ECE242_Project 2 Linked Lists and Queues

Apartment Assignments

Overview

The Family Housing of University of Massachusetts Amherst provides limited off-campus apartments to students. The housing system uses a waiting list for apartment assignments. According to Family Housing policy, when a match between an applicant and an apartment is found, the apartment is assigned to the applicant.

Introduction

In this project, you need to find matches between applicants and apartments based on a list of available apartments and a list of waiting students. Each apartment has a quality rating (an integer between 1 and 10) and three features: location, number of bedrooms, and laundry availability. To facilitate matches, each applicant indicates his/her quality threshold, desired location, number of bedrooms and desire for laundry.

The **assignment process** assigns students to apartments in a "first come first serve" manner. It picks the first student in the waiting list, and then scans the list of available apartments. If an apartment that satisfies the student's requirements is found, it is assigned to him/her, and it is removed from the available apartment list. The next student is then processed. If no apartment satisfies the student's requirements, the student remains unassigned and the process moves on to the next student. Note that an apartment satisfying a student's requirements matches all four student requirements (the quality value of the apartment is larger than or equal to the student's quality threshold; location, number of bedrooms, and laundry availability are exactly what the student desired, except when he/she doesn't care about these features). The assignment process finishes when either all students in the waiting list have been picked or there are no more available apartments.

Input files

You will be given an apartment file which includes a series of available apartments (each line describes one apartment) in the following format:

Apartment ID | location | # of bedrooms | includes laundry | quality rating

The following lines illustrate the syntax:

```
2001 Lincoln  2  Yes  6
2002 Lincoln  1  No   8
2003 NV       1  No   7
```

The first line, for example, indicates that the apartment 2001 is located at Lincoln, has

laundry as well as 2 bedrooms, and its quality rating is 9.

You will be also given a student file which describes students in the waiting list in the following format:

ID | Name | required location | required # of bedrooms | desire for laundry | quality threshold

The following lines illustrate the syntax:
10000001 Garnett Lincoln 1 No 7
10000002 James Any 2 Any 6

The first line indicates the first student in the waiting list. It indicates that Garnett (ID: 100000001) requires an apartment with quality rating at least 7 that is at Lincoln and has 1 bedroom and no laundry. The second line, which describes the second student, shows that James (ID: 10000002) requires an apartment with quality rating at least 6 that has 2 bedrooms and doesn't care about the other two features of the apartment ("Any" means the student doesn't care about the corresponding feature of an apartment).

Output

Print out assignment results which include: which apartment was assigned to whom, unhoused students (if any) and unassigned apartments (if any).

For example, the output of the assignment results based on the above available apartment list and waiting list should be as follows:
The apartment 2002 is assigned to Garnett (10000001).
The apartment 2001 is assigned to James (10000002).
There are no unhoused students.
The apartment 2003 is unassigned.

Project Details

- Implement a class called "**Apartment**" which should contain a constructor to initialize the apartment's ID, quality rating and its three features (location, number of bedrooms and whether having laundry). These are the basic features that your **Apartment** class must contain. You can also define other properties and methods in **Apartment** if you wish.

- Create a class called "**ApartmentList**" which stores a linked list of the **Apartment**s. Implement the following methods in this class: insert/add, remove, isEmpty. Like the previous step, you have free reign to include any additional methods.

- Implement a class called "**Student**" which should contain a constructor to initialize the student's ID, name, and his/her requirements (required location, required number of bedrooms, whether requiring laundry and the quality threshold). Add other

properties and methods if you wish.

- Use a queue to store all the students in the waiting list. Utilize a Doubly Linked List to implement the queue class (name it **WaitingStudentQueue**). Implement the following methods in this class: enqueue, dequeue, isEmpty. Feel free to include any additional properties and methods.

Notes

- You must write your own **ApartmentList** class and **WaitingStudentQueue** class for this project. Do not use the built-in LinkedList or Queue interface in Java.

- Make sure your solution is not limited to the sample input files. We may use different input files (with the same format) for grading.

Grading

You should submit your completed code by the date posted on the course website. You will receive 80 points for your code running accurately, and 20 points for the thoroughness and usefulness of your comments which should be included in the code.

- ✓ Implementing class Apartment (5 points)
- ✓ Implementing class ApartmentList (10 points)
- ✓ Implementing class Student (5 points)
- ✓ Implementing class WaitingStudentQueue (15 points)
- ✓ Parsing input files correctly (10 points)
- ✓ Assigning apartment correctly (35 points)
- ✓ Commenting thoroughly and usefully (20 points)