**Mini – Project**

**Medical Store Management System**

**Contributors:**

2303 Aparna Agrawal

         2312 Hardiki Sonchhatra

         2317 Khushbu Bora

         2326 Sudha Chintake

**Details:**

         This application has been built to reduce manual work of medical stores. Objective is to organize medical store details like medicines stocks, location, transactions

         This project has been designed for both admin and customer interface.

**Data Structure and Algorithm Used**

**Medical Inventory:** Avl Tree

·      Created on the basis of names of medicines

·      Time complexity of searching is reduced to O(logn)

·      No extra static memory space required

**Customer Database:** HashTable

·      Implemented using java hashmap

·      Maps key (customer id) to our value (corresponding customer object)

·      Insertion time complexity reduced to O (1)

**Classes**

1. **Medicine** (Implements serializable)
   1. Parameterized Constructor: Set values of instance variables.
   2. void accept\_diseases (): Accept disease ArrayList.
   3. void display (): Display medicine details.
2. **Medical\_avl**
   1. Constructor: Set root to null.
   2. private int height (Medicine m): Returns height of m.
   3. private int balance\_factor (Medicine m): Calculates balance factor.
   4. private Medicine rightRotate (Medicine m): Perform right rotation around m.

private Medicine leftRotate (Medicine m): Performs left rotation around m.

* 1. Medicine insert (Medicine root, Medicine m): Add m to the root

void create (): Create and accepts medicine details and add it to root using insert method.

* 1. void inorder\_display (Medicine m): Display medicine recursively in inorder fashion.

void inorder\_display (): Calls inorder\_display (Medicine m) by passing root.

* 1. void check\_stock (): Display required details of out-of-stock medicines.
  2. void checkExpiry (): Display required details of out-of-stock medicines.
  3. Medicine search\_name (Medicine m, String n): Recursively search and returns Medicine named n.

void search\_by\_name (Medicine root): Calls search\_name by passing root and display its details.

* 1. void search\_disease (String d): Display required details of medicines which cure disease d.
  2. void search\_supplier\_name (String s): Display required details of medicines supplied by supplier s.
  3. void restock (): Restock medicines
  4. Medicine deleteNode (Medicine root, String n): Recursively search and delete medicine named n from tree.
  5. private Medicine min (Medicine m): Returns minimum node.
  6. void customer\_order (Customer c): Takes input for purchased list of Customer c and updates stock of purchased medicines.

1. **Purchased** (implements Serializable)
   1. void accept (): Accept details of medicine to be purchased.
   2. void display (): Display details of the purchased medicine.
2. **Customer** (implements Serializable)
   1. void accept\_customer (): Accepts the details of customer.
   2. void display (): Display customer details.
   3. void create\_customer\_bill (): Adds up all the prices of purchased medicine and display bill.
   4. void update\_quantity\_of\_medicine (Medical\_avl m): Updates the quantity of medicine in purchased list and updates stock of purchased medicines.
   5. void remove\_medicine\_from\_bill (Medical\_avl m): Removes medicine from purchased list and updates stock of removed medicines.
3. **CustomerHash**
   1. static void insert (Customer c): Put customer c into static hashmap customerDatabase (consider c’s id as key).
   2. static void displayTotalSales (): Traverse through customerDatabase and display id, name and total bill of each customer and adds bill to display total revenue generated.
4. **MedicalMain**
   1. static void writeToFile (Medical\_avl m): Writes root of m to the file.

static void writeToFile (): Writes all customer object present in customer database in the file.

* 1. static void ReadFromFile (Medical\_avl m): Reads object from the file to root of m.

static void ReadFromFile (): Reads each customer object and puts it into a static hashmap customerDatabase.

* 1. public static void main (String args[]): Main method from where program execution starts.

**Future Scope**

* Customer Transaction History could be stored.
* Method to display day to day transaction of medical store could be added.
* This application can be implemented for any other store or multiple stores by creating multiple instances of tree.

**References**

* DSAII unit 2 and unit 6 pdf
* <https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/#:~:text=HashMap%20is%20similar%20to%20the,util>.
* <https://www.geeksforgeeks.org/avl-tree-set-1-insertion/>