1. Introduction

In this assignment you will write a Java program which implements the complete interface of the “minesweeper” game. In this first phase, you will concentrate on the completed interface. Correct game play will be completed in the follow-on assignment (Program 2).

The game starts by placing a number of “bombs” in unknown random locations in a 2-D grid of cells. The player sees a grid of blank cells - the location of the bombs is unknown at the start. Play proceeds by allowing the player to click on any cell in the grid. The goal is to avoid clicking on a cell where a bomb is located. If the player clicks on a bomb, the game ends and the entire board is revealed. Selecting a cell which is not a bomb should cause the cell face to change color and display a number that represents the number of its neighboring cells that contain bombs. Selecting a cell that does not border on any bombs should cause “clearing”, where all cells that are neighbors with the selected one and also do not border any bombs are automatically revealed. Again, game play as described above is not required in this first phase, just the interface and layout of the grid.

A cell has 8 neighbors: the north, south, east and west cells, together with the four diagonal ones. Cells on the boundary of the grid have fewer neighbors (only five for a cell on an edge and corner cells only have three). The player scores a victory by clicking on all cells which are not bombs without hitting a bomb.

The Java program you write to implement the game in this first phase must implement the complete layout of the game interface, including menu options (start new game, settings that control grid size and level of difficulty, quit the game, and help) and game timer. Minimum grid size is five by five; maximum is ten by ten. The height, width, and number of bombs must be able to be set independently. Height and width must be clamped in the range 5 to 10, and bombs from one to one half the number of spaces on the currently defined grid. In addition, the settings pane should let the user (alternatively) choose from three pre-set skill levels: Beginner (5x5 grid with 5 bombs); Intermediate (8x8 with 15 bombs); and Expert (10x10 with 30 bombs).

2. Details

You must implement your solution to the interface using Java. Implement the grid of cells and the methods to manage the cells for this game as Java classes. Use private data to store the state of the board, such as which cells have been selected and which contain bombs. With a good design (constructor, accessor, mutator, helper methods, private instance variables, and inheritance) the game can be controlled by a simple driver, which should allocate an instance of the class to set up the game. The constructor of the class must ultimately allow the driver code to assign a variable grid size, and a variable number of bombs. The class should assign bomb locations for the appropriate number of bombs randomly using the random number generator.

It is not necessary to implement actual game play for this assignment. At any point that a user event from the interface is required, design your code to issue a simple pop-up window that displays a message containing debugging information: name and type of calling object, actual parameters that triggered the event, intended action to be implemented. In particular, any click on a grid location should issue a popup window with the coordinates of the grid, the name of the object stored in that grid location, and the action to be called in response to the user’s event. Selection of any menu item should do the same. For
example, selection of the setup menu option should bring up your setup “widget”. Settings on the setup widget should trigger a pop-up that reports the settings that were selected. Underneath, in your code, it is not necessary at this stage to respond to the settings (e.g., not necessary to re-arrange the board to a new size, or re-set the number of bombs). The focus of this assignment is to get the interface correct without implementing any of the actual game play functionality underneath. You will implement the code to respond to the user settings in the second phase when you complete the assignment. The exception to this is quit. Selection of quit should exit the program.

3. What to Turn In
You must use the class webpage program in order to submit your Java files that implement this program. Be sure to name the driver file Bombs.java. The names of the other files you may submit can be anything. Your Java source code must contain internal documentation, and in addition to the Java source code, you must supply external documentation in the form of a text file named Bombs.txt, which describes the project specifications and the implementation details.

4. Example Interface Components