COMP1406 - Assignment #7

(Due: **Monday, March 10th @ 11:30pm)**

In this assignment you will create a user interface that uses the **GridLayout** manager as well as the **GridBadLayout** manager. You MUST use these layout mangers to receive your marks on this assignment. You will also get some experience using **MouseListeners** as well as populating **JTextFields** with data.

(1) The Model Classes:

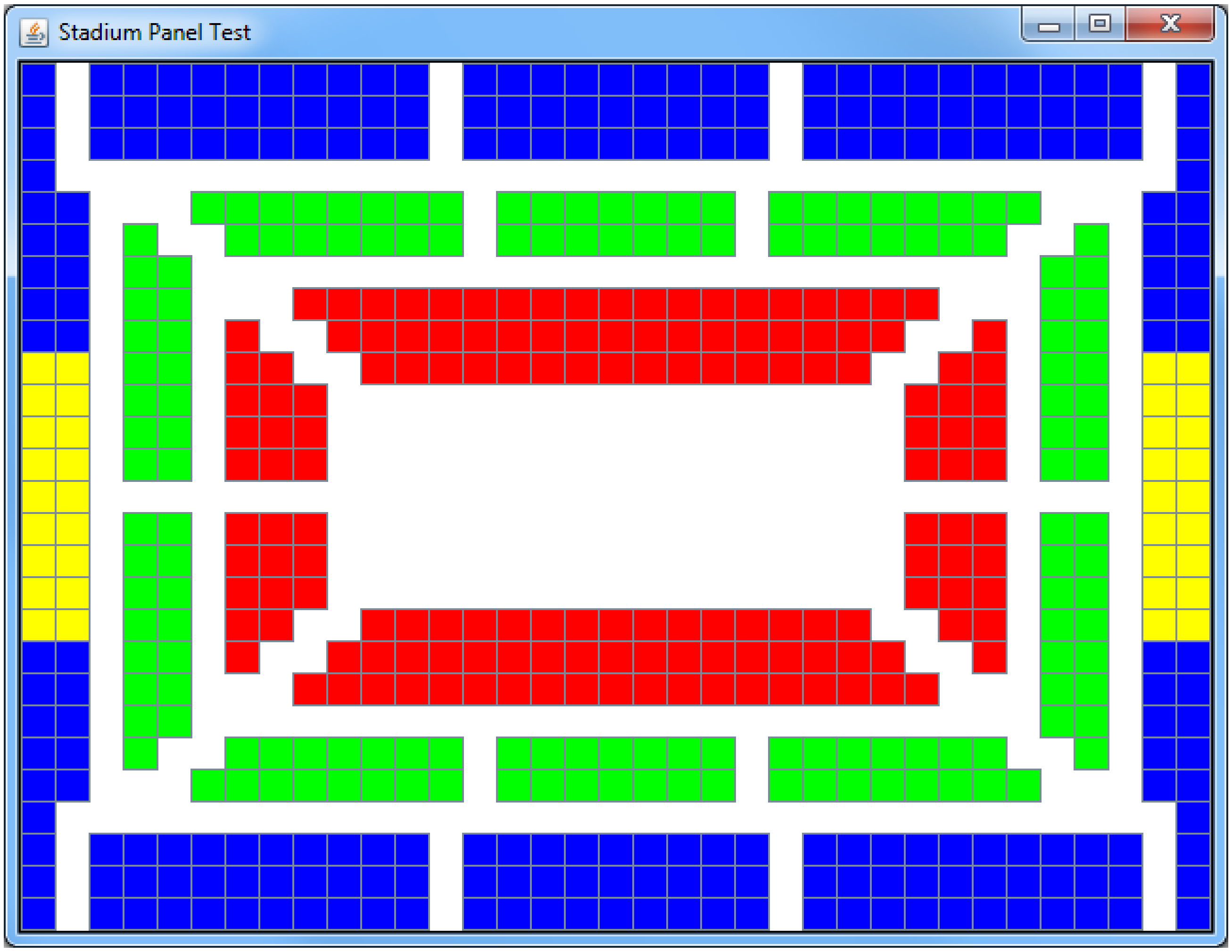
Define and compile the following two classes that will represent seats in a stadium:

```
public class Seat {
   public static int[] PRICING = {74, 47, 32, 19};

   private byte section;
   private char row;
   private byte number;

   public Seat(byte s, char r, byte n) {
      section = s;
      row = r;
      number = n;
    }

   public byte getSection() { return section; }
   public char getRow() { return row; }
   public byte getNumber() { return number; }
   public int getPrice() { return PRICING[section-1]; }
}

public class Stadium {
   public static int ROWS = 27;
   public static int COLUMNS = 35;

   private static String[] SEAT_NUMBERS = {
         "9 1234567890 123456789 1234567890 1",
         "8 1234567890 123456789 1234567890 2",
         "7 1234567890 123456789 1234567890 3",
         "6                                 4",
         "55   12345678 1234567 12345678   15",
         "44 8  1234567 1234567 1234567  1 26",
         "33 77                         12 37",
         "22 66   1234561234567123456   23 48",
         "11 55 0  12345123456712345  1 34 59",
         "99 44 98  123412345671234  12 45 11",
         "88 33 876 --------------- 123 56 22",
         "77 22 765|               |234 67 33",
         "66 11 654|               |345 78 44",
         "55       |               |       55",
         "44 87 543|               |456 11 66",
         "33 76 432|               |567 22 77",
         "22 65 321 --------------- 678 33 88",
         "11 54 21  432176543214321  89 44 99",
         "95 43 1  54321765432154321  0 55 11",
         "84 32   6543217654321654321   66 22",
         "73 21                         77 33",
         "62 1  7654321 7654321 7654321  8 44",
         "51   87654321 7654321 87654321   55",
         "4                                 6",
         "3 0987654321 987654321 0987654321 7",
          "2 0987654321 987654321 0987654321 8",
          "1 0987654321 987654321 0987654321 9"
   };

   private static String[] SEAT_ROWS = {
          "Z CCCCCCCCCC FFFFFFFFF IIIIIIIIII K",
          "Z BBBBBBBBBB EEEEEEEEE HHHHHHHHHH K",
          "Z AAAAAAAAAA DDDDDDDDD GGGGGGGGGG K",
          "Z                                 K",
          "ZY   BBBBBBBB DDDDDDD FFFFFFFF   JK",
          "ZY T  AAAAAAA CCCCCCC EEEEEEE  H JK",
          "ZY TS                         GH JK",
          "ZY TS   CCCCCCFFFFFFFIIIIII   GH JK",
          "ZY TS X  BBBBBEEEEEEEHHHHH  L GH JK",
          "DC TS XW  AAAADDDDDDDGGGG  KL GH AB",
          "DC TS XWV --------------- JKL GH AB",
          "DC TS XWV|               |JKL GH AB",
          "DC TS XWV|               |JKL GH AB",
          "DC       |               |       AB",
          "DC RQ XWV|               |JKL IJ AB",
          "DC RQ XWV|               |JKL IJ AB",
          "DC RQ XWV --------------- JKL IJ AB",
          "DC RQ XW  SSSSPPPPPPPMMMM  KL IJ AB",
          "XW RQ X  TTTTTQQQQQQQNNNNN  L IJ LM",
          "XW RQ   UUUUUURRRRRRROOOOOO   IJ LM",
          "XW RQ                         IJ LM",
          "XW R  OOOOOOO MMMMMMM KKKKKKK  J LM",
          "XW   PPPPPPPP NNNNNNN LLLLLLLL   LM",
          "X                                 M",
          "X TTTTTTTTTT QQQQQQQQQ NNNNNNNNNN M",
          "X UUUUUUUUUU RRRRRRRRR OOOOOOOOOO M",
          "X VVVVVVVVVV SSSSSSSSS PPPPPPPPPP M"
   };

   private static String[] SEAT_SECTIONS = {
          "3 3333333333 333333333 3333333333 3",
          "3 3333333333 333333333 3333333333 3",
          "3 3333333333 333333333 3333333333 3",
          "3                                 3",
          "33   22222222 2222222 22222222   33",
          "33 2  2222222 2222222 2222222  2 33",
          "33 22                         22 33",
          "33 22   1111111111111111111   22 33",
          "33 22 1  11111111111111111  1 22 33",
          "44 22 11  111111111111111  11 22 44",
          "44 22 111 --------------- 111 22 44",
          "44 22 111|               |111 22 44",
          "44 22 111|               |111 22 44",
          "44       |               |       44",
          "44 22 111|               |111 22 44",
          "44 22 111|               |111 22 44",
          "44 22 111 --------------- 111 22 44",
          "44 22 11  111111111111111  11 22 44",
          "33 22 1  11111111111111111  1 22 33",
          "33 22   1111111111111111111   22 33",
          "33 22                         22 33",
          "33 2  2222222 2222222 2222222  2 33",
          "33   22222222 2222222 22222222   33",
          "3                                 3",
          "3 3333333333 333333333 3333333333 3",
          "3 3333333333 333333333 3333333333 3",
          "3 3333333333 333333333 3333333333 3"
   };

   private Seat[][] seats;

   public Stadium() {
      seats = new Seat[ROWS][COLUMNS];

      for (int r=0; r< ROWS; r++) {
          String secString = SEAT_SECTIONS[r];
          String rowString = SEAT_ROWS[r];
          String numString = SEAT_NUMBERS[r];
          for (int c=0; c< COLUMNS; c++) {
             byte section = (byte)Character.digit(secString.charAt(c),10);
             char row = (char)rowString.charAt(c);
             byte number = (byte)Character.digit(numString.charAt(c),10);
             if (!Character.isLetter(row))
                seats[r][c] = null;
             else
                seats[r][c] = new Seat(section, row, number);
          }
      }
   }

   public Seat[][] getSeats() { return seats; }
   public Seat getSeat(int row, int col) { return seats[row][col]; }
}
```

When creating a new Stadium, the constructor above will fill a 27x35 two-dimensional array of **Seat** objects, where each **Seat** has a number, row and section.   There are locations in the stadium where no seat resides ... therefore in the 2D array, there will be some **null** locations.

(2) The **StadiumPanel** Class:

Create a subclass of **JPanel** called **StadiumPanel** which will display a 2D array of **JButtons** as follows:



The constructor for this class should take a single **Stadium** parameter and then, using a **GridLayout**, arrange a 2D array of **JButtons** as shown above. If there is a **Seat** at the given row and column of the **Stadium**, add a **JButton** at that location in the grid. If there is no **Seat** (i.e., there is a **null** in the array) at that location, then add a blank **JLabel** instead of a **JButton** at that location in the grid. The background of the **StadiumPanel** should be white and all **JButtons** should be colored according to the **Seat** section that they represent. Section 1 should be red, section 2 green, section 3 blue and section 4 yellow (as shown above). Add the following to the end of your **StadiumPanel** class to make sure that it displays properly as the window is resized:

```
public static void main(String args[]) {
   JFrame f = new JFrame("Stadium Panel Test");
   f.getContentPane().add(new StadiumPanel(new Stadium()));
   f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   f.setSize(649, 500);
   f.setVisible(true);
}
```

Save and compile the following **BoardPanel** class which represents a fixed-size **JPanel**:

```
import java.awt.*;
import javax.swing.*;

public class BoardPanel extends JPanel {
      static int WIDTH = 633, HEIGHT = 462;
      public int getWidth() { return WIDTH; }
      public int getHeight() { return HEIGHT; }
      public Dimension getSize() { return new Dimension(WIDTH, HEIGHT); }
      public Dimension getSize(Dimension rv) {
            rv.width = WIDTH; rv.height = HEIGHT; return rv;}
      public void setBounds(Rectangle r) {
            super.setBounds(new Rectangle(r.x, r.y, WIDTH, HEIGHT));
      }
      public void setBounds(int x, int y, int w, int h) {
            super.setBounds(x,y,WIDTH,HEIGHT);
      }
      public Dimension getMaximumSize() { return new Dimension(WIDTH, HEIGHT); }
      public Dimension getMinimumSize() { return new Dimension(WIDTH, HEIGHT); }
      public Dimension getPreferredSize() { return new Dimension(WIDTH, HEIGHT); }
      public void setSize(Dimension d) {  }
      public void setSize(int x, int y) {  }
}
```

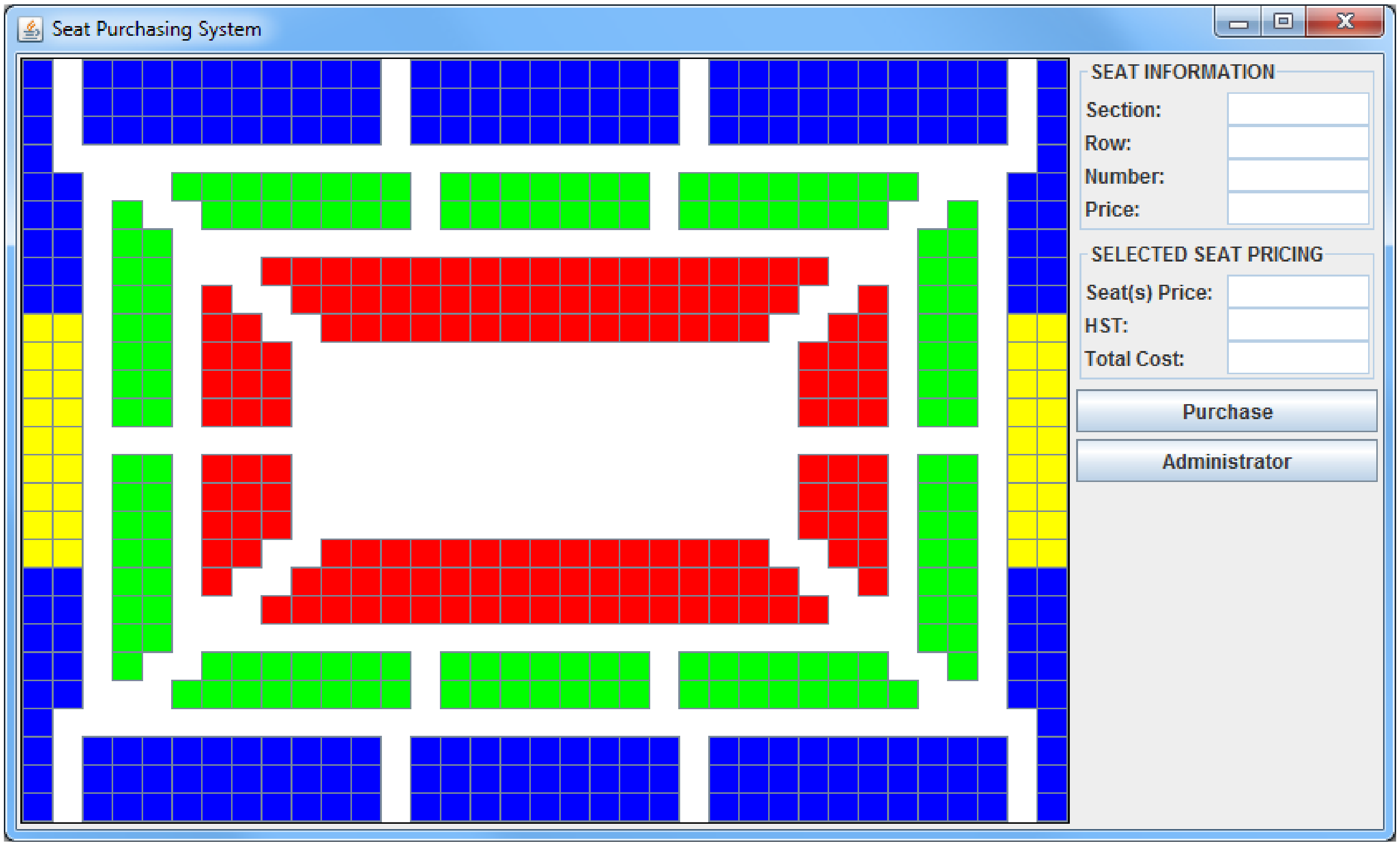
Change **StadiumPanel** to be a subclass of **BoardPanel**. Re-run the code to make sure that it no longer re-sizes as the window re-sizes.

(3) The **StadiumApp** Class:

Now you will create a **StadiumApp** class which represents the window shown on the next page. The window must use a **GridBagLayout** manager. The window has exactly 5 components:

1. a StadiumPanel
2. a JPanel that must use a **GridLayout** to display the section, row, number and price of a seat.
3. a JPanel that must use a **GridLayout** to display the price, HST and Total Cost for all selected seats.
4. a Purchase button
5. an Administrator button

Note that you MUST use the layout managers described above, otherwise you WILL NOT receive any marks for this part of the assignment. The window shown has a size of 840 x 505. The window may look weird when re-sized, but that is ok. Once you get it arranged nicely, use **setResizable(false)** to disable window resizing behavior. You may need to then adjust the window size to 840 x 494 or something like that to adjust for the margins which changes when you made it non-resizable.



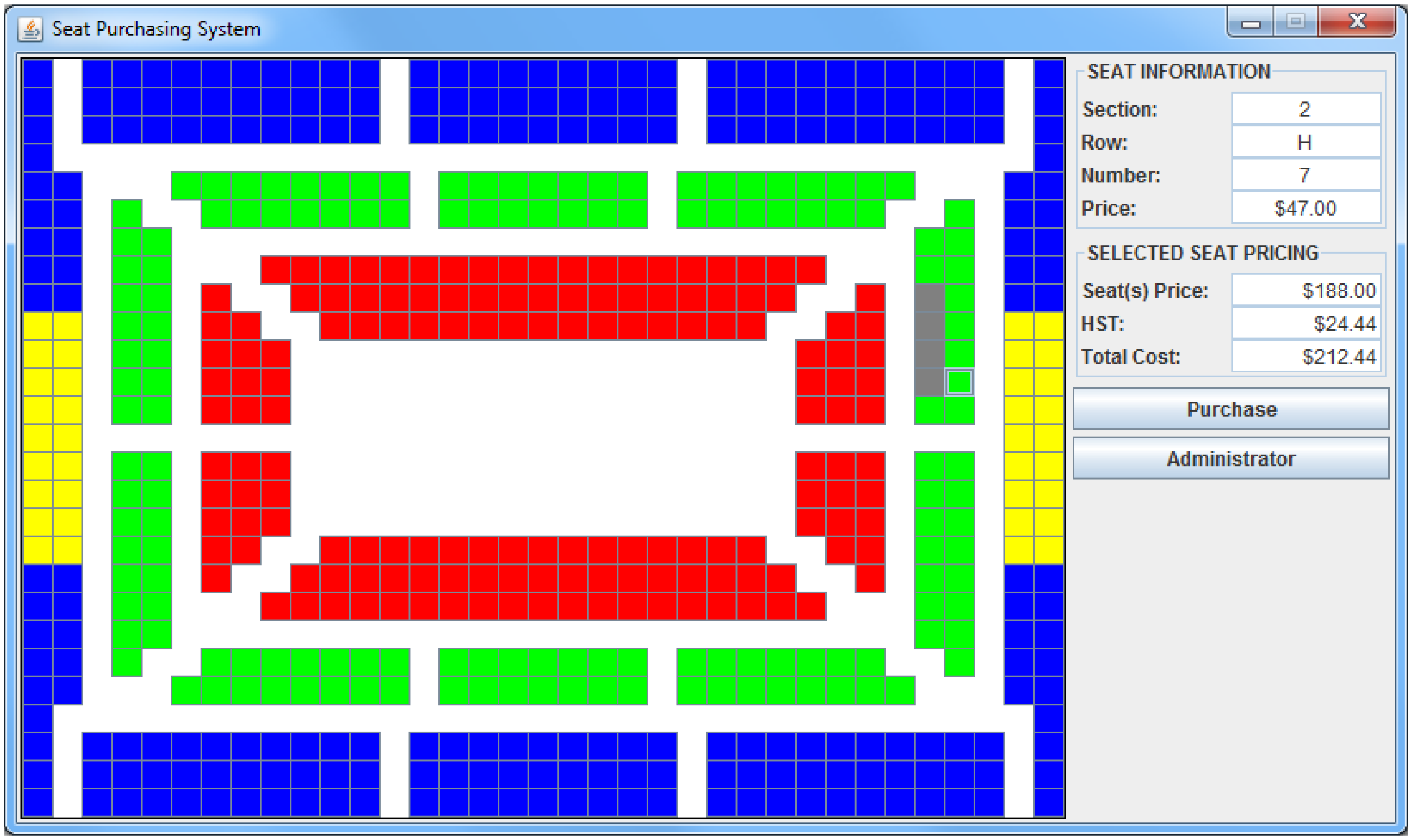
(4) Finishing Touches:

Add a **MouseListener** to each of the **JButtons** that represent a seat. Write the **mouseEntered(MouseEvent e)** and **mouseExited(MouseEvent e)** methods so that when the mouse hovers over a seat, the section, row, number and price of that seat is displayed in the **Seat Information** panel on the window. When the mouse leaves a seat, then those text fields should become blank again.

Add a private **selected** attribute to the **Seat** class along with public **isSelected()** and **setSelected(boolean s)** methods.

Add an **ActionListener** to each seat's **JButton** so that when pressed, the **Seat** clicked on becomes *selected*. Selected seats should be shown as Color.GRAY. You may want to write an **update()** method that simply goes through all seats and sets their color to either GRAY (if selected) or their default color as before. Then just call the **update()** from the **ActionListener** after you set the seat to be *selected* or *unselected*.

In the **update()** method that you wrote, go through all seats and total up the price for all currently selected seats and show the selected seats combined price in the SELECTED SEAT PRICING panel along with the HST and total Cost. These fields should ALWAYS reflect the totals for currently selected seats (see picture on next page).



Seat Purchasing System

SEAT INFORMATION

Section:	2
Row:	H
Number:	7
Price:	$47.00

SELECTED SEAT PRICING

Seat(s) Price:	$188.00
HST:	$24.44
Total Cost:	$212.44

NOTE: Submit all **.java** files needed to run. You **MUST NOT use packages** in your code, **nor projects**. Submit ALL of your files in one folder such that they can be opened and compiled individually in JCreator. Some IDEs may create packages and/or projects automatically. You **MUST** export the .java files and remove the package code at the top if it is there. Do NOT submit JCreator projects either. **JUST SUBMIT the JAVA FILES**. Note that if your internet connection at home is down or does not work, we will not accept this as a reason for handing in an assignment late ... so make sure to submit the assignment WELL BEFORE it is due !

Please NOTE that you WILL lose marks on this assignment if any of your files are missing. You will also lose marks if your code is not written neatly with proper indentation. See examples in the notes for proper style.