard** function is then called which prints the board to screen.

The final section of the main function continuously calls my **checkChanges** function and passes the current generation to this function. This will continue happening until the board contains only zeros and therefore every 'person' has died. Alternatively if the game is in the situation where it is infinitely doing the same thing this loop will never end.

Here is an overview of each function:

makeBlankBoard: this function of type 2 dimensional int array is passed a 2 dimensional array and makes each element of it zero, in terms of the game 'DEAD'. This function is used in the game initialization.

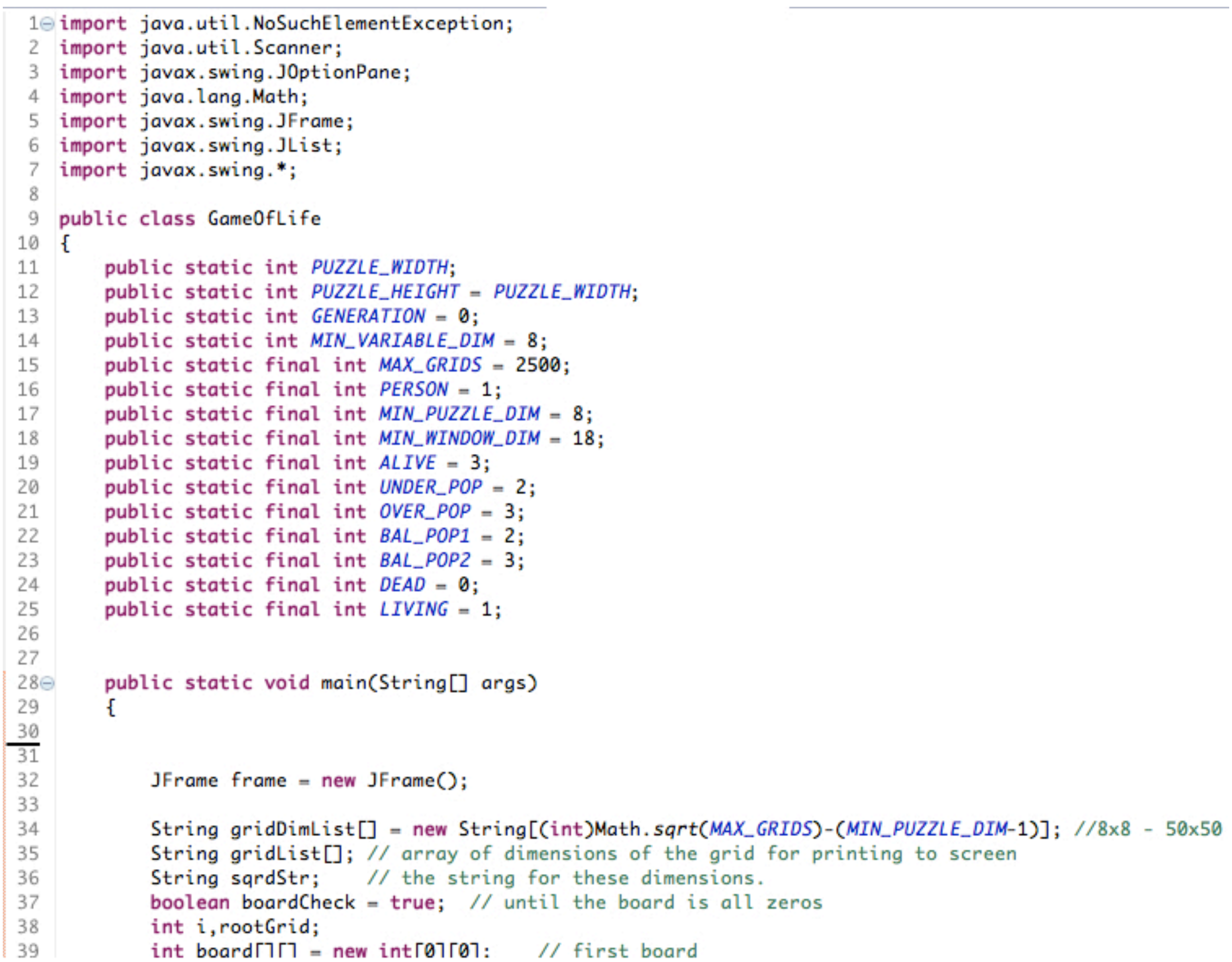
setUpPeople: this function of type 2 dimensional int array is passed a 2 dimensional array and a game index, finds the row and column number of this index, and makes the row and column position of the 2 dimensional array 1, eg. 'ALIVE'. This is used when the user selects the live cells in the game.

checkChanges: this function of type 2 dimensional int array is passed a 2 dimensional array. This is the main function in the program and it's purpose is the find the next generation in the Game Of Life. Each element of the array has all of the elements around it checked for 1. If any element is 1, the integer 'count' is incremented. The value of count determines what happens to the current cell. This determination is based on the rules of the game itself. This continues until each cell is dealt with.

printBoard: this function of type 2 dimensional int array is passed a 2 dimensional array and prints it out. Each live cell (== 1) is printed with a black square and each dead cell (== 0) is printed with a blank square.

Overall I found this to be a fun and moderately challenging assignment. The biggest problem I had was with the GUI as I was entering into new territory, but ultimately I got a lot from doing the assignment.

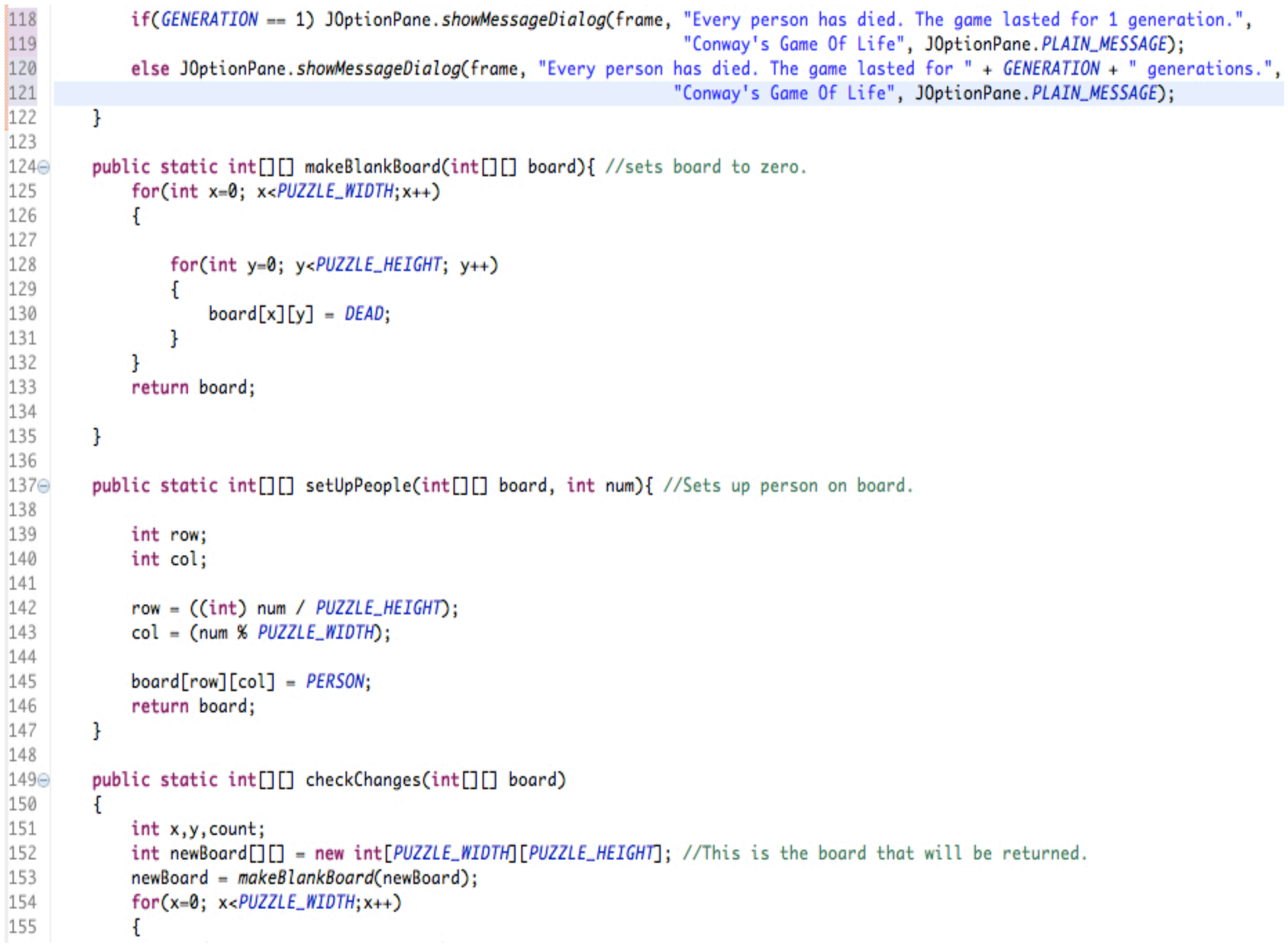
The following is the code used for the program:



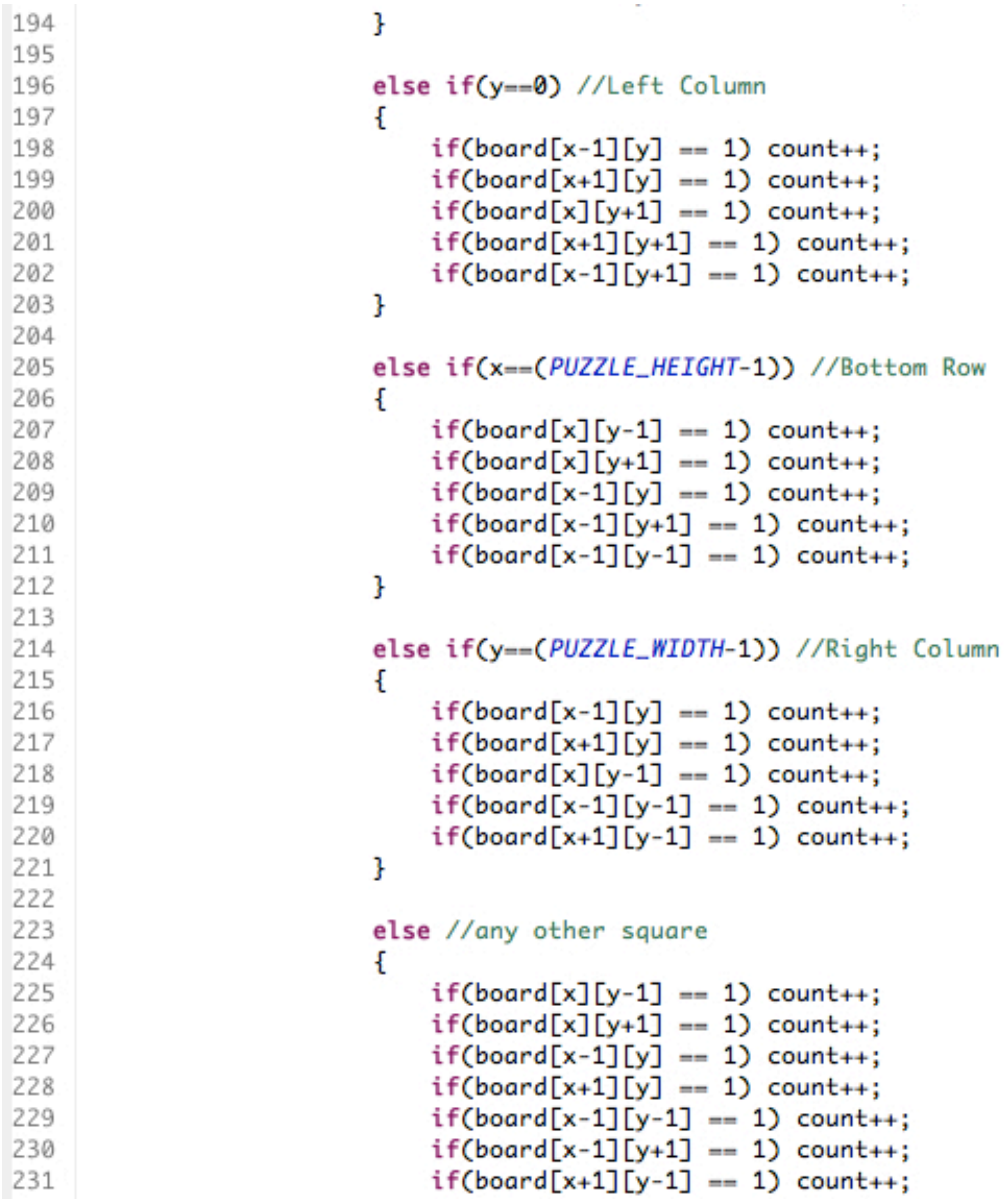
```java
import java.util.NoSuchElementException;
import java.util.Scanner;
import javax.swing.JOptionPane;
import java.lang.Math;
import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.*;

public class GameOfLife
{
    public static int PUZZLE_WIDTH;
    public static int PUZZLE_HEIGHT = PUZZLE_WIDTH;
    public static int GENERATION = 0;
    public static int MIN_VARIABLE_DIM = 8;
    public static final int MAX_GRIDS = 2500;
    public static final int PERSON = 1;
    public static final int MIN_PUZZLE_DIM = 8;
    public static final int MIN_WINDOW_DIM = 18;
    public static final int ALIVE = 3;
    public static final int UNDER_POP = 2;
    public static final int OVER_POP = 3;
    public static final int BAL_POP1 = 2;
    public static final int BAL_POP2 = 3;
    public static final int DEAD = 0;
    public static final int LIVING = 1;


    public static void main(String[] args)
    {


        JFrame frame = new JFrame();

        String gridDimList[] = new String[(int)Math.sqrt(MAX_GRIDS)-(MIN_PUZZLE_DIM-1)]; //8x8 - 50x50
        String gridList[]; // array of dimensions of the grid for printing to screen
        String sqrdStr;    // the string for these dimensions.
        boolean boardCheck = true;  // until the board is all zeros
        int i,rootGrid;
        int board[][] = new int[0][0];    // first board
```

```java
Scanner scanner;
String input;


for(i=0; i<gridDimList.length;i++){     //sets each position of gridDimList to a grid dimension
    sqrdStr = MIN_VARIABLE_DIM + " x " + MIN_VARIABLE_DIM;
    gridDimList[i] = sqrdStr;
    MIN_VARIABLE_DIM++;
}

try{

JOptionPane.showMessageDialog(frame, "Conway's Game of Life - 1970\nProgrammed by: Tadhg Riordan",
                                     "Conway's Game Of Life", JOptionPane.PLAIN_MESSAGE);
JOptionPane.showMessageDialog(frame, "Rules:\n1. Select a game size.\n2. Select live game cells (Hold CMD/CTRL to select multiple cells).\n\n"
                                     + "The game continues through a set of generations.\nA living cell with less than two neighbours dies of "
                                     + "underpopulation.\n" + "A living cell with two or three neighbours lives to the next generation.\n" +
                                     "A living cell with more than three neighbours dies of overpopulation." +
                                     "\nAny dead cell with exactly three neighbours comes alive.",
                                     "Conway's Game Of Life", JOptionPane.PLAIN_MESSAGE);

input = (String) JOptionPane.showInputDialog(frame,"Choose the number of grids for the game:",
                                            "Conway's Game Of Life",JOptionPane.PLAIN_MESSAGE, null, gridDimList, gridDimList[0]);
scanner = new Scanner(input);
rootGrid = scanner.nextInt();
PUZZLE_HEIGHT = rootGrid;
PUZZLE_WIDTH = rootGrid;

board = new int[PUZZLE_WIDTH][PUZZLE_HEIGHT];
board = makeBlankBoard(board);

gridList = new String[rootGrid*rootGrid];
for(i=0; i<gridList.length;i++)
{
        gridList[i] = "\u2B1C ";
}


JList list = new JList(gridList);
```

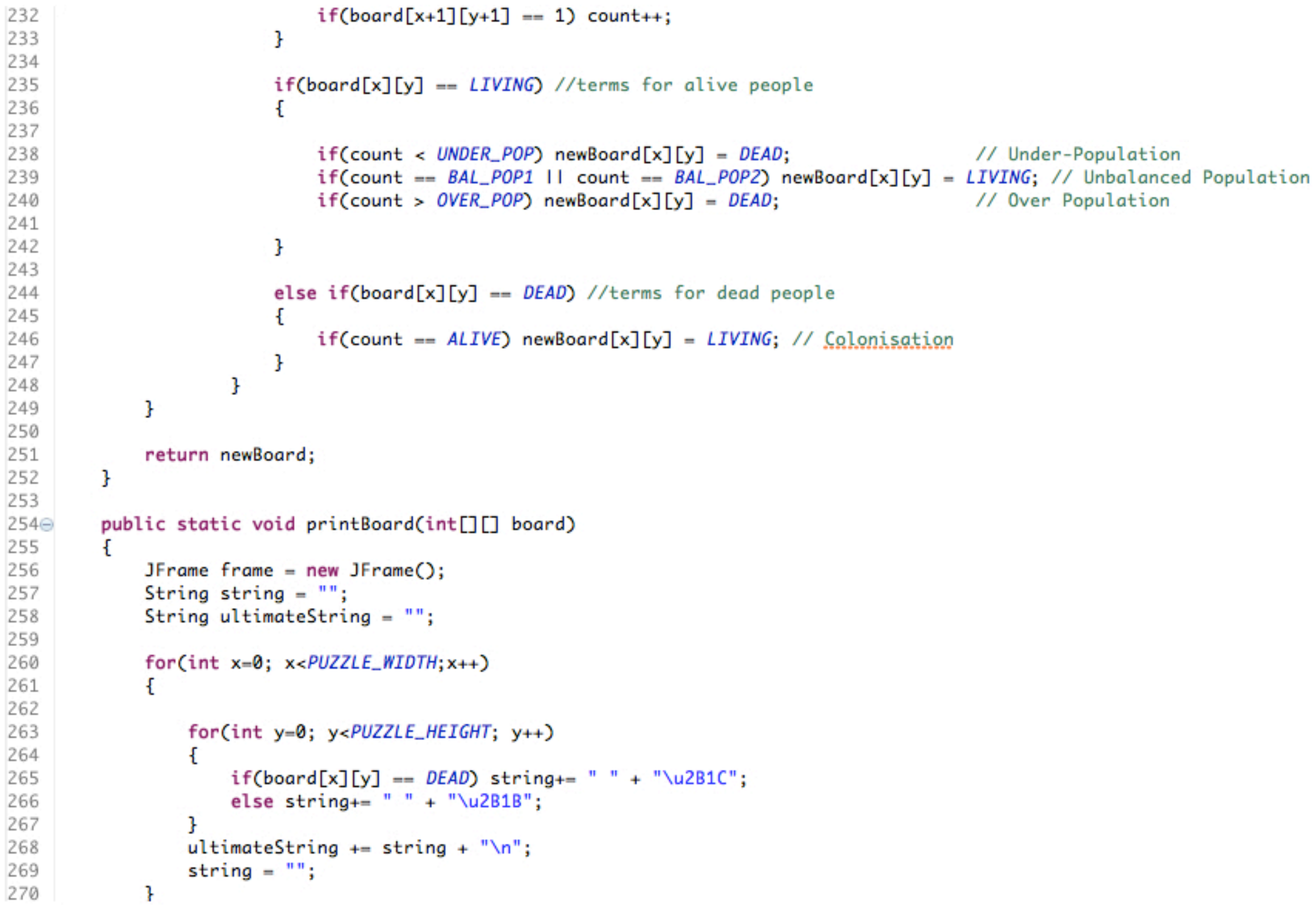
```java
        list.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
        list.setLayoutOrientation(JList.HORIZONTAL_WRAP);
        list.setVisibleRowCount(PUZZLE_HEIGHT);
        JScrollPane listScroller = new JScrollPane(list);

        JOptionPane.showMessageDialog(null, listScroller,"Conway's Game Of Life", JOptionPane.PLAIN_MESSAGE);
        int[] selected = list.getSelectedIndices(); //creates an array of the indexes of grids highlighted
        for(i=0; i<selected.length;i++)
        {
            board = setUpPeople(board,selected[i]);
        }

        printBoard(board);

        }catch(NoSuchElementException noElem){}
        catch(NullPointerException cancel){
            System.exit(0);
        }

        while(boardCheck == true)
        {

            GENERATION++;
            board = checkChanges(board);
            printBoard(board);

            boardCheck = false;
            for(int x=0; x<PUZZLE_WIDTH;x++)
            {
                for(int y=0; y<PUZZLE_HEIGHT; y++)
                {
                    if(board[x][y] == LIVING)
                    {
                        boardCheck = true;
                    }
                }
            }
        }

```



```
        if(GENERATION == 1) JOptionPane.showMessageDialog(frame, "Every person has died. The game lasted for 1 generation.",
                                                                "Conway's Game Of Life", JOptionPane.PLAIN_MESSAGE);
        else JOptionPane.showMessageDialog(frame, "Every person has died. The game lasted for " + GENERATION + " generations.",
                                                               "Conway's Game Of Life", JOptionPane.PLAIN_MESSAGE);
    }

    public static int[][] makeBlankBoard(int[][] board){ //sets board to zero.
        for(int x=0; x<PUZZLE_WIDTH;x++)
        {

            for(int y=0; y<PUZZLE_HEIGHT; y++)
            {
                board[x][y] = DEAD;
            }
        }
        return board;

    }

    public static int[][] setUpPeople(int[][] board, int num){ //Sets up person on board.

        int row;
        int col;

        row = ((int) num / PUZZLE_HEIGHT);
        col = (num % PUZZLE_WIDTH);

        board[row][col] = PERSON;
        return board;
    }

    public static int[][] checkChanges(int[][] board)
    {
        int x,y,count;
        int newBoard[][] = new int[PUZZLE_WIDTH][PUZZLE_HEIGHT]; //This is the board that will be returned.
        newBoard = makeBlankBoard(newBoard);
        for(x=0; x<PUZZLE_WIDTH;x++)
        {
            for(y=0; y<PUZZLE_HEIGHT;y++)
            {
                count = 0;
                    if(x==0 && y==0)        //top left
                    {
                        if(board[x][y+1] == 1) count++;
                        if(board[x+1][y] == 1) count++;
                        if(board[x+1][y+1] == 1) count++;
                    }

                    else if(x==(PUZZLE_WIDTH-1) && y==0) //bottom left
                    {
                        if(board[x-1][y] == 1) count++;
                        if(board[x][y+1] == 1) count++;
                        if(board[x-1][y+1] == 1) count++;
                    }

                    else if(x==0 && y==(PUZZLE_HEIGHT-1))//top right
                    {
                        if(board[x+1][y] == 1) count++;
                        if(board[x][y-1] == 1) count++;
                        if(board[x+1][y-1] == 1) count++;
                    }

                    else if(x==(PUZZLE_WIDTH-1) && y==(PUZZLE_HEIGHT-1)) //bottom right
                    {
                        if(board[x-1][y] == 1) count++;
                        if(board[x][y-1] == 1) count++;
                        if(board[x-1][y-1] == 1) count++;
                    }

                    else if(x==0) //Top Row
                    {
                        if(board[x][y-1] == 1) count++;
                        if(board[x][y+1] == 1) count++;
                        if(board[x+1][y] == 1) count++;
                        if(board[x+1][y+1] == 1) count++;
                        if(board[x+1][y-1] == 1) count++;
```



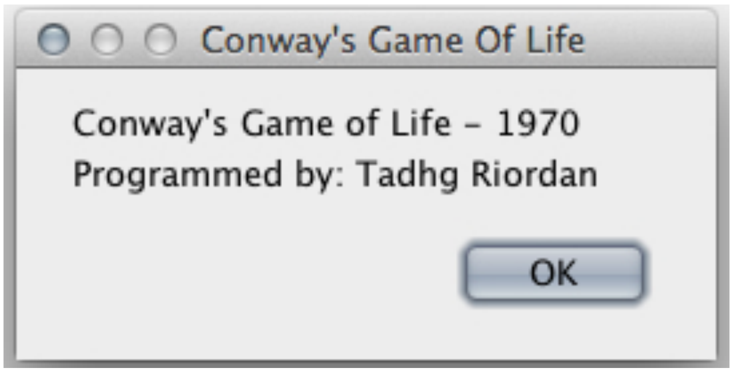
```
}

else if(y==0) //Left Column
{
    if(board[x-1][y] == 1) count++;
    if(board[x+1][y] == 1) count++;
    if(board[x][y+1] == 1) count++;
    if(board[x+1][y+1] == 1) count++;
    if(board[x-1][y+1] == 1) count++;
}

else if(x==(PUZZLE_HEIGHT-1)) //Bottom Row
{
    if(board[x][y-1] == 1) count++;
    if(board[x][y+1] == 1) count++;
    if(board[x-1][y] == 1) count++;
    if(board[x-1][y+1] == 1) count++;
    if(board[x-1][y-1] == 1) count++;
}

else if(y==(PUZZLE_WIDTH-1)) //Right Column
{
    if(board[x-1][y] == 1) count++;
    if(board[x+1][y] == 1) count++;
    if(board[x][y-1] == 1) count++;
    if(board[x-1][y-1] == 1) count++;
    if(board[x+1][y-1] == 1) count++;
}

else //any other square
{
    if(board[x][y-1] == 1) count++;
    if(board[x][y+1] == 1) count++;
    if(board[x-1][y] == 1) count++;
    if(board[x+1][y] == 1) count++;
    if(board[x-1][y-1] == 1) count++;
    if(board[x-1][y+1] == 1) count++;
    if(board[x+1][y-1] == 1) count++;
```



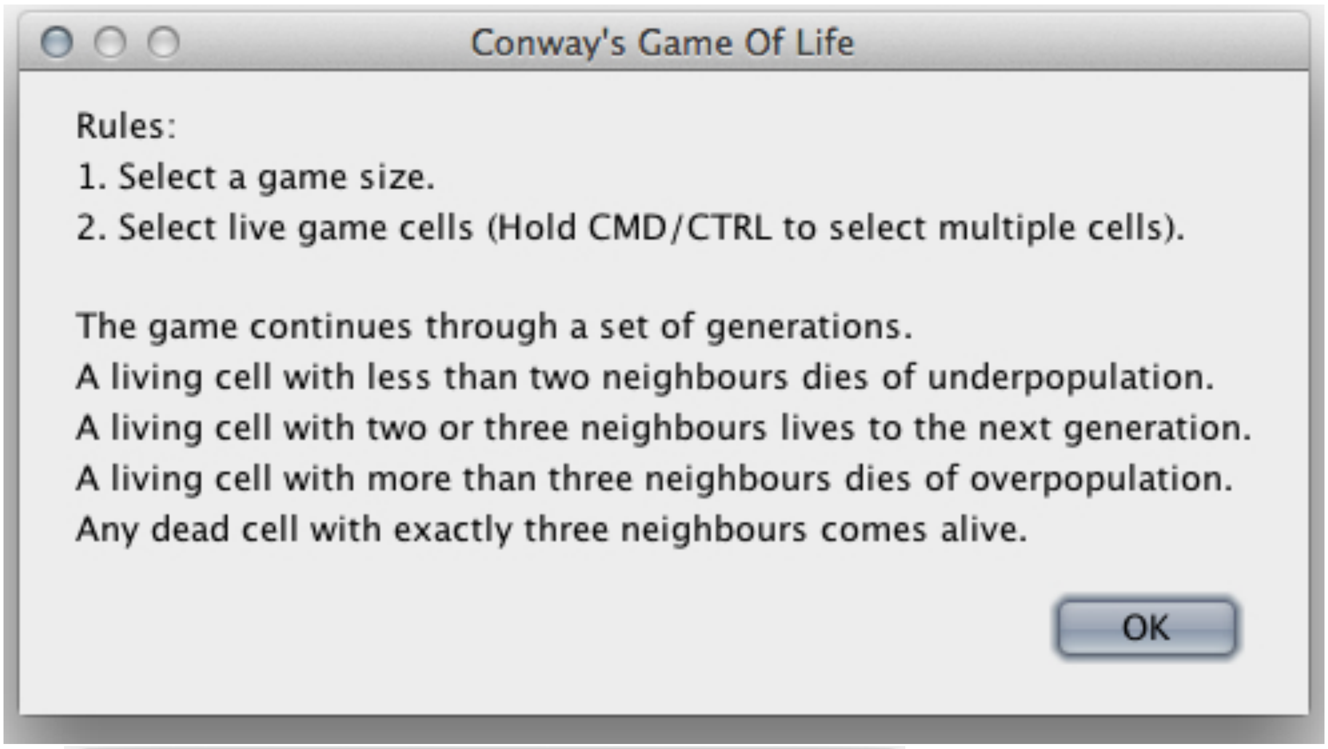
```java
                        if(board[x+1][y+1] == 1) count++;
                    }

                    if(board[x][y] == LIVING) //terms for alive people
                    {

                        if(count < UNDER_POP) newBoard[x][y] = DEAD;                // Under-Population
                        if(count == BAL_POP1 || count == BAL_POP2) newBoard[x][y] = LIVING; // Unbalanced Population
                        if(count > OVER_POP) newBoard[x][y] = DEAD;                 // Over Population

                    }

                    else if(board[x][y] == DEAD) //terms for dead people
                    {
                        if(count == ALIVE) newBoard[x][y] = LIVING; // Colonisation
                    }
                }
        }

        return newBoard;
    }

    public static void printBoard(int[][] board)
    {
        JFrame frame = new JFrame();
        String string = "";
        String ultimateString = "";

        for(int x=0; x<PUZZLE_WIDTH;x++)
        {

            for(int y=0; y<PUZZLE_HEIGHT; y++)
            {
                if(board[x][y] == DEAD) string+= " " + "\u2B1C";
                else string+= " " + "\u2B1B";
            }
            ultimateString += string + "\n";
            string = "";
        }
```

```java
JOptionPane.showMessageDialog(frame,"Generation: " + GENERATION + "\n"
+ ultimateString, "Conway's Game Of Life", JOptionPane.PLAIN_MESSAGE);
 }
}
```

The following images show a run of the program having selected a 12x12 grid.

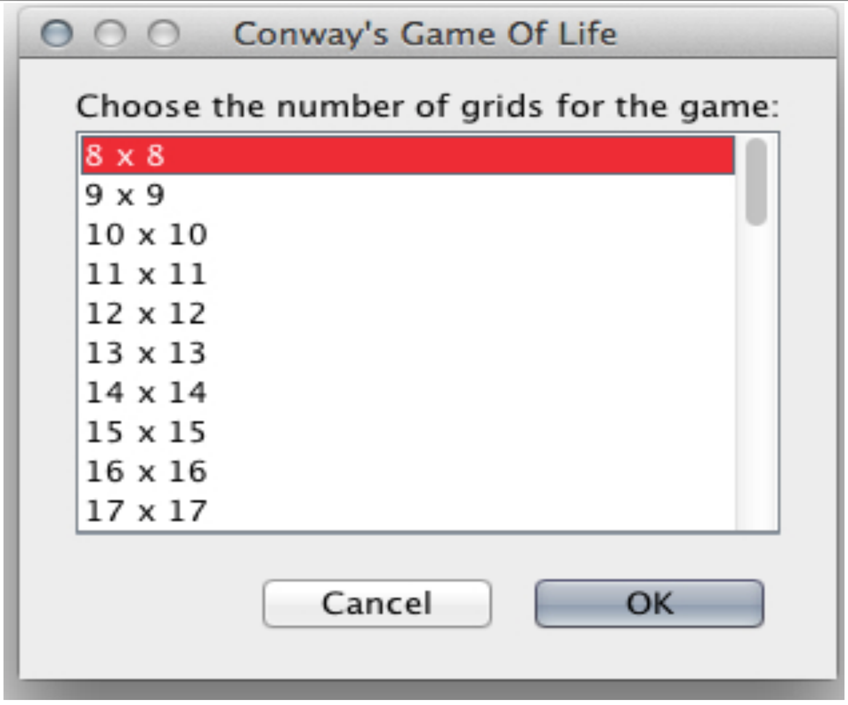

Conway's Game Of Life

Conway's Game of Life – 1970
Programmed by: Tadhg Riordan


Conway's Game Of Life

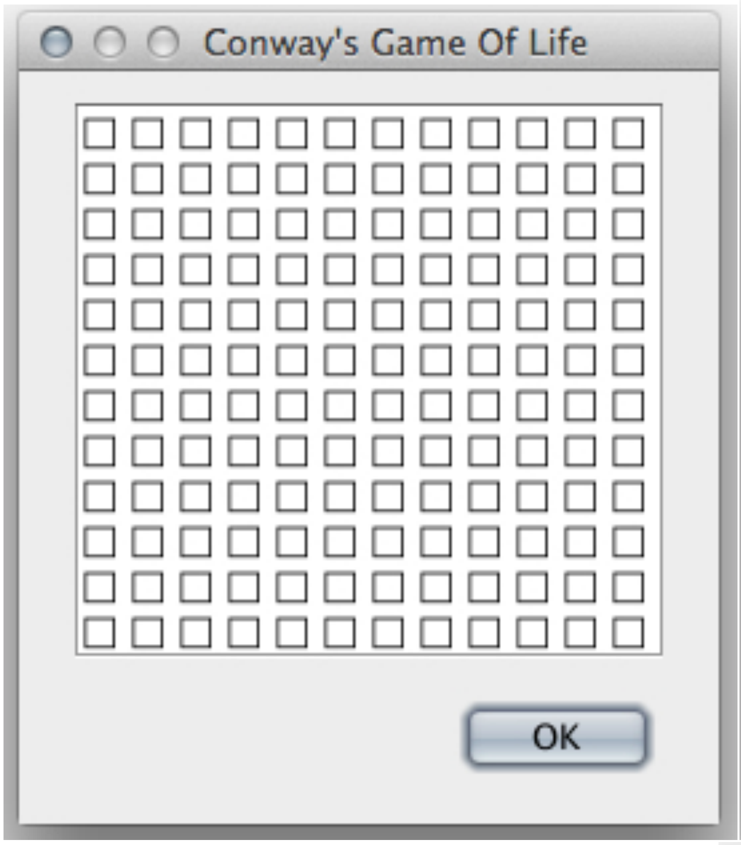
Rules:
1. Select a game size.
2. Select live game cells (Hold CMD/CTRL to select multiple cells).

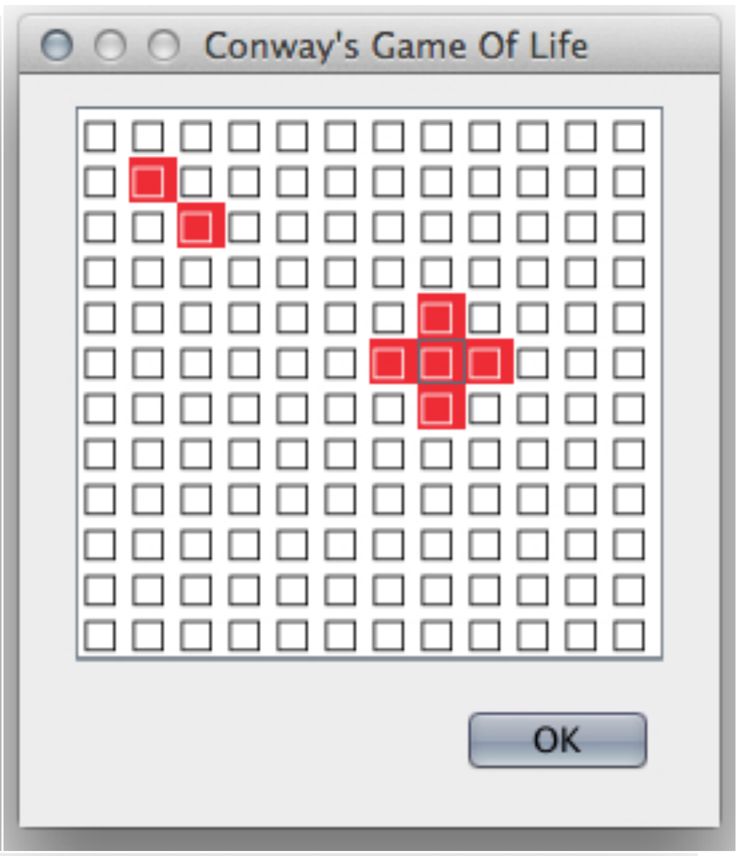
The game continues through a set of generations.
A living cell with less than two neighbours dies of underpopulation.
A living cell with two or three neighbours lives to the next generation.
A living cell with more than three neighbours dies of overpopulation.
Any dead cell with exactly three neighbours comes alive.

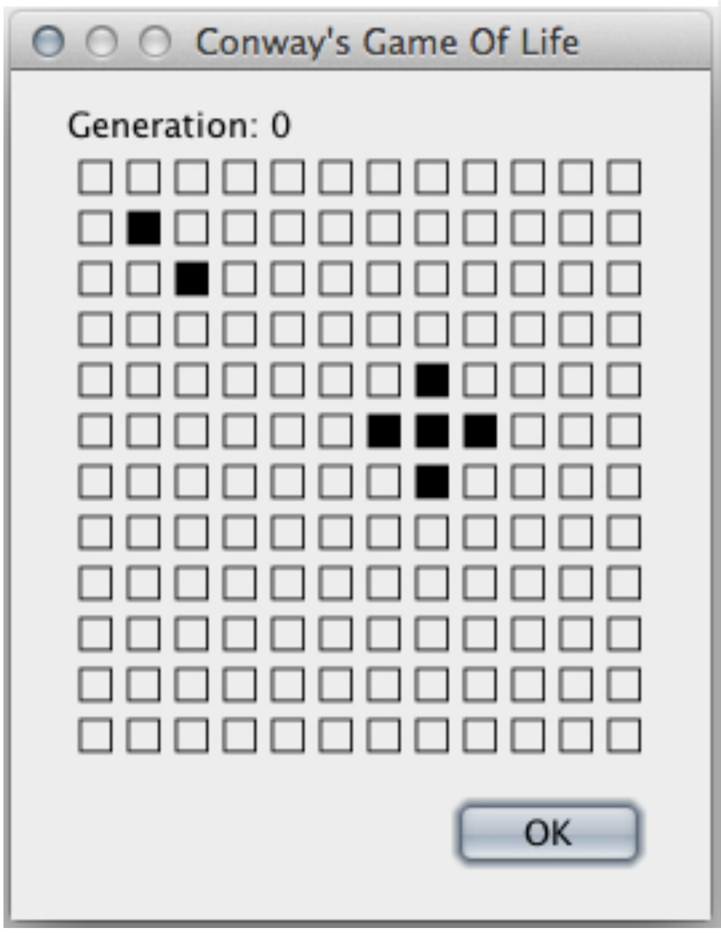

Conway's Game Of Life

Choose the number of grids for the game:

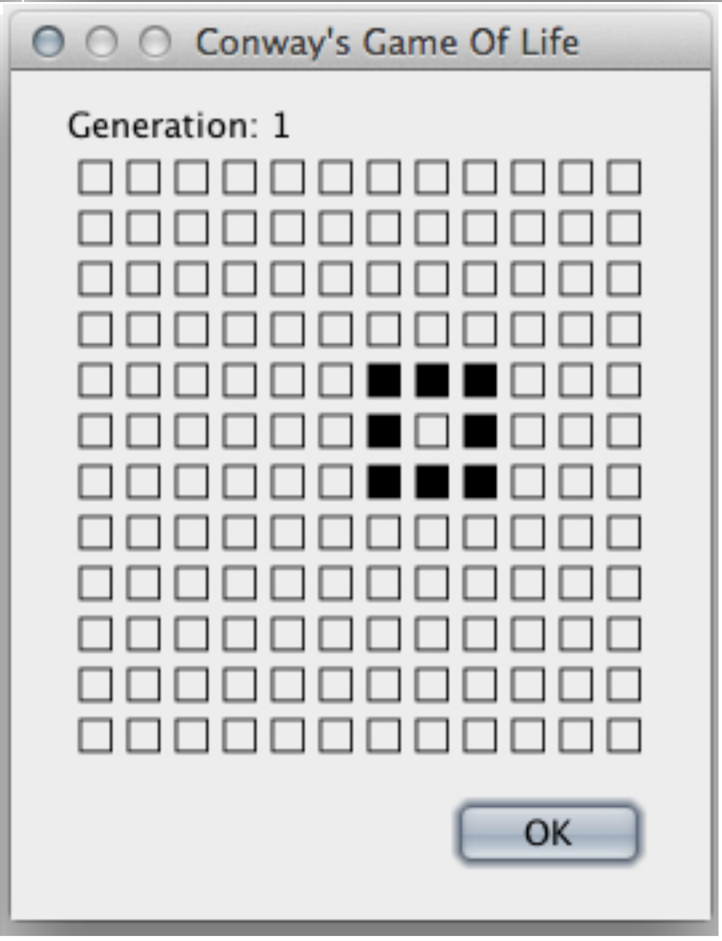
8 x 8
9 x 9
10 x 10
11 x 11
12 x 12
13 x 13
14 x 14
15 x 15
16 x 16
17 x 17



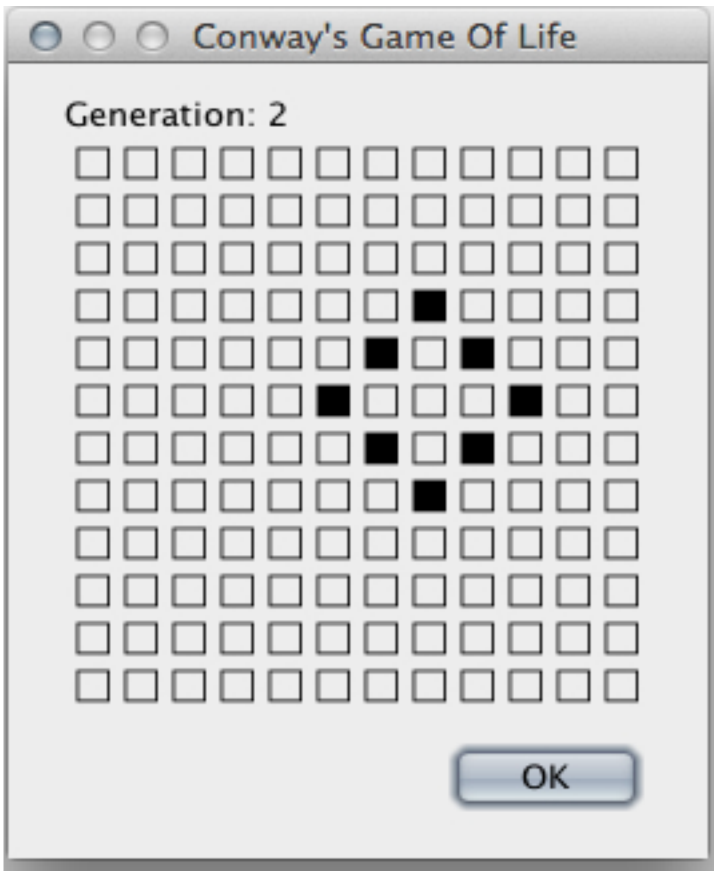


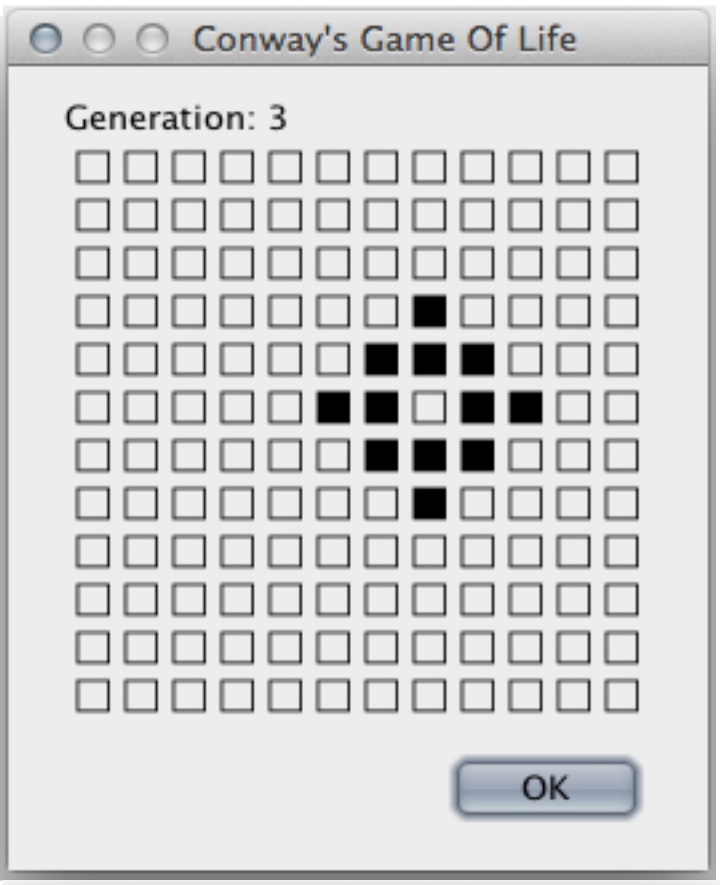


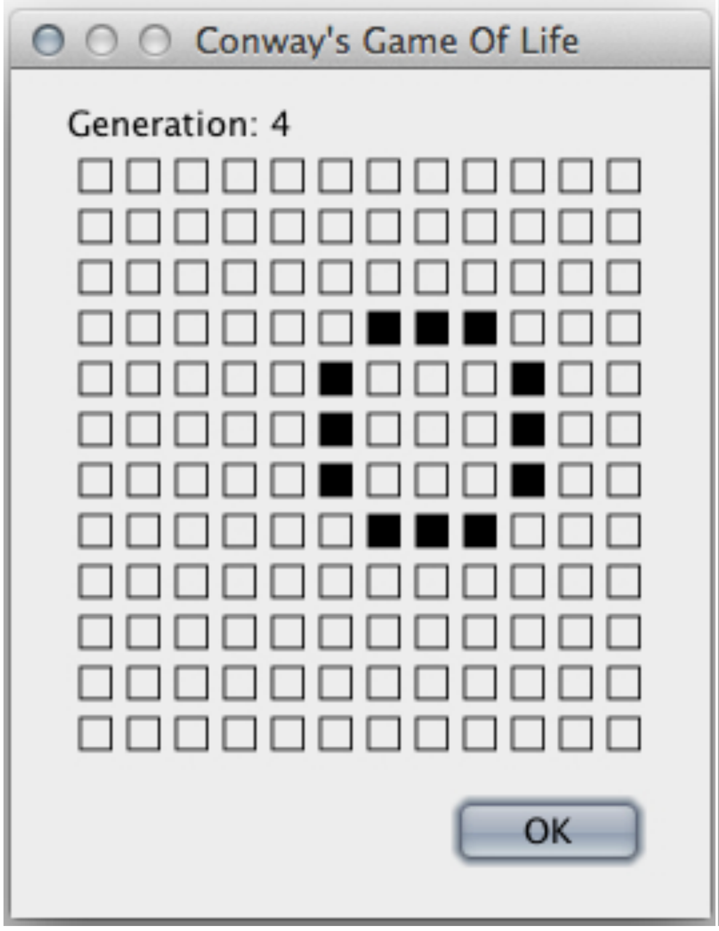
Generation: 0

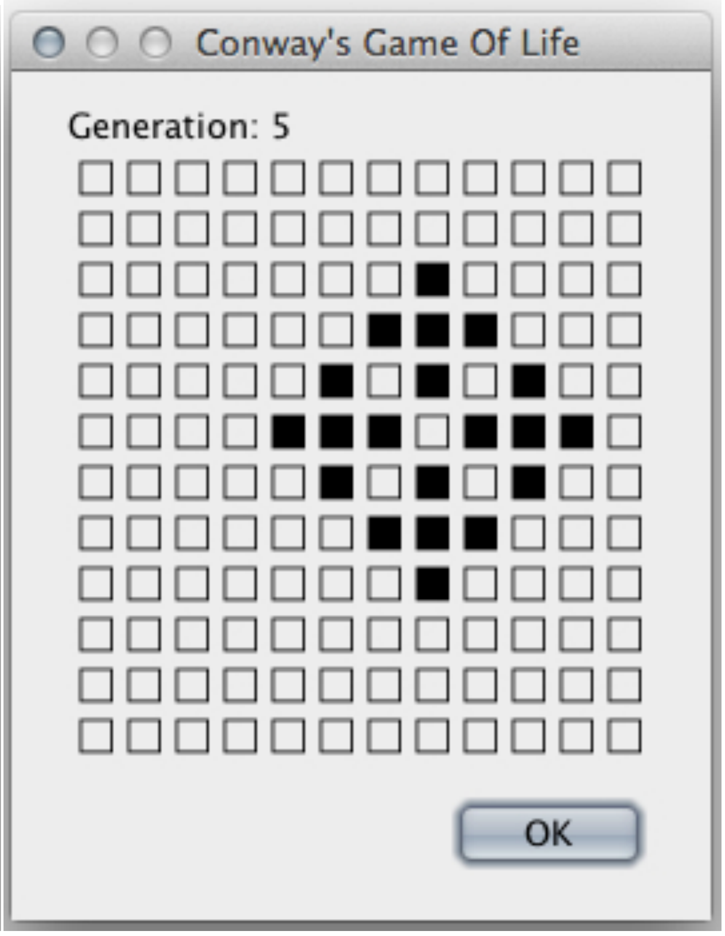


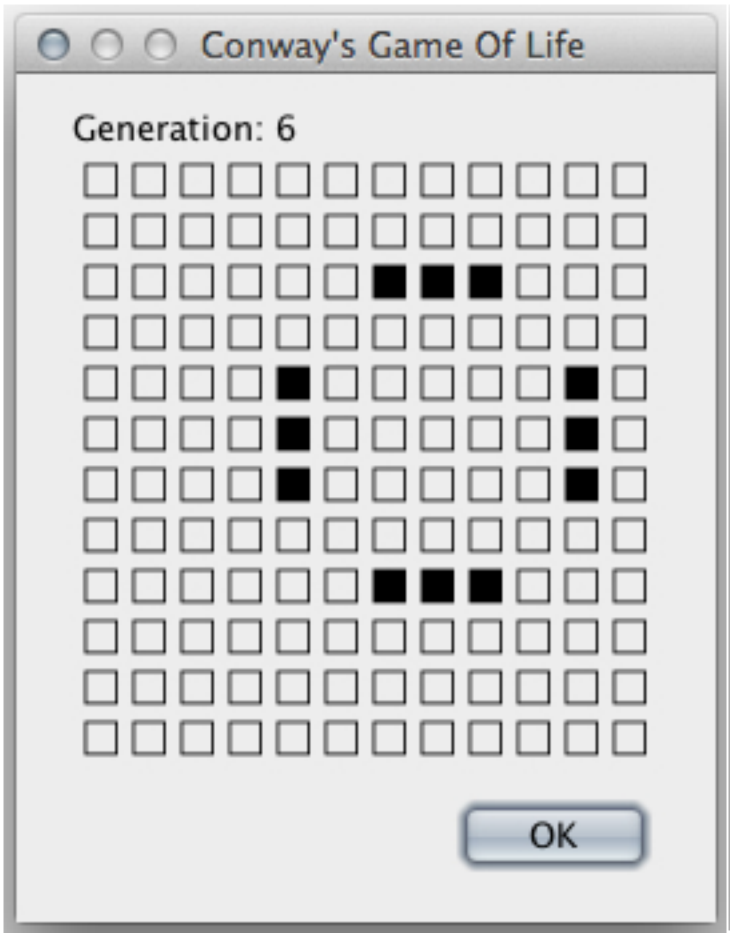
Generation: 1

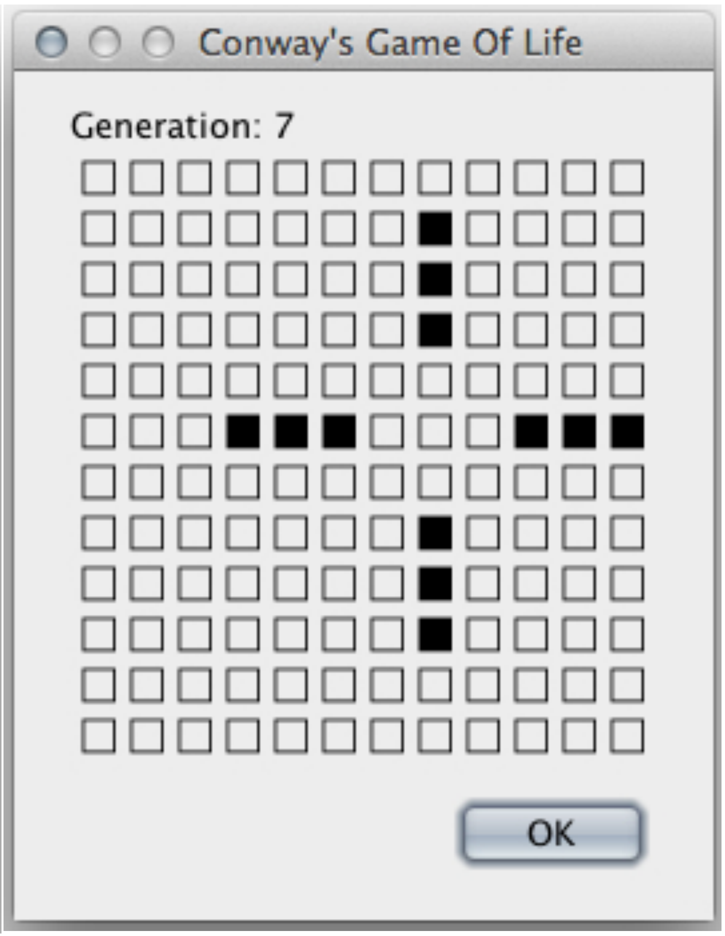

Conway's Game Of Life
Generation: 2
OK


Conway's Game Of Life
Generation: 3
OK


Conway's Game Of Life
Generation: 4
OK


Conway's Game Of Life
Generation: 5
OK


Conway's Game Of Life
Generation: 6
OK


Conway's Game Of Life
Generation: 7
OK