CIS263AA Final Project Documentation

Fill in requested information below and submit this document as part of your final project zip file.

# General Information

Your Name: Lina Breunlin

## Overview

In 2-3 sentences, describe your final project. What is it? What does it do?

* My final project is a ticket reservation system for a small theatre that offers general admission seating. The GUI allows the theatre patron to input their first and last name and reserve the appropriate type of ticket.
* Once the patron has reserved their ticket, the application assigns them an auditorium entrance number based on how far in advance they made their reservation, which determines the order they would be let into the auditorium on the night of the performance to pick their seat.

## Directions

Explain how I should use/test your project.

* First select the appropriate type of ticket, once this is done, input the patron’s first and last name in the text fields. After that click on the “reserve” button which will create the appropriate type of ticket object based on your selection using the input from the text fields and add it to the queue.
* After you have clicked reserve, click either “receipt” to send the information to an external file (receipt.txt) and start over or click the clear button. Once all the tickets have been reserved, click the “Entry List” button. This will give you the order in which the patrons reserved their tickets and tell you the order that they can enter the theatre on the night of the show.

## Sources/Resources

List any outside sources you used to solve problems, including complete URLs. You do not need to list sources from the course (the textbook, videos and resources linked in the Module Overview pages, etc.). *Note: Failure to identify a source will be considered plagiarism.*

* YouTube Video on populating combo boxes: <https://www.youtube.com/watch?v=YO9PQfTdbFI>
* Online tutorial for using the getValue() method for combo boxes

<https://www.programcreek.com/java-api-examples/?class=javafx.scene.control.ComboBox&method=getValue>

* Setting a default value for a comboBox

<https://stackoverflow.com/questions/34949422/how-to-set-default-value-in-combobox-javafx>

* Online tutorial for using the DecimalFormat class: <https://javarevisited.blogspot.com/2012/03/how-to-format-decimal-number-in-java.html>
* How to convert a Queue to an arrayList

<https://stackoverflow.com/questions/12779839/convert-a-queue-to-list>

* Online tutorial on printing the contents of an arrayList

<https://www.interviewsansar.com/how-to-print-arraylist-elements-in-java/>

* Online tutorial for writing to a file

<https://howtodoinjava.com/java/io/java-write-to-file/>

* Clearing a text field in a GUI

<https://stackoverflow.com/questions/28516654/javafx-how-to-clear-textfield-when-mouse-is-clicked-on-it>

* Disabling a button

<https://stackoverflow.com/questions/17871329/disabling-a-button-in-javafx>

* Formatting new line

<https://stackoverflow.com/questions/29038627/how-can-i-print-a-new-line-in-a-tostring-method-in-java>

# Project Requirements

## Object-oriented Design

Explain how you organized your application into classes and objects, including how you chose to implement inheritance. You may supplement your explanation with a UML-style class diagram if you wish.

* The fictitious theatre that this application is built for sells four types of tickets, Adult, Senior, Student and Child. Each ticket costs a different amount and has a different description of who is applicable to purchase it. Otherwise, all the other information that pertains to the tickets is the same.
* Because of this I chose to create an abstract super class called “tickets” because I wanted the fields and methods to be inherited by all the subclasses but the theatre does not sell just a “ticket”, it has to be one of the four types mentioned above, so I thought it made more sense to use an abstract class rather than an interface.
* The subclasses do not override any methods, but each has its own constructor which populates with several default values. I chose to have the constructors supply the default values because those things would always be the same (i.e. an Adult ticket is always $12.00).

## Use of a dynamic or “advanced” data structure

Explain where I can find an example of a meaningful/significant use of a dynamic or “advanced” data structure, including specific file names and/or line numbers. *Note: an array will not meet this requirement.*

* The application uses a queue to keep track of the patrons who are reserving tickets. The patrons are incentivized to reserve their tickets far in advance because the theatre does not assign seats.
* Patrons are assigned an auditorium entrance number which is the order on show night they are let into the auditorium to pick their seat, so the earlier they reserve a ticket, the better chance they have of getting a good seat. A queue made sense for this because it is a first in first out data structure and I was picturing patrons on the night of the show waiting in line to go into the auditorium.

## JavaFX

Give a brief overview of the decisions you made regarding you GUI. How did you decide on widgets/controls? Did you code by hand or use Scene Builder (or another tool)?

* For the GUI, I opted to have three prepopulated labels that would tell the user what they are reserving a ticket for.
* I chose to have a combo box for the ticket type because there are only those four options and the user only gets to pick one. Stylistically, I liked the look of the drop down better than a radio button for choosing only one option.
* The other two fields are the text entry fields for first and last name which are used to collect input from the user that is used for creating ticket objects and gets passed into the queue.
* I created the GUI using scene builder because I thought that was a more efficient use of time than trying to code by hand. I could lay it out relatively quickly and then spend more time on the business logic rather than the user interface.

## Multi-threading, file I/O, and/or network functionality

Explain where I can find use of multi-threading, database access, or network connectivity, including specific file names and/or line numbers.

* The application writes to two different files. After a patron has reserved their ticket, there is a receipt button that takes all the information in the ticket object’s overridden toString() method and writes it to an external file called “receipt” (line 193).
* Once all the tickets have been placed into the queue (or if you want to check part way through, you can do that too) there is a button on the GUI called “Entry List” that converts the queue to an array and then prints the patron’s name with their corresponding entry number to an external file called “patronEntry.txt” (line 238).