Arman Mottaghi

Pro. David Duvall-Early

Computer Science I

05/09/2014

Minesweeper Project Logic

**public int[][] mines;**

**public int[][] tiles;**

We are instantiating integer arrays mines and tiles which represent the board of the game. They are public because we will need to access them from a GUI class (I haven’t done it yet!).

**public Minesweeper()**

**public Minesweeper(int newRows, int newCols)**

They are simply constructors which will build a game board either with a default size (ie. 9 x 9) or with a user-defined size which will be passed into the constructor’s parameter(s).

**Methods of the Game**

**GetStatus method**

It will determine whether the game is ended or not.

**GetRows & getCols methods**

They are just a getters for the board size.

**getMines method**

This method will return the value of the given index of the mines array (passed into the method parameters). It will return -1 if it’s not a valid tile.

**getBoard method**

It will get a status of a given tile and then print a character depending on what the value is.

**MarkTile method**

The program will use this method when user clicks on a specific tile to change the value of the tile array (i.e put a question mark on a tile that might have a mine).

**toStringMines , toStringTiles & toStringBoard methods**

These methods will print out the board, mines, and tiles of game. We mostly use it as a test before we actually implement the GUI.

**initGame(int newRows, int newCols) method**

This is a method that creates a new game.

**resetTiles method**

It will reset the value of all tiles. It happens when user decides to start a new game.

**placeMines method**

In this method, we have an int variable numMines which keeps track of how many mines that are in the game. It will prevent the game from placing too many mines and making sure that we will have enough mines in the game. While loop runs until we get the max number of mines in the game and at the same time for each row and column (each tile), we generates a random number and if it’s equal to 100, the program places a mine at that specific tile. In this manner, we are randomly distributing mines on the game board.

**calculateClues method**

After we placed all of the mines on the board, this method will go through each tile and if it finds a mile, it will increment the value of nearby tiles in the mines array which will indicates the presence of mines around each tile. In the process of looping, the program will check coordinates of each tile to make sure the nearby tiles are on the board and we will not increment the value of those tiles.

**validIndex(int x, int y) method**

It will check the given coordinates to see whether it’s off the board or not.

**gameWon method**

It will check the status string variable to see if the user has won the game.