ECE 251 Object-Oriented Programming

Project

****

NOTE:

1. Programs *must* compile without error. If your program does not compile, you will lose **50%** of the points. Even if your program is "essentially" correct, you will still lose 50% of the points; there is no reason to turn in a program with syntax errors.
2. To get full credit for an assignment your program must completely solve the stated problem.
3. Your program should contain a reasonable amount of comments. At the very beginning of each file, there should be a comment containing the course number, your name, the assignment number, and the date. It is good practice to insert comments into the source code, as you do when programming in any other languages.
4. You should follow the programming styles and guidelines set forth in class. In particular, use meaningful names for variables. For each coding question you should also write a test class and test your code thoroughly.
5. Compress all .java files into one zip file and name it with your full name firstname_lastname_project.zip. Submit the zip file on blackboard.

****

A new **"Stadium"** has just opened and needs a system for selling tickets for the NBA playoffs. You will be given some files which must be used to design such a system.

The Model:

Each seat in the stadium is identified by 3 pieces of identification as follows: the section number (1 - 4), the row letter (A to Z) and the seat number within the row and section (1 to 9). A seat must be identified by all three pieces of information. The stadium is essentially arranged as a grid (i.e., two dimensional array) containing seats. The grid is fixed at 35 rows by 27 columns in size (i.e., use static variables for the size). Not all of the locations in the grid represent seats. Some of the locations represent aisles and the rink itself. The seat costs are as follows: section 1 = $74, section 2 = $47, section 3 = $32 and section 4 = $19.

Get the following text files from the webpage: "**sections.txt**", "**rows.txt**" and "**numbers.txt**". The "**sections.txt**" file contains a layout of characters representing the section numbers for all seats. Notice that some characters are spaces, dashes or vertical bars. These are aisle and rink locations and so they do not contain seats. The "**rows.txt**" file contains a similarly laid out set of characters representing the row letters. The "**numbers.txt**" file contains yet another arrangement of numbers representing the seat numbers. Note that the files are merely stored as characters.

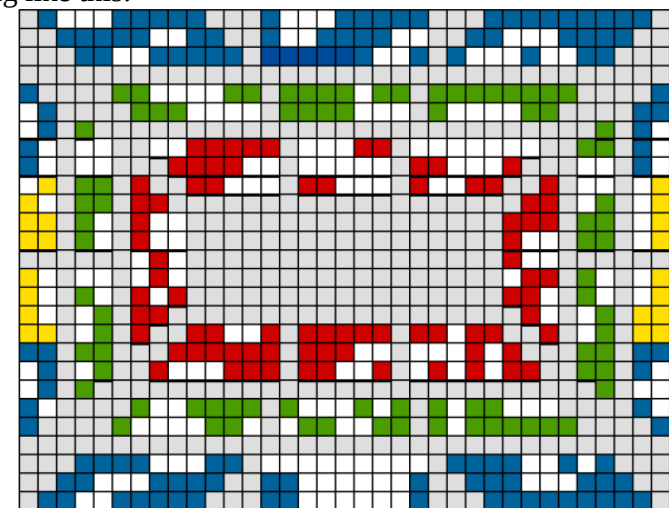
Create a **Seat** class to represent each seat (with at least section, row and number instance variables). Create a **Stadium** class as well to represent the stadium which should maintain a 2 dimensional array of **Seat** objects. When a **Stadium** is created (i.e., in the constructor), it should read in the three text files and build up the **Seat** information according to the information in the files. That is, for each of the 35 x 27 characters, you'll have to create a Seat object and set its instance variables to the

corresponding character or number from the file. In the case where there is a blank character in the file, you should probably NOT make a **Seat**, instead...store **null** in the array at that position.

The User Interface:

OK. Now we'll make an interface to show the stadium seating plan and allow the user to select and purchase seats. Your interface must have this:

- a way to display the seating availability information in one of these two ways: (a graphical representation (i.e., draw the seats) or as a grid panel of buttons or labels). For instance, it may look something like this:



- Here we see that the seats are displayed as squares (don't forget that you can simply use buttons or labels). The gray areas are places where there are no seats. The display should show seats with different colors corresponding to the different sections. We have 4 main colors shown above (red, green, blue and yellow). The white seats are those which are not available (i.e., already purchased for that game).

- seat prices should be shown on your GUI with the appropriate seat colors that match the ones in the main window.

- a way for the user to select seats to be purchased (clicking is best, but text fields are ok too). When the user selects seats, they are added to some kind of shopping cart (probably just an arraylist). The selected seats information and the total price should be shown on your GUI. Your interface MUST also indicate which seats have been selected and placed in the shopping cart (use different colors).

- there should be a **Purchase** button on your GUI. If the **Purchase** button is pressed (seats are purchased) the purchased seats must appear as unavailable (use a different color such as white) in the main window and no user may be able to purchase them again.

- a button called **Empty Shopping Cart** should be on the GUI that empties everything in the shopping cart.