# Project Report:

# Portfolio Project – Custom Queue ADT and Quicksort

Alejandro Ricciardi

Colorado State University Global

CSC400: Data Structures and Algorithms

Professor: Hubert Pensado

October 6, 2024

**Project Report:**

**Portfolio Project – Custom Queue ADT and Quicksort**

This documentation is part of the Project Report Assignment from CSC400: Data Structures and Algorithms at Colorado State University Global. This Project Report is an overview of Milestone 1 containing Module 2 and Module 4 programs (Program 1 and 2), and Milestone 2 containing Module 5 and Module 6 programs (Program 3 and 4). It also showcases the final program (Custom Queue ADT and Quicksort) the program's functionality and testing scenarios including console output screenshots. The programs are coded in Java JDK-22.

**The Assignment Direction:**

Option #1: Person Class
Your Portfolio Project for CSC400 consists of the following:

- Milestone 1 (due in Module 5): Java source code (with corrections if necessary) for programs created in Module 2 and Module 4.
- Milestone 2 (due in Module 7): Java source code (with corrections if necessary) for programs created in Module 5 and Module 6.
- Lessons Learned Reflection
- Final Program

Lessons Learned Reflection:
Write a 2-page summary that outlines the lessons you have learned in this programming course. Reflect on how these lessons can be applied towards more effective coding.

Final Program:
Write a program that creates a Person class that contains strings that represent the first and last name of a person and their age. You will need to create a Queue class that will store each person in the queue and can sort the queue based on last name or age.
Prompt the user of the program to add five people to the queue. Your program should provide the contents of the queue and then sort the queue using the quick sort in two ways:

1. Descending order by last name
2. Descending order by age

Assemble your Lessons Learned Reflection, your source code, and screenshots of the application executing and results into a single document. Submit your completed Portfolio Project by the posted due date.

⚠️ **My notes:**

- No corrections were needed for the programs in Milestones 1 and 2.
- The Milestones 1 programs' files can be found in the directory Milestones-1.
- The Milestones 2 programs' files can be found in the directory Milestones-2.
- The final program's Java code source files can be found in the folder Final-Program.
- Screenshots for the final program can be found in the Screenshots folder.
- I used the generic data type in my implementations of the queue and Quick Sort in my final program.

**Git Repository**

I use GitHub as my Distributed Version Control System (DVCS), the following is a link to my GitHub, Omegapy.
My GitHub repository that is used to store this assignment is named My-Academics-Portfolio and the link to this specific assignment is: https://github.com/Omegapy/My-Academics-Portfolio/tree/main/Data-Structures-and-Algorithms-CSC400/Portfolio-Project

---

## Final Program

**Program Description:**

This program implements in Java a generic Linked-list queue and sorts the queue using a quicksort algorithm.
The queue stores Person objects representing a person's first name, last name, and age.
The Person objects in the queue can be sorted by last name or age.

Quicksort algorithm notes:

- The quicksort algorithm implements a Hoare partition to partition the queue. Meaning that the head node of the linked-list queue is picked as the pivot.

- Additionally, a queue ADT sort is expected to be stable, preserving the relative (entry) order of elements with equal values.

- To partition a linked-list, the element needs to be traversed element by element. Dividing the list into three parts (left, equal, and right) helps simplify the recursion of traversing the list element by element and preserving elements with equal values in order of entry. This also avoids moving elements around in memory, as is required in array-based implementations of quicksort.

**Classes Description:**

- **Person Class**
  The class represents an individual with a first name, last name, and age.

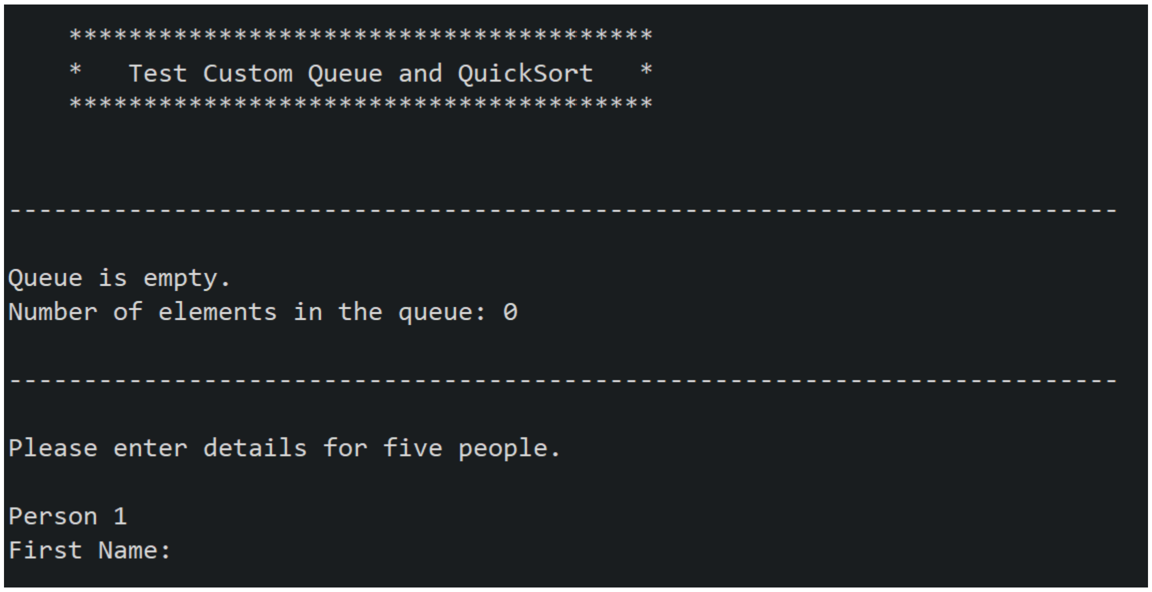
- **MyQueue<T> Class**
  Generic queue ADT (Abstract Data Type) using a linked-list.

- **MyQuickSort Class**
  A generic implementation of the quicksort algorithm on a linked-list queue that returns a sorted LinkedList without modifying the original queue. See Program Description's notes for more information about the quicksort algorithm type used in this program.

- **Main Class**
  Test the queue and the quick sort. It prompts the user to enter data for five Person objects, adds them to the queue, sorts the queue based on last name and age, and displays the contents of the queue.

**Screenshots**

**Figure 1**
*Welcome Screen*

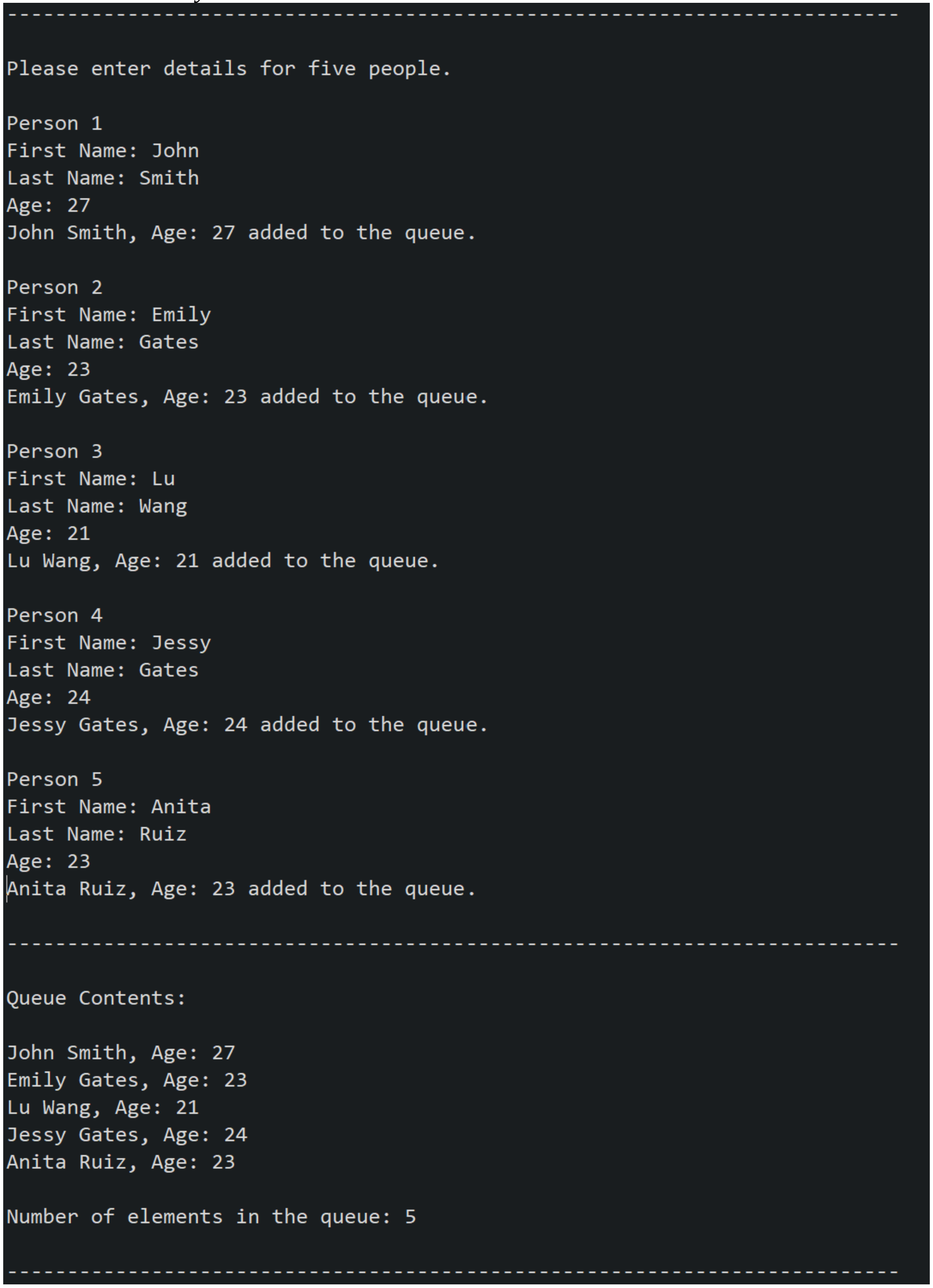


```
    ***************************************
    *   Test Custom Queue and QuickSort   *
    ***************************************



---------------------------------------------------------------------------

Queue is empty.
Number of elements in the queue: 0

---------------------------------------------------------------------------

Please enter details for five people.

Person 1
First Name:
```

*Continue next page*

**Figure 2**

*Data Persons Entry*

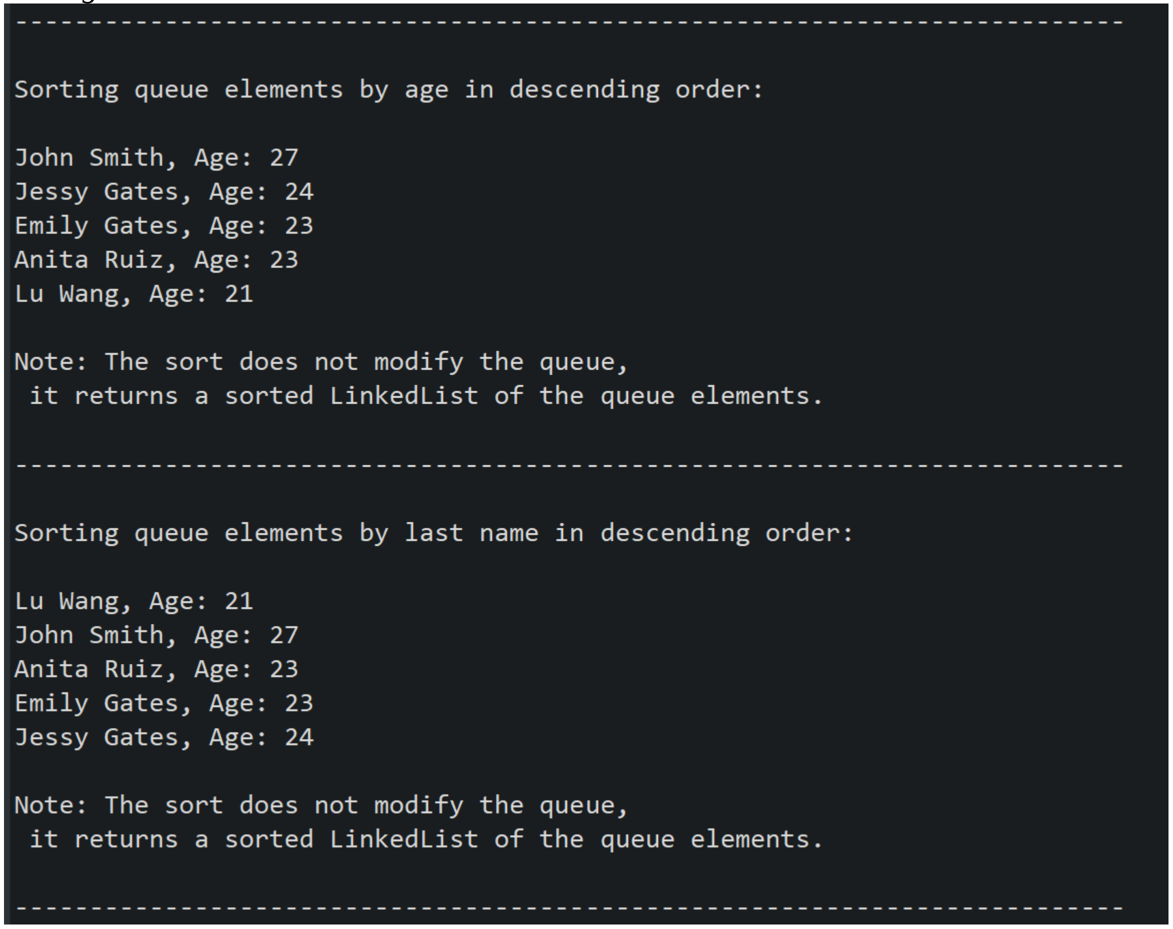


```
---------------------------------------------------------------------------

Please enter details for five people.

Person 1
First Name: John
Last Name: Smith
Age: 27
John Smith, Age: 27 added to the queue.

Person 2
First Name: Emily
Last Name: Gates
Age: 23
Emily Gates, Age: 23 added to the queue.

Person 3
First Name: Lu
Last Name: Wang
Age: 21
Lu Wang, Age: 21 added to the queue.

Person 4
First Name: Jessy
Last Name: Gates
Age: 24
Jessy Gates, Age: 24 added to the queue.

Person 5
First Name: Anita
Last Name: Ruiz
Age: 23
Anita Ruiz, Age: 23 added to the queue.

---------------------------------------------------------------------------

Queue Contents:

John Smith, Age: 27
Emily Gates, Age: 23
Lu Wang, Age: 21
Jessy Gates, Age: 24
Anita Ruiz, Age: 23

Number of elements in the queue: 5


---------------------------------------------------------------------------
```

*Note:* Showcase the enqueuing functionality of the queue

**Figure 3**

*Sorting*



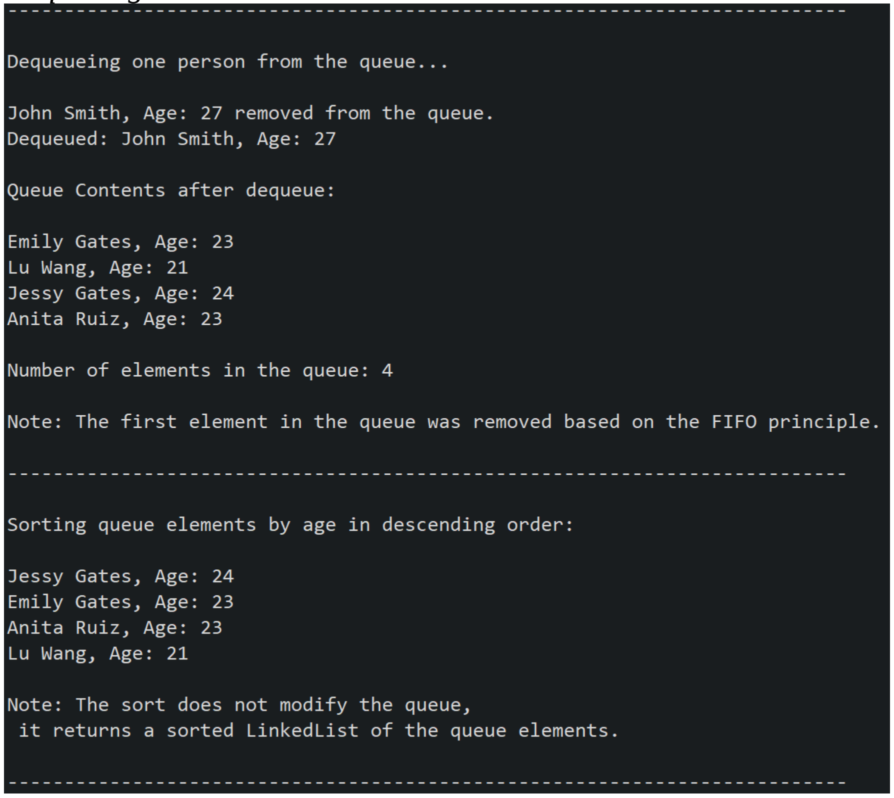
```
--------------------------------------------------------------------------

Sorting queue elements by age in descending order:

John Smith, Age: 27
Jessy Gates, Age: 24
Emily Gates, Age: 23
Anita Ruiz, Age: 23
Lu Wang, Age: 21

Note: The sort does not modify the queue,
 it returns a sorted LinkedList of the queue elements.

--------------------------------------------------------------------------

Sorting queue elements by last name in descending order:

Lu Wang, Age: 21
John Smith, Age: 27
Anita Ruiz, Age: 23
Emily Gates, Age: 23
Jessy Gates, Age: 24

Note: The sort does not modify the queue,
 it returns a sorted LinkedList of the queue elements.

--------------------------------------------------------------------------
```

*Note:* The sort does not modify the queue, it returns a sorted LinkedList of the queue elements. This preserves the FIFO principle when removing an element from the queue. Also, the sorting is stable, it is preserving the relative (entry) order of elements with equal values.

*Continue next page*

**Figure 4**

*Dequeuing Element*

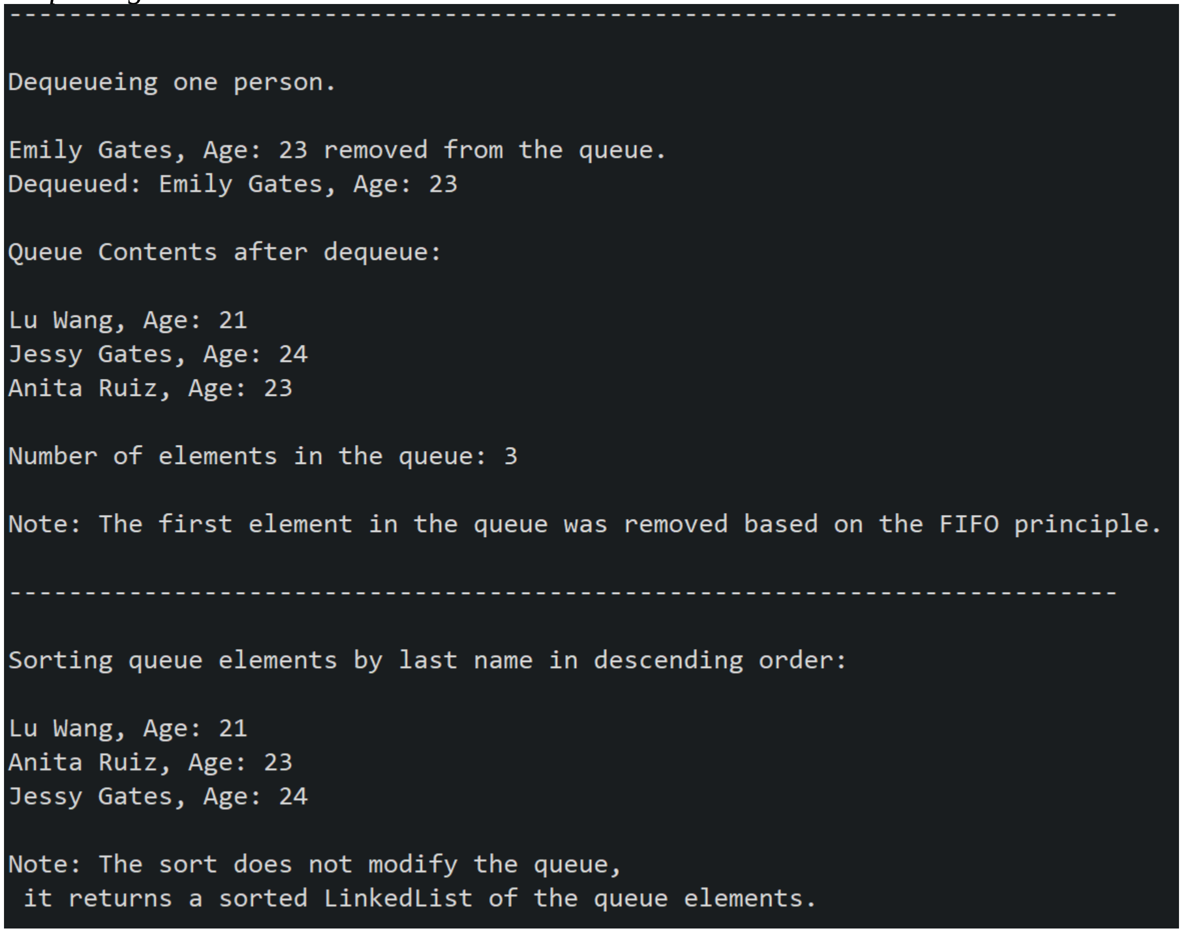


```
--------------------------------------------------------------------------

Dequeueing one person from the queue...

John Smith, Age: 27 removed from the queue.
Dequeued: John Smith, Age: 27

Queue Contents after dequeue:

Emily Gates, Age: 23
Lu Wang, Age: 21
Jessy Gates, Age: 24
Anita Ruiz, Age: 23

Number of elements in the queue: 4

Note: The first element in the queue was removed based on the FIFO principle.

--------------------------------------------------------------------------

Sorting queue elements by age in descending order:

Jessy Gates, Age: 24
Emily Gates, Age: 23
Anita Ruiz, Age: 23
Lu Wang, Age: 21

Note: The sort does not modify the queue,
 it returns a sorted LinkedList of the queue elements.

--------------------------------------------------------------------------
```

*Note:* that the person dequeued is ‘John Smith’ the first entered in the queue. This is in accordance with the FIFO principle.

*Continue next page*

**Figure 5**
*Dequeuing Another Element*



```
---------------------------------------------------------------------------

Dequeueing one person.

Emily Gates, Age: 23 removed from the queue.
Dequeued: Emily Gates, Age: 23

Queue Contents after dequeue:

Lu Wang, Age: 21
Jessy Gates, Age: 24
Anita Ruiz, Age: 23

Number of elements in the queue: 3

Note: The first element in the queue was removed based on the FIFO principle.

---------------------------------------------------------------------------

Sorting queue elements by last name in descending order:

Lu Wang, Age: 21
Anita Ruiz, Age: 23
Jessy Gates, Age: 24

Note: The sort does not modify the queue,
 it returns a sorted LinkedList of the queue elements.
```

*Note:* The person dequeued is ‘Emily Gates’, who is the person in the front of the unsorted queue after the first person was dequeued and the second person entered in queue. This is in accordance with the FIFO principle.

As shown in Figures 1 to 5 the program runs without any issues displaying the correct outputs as expected.