Project: Land Mine!

Goals: There are three goals for this project.

1. Create a Land Mine computer game
2. Work as a team of software developers
3. Use Github for a source control system as described in the Github lesson. Let Mr. Jaffe know if you have trouble and we'll figure it out.

Before starting, you may need to refresh your memory about concepts we've discussed a few weeks ago:

- Go through these tutorials about Netbeans
- Re-watch the video about Messaging
- Re-read about MVC Framework
- Re-watch the MVC videos (Part 1: Code, Part 2: Demonstration)

Github: Since you are working in groups, be sure you follow the Github directions from the Github Lessons to maintain version and collaboration control. Here are links to those lessons if you need to review them:

- Part 1: Introduction
- Part 2: Making an Organization and forking the class repo
- Part 3: Creating individual repos and forking the Organization repo
- Part 4: Commiting your changes, pushing them to your repo, and merging into the Organization repo

Code template: The MVC framework that you experimented with in the last unit will be the code template for this project. Fork your template code from https://github.com/JaffeAPCS1415/MVCExample to create your repository just like you did in the Github project from Unit 4.

Game description: In this project you and your team will create a video game called Land Mine! Here's how the game works:

1. The Board is arranged in an 8x8 grid (like a checkerboard).
2. The 8x8 board will create 64 individual cells
3. Each cell will either be empty, or contain a landmine that will explode when the cell is uncovered.
4. There should be 10 landmines placed randomly on the board (see note below about random numbers)
5. The player uncovers one cell at a time.
    a. If the cell is empty, the player receives one point

    b. If the cell contains a landmine, the player explodes and loses one life
6. When the player loses 3 lives, the game is over
7. The game should have a leaderboard that is displayed between games
    a. If a player gets a high score, the game should ask for his or her name which will be displayed on the leaderboard.

Group planning and discussion: Before you write a single line of code, you need to get together as a group and decide on the methods and instance data that will be required by the Model, View, and Controller classes.

For instance, the Model class will need a method called (for example) `clearBoard` which will set all the cells in the array to false. You will also need a method called `randomizeMines(number)` which will randomly place <number> of mines on the board.

You'll also need to agree on the messaging protocol that will be used to communicate in the MVC framework. For instance, the View could send the message `exposeCell` with a message payload of the cell number as so that the Controller could tell the Model to modify the app's data accordingly.

Division of effort: Each team will consist of three people with the following responsibilities

- Team member 1 will be in charge of writing and unit testing the Controller class
- Team member 2 will be in charge of writing and unit testing the View class
- Team member 3 will be in charge of writing and unit testing the Model class
- All team members are responsible for integration testing (putting the classes together and testing the final game)
- What the game's UI looks like is up to you. Use your team's gaming experience to guide you. Be creative and use your imagination.

Team meetings: At the beginning of the period each day your team will have a short (5 minutes or so) meeting where you will describe to each other the work you did for the last 24 hours, the work you plan to do today, and how your Java class (Controller, View, or Model) will interact with the other two classes. This is your time to make sure that you are all on the same page and understand what the others are doing.

Note about random numbers: We will discuss random number generation in more detail in Lesson 29 (Unit 8), but for now, you can generate a random number between `0` and `n` using the `Random.nextInt(n)` method.

Grading: Your game will be assessed on the following things:

- Does the game work?
- Appropriate documentation using JavaDoc styling

- Coding style and structure (Indentation, inline comments where needed)
- Overall “coolness” of your app
- Correctly submitting your code for review (see syllabus for instructions)
- How well you worked with the others in your group (assessed by the others in your team)

Programs that meet the requirements of the game description above can get a maximum of a ‘3’ on the Overall “coolness” score. Some things that your team can do to get a ‘5’ on “coolness” are:

- Let the player choose the size of the board
- Let the player choose how many mines are on the board
- Animate explosions when the mines are uncovered
- Supply encouraging messages to the player when empty spaces are uncovered.