**Game Of Life**

**Introduction**

As an adaptation of Conway's Game of Life for two players, this game has a few different variants. The Game of Life is played on a grid of cells, and each cell in the grid determines whether the player is alive or dead. A series of rules is used to decide whether a cell is alive or dead during each "turn" or "generation" of the game. The game iterates over all of the cells and determines whether a cell is alive or dead based on these criteria. These are the rules that apply to the normal version of the game for a single player:

* Live cells with fewer than two neighbors perish (under population).
* Any live cell with two or three live neighbors lives.
* Any live cell with more than three neighbors dies (over population)
* Any dead cell with exactly three neighbors becomes a live cell.

Cells next to the cell (up to 8 cells) are called neighbors. All cells are told at the same time whether they are living or dead. This project is for a two-player Game of Life. Each cell is either dead or holds a live cell that belongs to Player 1 or Player 2. So, here are the rules for the Game of Life for two people:

* Any live cell with fewer than two live neighbors of either player dies.
* Any live cell with two or three live neighbors of either player lives.
* Any live cell with more than three neighbors of either player dies.
* Any dead cell with exactly three neighbors becomes a live cell owned by the player who has the majority of live neighboring cells.

The game generates new generations until one player has no living cells. The victor has living cells. A tie occurs when both players' cells die simultaneously.

**Technical Requirements**

This is a Java program that will play the two player Game of Life.

The format of the initial generation file is as follows:

1. The first line states the number of rows and columns in the grid, separated by a space (e.g., 5 6)

* 2. The remaining lines describe the initial generation stage. Each grid row has a character for each cell's state. The grid has characters for each column. Specifying cell state:
* . denotes a dead cell.
* 1 denotes a live cell owned by player 1.
* 2 denotes a live cell owned by player 2.

Here is an example of the initial generation file: 10 10

..........

..........

...1......

..111.....

..........

.......22.

......22.2

.......22.

..........

..........

At startup, the application checks the initial generating file for accuracy. An invalid initial generation file does not meet the standard. If the initial generating file is invalid, the application exits with an error message. After reading the initial generation file, the application prints generation status. The generation state includes:

1. The generation number. The initial generation is considered generation 0.
2. The number of live cells owned by player 1.
3. The number of live cells owned by player 2.
4. The grid of live and dead cells following same convention for the state of the cells used for the initial generation file.

For the example initial generation file above, the program prints out the following state:

Generation #0

Player 1 Cells: 4

Player 2 Cells: 7

..........

..........

...1......

..111.....

..........

.......22.

......22.2

.......22.

..........

..........

The program then generates the next state given the initial state and prints out the state in the same format as the initial state:

Generation #1

Player 1 Cells: 4

Player 2 Cells: 7

..........

..........

....1.....

..111.....

..........

.......22.

......22.2

.......22.

..........

..........

The software examines if a player has no living cells. The programme displays the winner and how many cells they have left:

Congratulations, Player 1 Wins! with 11 live cells

Game Over!!

The computer software will print out that there is a tie and then exit the program if neither of the participants has any living cells left with them. In the event that this does not occur, the program will continue to generate new states, print out the state, and determine whether or not there is a winner until either a winner is proclaimed or the program is terminated.