CS 2336 – PROJECT 5 – GotG2 Ticket Reservation System (Maintenance Project 3)

Pseudocode Due: 4/14 by 11:59 PM

Project Due: 4/30 by 11:59 PM

Submission and Grading:

- All project deliverables are to be submitted in eLearning.
- The pseudocode should be submitted as a Word or PDF document and is not accepted late.
- Zip all of the source files into a single zip file for submission.
- Projects submitted after the due date are subject to the late penalties described in the syllabus.
- Programs must compile and run with JDK 8.
- Each submitted program will be graded with the rubric provided in eLearning as well as a set of test cases. These test cases will be posted in eLearning after the due date. Each student is responsible for developing sample test cases to ensure the program works as expected.
- **Type your name and netID in the comments at the top of all files submitted.**
- **Your main class file must be named Main.java**
    - **Keep Main.java in the default package**
- **You may create any packages that you feel are necessary. Just make sure that Main.java is not in a package.**
- **Zip the contents of the src subdirectory into a single zip file (not .rar, not .tar, not .7z).** ***Do not zip the src directory, only its contents.***

Changes from projects 1 and 3 are displayed in blue.

Objectives: Use a hashmap to access and store various pieces of data. Create a comprehensive professional application.

Problem: In preparation for the release of Guardians of the Galaxy 2, you have been hired by the owner of a small movie theater to develop the backend for an online ticket reservation system. Patrons will be able to reserve seats in one of three auditoriums. Once the patron has selected an auditorium, the program should display the current seating arrangement and allow the patron to select seats. A report should be generated at the end of the program to specify for each individual auditorium and overall for all auditoriums how many seats were sold/unsold and how much money was earned.

The theater owner understands that to be competitive, the reservation system needs to track customer orders. The customers need the ability to login to the system and access their orders. The customers should have the ability to add orders, change orders or cancel orders as well as the ability to see the receipts for their orders. The reservation system should also have an administrator interface.

Details

- To avoid potential errors when grading, do not create multiple Scanner objects for System.in. If the Scanner object is needed in multiple functions, please pass the Scanner object into the function.
- The user account information will be stored in a HashMap

    - You may use the pre-built Java HashMap object
- Each entry in the HashMap should contain the customer username, password and all orders.
    - An order consists of the auditorium, seats, and number of tickets per ticket type
    - I suggest using the username as the key value
- The seating arrangement for each auditorium will be stored in separate files. These files will be named *A1.txt, A2.txt* and *A3.txt* for auditorium 1, 2 and 3 respectively.
- Each line in the file will represent a row in the auditorium. The number of rows in each auditorium is unknown to you.
- The number of seats in each row of a **single** auditorium will be the same. For example, if the first line of the file has 15 seats, then every subsequent row in the theater will also have 15 seats. This does not mean that each auditorium has the same number of seats in each row. One auditorium may have 15 seats per row and another may have 20 seats.
- Empty seats are represented by a pound sign (#).
- Reserved seats are represented by a period (.).
- Use the reservation system from project 1 or 3.
    - Feel free to tweak the reservation system as you feel is necessary.
- The user will enter individual seats for reservation and can reserve seats that are not adjacent to the initial seat selected.
- If all the desired seats are not available, offer the user the best available consecutive seats in the entire auditorium equal to the number of tickets they wish to purchase. The best available seats are the seats closest to the middle of the auditorium.
        - Think of the distance between 2 points
        - Use the distance between the **middle seat** of the section and the center of the auditorium.
            - With a selection containing an even number of seats, round down to find the middle seat
        - In the event of a tie for distance, the row closest to the middle of the auditorium should be selected.
        - You may develop this piece any way that you see fit. Please try to be as efficient as possible with the memory.
    - State to the user what the best available seats are and then ask if they would like those seats.
    - If the user declines the best available seats, return the user to the main menu.
    - If the user accepts the best available seats, reserve them and display a conformation to the screen.
- After seats are reserved, please display a confirmation of this on the screen.
- Tickets can only be reserved the day of the screening and all screenings are at the same time.
- Tickets are now priced as follows:
    - Adult - $10
    - Senior - $7.50
    - Child - $5.25
- When prompting the user for input, expect anything. Do not assume any input will be valid. If you ask for a number, the user could put in a floating point number, symbols or maybe even the Gettysburg Address (or at least a sentence or two). Make sure that your program handles proper validation. If invalid input is

entered, loop until valid input is received. Consider using exception handling for that task (but it is not required)

Filling the HashMap:

- All user data will be in a file named `userdb.dat`.
- The program should read the file and fill in the hashmap before starting the user interface.
- The format of the file is as follows:
  `<username><space><password><newline>`
  - This should be encrypted, but you have enough to do without adding in decryption
- The last line in the file will not contain a newline character at the end of the line

Customer User Interface and Workflow: Present a user-friendly system for the user to reserve seats.

- **Starting Point:** Before the customer can use the system, a login is required. The starting point of the interface is to ask the user for a username. After the username is entered, prompt for the password. Verify that the password is correct. If the password is not valid, prompt the user to enter the password again. If the password is entered incorrectly a total of 3 times, return to the starting point. Once the user has successfully logged in, display the main menu.

- **Main Menu**

```
1. Reserve Seats
2. View Orders
3. Update Order
4. Display Receipt
5. Log Out
```

- **Reserve Seats:** When the user reserves seats, display the auditorium submenu.

```
1. Auditorium 1
2. Auditorium 2
3. Auditorium 3
```

  After the auditorium is selected, display that auditorium.

```
  12345678901234567890
1 ...##..#####........
2 ########....####..##
3 .........##.........
4 #.#.#.#.#.#.#.#.#.#.
5 ########.#####.#####
```

  The seats are numbered sequentially from left to right and only the ones digit is displayed above each column to make it easier to display the chart. It is understood that the second set of digits from 1-0 are for the numbers 11-20 in the above example.

  Ask the user for the number of tickets in the following categories (use this order for your prompts):

- Adult
- Senior
- Child

For the total number of tickets entered, prompt the user to enter the row and seat for each ticket. Ask for the seats of each category in the same order as the prompts for ticket amounts. You will need to track which type of person is sitting in each seat so that if a seat is removed from an order, the receipt and report totals will be accurate.

Check that each ticket is available. Do not reserve any seats unless all seats are available. If all seats are not available, search for the best available (see criteria above). If best available is located, prompt the user to enter a **Y** to reserve the best available or **N** to refuse the best available. Once the selection has been processed, return to the main menu.

- **View Orders:** Develop an easy to read interface that displays all orders by the user that is logged in. As you develop this interface to display orders, consider that the interface will also be utilized when the user wants to update the order. Remember that the order consists of the auditorium, seats, and number of tickets per ticket type. You do not have to specify which type of person is sitting in each seat. Return to the main menu once the orders are displayed.

- **Update Order:** Display the orders in a numerical menu system so that the user can enter the number of the order that should be updated. List each order on a separate line and be descriptive of the order. Do not just list Order 1, Order 2, etc.

  After the order menu has been displayed and the user has selected an order, display the menu below and prompt the user for an update action:

```
1. Add tickets to order
2. Delete tickets from order
3. Cancel Order
```

  If the user wishes to add tickets to the order, step the user through the reservation process. **All seats reserved this way will be added to the current order.** When finished adding seats, return to the main menu.

  If the user wishes to delete tickets from an order, display a numerical menu listing each individual seat reserved for that order. Along with the row and seat, also display the type of ticket so that the reports will be accurate if the seat is removed. The last entry in the menu should be “Exit”. The user will select a seat from the menu. Remove that seat from the order and mark the seat as open instead of reserved in the auditorium. If there are no tickets left in the order, remove the order from the user’s account and return to the main menu. Loop back to the individual seat menu until the user decides to exit the process and return to the main menu.

  If the user wishes to cancel the order, mark all seats in the order as available and remove the order from the user's account. After the order has been cancelled, return to the main menu.

- **Display Receipt:** Create a formatted receipt for the user. Display each order in detail (auditorium, seats, and number of tickets per ticket type), the amount of each order and the overall amount of all orders. Once displayed, return to the main menu.

- **Log Out:** Return to the **Starting Point**

Admin Interface and Workflow:

- **Starting Point:** Like the customer, a login is required for the admin. Both the customer and admin will begin at the login prompt. The admin username is `admin` (*Please note this is a horrible idea professionally, but that is the network admin's problem*). After the username is entered, prompt for the password. Verify that the password is correct. If the password is not valid, prompt the user to enter the password again. If the password is entered incorrectly a total of 3 times, return to the starting point. Once the user has successfully logged in, display the Admin main menu.

- **Admin Main Menu**

```
1. View Auditorium
2. Print Report
3. Exit
```

- **View Auditorium:** Display the auditorium submenu:

  1. Auditorium 1
  2. Auditorium 2
  3. Auditorium 3

  Prompt the user for the auditorium and display the current state of the auditorium. Display the auditorium in the same format that is displayed to the customer. Return to the Admin main menu.

- **Print Report:** Display a formatted report to the console. The report should consist of 7 columns:
  - Column 1 – labels
    - Auditorium 1
    - Auditorium 2
    - Auditorium 3
    - Total
  - Column 2 - **Open seats**: number of all open seats for each label
  - Column 3 – **Total reserved seats**: number of all reserved seats for each label
  - Column 4 - **Adult seats**: number of all adult seats reserved during this session for each label
  - Column 5 - **Senior seats**: number of all senior seats reserved during this session for each label
  - Column 6 – **Child seats:** number of all child seats reserved during this session for each label

    - Column 7 – **Ticket sales:** total amount of ticket sales during this session for each label

- **Exit:** Store the auditoriums back to the files and end the program. The auditorium files should only contain # and . characters. There should be no newline character after the last line in the file so that the files can be used again for another session.