Bluegrass Community and Technical College

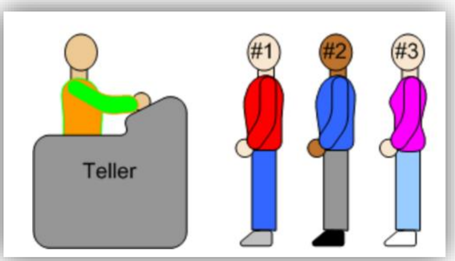
Programming Requirements Document

Voting Queue using CircularArrayQueue Class and QueueADT Interface

NARRATIVE DESCRIPTION

A Queue is an abstract data structure in which elements are added to the rear and removed from the front. This structure is often referred to as a FIRST IN FIRST OUT (FIFO) structure.

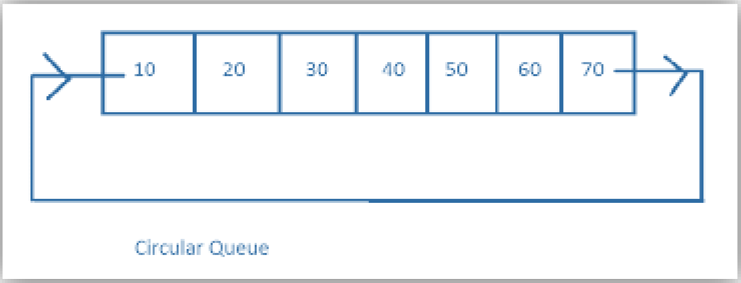
There are many examples of queues in the "real world": (1) waiting lines – such as standing in line for movie tickets, (2) our digestive system, and (3) cars driving on a one lane one-way street.



Make sure you understand the typical actions of a queue, such as enqueue(), dequeue(), size(), isEmpty(), before you begin.

This assignment has an accompanying ZIP file that was provided in Blackboard. Download the ZIP file and unzip the contents. Do not move any of the files around within the folder created when the file is unzipped. The zipped file contains:

1. A VotingQueue.java file
2. A Voters.csv file
3. A jsjf folder



You will start with the **VotingQueue.java** file to complete your work.

VotingQueue.java will need to use the **CircularArrayQueue** class and the **QueueADT** interface (provided in the jsjf folder and discussed in the Lewis chapter you read this week), and the **Voting.csv** file. <u>These Lewis files allows you to implement and use a queue.</u>

The **Voting.csv** file contains a list of cars that arrive at a voting precinct to vote in a local election. Each car group has various number of voters. For example, the first few records in the file are:

```
1,3
2,4
3,2
4,1
5,2
```

The first number on a line is a <u>car number</u> (1, 2, 3, 4, 5, …)

The second number on a line is the <u>number of voters in the car</u> (3 in the first car, 4 in second car, 2 in the third car, 1 in the fourth car, 2 in the fifth car, etc.)

You will create two short programs this week. Steps 1-3 below will be included in the first program and it simply insures you can enqueue and dequeue items in a queue.

Step 4 will be included in the second program and it functions like a typical queue.

Step 1: Create a queue of the Voters

You are working at a local voting precinct and arrive about 15 minutes before voting opens. You see there is already a line of voters ready to cast their ballot! In other words, the voters are already waiting in line. Remember waiting lines are a type of queue. Each voter group (car group) has been given a number for their place in line (1, 2, 3, etc.).

You will create a queue of integers (the number of voters in each car). Read the records from the **Voting.csv** file and add the number of voters in the cars to a queue. Looking back as the sample data on the previous page, the first car has 3 voters so place 3 in the queue. The second car has 4 voters in the car, so place 4 in the queue, etc. Note: You are not doing anything with the car number, you are simply placing the number of voters from the car in the queue.

Step 2: Remove items from the queue and process the voters. Create a report of all voters and time to vote.

After the waiting cars groups in **Voting.csv** have been placed in the queue, use the queue to:

- Remove an integer from the queue and process each of the voters in the vehicle.
- To process the voters:
    - Assign a voter number to each voter. Start at 1 and increment by 1 for each voter. This a sequential number for all voters. Do not start over at 1 again with each new car.
    - Generate a random number from 60 – 900 for each voter. This represents the length of time (in seconds) it took the voter to vote.
    - For example, if there are 3 voters in the first car, their voter numbers will be 1, 2 and 3. You will generate time for each of them.
- Create a report (to a text file) with headings, date, and details of voting time. This will include columns for voter number and voting time for each voter. Make sure this is in a professional format and columns are aligned.
- *NOTE: You may temporarily display the voter number and time on the screen while you are testing your program but ultimately, you need to create a report with headings and columns written in a file. Sample were provided in previous weeks of how to create an output file. While the program is processing the data, make sure the user knows processing is occurring. The user should never guess as to what is occurring in the program.*

Step 3: Display statistics

At the end of the report include the count of voters, the total time it took for all to vote, and the average time it took all to vote. You are welcome to display on the screen as well.

Test Steps 1-3 and save your files before proceeding to Step 4.

Step 4: Process data like a queue

Copy your program and create a second version. Name the second version **VotingQueue2**.java.

When you use a queue, you typically do not enqueue all data at once and then dequeue all at once. You allow data to be added and removed as needed. Think of a line to buy movie tickets. People arrive and line up at the end of the line. People at the beginning of the line purchase tickets. There is a constant flow of processing from the front of the line while people line up at the end.

Modify the program so the you can enqueue (add) and dequeue (remove) data as voters arrive.
The program should ask the user what they wish to do. The user can select any of the following in any order:

A. Add a carload of voters to the queue (enqueue data)
    - The user will need to input the number of voters in the car
    - Add the number to the queue
B. Process voters from a car and document the results in a report(dequeue data)
    - Process each of the voters from the car as described in step 2. You will take the next item from the queue and process that many voters. You are creating a report as before.
C. Display statistics on the screen
    - Display the people who have voted so far (only those who actually voted)
    - Display the total time it took them to vote
    - Display the average voting time it took them to vote
    - Display the total number of items remaining in the queue
D. End the program
    - You will need to add statistics to the end of the report (same data as in C above)
        - Total number of voters
        - Total time it took the voters to vote
        - Average time it took the voters to vote
        - Remaining number of cars in the queue when the program ended.

Step 4 allow you to see the normal functioning of a queue (some items go into the queue, some come out, but not necessarily all at the same time). Notice in Step 4 you are not removing all voters at once. You only remove one item from the queue at a time.

For grading, zip the jsjf folder, the Voting.csv file, the VotingQueues.java class, and the VotingQueues2.java class and submit in Blackboard.

New Concepts Assessed and Illustrated (in addition to any previously learned)

- Queues

Software Requirements

R1: A queue of voters is properly created using the CircularArrayQueue Class and QueueADT Interface.
R2: Voters are properly processed and displayed.
R3: Statistics are properly calculated and displayed.
R4: A professional report is created of the voters with headings, date, and statistics.
R4: User can process voters like a queue (step 4).

Security Considerations

The public and private access modifiers have security implications. Use a private modifier for all instance data items. Create accessors for instance data items only when absolutely needed. Do not include sensitive or private information in data returned by the toString() method.