**Java Chapter 30 Lab B**

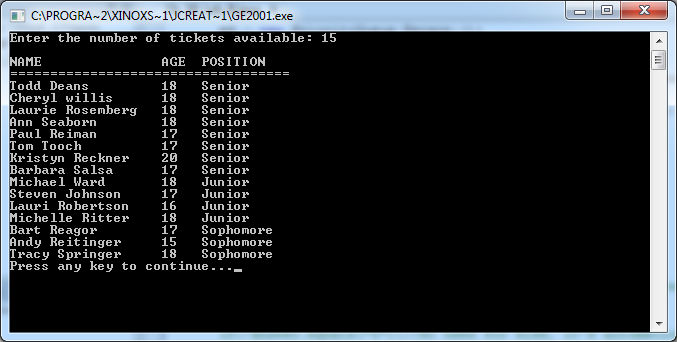
The purpose of this lab is to demonstrate your knowledge of Java’s **PriorityQueue** class by making a list of students and teachers from a high school. The list is ordered by position (**Teachers** are fist, then **seniors**, then **juniors**, etc). There will also be a secondary priority that orders the list by age (The older seniors are placed before the younger seniors, etc), but that will be done later.

You will be provided with a shell of a program. You must read a list of people from a text file called *students3b.dat* located on the P:\ drive. The list contains three pieces of information about a person: their **name**, their **age**, and their **position**. You will then write your own class to order the list of people in the school. You will enqueue the three values into the **PriorityQueue** as a single object.

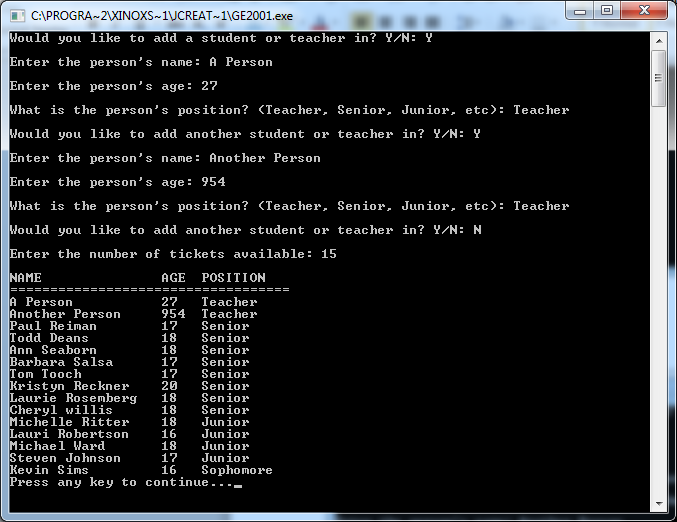
The methods in the main class are shown below:

|  |
| --- |
| *import java.io.\*;*  *import java.util.\*;*  *public class Ch\_30\_Lab\_B*  *{*  *static PriorityQueue<Person> pQ;*  *public static void main(String[] args) throws IOException*  *{*  *getList();*  *insertPerson();*  *displayTicketOwners();*  *}*  *//This method reads in the information from students3b.dat*  *public static void getList() throws IOException*  *{*    *}*  *//This method asks the user if they want to add another person to the list.*  *public static void insertPerson()*  *{*    *}*  *//This method displays all of the students/staff that received a parking permit*  *public static void displayTicketOwners()*  *{*    *}*  *}* |

For the first part of the program, you only need to order the list by position. To do this, your class needs to implement the *Comparable* interface so that the **compareTo** method is present. You don’t need to be able to add a new person quite yet. Your output should look like this:



For the next part of the program, use the **insertPerson** method to ask the user if they want to add in another student or staff. When adding new people, make sure that the order is first come, first serve like a queue would (As long as they have the same position; a teacher should still be prioritized over a senior.). Your output should be similar to:



Finally, you must implement a secondary priority, where a person’s age determines the indices between two people of the same position. The two teachers added in the second part are still added in the following output. The final outcome of the program should look similar to:

