CSSE 220 Minesweeper Programming assignment – team min-project

This assignment will be done by three-person teams. If there the number of students in your section is not a multiple of three, there may be a team of two. My intention is not that you "divide and conquer" as much as that you have someone to talk with as you write and test this program. If you have not already done so, read this short article on Pair Programming and discuss it with your partners: http://en.wikipedia.org/wiki/Pair_programming . In particular, note what it says about who should be the driver if you are a "mismatched pair".

Everything that you submit for this project should be understood by all team members. It is your responsibility to (a) Not submit anything without first discussing it with your partners, and (b) not let something your partners write go "over your head" without making a strong effort to understand it, including having your partners explain it to you of course.

You have had practice with our "short cycles and user stories" approach. Hopefully by doing it again, you will get even better at it.

Grading note: Usually all team members will receive the same score for this project. But if there is ample evidence that one person did not fully participate in the learning and the doing, I reserve the right to give different grades. A peer evaluation survey at the end of the project will help me to determine this. If the survey or my observations indicate the need, I may ask you to explain parts of your project to me.

Game description pointer: You will write a game that is patterned after the popular Minesweeper game that has been available with the MS Windows operating systems for many years. You can read the Help that is provided with this program to get an idea of how it should work. You do not need to have a help menu in your program.

Repository name: http://svn.cs.rose-hulman.edu/repos/csse220-200920-mineXY, where XY is your team number. You should check out the project from this repository, and all subsequent work should be placed in this project folder and committed back to your repository.

Milestones (all due at 8:05 AM:

Wednesday, Feb 11	UML class diagram, CRC cards, User Stories for Cycle 1
Thursday, Feb 12	Cycle 1 code and progress report, user stories for Cycle 2
Monday, Feb 16	Cycle 2 code and progress report, user stories for Cycle 3
Wednesday, Feb 18	Cycle 3 code and progress report, user stories for Cycle 4
Thursday, Feb 19	Final Code, Cycle 4 progress report
Thursday, Feb 19 (5 PM)	Team member evaluation survey

UML Class Diagram and CRC cards

You should go through the same process that we used in an in-class exercise and for Vector Graphics:

1. Brainstorm possible classes.
2. List the responsibilities of the Minesweeper program.
3. Assign responsibilities to classes, determine what classes need to collaborate in order to carry out those responsibilities, and what responsibilities those collaborating classes need to have. Keep itrating this until all of the program's responsibilities have been assigned to classes.
4. Turin in your CRC cards at the beginning of Wednesday's class.

Save your diagram as a PNG file, so it can be viewed without Violet. Put both the PNG and Violet files in your reposiotory.

The rest:

Begin implementing, commenting, and testing your code, cycle by cycle.

Create JUnit tests for any parts of the program that can be tested without the GUI.

Commit your project often.

You should include as much of the Minesweeper functionality as you can in the time allowed. If you can't get it all done, here are some things that I consider lower priority than other things (the least important things to implement are at the top) of the list.

1. Everything in the Help menu.
2. Sound.
3. The ability to place the ? mark on squares.
4. Colors.
5. Best Times list.

An "A" project should do most of the things on this "least important" list - perhaps all of them.

One important thing for everybody's project is the recursive behavior, Clicking on a square with no neighboring mines is equivalent to clicking on each of its neighbors. Simultaneously left-right clicking on a revealed numbered square that already has all of its neighboring mines correctly marked. is equivalent to clicking on each of its remaining unclicked neighbors.