**CMPE223 - HOMEWORK 3**

UMUT UYGUR ID: 13474078970 Section: 01

**Problem Statement and Code Design**

The task was to create a database for a hospital using a binary search tree (BST) to store patient details such as:

* Dates
* Patient names
* Doctor names
* Care-team details (stored in an internal BST)

The assignment required implementing database management methods including **add**, **remove**, and **show** (in ascending order). The solution involved two interconnected BST implementations:

1. **First BST**:
   * Key: Patient Name
   * Value: Patient Object

A diagram of a company

Description automatically generated

1. **Second BST**:
   * Key: Date
   * Value: Patient Name

A diagram of a company

Description automatically generated

**Classes and Descriptions**

* **BSTREE.java**: Binary search tree implementation based on the textbook with minor adjustments.
* **Date.java**: Class for storing and comparing dates.
* **HospitalDB.java**: Core part of the assignment containing database management methods and storing the BST.
* **Main.java**: Tester class for testing the database and comparing results to expected outputs.
* **MedicalStaff.java**: Class for storing care-team information.
* **Patient.java**: Class for storing patient information and managing comparisons.
* **Queue.java**: Simple queue implementation.
* **Node.java**: Simple node implementation.
* **Stack.java**: Simple stack implementation.

**Implementation and Functionality**

**1. BSTREE Class**

This class implemented the BST from the textbook with an added count variable to simplify the addMember method.

**2. Date Class**

* Converts a given date into its object representation.
* Implements the Comparable interface.
* **compareTo Method Logic**:
  + Compare years: Return 1 if greater, -1 if smaller.
  + If years are equal, compare months.
  + If months are equal, compare days.
  + Return 0 if dates are identical.

**3. Patient Class**

* Stores patient details.
* Contains an inner BST to store each patient’s care team.
* Implements Comparable for comparison by names.

**4. MedicalStaff Class**

* Stores details about the medical staff such as name and role.

**5. HospitalDB Class**

* Maintains two interconnected BSTs:
  1. First BST (Key: Patient Name, Value: Patient Object)
  2. Second BST (Key: Date, Value: Patient Name)

**Methods in HospitalDB:**

* **addPatient(String patientName, String doctorName, int visitDay, int visitMonth, int visitYear)**:
  + Checks if there is space at the end; removes if necessary.
  + Adds a new patient if not already present; overrides if existing.
* **removePatient(String patientName)**:
  + Removes the patient if they exist; otherwise, prints an error.
* **addMember(String patientName, String memberName, String memberRole)**:
  + Adds a member to the care team if the patient exists; otherwise, prints an error.
* **removeMember(String patientName, String memberName)**:
  + Removes a care team member if both patient and member exist; otherwise, prints an error.
* **showAllPatients()**:
  + Iterates through the second BST (sorted by year) to display all patients.
* **showPatient(String patientName)**:
  + Displays patient details (name, date, doctor, care team).
* **showDoctorPatients(String doctorName)**:
  + Displays all patients under a specific doctor. Prints an error if none found.
* **showPatients(int visitYear)**:
  + Iterates through the second BST to display all patients for a given year.

**6. Main Class**

* Contains test cases to verify the correctness of the program.

**Testing**

The code was tested using inputs from the assignment PDF. Testing outcomes were as follows:

* All functions worked as intended.
* Minor discrepancies in date and name formatting compared to expected results.
* Issues such as overwriting patients with matching years were resolved by using the date as a key.

**Bugs Identified and Fixed**

1. Overwriting patients with matching years: Resolved by using a date object as the key.
2. Show method displaying unordered results: Implemented a second BST for sorting and used a stack to reverse the order when needed.

**Final Assessments**

* **Challenges**:
  1. Understanding the assignment requirements.
  2. Determining appropriate keys for the BSTs.
* **Learning Outcomes**:
  1. Extensive practice with BST implementation.
  2. Real-life problem-solving experience with database structures.
  3. Improved understanding of data storage and retrieval using trees.

**Conclusion**

This assignment provided valuable insights into database management using binary search trees. The practical nature of the problem enriched our understanding of data structures and their applications.