Assignment 02: building a custom view to implement the game of minesweeper

introduction:

Note: read the whole assignment brief first before implementing it contains very important information.

In this assignment you will be tasked with building a custom view that will render and implement the rules of minesweeper. Assuming you are not already familiar with minesweeper the premise of the game is simple. You are given a two dimensional grid representing a minefield. However all of the cells are covered so you cannot see what is underneath. To sweep the minefield you must uncover cells one after the other until all non-mined cells are uncovered. Non-mined cells will include a number indicating how many adjacent cells contain a mine relative to that cell. If there are no adjacent mines a number will not be displayed.

Players should be able to use a combination of a logical reasoning and a little bit of guesswork to determine what cells contain mines. If a cell contains a mine they should be able to mark it as such. Any time a player clicks on a marked cell (usually by accident) it will not uncover. Unless the user explicitly unmarks it.

You will be tasked with building a fully working game of minesweeper runing on a 10x10 board with 20 mines. Players should be able to reset the game, switch between marking and uncovering modes and have some information displayed to them about how many mines they have marked and how many mines there are in total.

If you are not familiar with Minesweeper I would strongly suggest that you play it for about 20 to 30 minutes to make yourself familiar and then get to working on the application.

Please read through the rest of this assignment before you start coding straight away. In the words of Scott Meyers “Weeks of coding saves hours of planning”.

Notes:

Take note with the milestones. If you skip a milestone or have serious errors with a milestone, further milestones will not be considered for correction. e.g. if you skip milestone 03 or have bad errors with it then milestones 04,05,06,…. will not be considered for correction. With regards to errors, depending on their severity I will either deduct 5, 10, or 15% for each bug. Thus it's possible that if you implement all milestones but have 20 small bugs you can end up with a score of 0%. This is to encourage you to build robust software free from bugs.

No structure will be provided for the application so it is up to you to determine what datastructures and algorithms you will need to complete this assignment. You will be expected to use the user login service, JDO objects, and JSPs in order to complete this assignment

For submission you are required to upload an archive of your complete eclipse project to the moodle by 2016-04-10 at 23:55:00. The accepted archive formats are the following six: 7z/zip/rar/tar.gz/tar.bz2/tar.xz.

You are also expected to have a git repository with your work. To use git install the Egit package in eclipse. Git is a source code manager that will serve as a backup of your work and also a rollback facility if you mess up your source. You are expected to make an initial commit immediately after the project is created and at least 1 commit for each milestone (when you finish a milestone at the beginning of your commit comment write “Milestone X:” where X is the number of the milestone.

Penalties:

There will be penalties applied to your score if any of the following conditions are not met. These are silly things that slow down my correction of your work. Thus if you want your result back in the quickest possible time make sure you do not fall foul of these penalties.

- -40% for a missing git repository. Make sure when you go to submit your work that there is a .git/ directory in your project before archiving. I like to see development history on assignments to see how it was made. You will also lose marks for missing commits.
- -30% for non compiling code. I should be able to open your project have it compile and run immediately. You are final year students, non compiling code is a grave sin at your level.
- -10% for a non supported archive format. I have a python script that can unpack and sort all projects downloaded from moodle. Anything not in those archive formats I have to unpack and sort by hand.
- Standard late penalties: submission even a second late after 23:55:00 on submission day will have the late submission penalties applied.

Milestones:

As noted before make sure that you complete milestones properly before moving onto the next one. The percentage in brackets indicates the maximum score you can achieve if you pass that milestone.

01) Define the shell of a custom view class and generate a layout for the main activity consiting of the custom view, two buttons, and two textviews. (5%)

02) Force the custom view to restrict itself to a square size (10%)

03) Draw the initial state of the game board with all cells covered. Black should be used as the fill colour and white lines should separate the cells (20%)

04) Implement basic touch behaviour that will uncover a cell. When a cell is uncovered it should change from a black colour to a grey colour. (35%)

05) Implement methods to place 20 mines randomly in the minefield and render the mines when a cell is uncovered. A cell containing a mine should have a red background with a black M in the foreground. You will fail this milestone if: (50%)

- less than 20 mines are placed (it's possible random number generation will give you the same coordinates)
- the mines are not rendered

06) Implement methods to render the individual numbers that may appear in a cell. 1 should be blue, 2 should be green, 3 should be yellow, and four and above should be red. Also write code to determine the number that should be displayed in each cell when the game is initialised for the first time. (75%)

07) modify the touch behaviour to stop accepting input when a mine is uncovered. It should only reenable input when the user has clicked the reset button. (80%)

08) The other button should switch between displaying "uncover mode" or "marking mode" each time it is clicked. In uncover mode each touch should result in a cell being uncovered. In marking mode each time a covered cell is touched it should change to yellow to denote it is marked, or back to black to denote it is unmarked. Use the two text fields to denote the total number of mines and the number of mines marked. In uncover mode touching a marked cell should do nothing.(100%)