Program 5: RoboTickets System Complete

CSCI 2210 Java Programming Spring 2018

1 Goals.

1. To use a ~~text file~~ object file for output and after for input.
2. To use exceptions and write an exception handler that does more than abort execution.
3. To use a static class variable.
4. To use a boolean variable and a Date variable.
5. To use an ArrayList of objects.
6. Create a GUI Window that displays a message and contains a button.
7. To implement ~~Phase 1~~ finish the two phase well-structured, multiclass system using the MVC design pattern.

In part 2 you will:

7. To add inheritance to the Model Class for payment type.
8. Save data to a binary (object) file. See L7 for instructions on working with Binary Files.
9. Implement whole system as GUI.

You will need **8 scenes or more for this system**. You will need to switch scenes as needed to manage the user interactions. I recommend drawing out the system with several scenarios (Use Cases) of user interaction to make sure your system will do what is required. I have included some challenges for those of you who have more programming experience and want to explore advanced ideas at the end of this document.

Here is a Basic Implementation – with no frills. Your system should look better than this. See the image on the last page.

5.3 The buyTickets() function

Allow the user to select tickets to order. First have them pick the type of event: sports, theater/performance, or concert/event using a ComboBox. Based on the type of event the appropriate age groups should be offered (radio button to select this or use arrows with ticket type like the Movies do), use the prices below. The user should be able to buy tickets from each age group offered. Calculate the total sale and show it on the scene: Fees are based on the ticket type (age group) and event type. You should use a method to calculate the fees, call it chargeFee() and call the User paidAmount() to add this amount to the paidToDate attribute for the User.

1. Adult Ticket Prices:
   a. Sports event ticket are $50
   b. Theater/Performance tickets are $125
   c. Concert/Event Tickets are $150
2. Children pay 50% of Adults.
3. Seniors pay 75% of Adults for all types.

1 Start-up and Welcome Scene – depends on if there are already users in the file.

When you start the application, you should read the data from the file and put it into the ArrayList. **<u>If there are no users</u>** (there is no file or the file is empty), you can send the user to the addUser Scene to add themselves as the owner. It would be good form to have them login at this point, or you can make the owner the current user and proceed to the owner scene.

If the file has users in it, then the first scene should be the Welcome/Login scene. It should have the logon fields. For login name use a textField and for password use the PassWordField (it shows dots instead of the characters entered, it is a subclass of TextField).

2 Owner Scenes – when the owner (userID = 0) is the currentUser.

Keep track of who is logged in by putting their index from the ArrayList (which should be the same as their userID). I like -1 as the currentUser when nobody is logged in, like at startup.

If the Owner is logged in, proceed to the Owner Menu. They should be able to choose to add users, list users and logout from here. I am using the X button on the title bar as the exit button.

If the owner chooses to list users they should proceed to the list users scene. This scene is only accessible by the owner, so it is convenient to allow them to select the user from the ListBox and click edit or go to an edit screen and have them enter the data and find the user to edit that way. This screen also needs a back button. This screen can be combined wit the first owner screen eliminating the need for a list users button if you would like.

You need a separate scene for editing user data by the owner. It would be tricky to combine it with the user scene since the owner and user have different editing privileges. The owner can change username, loginName and totalPaid and dateJoined. They cannot change the password or the payment information. Nobody can edit the UserID, it is final.

3 User Scenes – when the owner is not the currentUser (userID ≠ 0).

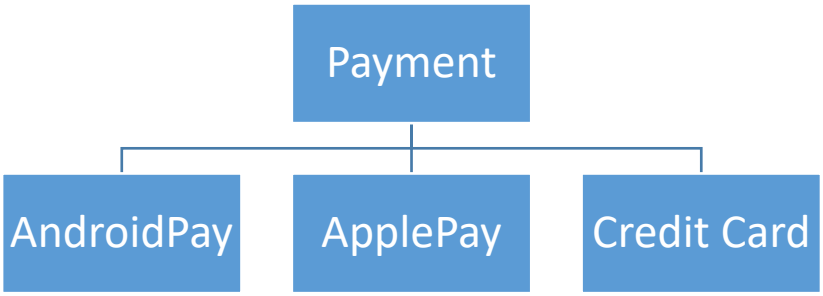
Keep track of who is logged in by putting their index from the ArrayList (which should be the same as their userID). I like -1 as the currentUser when nobody is logged in, like at startup.

If a user who is not the owner is logged in, proceed to the User Menu. They should be able to choose to Edit their own data, order tickets and logout from here. I am using the X button on the title bar as the exit button.

If the user chooses to order tickers they should proceed to the order tickers scene. This scene is only accessible to users. They can select the event from the ListBox or ComboBox and either use radio buttons to pick the type of ticket of have them add tickets of each type like they do at a movie theater. This screen needs a back button. This screen can be combined with the first user screen eliminating the need for an order tickets button if you would like. You must be able to determine the event and type of ticket. ***See challenge options below.***

You need a separate scene for the user to edit their own data. It would be tricky to combine it with the owner scene since the owner and user have different editing privileges. The user can change username, loginName, password and payment info. They cannot change the totalPaid or the dateJoined. They can also close their account on this screen. If they close their account, change isUser to false and don't allow

them to login anymore. Nobody can edit the UserID, it is final. For the Payment Options, implement Android Pay, Apple Pay and Credit Card. Andoid Pay and Apple Pay require a user name and password to be saved. Credit cards require a user name, account number, code and expiration date to be saved. Display only the last four digits of their credit card/accounts for safety. Use inheritance for the payment types.



6 Testing.

When you are sure your program works, delete the input and output files. Then follow the testing procedure below. Add at least 3 more test cases to make sure that all of your code is executed in all options.

(1) Start the program with no existing data file (start new) and enter your own logon, name and Date Joined (it can be any date). Enter this user as the Owner.
   (a) Add a second user you made up.
   (b) Have the second user purchase tickets as follows:
      (i) User ID 1 should purchase two of one type of ticket and 3 of another.
   (c) Exit the system.
   (d) Make a copy of the database file and make its name run1.txt and copy and paste your console output to a file called console1.txt (you can use notepad to save this).

(2) Start the program again (it should read the existing users from the file and add them to your arraylist). Enter two more users – make them up. This makes four users in total.
   (a) Ticket Purchases:
      (i) Have the third user purchase an adult and two child tickets of the remaining type.
      (ii) Have the last user purchase more than one senior ticket.
      (iii) Have the second user purchase a selection of tickets.
      (iv) Test some error conditions.
   (b) Exit the system.
   (c) Make a copy of the database file and make its name run2.txt and copy and paste your console output to a file called console2.txt (you can use notepad to save this).

(3) Create additional test cases so that every line of code in your program is tested. Don't forget to have test runs to test errors in entering the data.

7 Submission.

Due by 11:59 PM on the Tuesday, May 2. Send to my email progPageUNH@gmail.com .

Follow the style guide for your program, put your name at the top, greet and end your program gracefully. Make all interactions clear and check data.
When you are sure your program works, delete the input and output files. Then follow the testing procedure above.

(4) Finally, hand in a zipped folder containing:

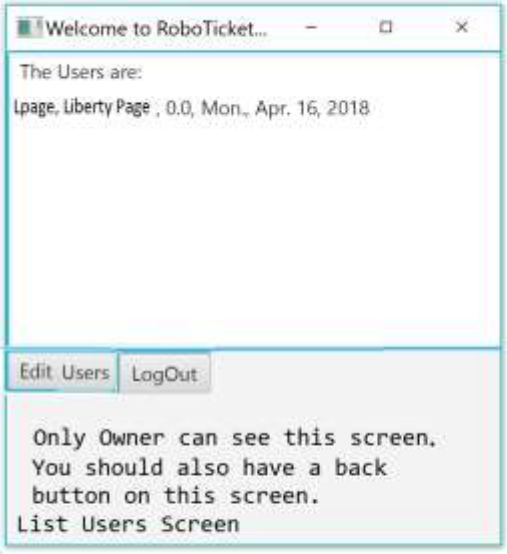
- The .java files for all the classes in your package (the RoboTickets and User class and Main Class (if you created one).
- A screen shot of your GUI windows.
- The clearly labeled data files from each of the test runs.
- Console output from each test run of your code, you can copy and paste them in to the same file, but clearly mark them as run 1, run 2 and so on.

The screen and console output that you hand in must correspond to the file output. Use a simple text editor (such as Notepad) for the console output; DO NOT USE WORD. (Word destroys both code and output by adding things and changing things.)

Please send me only the things I have asked for. I would rather have you organize your work than do it for you. It is incredibly time consuming for me to do this job for the three programming classes.

Send only working code, if you attempted something but cannot make it work, **remove it**. You can send your attempt separately ONLY if it is clearly marked as not working in the name of the program. I might have time to look at it, but I cannot promise anything. I cannot use your projects and I cannot run your compiled code because of incompatible major/minor version problems. (DO NOT believe that Java is a highly portable language!)

Program 5: RoboTickets System Part 2



You need at least three more scenes:

You need a scene to edit the member's data by a member, they can change their password and set up and edit their payment information.

You need another scene for the owner to edit a member's data. The owner cannot change payment or password.

You need another scene to order tickets.

NAVIGATION: The X button (to close the window) in the title bar can be used as the exit button for your program. **You must save the data in the ArrayList to a file before you end your program.** If you have trouble with the X button, you can add your own exit button to each screen.

Screens should have back buttons ⮌ to return to the previous screen. If the previous is the Welcome Screen (login), you can call it Logout.

Challenge Options (not required):

After you have implemented the basics, check out ticketing websites and see how much you can add. Make this is separate project and share what you figured out with the class at our last class. For example, above is a page from ticketron.com, you can pick your seat and see the view. For your project, you can make every stadium the same and maybe have three image files to choose from? **This will prove you can do it**, but you can't spend 10,000 hours needed to implement every detail. This should be fun and exciting.

Use SHA-512, 3-DES or similar hashing for the password, save the hashed password only and use it to verify the user.

You can propose your own ideas. All of it must be done in Java 8 and Java FX.