ITI 1120 Fall 2013 - Assignment 4

Available: Nov 6, 2013
Due: Sun, Nov 24, 22:00

Instructions

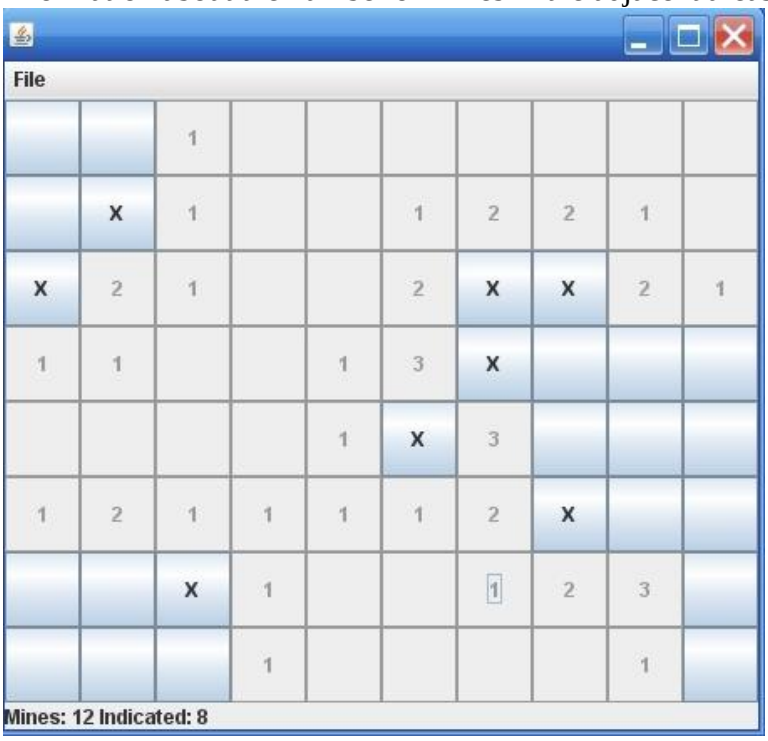
This assignment is to be done in TEAMS OF TWO PEOPLE. The assignment should be submitted through the Virtual Campus (Backboard Learn). Create a group in the online system for the two students in the group. Only one member of the team needs to submit, but please ensure that the identification material contains the information for both team members (as a comment in the Java file). If both members of the team submit, they will post in the same group, and the most recent version will be marked. Submit only the file MineSweeperLib.java

Marking Scheme (total 100 marks)

- **Regulations and Standards: 10 marks**
- Question 1: 30 marks
- Question 2: 35 marks
- Question 3: 25 marks
- Bonus: maximum 5 marks

Introduction to a game of MineSweeper

The MineSweeper game has the goal of locating mines hidden in a rectangular field, using only information about the number of mines in the adjacent areas.



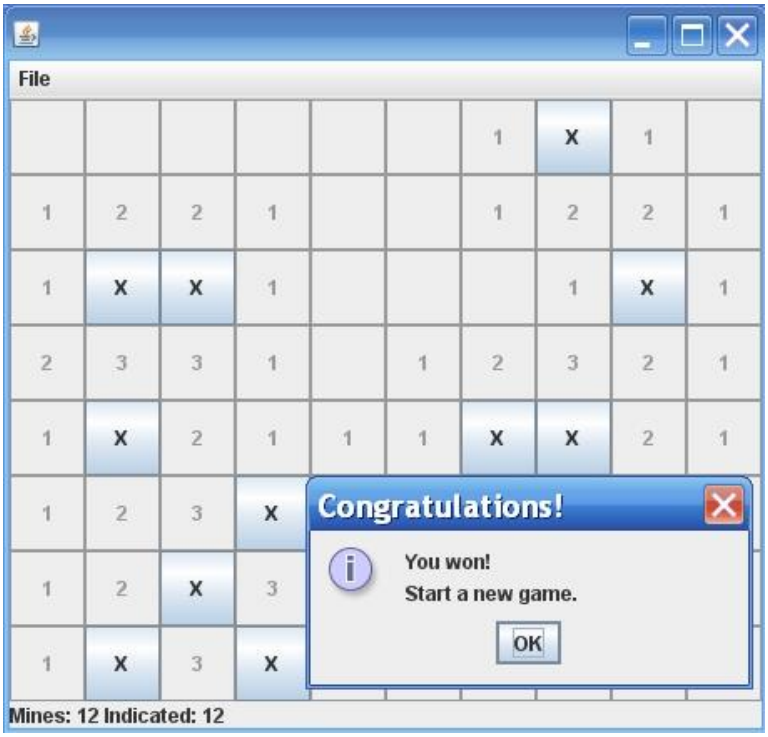
The image above shows a mine field, where **X** represent the tentatives of the player to guess where true mines are located, and the digits (provided by the game) represent the number of real mines detected in the adjacent areas. The player uses the mouse to play (left click to open a cell and right click to mark it with X as a probable mine).

The field is rectangular and its length and width are determined by the player. The number of mines is also chosen by the player:

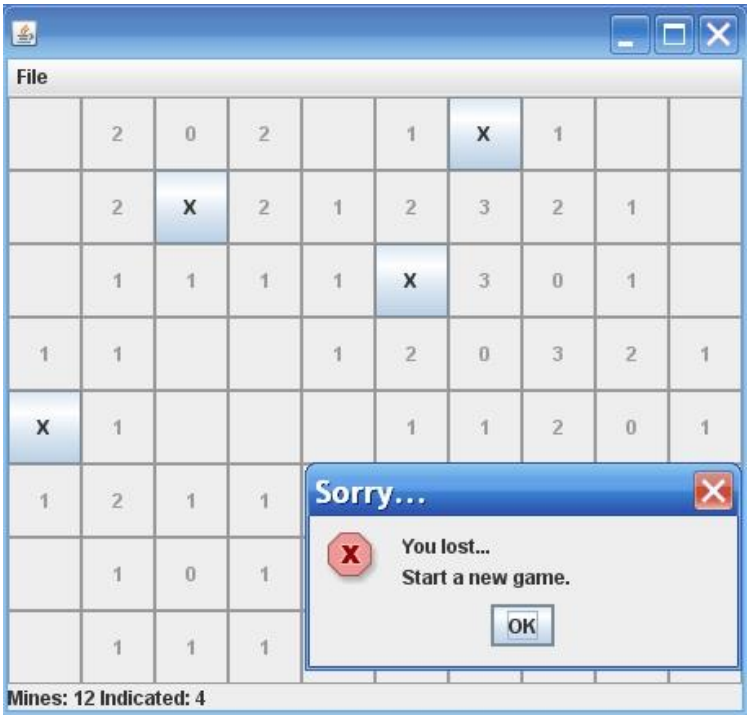
```
ITI1120 Assignment 4
MineSweeper game!
----------------

Please enter the length of the field (between 1 and 20) :8
Please enter the width of the field (between 1 and 20) :25
Invalid value...
Please enter the width of the field (between 1 and 20) :10
Please enter the number of mines (between 1 and 80) :0
Invalid value...
Please enter the number of mines (between 1 and 80) :12
Super! We can start...
```

The game will indicate if you succeed to detect all the mines correctly...



...or, if your guess was wrong and you stepped on a mine! (The mine field is then displayed, and '0' values indicate the positions of the remaining mines).



Your task consists in completing several methods in a given incomplete code for the MineSweeper game. The headers of the methods are provided in MineSweeperLib.java (with the return values temporarily empty so that the program can compile). There are also several JUnit tests that allow you to test each of your methods.

There are five files provided:

- http://www.site.uottawa.ca/~diana/iti1120/A4_Fall2013/MineSweeper.java : The main program, which you should not modify. It interacts with the user (the player) for determining the size of the mine field and the number of mines (note the generic method for validation!) and creates a graphical interface of the right size.
- http://www.site.uottawa.ca/~diana/iti1120/A4_Fall2013/MineSweeperGUI.class : Takes care of the graphical interface of the game. It calls concepts from the user interface and event management that we did not cover in the course. For this raison, the methods in this class are not given as source code, but they are provided in compiled form only. In their turn, they call methods from the library class that you will complete.
- http://www.site.uottawa.ca/~diana/iti1120/A4_Fall2013/MineSweeperLib.java : (***YOU HAVE TO COMPLETE IT***) A library class with methods for generating a matrix of boolean values indicating the (random) positions of the mines in a field (Q1), for generating a matrix of integers where each cell indicates the number of adjacent mines (Q2), and for determining if all the mines in the field were found (Q3).
- http://www.site.uottawa.ca/~diana/iti1120/A4_Fall2013/MineSweeperLibTest.java : A set of tests for the methods in your library class. You can add more tests, but the provided tests **must be** passed as they are.
- http://www.site.uottawa.ca/~diana/iti1120/A4_Fall2013/MatrixLib.class : Three generic methods for manipulation matrices, used in the tests
    - compareInt: tests the equality between two matrices of integers;

- compareBool: tests the equality between two matrices of Boolean values;
- countTrue: Count the number of "true" in a matrix of Boolean values.

Question 1 (30 points)

Complete the following method from the class MineSweeperLib:

public static boolean [][] generateMineField (int length, int width, int numberMines)

This method generates a new mine field as a matrix of boolean values where a cell is set to "true" to indicate the presence of a mine. The length and the width of the field are provided, and also the desired number of mines (numberMines).
Hint: Use **Math.random()**, **Math.floor()**, and type cast to generate random positions for the mines.

Question 2 (35 points)

Complete the following method from the class MineSweeperLib:

public static int [][] calculeProximity (boolean [][] mineField)

This method generates a matrix of integers where each cell indicates the number of adjacent mines (i.e., that touch this cell directly). Use 0 for a position that corresponds to that of a mine in a field. The method should call (multiple times) a helper method that is given the reference to the mineField matrix, an index for the row and an index for the column, and checks if a mine exists at the position given by these indexes. The method also tests if the indexes are valid – if they are not, it returns zero. If they are valid, the matrix element is tested to return either 0 (the element is false, i.e., no mine is there) or 1 (the element is true, i.e., the mine exits in that position). *Hint:* Pay attention to the boundaries of the mine field.

Question 3 (25 points)

Complete the following method from the class MineSweeperLib:

public static boolean allFound (boolean[][] mineField, boolean[][] tentatives)

This method determines if all the mines were found. The matrix of boolean "tentatives" represents the guesses of the player (*true* if a position represents a potential mine, for example an X in the game). The two matrices have the same size (no need to verify).

Bonus (recursivity)

- At least one of the above methods has to be recursive; if not you will **loose** 5 marks.
- If you have a second recursive method that works, you will get **5 bonus points**!

Enjoy the game!