# CPSC 1110 – LAB 1B

More Arrays (2D)

This lab will deal with using arrays in java to implement a seating chart for a theater. Users will be able to purchase a seat based on either location, or price. You may use BlueJ to complete the lab. (If you want to use some other IDE please talk to me and or TA about it). **PLEASE COMMENT YOUR CODE.** You will have points taken off if you do not comment your code. You can see sample comments in my starter code for how you should comment your code. Keep your code neat.

Zip your .java files and a PDF containing your screen shot of your output verifying your program works correctly for submission. A program that does not compile will receive a 0.

**Some useful links:**

BlueJ tutorial [www.bluej.org/tutorial/tutorial-201.pdf](http://www.bluej.org/tutorial/tutorial-201.pdf)

Java tutorial home page: <http://docs.oracle.com/javase/tutorial/>

Start here: <http://docs.oracle.com/javase/tutorial/java/index.html>

variables <http://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html>

data types <http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>

relational operators <http://docs.oracle.com/javase/tutorial/java/nutsandbolts/op2.html>

if-then <http://docs.oracle.com/javase/tutorial/java/nutsandbolts/if.html>

java math library <http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html>

Arrays <http://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html>

**Some helpful tips:**

1. Compile often – do it.
2. Perform the tasks by hand to verify your work. Run the algorithms on paper with small input sizes to make sure your algorithm works.
3. It may be helpful to use the Debugger or print statements to check your work.

## Tasks: Follow the directions below to complete your lab assignment

For today's lab we will be completing ***Programming Project P7.5*** from the book with some slight modifications. (See below).

***P7.5*** – A theater seating chart is implemented as a two-dimensional array of ticket prices, like this:

10 10 10 10 10 10 10 10 10 10

10 10 10 10 10 10 10 10 10 10

10 10 10 10 10 10 10 10 10 10

10 10 20 20 20 20 20 20 10 10

10 10 20 20 20 20 20 20 10 10

10 10 20 20 20 20 20 20 10 10

20 20 30 30 40 40 30 30 20 20

20 30 30 40 50 50 40 30 30 20

30 40 50 50 50 50 50 50 40 30

(Note that there are 9 rows, and 10 columns – these values will be stored as constants in our class).

Write a program that prompts users to pick either a seat or a price. Mark sold seats by changing the price to 0. When a user specifies a price, find any seat with that price (iterate over the array one element at a time). Alternately, users can select a seat by specifying a row and a column.

I have again given you a starter shell code for this lab assignment. It is posted in **Labs\Lab1B** folder as TheaterSeatSeller.java. There are **FOUR** (4) methods that you must complete, as detailed below.

***First*** you must complete the constructor public **TheaterSeatSeller(){…}**. In this method, you need to fill in the pricesAvailable[] array (instance variable of our class). You will do this by iterating over the seats[][] array. When you encounter a value of 10 in seats[][], you increment pricesAvailable[0]. When you see a value of 20, you increment pricesAvailable[1], and so on up to a value of 50 and pricesAvailable[4]. At the end of this method, pricesAvailable[] should contain an entry for the number of tickets available at each prices ([0] → # of $10 tickets, [1] → 20, [2] → 30, [3] → 40, [4] → 50).

***Second*** you must add to the **printSeats(){…}** method. This method already prints out the entire theater showing ticket prices, and available seats. (Available seats show the price, unavailable seats show a 0). Your task for this method, is to add some information to the printout. After the seating chart has been printed, you must add a line (or lines) that display the number of each priced seat available. (recall, we are storing this information in the pricesAvailable[] array). For example, you might print out the following after the seats have been printed:

Tickets available by price:

$10: 25

$20: 6

$30: 3

$40: 4

$50: 5

***Third*** you must complete the **public** **boolean** getByPrice(**int** price){…} method. If there is an available ticket with the given price, then you will mark that location in the seats[][] array with a 0, and return true. If there is no ticket left with the given price, do not change the seats[][] array, and return false.

***Fourth*** you must complete the **public** **boolean** getByLoc(**int** row, **int** col){…} method. If there is a seat available indicated by parameters row and col, then you will mark that location in the seats[][] array with a 0, and return true. Otherwise, make no changes to the seats[][] array and return false.

You should test all aspects of your program to ensure that it works correctly. I will also provide a file called tests.txt which will run through multiple tests in a single batch operation (Instructions on how to use this file for input are given below in the next paragraph). The provided file buys all $50 tickets, and then attempts to keep buying. (Program should not crash, but indicate no more tickets are available at this price). Then the tests.txt file tries to buy 0,0; 0,1; 0, 2; and 0, 4. Finally, the tests.txt file uses a Q to quit the program. The text that you enter into the tests.txt file is simply the menu commands and inputs that you would give to the program if you were interacting with it in real time. Look at the tests.txt file so that you can add your own tests. (for example, you should try to buy 0,0; then try to buy it again, and make sure your program sells the ticket on the first try, and reports that the ticket is not available on the second try).

To use the file tests.txt you must have your TheaterSeatSeller.class file and tests.txt file in the same directory/folder. (note this is the ***.class*** file, NOT the ***.java*** file – usually contained in a “bin” directory in eclipse, and it is in your main project directory in BlueJ). Next, you need to navigate through the command line to the folder where both of these files are located. (There is a shortcut in windows, <shift> right click the folder where you want to open a command line, and you get an option to “Open command window here.” You can then type:

java TheaterSeatSeller < tests.txt

This will run your program, and enter all the text from your tests.txt file into the program as if you had entered it in manually. You must test many aspects of your program, so this will allow you to create a “batch test” where you can run a test with a single command, fix bugs, and then run the test again. I will be creating an ultimate batch test that will test all aspects of your methods to help me to determine your final score.

Note that if you want to save your test results, you can use the following command.

java TheaterSeatSeller < tests.txt > myResults.txt

The results will be stored in a text file named “myResults.txt”

Capture a text file of a sample run of program (you don't need a screen-shot this time, you can simply capture the text output from your program). Submit your entire project folder and program output as a single .zip file to UTCOnline.

## To Turn In via UTCOnline

You should turn in 1 .ZIP file containing your java files and a PDF document with output from your program. 1 file should be uploaded to UTCOnline. ***IMPORTANT!!!*** You should name your file in the following manner. lastname-firstname-lab1B.zip. So, John Smith would submit smith-john-lab1B.zip.