Fall 2026 COP 3330 Program #3: Ticketing System

In the last part of the previous assignment, we printed out a "simulated" seating chart. Since we hadn't learned arrays yet, we couldn't store a record of the seating chart in our program. Now that arrays have been taught, we can do a program which manages a seating chart!

To make life a bit easier, we'll have a fixed seating chart. We'll make our seating chart with 26 rows, each with 60 seats. The rows will be labeled A – Z, from front to back and the seat numbers will go 1 to 60 from left to right.

We'll split the seats into 6 regions and the cost of every seat within a region will be the same.

Here are the regions and the cost of the seats in that region:

Region	Seats in Region	Cost/Seat in Region
1	Rows A – M, Seats 1 – 20	$200
2	Rows A – M, Seats 21 – 40	$300
3	Rows A – M, Seats 41 – 60	$200
4	Rows N – Z, Seats 1 – 20	$100
5	Rows N – Z, Seats 21 – 40	$150
6	Rows N – Z, Seats 41 – 60	$100

Whenever a user wants to buy a seat, you'll ask them which region (1 – 6) they want to buy a seat, and how many seats they want to buy.

If there are enough open seats in the region, automatically select the seats in order by row (from earlier letter row to later letter row) sorted by seat number (lower seat numbers first). You should print out to the user which seats they are receiving as well as total cost of the purchase.

If there are not enough open seats in the region, then you inform the user that this is the case and do not perform a transaction.

The user will also have an option to view the whole seating map, showing which seats are taken and which are still open.

The third option the user will have is to view the total number of tickets sold so far as well as the total revenue.

When the user quits the program, print out a final report with the final seating chart, the number of tickets sold, and the total revenue collected from the ticket sales.

Required Static Methods (other than main)

To make this a bit easier for you, and also so the graders aren't dealing with programs that are poorly designed, several static methods will be required with their behavior will be specified below. You may write MORE static methods that are listed here, but these are the bare minimum requirements:

```
// Returns a character array with 26 rows and 60 columns
// representing the seating chart for the concert. The
// array should be initiated so that all cells store 'O'.
public static char[][] makeEmptyChart();

// Asks the user to enter a menu choice and continues to prompt
// the user until they enter a valid choice and returns that
// choice. (Choices are 1 = buy, 2 = print chart,
// 3 = list tickets sold, total revenue, 4 = quit.
public static int getMenuChoice(Scanner stdin);

// Asks the user for the region in which they want to buy
// tickets and continues to prompt the user until they enter
// a valid choice and returns that choice (1 to 6)
public static int getRegion(Scanner stdin);

// Prints out the current seating chart for the concert.
// Please follow the format shown on the sample output.
public static void printChart(char[][] seats);

// Returns the total cost of buying numTickets number of tickets
// in region number region using the seating chart seats. If
// too few tickets are available in the region, 0 is returned
// and no tickets are issued. Otherwise, the method lists each
// ticket sold (follow sample output for format) and returns the
// total cost of all of the tickets.
public static int buySeats(char[][] seats, int region, int
numTickets);

// Returns the number of open seats in the seating chart seats
// in region, region. region is guaranteed to be in between 1
// and 6.
public static int numOpenSeats(char[][] seats, int region);
```

Required Constants
For full credit, you must use these constants in all relevant places throughout your code instead of the corresponding integer literals:

```
final public static int TOTAL_ROWS = 26;
final public static int TOTAL_COLS = 60;
final public static int REGION_ROWS = 13;
final public static int REGION_COLS = 20;

final public static int[] COST = {200, 300, 200, 100, 150, 100};
```

The first two specify the dimensions of the whole seating chart.
The last two specify the dimensions of each region.

Note that for full credit, you'll have to use these in such a way that if the values of TOTAL_ROWS and TOTAL_COLS were to change to be different multiples of REGION_ROWS and REGION_COLS, that your code would work perfectly without any other changes. (The exception of course is the A – Z printing, since that doesn't extend past 26...)

Input Specification
For the menu choice, the user will always enter values that fit into the integer data type and will eventually enter an integer in between 1 and 4, inclusive.

For option #1 of the menu, the user will enter values that fit into the integer data type for region and number of tickets and will eventually enter an integer in between 1 and 6 for region.

Output Specification
This is fairly lengthy, so instead of spelling this out here, multiple files with sample output will be given as examples to follow.

Sample Program Runs
These are very lengthy so they'll be included in separate files.

Deliverables
Please submit one file, ticketing_v1.java for your solution to this problem via WebCourses by the designated due date.

Please make sure to include a header comment and internal comments in your code and indent when appropriate.

Note: This program is substantially longer than P1 and P2. The sample solution is more than 200 lines with comments. So, get started on this early!!!