California State Polytechnic University, Pomona
Computer Science Department
[redacted]

Write a program titled "GameOfLife.java" which will implement Conway's Game of Life. Life is a simulation of simple one-celled organisms and therefore typically played on a 2D grid of square cells. Each cell has two possible states: dead or alive. To calculate each new generation of the board, use the following rules:

For spaces already "alive":

- Each cell with less than two neighbors, who are alive, dies of loneliness in the next generation.
- Each cell with more than three neighbors, who are alive, dies of overpopulation in the next generation.
- Each cell with two or three neighbors, who are alive, continues to be alive in the next generation.

For spaces already "dead":

- Each cell with three neighbors, who are alive, comes to life in the next generation.

Note: Every cell has eight neighbors which are the cells that are diagonally, vertically, or horizontally adjacent.

For your program, use a 2D char array to represent the board. Use '0' to represent dead cells of the board and 'X' to represent cells which are alive. Your program will load game board data from a file to begin the game, with the first line of the input file consisting of two ints representing the number of columns and rows for the board.

For example, a sample input file may appear as follows:

```
9 10
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0
0 0 1 0 0 0 0 1 0 0
0 0 1 0 0 0 0 1 0 0
0 0 1 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0
```

Methods to implement:

- The constructor which initializes a new game board by prompting the user for the file name and loading the game board data from the file.
- public int getColumns(), return the number of columns in the game board.
- public int getRows(), return the number of rows in the game board.
- public int getCell(int column, int row), get the value of the cell at given column and row, returning 0 if either the column or the row is outside the bounds of the game board.
- public void setCell(int column, int row, int value), set the value of the cell at given column and row. This method does not have to handle out-of-bounds column or row numbers.

- public void computeNextGeneration(int generation), creates a temporary 2D array to compute the next iteration of the board containing the next generation of organisms, as determined by the Rules of Life. Then updates the board to represent the next generation. The argument passed in represents the number of generations the user wants to compute. To compute each generation, the method should recursively call itself and decrement the integer until it terminates when there are no more generations left to compute.
- public void print(), which prints out the board to the console.

The program should prompt the user for the file name and how many generations to compute, then print out each generation to the screen.

The following shows an example of the program running
(bolded areas represent the user's input):

```
Enter file name: sample.life
Enter how many generations to compute: 4

Generation 1

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0
0 0 1 0 0 0 0 1 0 0
0 0 1 0 0 0 0 1 0 0
0 0 1 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0

Generation 2

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 1 1 0 0 0
0 0 0 1 0 1 0 1 0 0
0 1 1 1 0 0 1 1 1 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0 0 0

Generation 3

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 1 0 1 0 0 1 0
0 0 1 1 1 0 1 1 1 0
0 0 1 0 0 1 0 0 0 0
0 0 0 0 0 1 1 0 0 0
```

```
0 0 0 0 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0

Generation 4

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 1 1 0 1 0 0 1 0
0 0 1 0 0 0 1 1 1 0
0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0
```

Run the program with one of the inputfiles provided as well as one of your own (with a reasonable amount of generations) to make sure it runs correctly.