ECE242 PROJECT 2: Apartment Assignments

Due: October 16, 2014, 11PM on Moodle

Introduction

The Family Housing at University of Massachusetts Amherst provides limited off-campus apartments to graduate 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.

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 rent and three features: location, number of bedrooms, and laundry availability. To facilitate matches, each applicant indicates his/her desired rent, desired location, number of bedrooms and desire for laundry.

The **assignment process** assigns apartments to students 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 unhoused and the process moves on to the next student. Note that an apartment satisfying a student's requirements matches all of the four requirements (the rent of the apartment is less than or equal to the rent desired by the student; location, number of bedrooms, and laundry availability are exactly what the student desired, except in cases where 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 | including laundry | rent (dollars)

The following lines illustrate the syntax:

```
2001 Lincoln  1  No   700
2002 Lincoln  2  Yes  800
2003 NV       1  No   800
```

The first line, for example, indicates that the apartment 2001 is located at Lincoln, with no laundry as well as 1 bedroom, and its rent is 700 dollars.

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 | desired rent (dollars)

The following lines illustrate the syntax:
10000001 Garnett Lincoln 1 No 600
10000002 James Any 2 Any 900

The first line indicates the first student in the waiting list. It indicates that Garnett (ID: 100000001) requires an apartment with rent no more than 600 dollars, located at Lincoln, with 1 bedroom, and without laundry. The second line, which describes the second student, shows that James (ID: 10000002) requires an apartment with rent no more than 900 dollars, and with 2 bedrooms. James 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). Note that "Any" may show in three fields: "required location", "required # of bedrooms" and "desire for laundry".

Task Overview

In this assignment, you will perform the following specific tasks.

1. Read in the apartment file and the student file.

2. Print out all the information of the available apartments and all the information of the waiting students that you just parsed from files.

3. Implement the following classes in order to perform the assignment process:

   - Implement a class called "Apartment" which should contain a constructor to initialize the apartment's ID, rent and its three features (location, number of bedrooms and whether having laundry). These are the basic properties 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 Apartments. Utilize a Doubly Linked List to implement this class and include the following three methods: insert/add, remove, isEmpty. Like the previous step, you can include any additional methods if you wish.

   - 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 desired rent). Add other properties and methods as needed.

   - Use a queue to stores all the students in the waiting list. Utilize a singly

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.

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

Getting started

The best way to complete any programming assignment (including this one) is to take tasks step-by-step. The first decision you need to make is the number of classes you will write. For this assignment, you should have at least five classes, including the four classes described in Task Overview and a driver class. Please do not place all the methods for your entire project in one class.

Required Output

1. Print out all the information of the available apartments and all the information of the waiting students before starting the assignment process.

2. 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:
There are no apartments satisfying Garnett (10000001)'s requirements.
The apartment 2002 is assigned to James (10000002).
The apartment 2001 is unassigned.
The apartment 2003 is unassigned.

Hints and suggestions

Successfully completing the project and achieving a good grade requires completing the project as described above and clearly commenting the code. As always, it makes sense to start the project early. Build your project code step by step.

- Verify that you have successfully read data in the files before attempting the assignment process. You might want to reuse some code in Project 1, especially the code in the "loadBooksToArray" method, to load files. Note that in this project, you can simply use a white space as the delimiter to split a line.

- If you parse the "required # of bedrooms" field in the student file as integers, make sure your code can handle the case that the field contains "Any".

- You can assume the "rent" field in the apartment file and the "desired rent" field in the student file include only integers.

- For printing out assignment results, one convenient way is to print out the assignment result for a student when you try to assign an apartment to him/her and then print out unassigned apartments (if any). Of course, you can also print out the assignment results in any other order you like.

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.

What to submit:

Please submit all .java files for your project and the screenshots for all the print out results. All code should be well commented.

Reminder: The course honesty policy requires you to write all code yourself. Your submitted code will be compared with all other submitted code for the course to identify similarities. Note that our checking program is not confused by changed variable or method names.

Grading

- Code works (50%)
- Comments (20%)
- Program structure (20%)
- Readability (10%)