CSC 115: Fundamentals of Programming II
Assignment #4: Hospital Admissions

Due date

Monday, March 26 at 09:00 am via submission to conneX.

How to hand in your work

Submit the requested files through the Assignment #4 link on the CSC 115 conneX site. Please make sure you follow all the required steps for submission (including confirming your submission).

Learning outcomes

When you have completed this assignment, you are expected to be able to:

- Use a BinarySearchTree data structure to
  - store and manipulate comparable items,
  - traverse the tree to produce an ordered list.
- Continue to apply good programming practice in
  - designing programs,
  - proper documentation, and
  - testing and debugging problems with code.

Hospital Admission Information System Description.

The local hospital Admissions Department needs to keep a record of current patients and room numbers. During the day, the Admission staff update, insert, delete and retrieve patient information from one of the workstations in the Admissions Department. Visitors also ask for information about the location of their loved ones. In the evening, when the Admissions Desk staff go home, two volunteers take over the job of locating patients for the evening and night visitors. However, the volunteers to do not have access to a workstation and therefore work from a printout of ordered patient names and room numbers. This list is printed daily for them by one of the Admission staff just before she goes home.

Problem description

We are asked to design a system for the Admissions Department. The hospital generally deals with thousands of admitted patient information and patient enquiries need to be handled quickly. We elect to use a BinarySearchTree data structure for relatively quick and easy access to the patient records, and because the *inorder* traversal easily processes records in their natural order, the Admissions Desk can produce an ordered printout for the volunteers at night.

Programming requirements

You will write the source code for the following single class:

- *AdmittedPatients.* The underlying data structure will be a reference-based BinarySearchTree.
    - Note that you are not required to create the complete BinarySearchTree ADT. In this assignment, the required methods are very specific to the *HospitalPatient* class, which are the elements of this data structure.

The following fully-developed Java files are provided to support the code you write:

- *HospitalPatient.java.* This defines the hospitalized patient's record. Since these are the elements of the tree structure, you need to be familiar with the content and the methods.
- *TreeNode.java.* This is a basic node for any BinaryTree. However, this version is not generic, but specific to the admitted patient records that are stored inside the AdmittedPatients tree.
- *SimpleDate.java.* Every admitted patient must have an admission date. This is as simple as can be for the Admissions Department's needs. The GregorianCalendar that is part of Java's API is too complicated for the hospital's needs.
- *ViewableTree.java.* This is provided for you so you can track the BinarySearchTree's structure, during the implementation and testing phase.

Implementation details

All the specifications documentaion can be accessed via the following link. (Make sure you are logged onto conneX when accessing the link directly.)

To get you started, we recommend the following:

1. Download the files associated with this assignment into a fresh working directory (csc115/assn4 for example).
2. Read through the TreeNode, SimpleDate, and HospitalPatient source code. Keep a copy of the specifications close by when developing the code that you are writing.
3. Create the minimal class for AdmittedPatients, copying and pasting the required specified methods.
4. Compile each of the files.
5. Refer to the textbook's description of the various parts of the BinarySearchTree ADT when you start to fill in the methods.
6. Start with the *admit* method.
7. Implement the *discharge* method last.

- There are 3 cases for the position of the node to delete. These are described in the specification details. Start with the simplest case.

You are welcome to use any number of extra *helper* methods to make your code easier to understand and handle. You may not create helper classes.

Recursion vs. non-recursion

The *getPatient* and *printLocations* methods are fairly straight-forward with implementations that use recursion. The *admit* and *discharge* method can be implemented recursively or non-recursively. The non-recursive management requires the maintenance of a *parent* reference for each of the *TreeNode* objects. The recursive management does not need to use this reference.

The textbook descriptions of the tree structure and method implementations use recursion. However, some textbooks also describe non-recursive insertion and deletion methods for BinarySearchTrees. You are welcome to use whatever makes sense.

Internal testing:

The AdmittedPatients class will require testing to make sure that each of the methods work. Specifically the admit and discharge methods will cause changes in the tree structure that are very specific, and it is easier to see the results than to print them all out.

The following internal testing example is not complete, but is meant to get you started. The constructor calls are spread over 2 lines to prevent word-wrapping on this document.

```
AdmittedPatients admitted = new AdmittedPatients();
HospitalPatient p1 = new HospitalPatient(
 new SimpleDate(2018,2,27),"Duck","Donald",'C',123);
HospitalPatient p2 = new HospitalPatient(
 new SimpleDate(2018,1,15),"Mouse","Minnie",'B',307);
HospitalPatient p3 = new HospitalPatient(
 new SimpleDate(2018,3,1),"Dog","Goofie",'A',422);
HospitalPatient p4 = new HospitalPatient(
 new SimpleDate(2017,12,25),"Newman","Alfred",'X',111);
admitted.admit(p1);
admitted.admit(p4);
admitted.admit(p3);
admitted.admit(p2);
admitted.printLocations();
 // these two lines cause the tree to be viewed in a little
 // resizable window.
ViewableTree dbt = new ViewableTree(admitted);
dbt.showFrame();
```

The single file to submit:

1. *AdmittedPatients.java*

Grading

- Submissions are expected to follow all of the requirements for the submitted files in the specification documents.
- Evidence of internal testing must be present: you may comment out any testing the main methods if you wish.
- The coding conventions, specifed in the codingConventions.pdf document on conneX, must be followed.
- Please post questions to the forum set up for this assignment; any clarifications posted there are part of the overall specifications. If there are any major changes, this will be announced and you will receive an email.