

**BCIS 3630**  
**Object-Orient Programming**  
**Problem #6: Arrays and Collections**

Extend the last problem to provide manipulations that involve the following logical functions: (NOTE: The phrase “an array” will appear in many instances below. You may interpret this phrase to read “one or more arrays.”)

1. Past versions of this application have been limited in the sense that you had to create output for lodgers and reservations as the data was entered. For this application, data entered for each of the reservation is to be loaded into an array and retained for later output to the screen via a pane.
2. Accumulation, manipulation, production and output of all detailed and summary information you produced in past applications is to be “delayed” until all data has been entered. These “statistics” are then to be produced based on the data retained in an array and generated as a value only after all data has been accepted for processing. This means that no output is to be produced until all input has been completed and the entry for more reservations has been answered “no.”
3. The output generated by this application should be arranged such that the information about the nearest reservation is listed first. (In other words we want to have the most imminent reservation nearest the top of the output list. Each subsequent reservation should be for lodgers reservations further into the future. If two or more dates are identical, then you are to arrange the reservations alphabetically by renter name. For purposes of this application you may assume that the initial reservation date for a given lodger will not be later superseded by a date entered later.
4. Summary output information is to be extended to include the revenue (rental amounts) and a count of the number of reservations by individual lodging units (e.g., for the Old Faithful Inn, how many times did it get rented during this session. This list should be alphabetically ordered by unit name (and should only show the actual units rented—i.e., no unit should be show that has zero rentals).
5. You are to employ at least one try/catch block to insure processing does not extend beyond the end of an array.
6. At least one additional class should be created to handle some aspect of the array processing of your application.
7. Your application should utilize features associated with a “collection class.”

Other requirements:

1. You are to supply standard elements (hardcopy, softcopy, sample output generated, etc.) for this problem.
2. Your problem should also be fully documented (javadoc and Visio).
3. Your application should be developed and deployed in NetBeans.

4. Your “startup” code should be executable from a file designated as “**Prob6Driver**”. This driver should serve only the function of launching the application.

**Due Date: December 3<sup>rd</sup>**