1. Introduction

1.1 Overview of the application

Our selected application will be the Clinic Wong Fei Hung, which is a clinic service system. The purpose of this system is for the clinic to serve and manage patients more efficiently so it can save time and make it easier for staff to handle patients. The Clinic Service System is classified into 5 modules, the Consultation Management Module, Billing Module, Medicine Management Module, In-Patient Management Module, and the Staff Management Module.

1.2 Assignment Scope

Medicine Module
The Medicine module will be referring to the inventory module for the clinic, where the user is able to add, remove, modify and delete the medicine details.

PrescribedMedicine Module
As for the PrescribedMedicine module, it will act as a connection between the Consultation module and Medicine module, by providing the required function to obtain data from the Medicine module and returning data to the Consultation module that required the data. For example, when the staff want to add medicine into the prescribed medicine list, the module will be getting the medicine information and add it into the prescribed medicine list. Once the prescribed medicine has been confirmed, it will be updating the data from the medicine data list and the medicine’s history will be incremented by 1 which contains the output date and quantity.

2. Abstract Data Type (ADT) Specification

ADT Sorted List
A sorted list will be a linear collection of entries of a type T that have been sorted according to specific elements. The entry will be added into the list, and find the suitable index to store the entry in the list. There will be 8 operations in total and the index number will be started from 0.

boolean add(T newEntry)
Description : Add the newEntry and sort it at the same time into the sorted list.
Post-condition : The newEntry is added into the sorted list at sorted position.
Return : Return true if successfully added, false otherwise

T search(T anEntry)
Description : Search and return the first anEntry in the sorted list
Post-condition : The sorted list remains unchanged
Return : Return the first object that is matched with anEntry in the sorted list, null otherwise

boolean remove(T anEntry)
Description : Remove the first occurrence of the anEntry from the sorted list
Post-condition : The newEntry is removed from the sorted list, if the anEntry is not found in the sorted list, the sorted list will remain unchanged.
Return : Return true if successfully removed, false otherwise

boolean contains(T anEntry)
Description : Check if the anEntry is available in the sorted list
Post-condition : The sorted list remains unchanged.
Return : Return true if the anEntry is founded in the sorted list, false otherwise

void clear()
Description : Remove all the elements in the sorted list.
Post-condition : The sorted list is empty without any object.

int getNumberOfEntries()
Description : Get the number of entries available in the sorted list.
Post-condition : The sorted list remains unchanged.
Return : The current total entries that are available in the sorted list.

boolean isEmpty()
Description : Check if the sorted list is empty.
Post-condition : The sorted list remains unchanged.
Return : Return true if the sorted list is empty, false otherwise

Iterator<T> getIterator()
Description : Get the list iterator of the sorted list
Post-condition : The sorted list remains unchanged
Returns : Return list iterator of the sorted list

3. ADT Implementation

3.1 Overview of ADT

The collection that has been chosen to implement is the Sorted ArrayList. This is because it is suitable in the case where the list is required to be sorted. For example in this case, there will be two types of medicine and both of them have different id prefix and therefore, it will be needed to sort the medicine list when a new medicine is added into the list for the user to browse the list accordingly. First of all, this collection will be having the add function, where adding the object into the list while being sorted, then will have remove function, which according to the index and object which will be used according to the situation given, and lastly the search function that will search the similar object within the list and return the result that we needed.

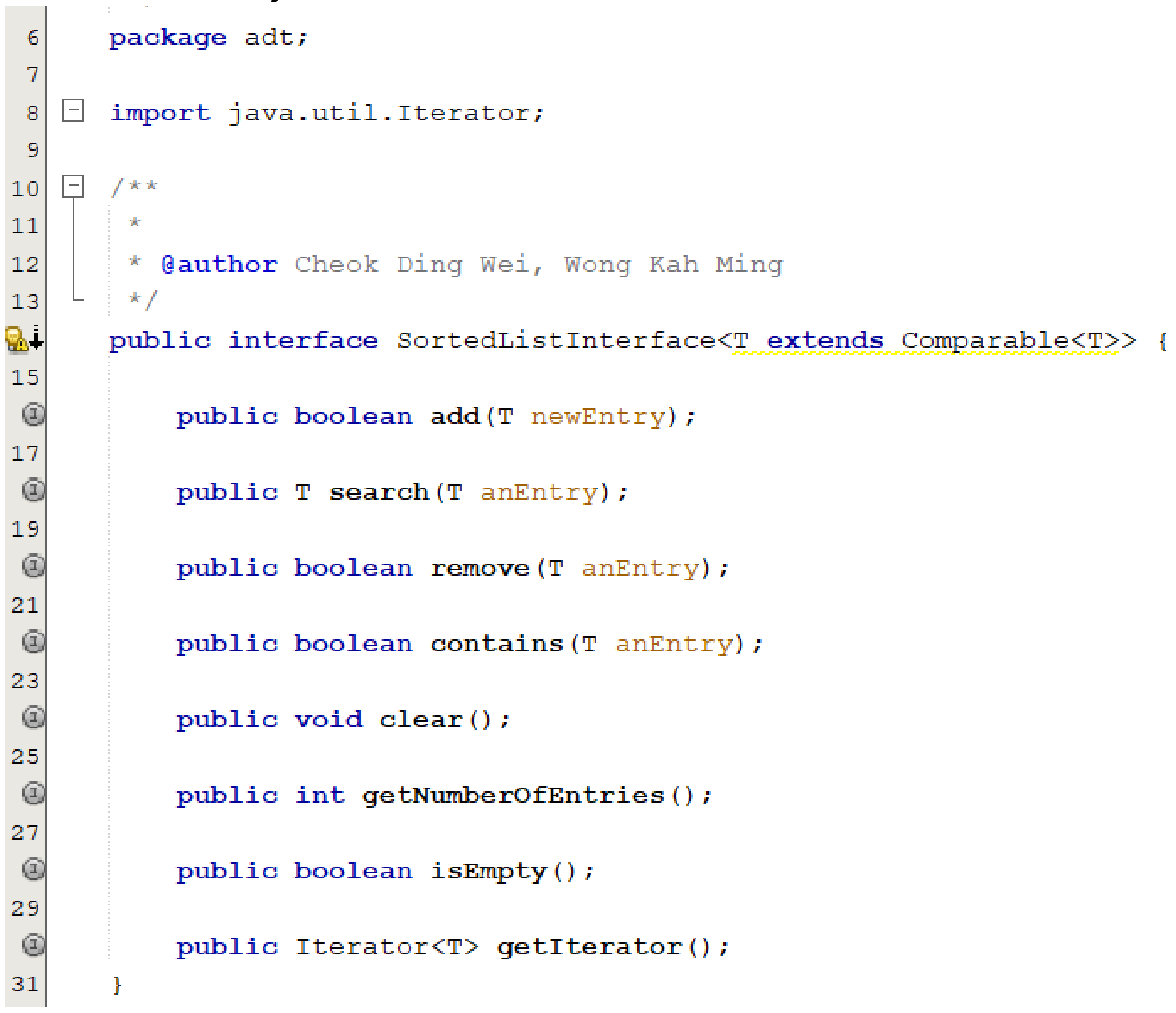
To summarize for the available operations that been used in the client program will be,

1. add() function to add the medicine object and sort it according to the medicine id
2. search() function to search the medicine object that have the same id within the sorted array list
3. binarySearch() function to search the medicine object and return both object and index
4. remove() function to remove the medicine object that have the same id within the sorted array list
5. getEntry() function to return the entry from the sorted array list with the given index
6. getNumberOfEntries() to return the number of objects available in the sorted array list.

In addition, a Map class will be used to store and return multiple values that are needed. Which in this case, it will be used to store the entry and its index in the sorted list so it can be used for multiple usage. For example, through the Map<T> binarySearch(T searchEntry) function, I can obtain 2 information through one operation which is the object that matched with the searchEntry as well as its index number in the sorted array list.

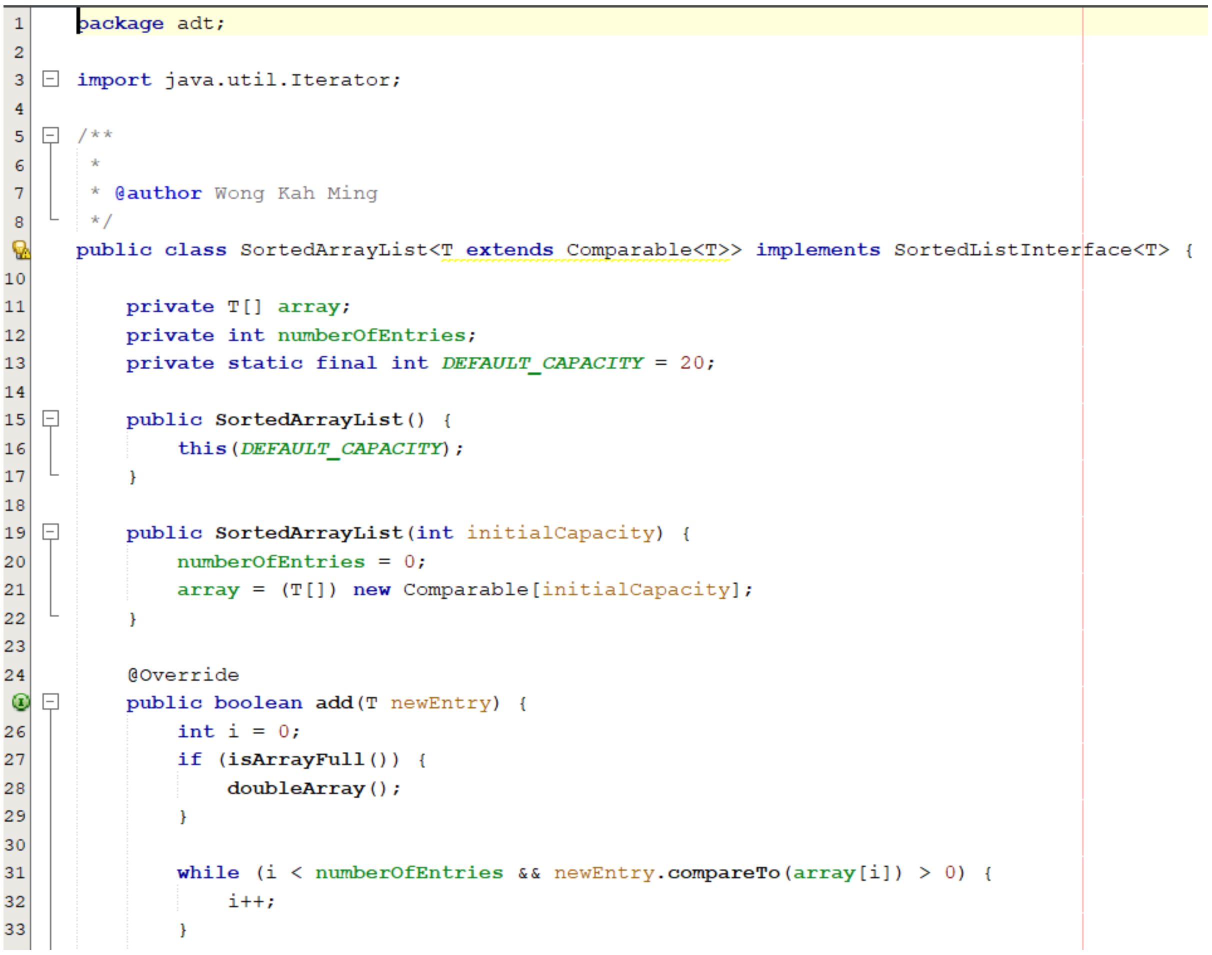
3.2 ADT Implementation

SortedListInterface.java

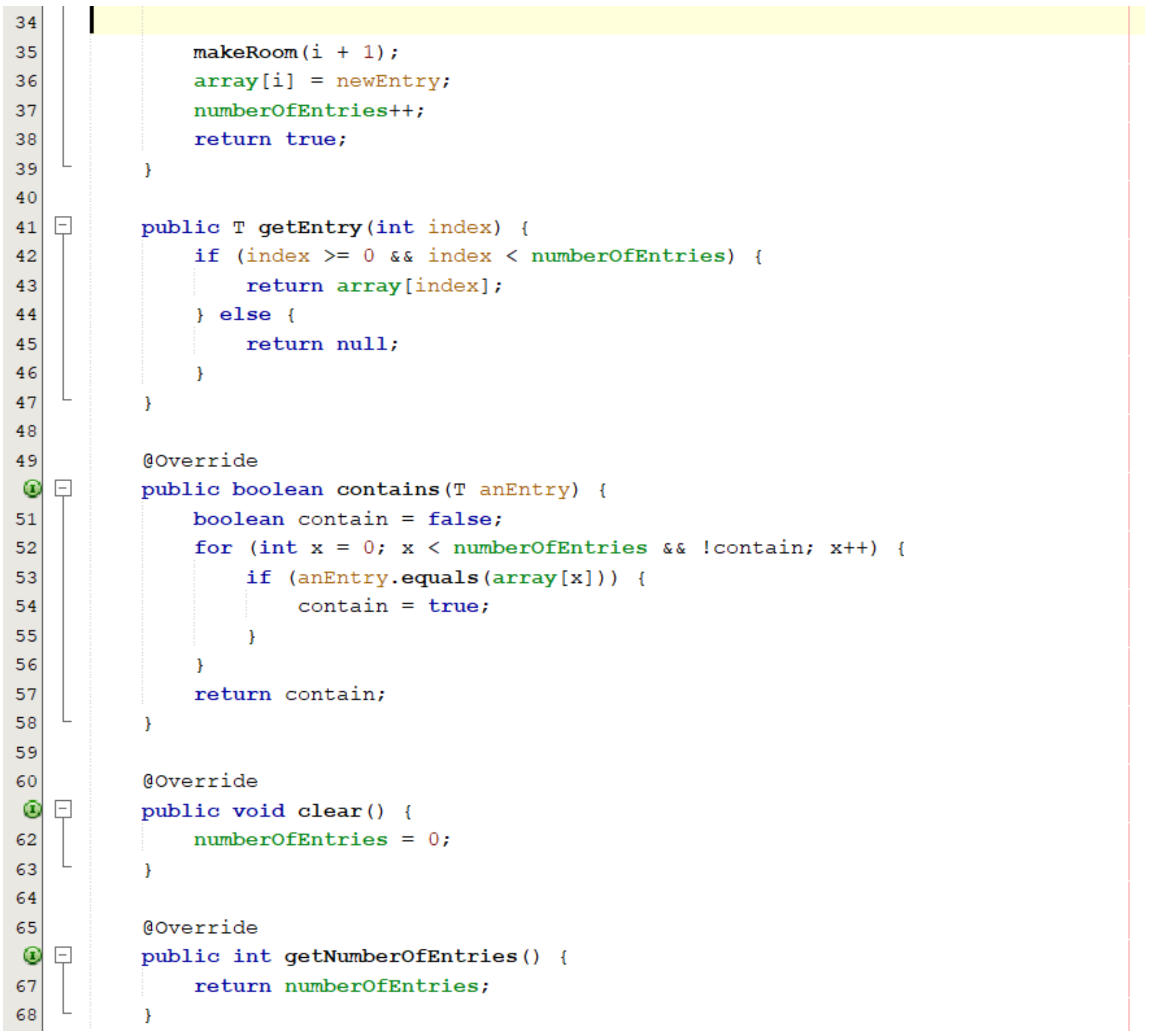


```java
package adt;

import java.util.Iterator;

/**
 *
 * @author Cheok Ding Wei, Wong Kah Ming
 */
public interface SortedListInterface<T extends Comparable<T>> {

    public boolean add(T newEntry);

    public T search(T anEntry);

    public boolean remove(T anEntry);

    public boolean contains(T anEntry);

    public void clear();

    public int getNumberOfEntries();

    public boolean isEmpty();

    public Iterator<T> getIterator();
}
```

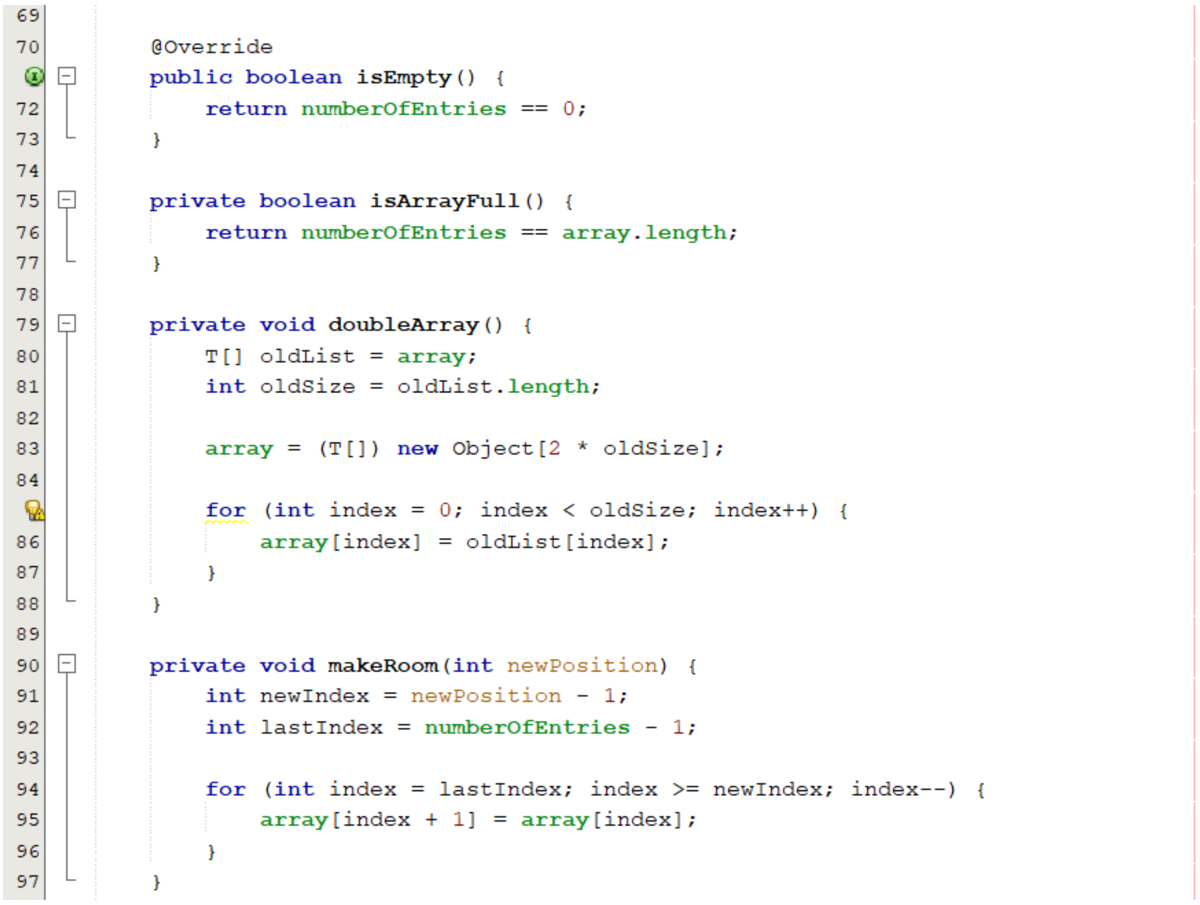
SortedArrayList.java



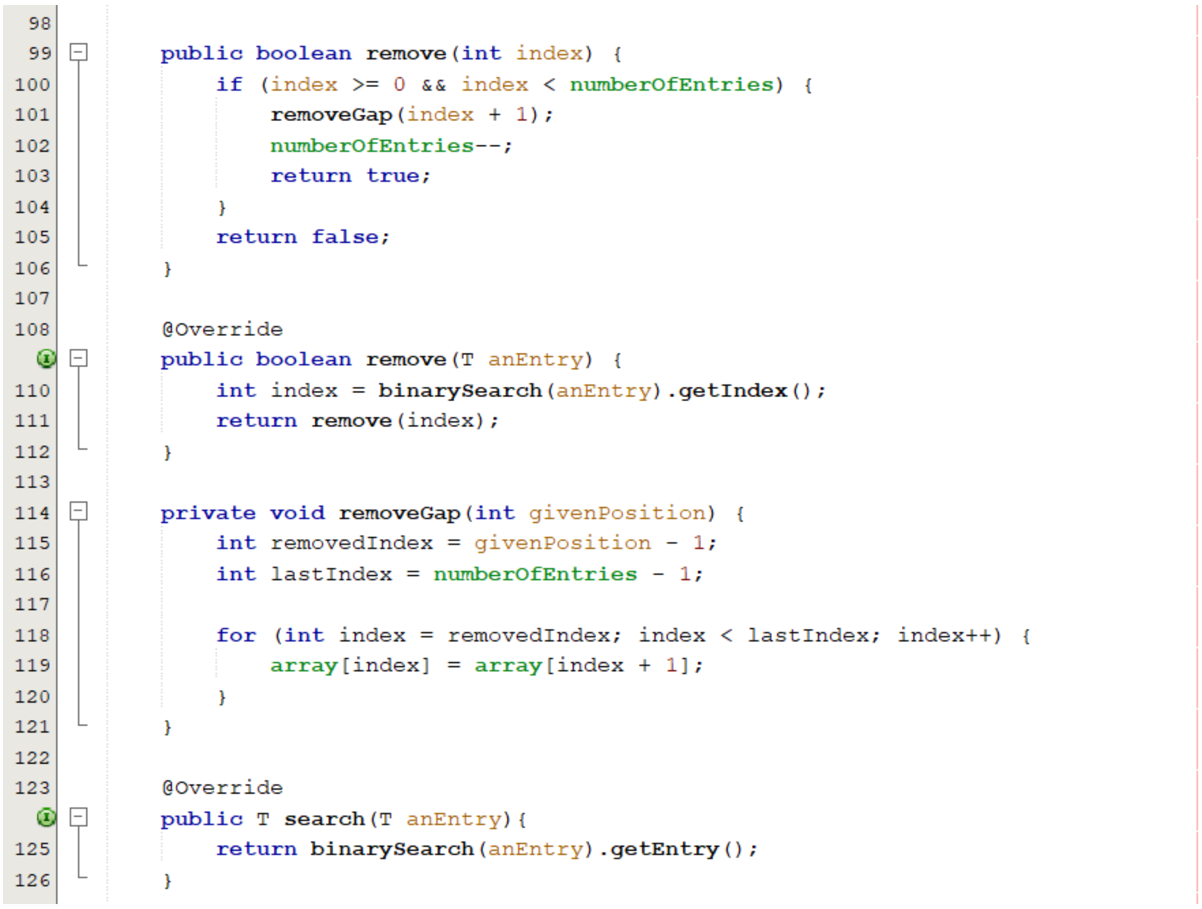
```java
package adt;

import java.util.Iterator;

/**
 *
 * @author Wong Kah Ming
 */
public class SortedArrayList<T extends Comparable<T>> implements SortedListInterface<T> {

    private T[] array;
    private int numberOfEntries;
    private static final int DEFAULT_CAPACITY = 20;

    public SortedArrayList() {
        this(DEFAULT_CAPACITY);
    }

    public SortedArrayList(int initialCapacity) {
        numberOfEntries = 0;
        array = (T[]) new Comparable[initialCapacity];
    }

    @Override
    public boolean add(T newEntry) {
        int i = 0;
        if (isArrayFull()) {
            doubleArray();
        }

        while (i < numberOfEntries && newEntry.compareTo(array[i]) > 0) {
            i++;
        }
```



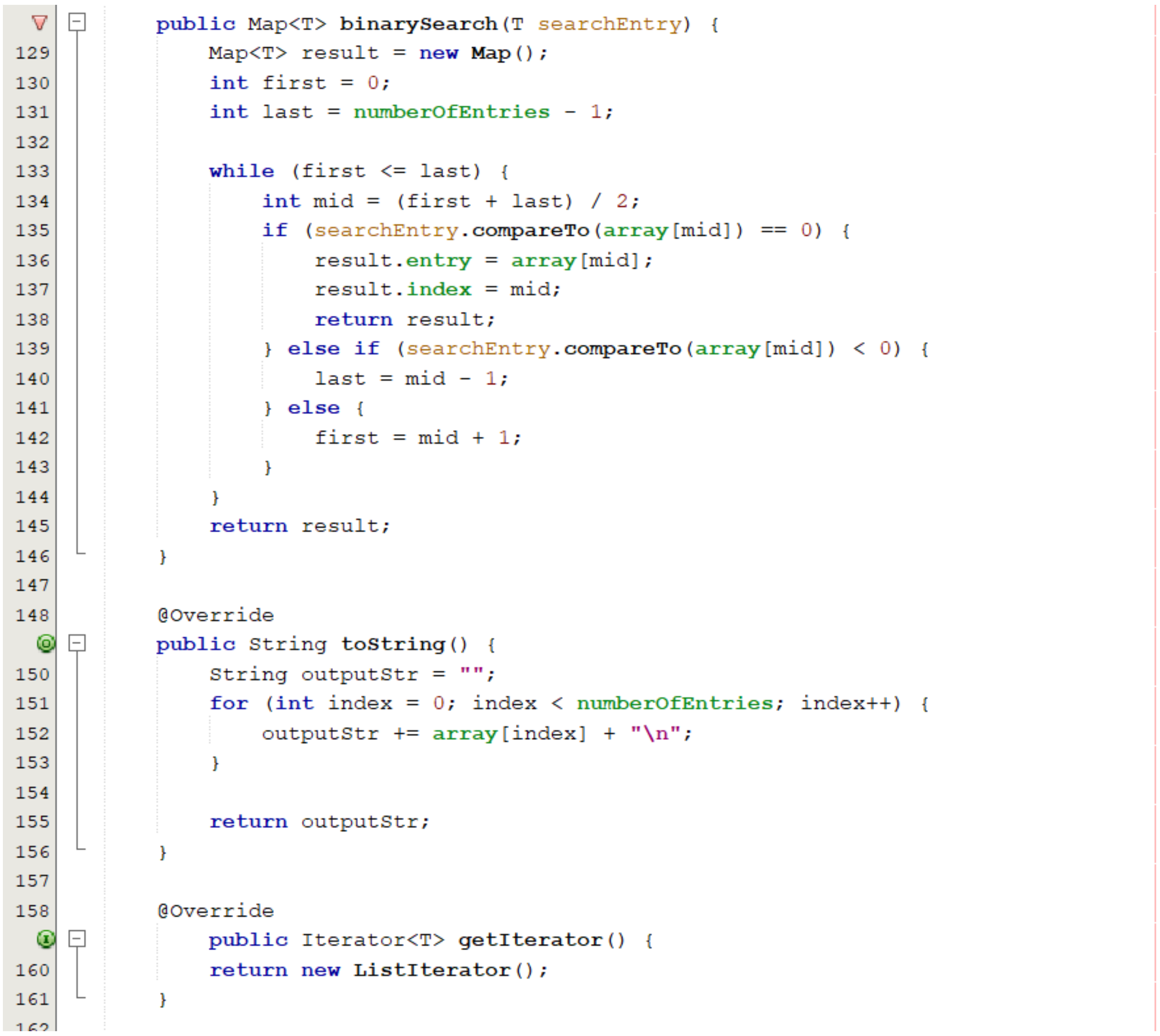
```java

        makeRoom(i + 1);
        array[i] = newEntry;
        numberOfEntries++;
        return true;
    }

    public T getEntry(int index) {
        if (index >= 0 && index < numberOfEntries) {
            return array[index];
        } else {
            return null;
        }
    }

    @Override
    public boolean contains(T anEntry) {
        boolean contain = false;
        for (int x = 0; x < numberOfEntries && !contain; x++) {
            if (anEntry.equals(array[x])) {
                contain = true;
            }
        }
        return contain;
    }

    @Override
    public void clear() {
        numberOfEntries = 0;
    }

    @Override
    public int getNumberOfEntries() {
        return numberOfEntries;
    }
```



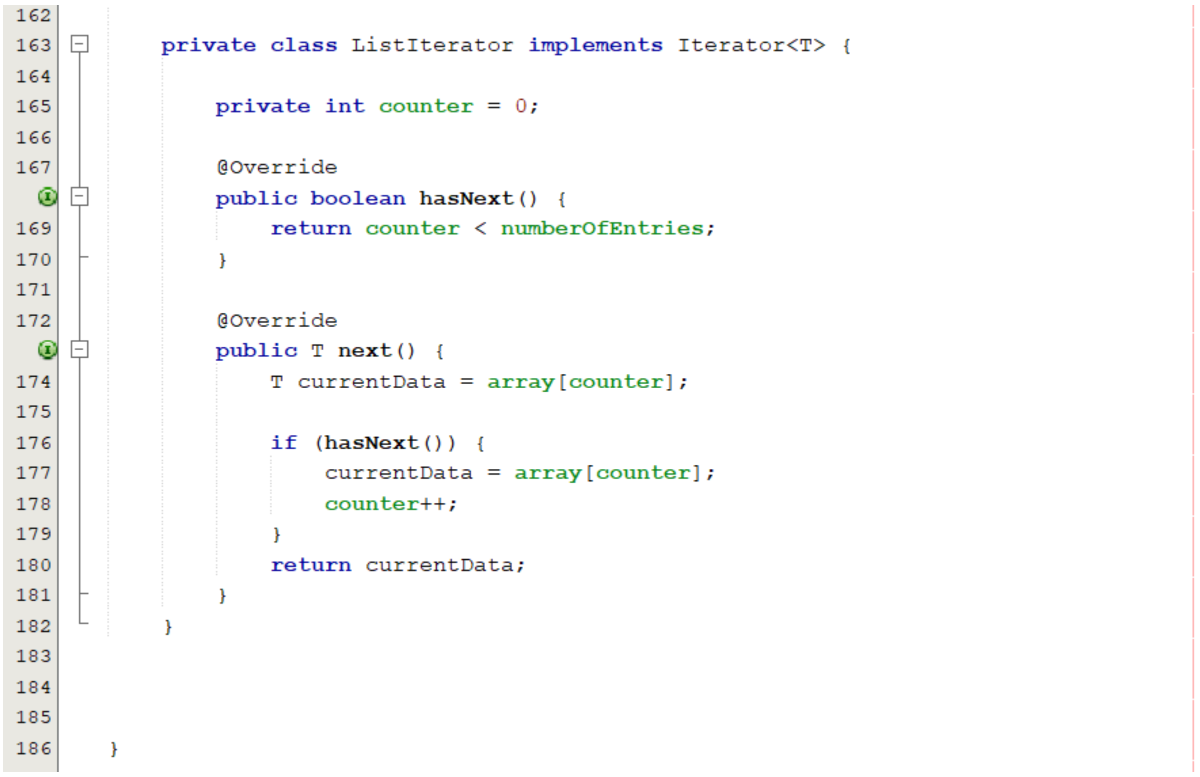
```java

    @Override
    public boolean isEmpty() {
        return numberOfEntries == 0;
    }

    private boolean isArrayFull() {
        return numberOfEntries == array.length;
    }

    private void doubleArray() {
        T[] oldList = array;
        int oldSize = oldList.length;

        array = (T[]) new Object[2 * oldSize];

        for (int index = 0; index < oldSize; index++) {
            array[index] = oldList[index];
        }
    }

    private void makeRoom(int newPosition) {
        int newIndex = newPosition - 1;
        int lastIndex = numberOfEntries - 1;

        for (int index = lastIndex; index >= newIndex; index--) {
            array[index + 1] = array[index];
        }
    }
```



```java

    public boolean remove(int index) {
        if (index >= 0 && index < numberOfEntries) {
            removeGap(index + 1);
            numberOfEntries--;
            return true;
        }
        return false;
    }

    @Override
    public boolean remove(T anEntry) {
        int index = binarySearch(anEntry).getIndex();
        return remove(index);
    }

    private void removeGap(int givenPosition) {
        int removedIndex = givenPosition - 1;
        int lastIndex = numberOfEntries - 1;

        for (int index = removedIndex; index < lastIndex; index++) {
            array[index] = array[index + 1];
        }
    }

    @Override
    public T search(T anEntry){
        return binarySearch(anEntry).getEntry();
    }
```

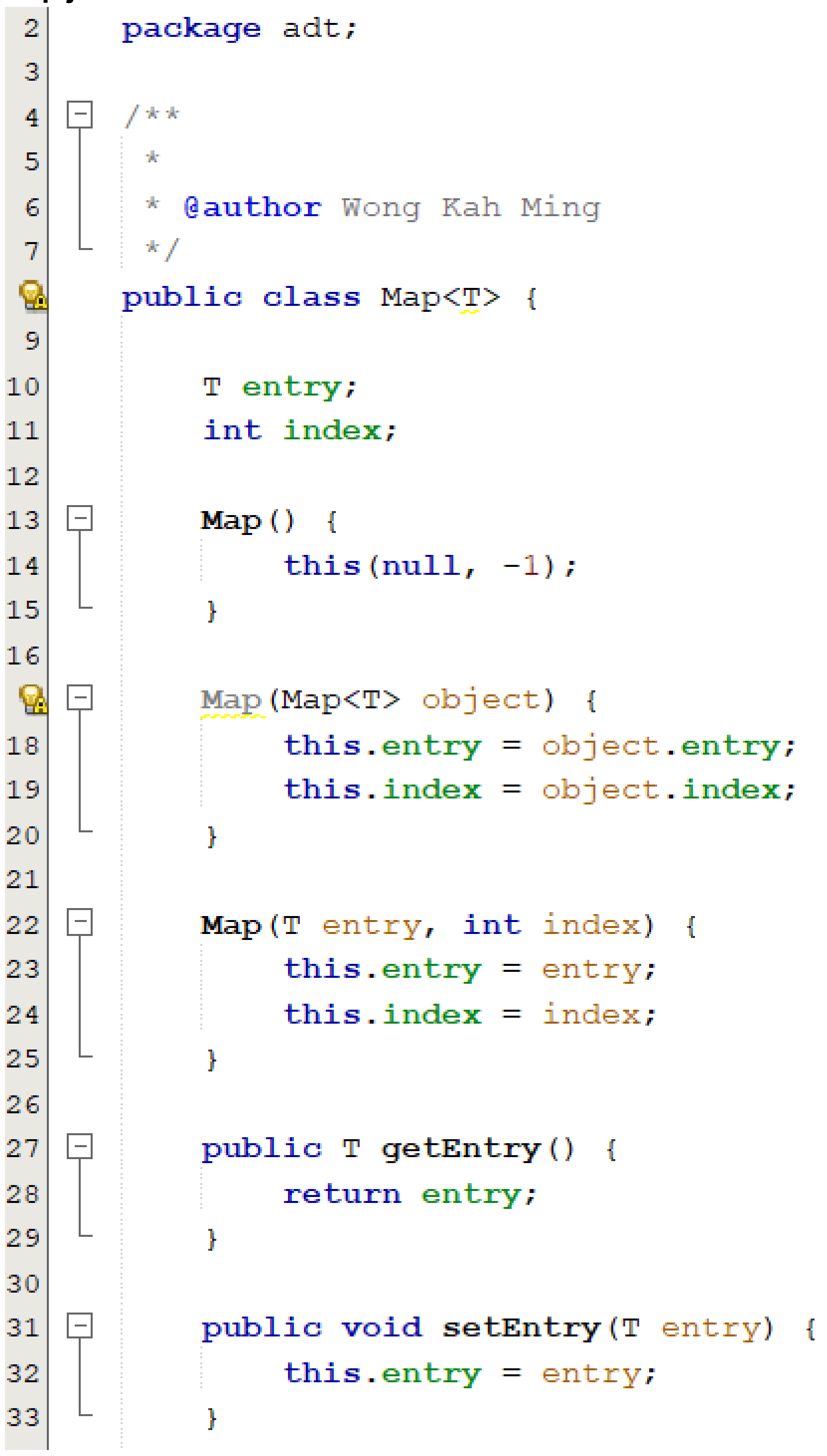


```
    public Map<T> binarySearch(T searchEntry) {
        Map<T> result = new Map();
        int first = 0;
        int last = numberOfEntries - 1;

        while (first <= last) {
            int mid = (first + last) / 2;
            if (searchEntry.compareTo(array[mid]) == 0) {
                result.entry = array[mid];
                result.index = mid;
                return result;
            } else if (searchEntry.compareTo(array[mid]) < 0) {
                last = mid - 1;
            } else {
                first = mid + 1;
            }
        }
        return result;
    }

    @Override
    public String toString() {
        String outputStr = "";
        for (int index = 0; index < numberOfEntries; index++) {
            outputStr += array[index] + "\n";
        }

        return outputStr;
    }

    @Override
        public Iterator<T> getIterator() {
        return new ListIterator();
    }
```

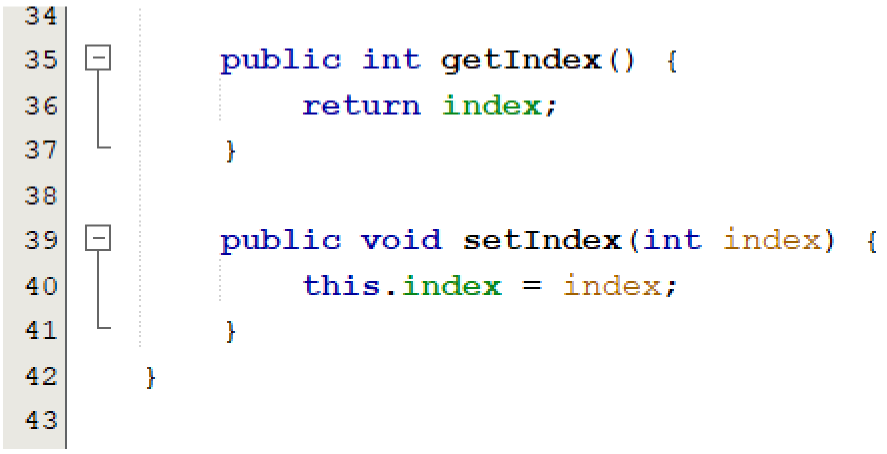


```

    private class ListIterator implements Iterator<T> {

        private int counter = 0;

        @Override
        public boolean hasNext() {
            return counter < numberOfEntries;
        }

        @Override
        public T next() {
            T currentData = array[counter];

            if (hasNext()) {
                currentData = array[counter];
                counter++;
            }
            return currentData;
        }
    }



}
```

Map.java



```java
package adt;

/**
 *
 * @author Wong Kah Ming
 */
public class Map<T> {

    T entry;
    int index;

    Map() {
        this(null, -1);
    }

    Map(Map<T> object) {
        this.entry = object.entry;
        this.index = object.index;
    }

    Map(T entry, int index) {
        this.entry = entry;
        this.index = index;
    }

    public T getEntry() {
        return entry;
    }

    public void setEntry(T entry) {
        this.entry = entry;
    }
```

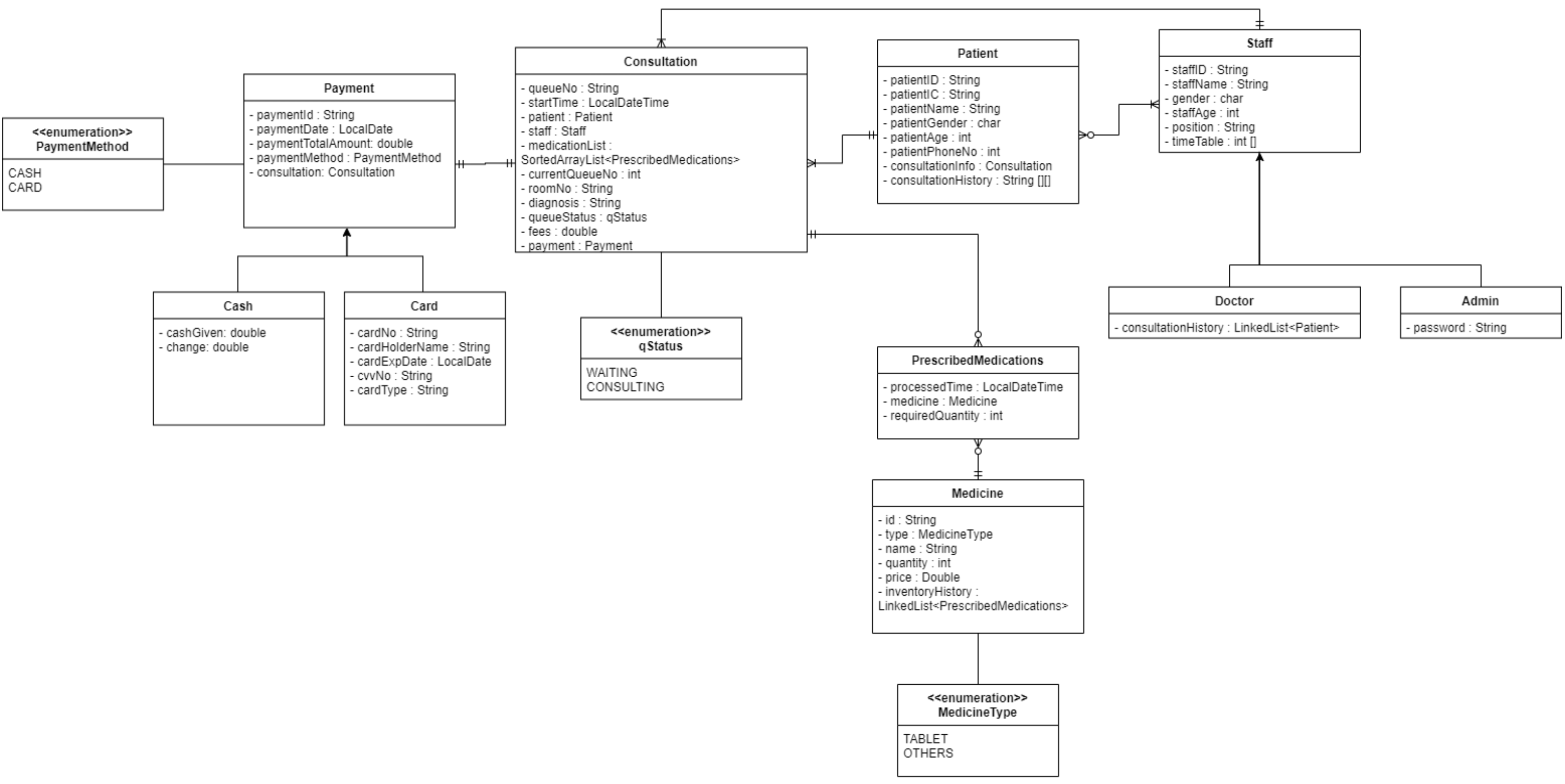


```java

    public int getIndex() {
        return index;
    }

    public void setIndex(int index) {
        this.index = index;
    }
}

```

4. Entity Classes

4.1 Entity Class Diagram

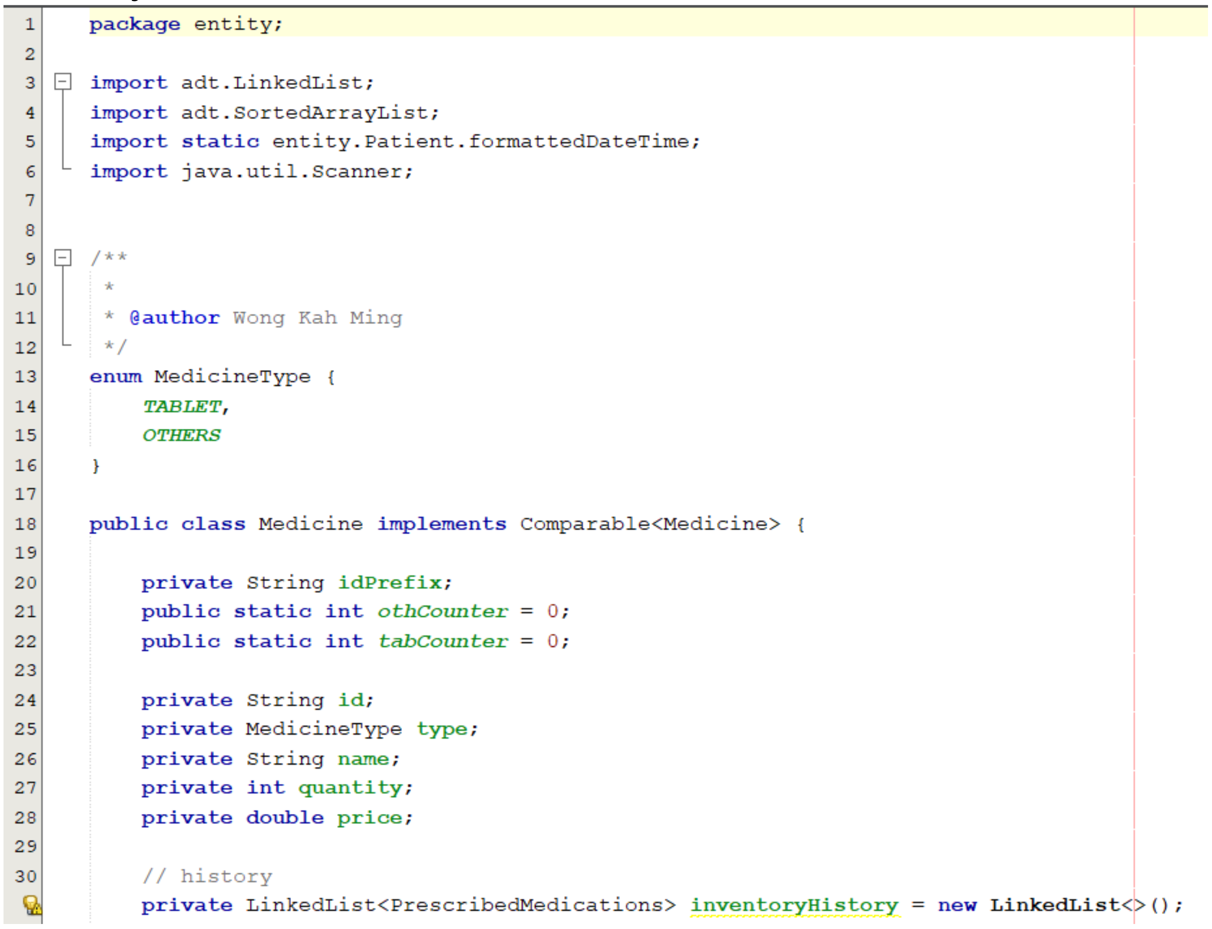
Diagram below shows the entity class diagram of the Clinic Wong Fei Hung system.



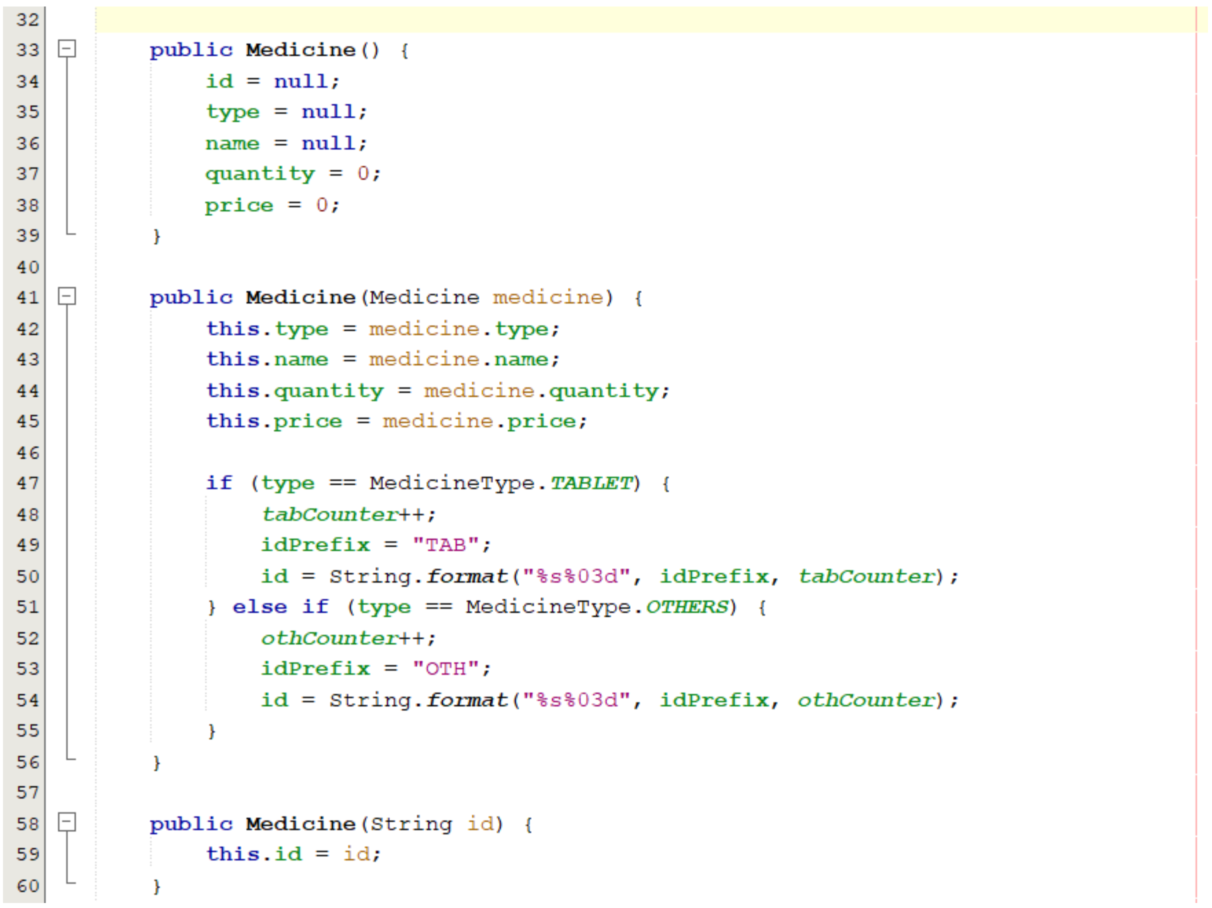
4.2 Entity Class Implementation

Diagrams below will be showing the source code for both Medicine and PrescribedMedication modules.

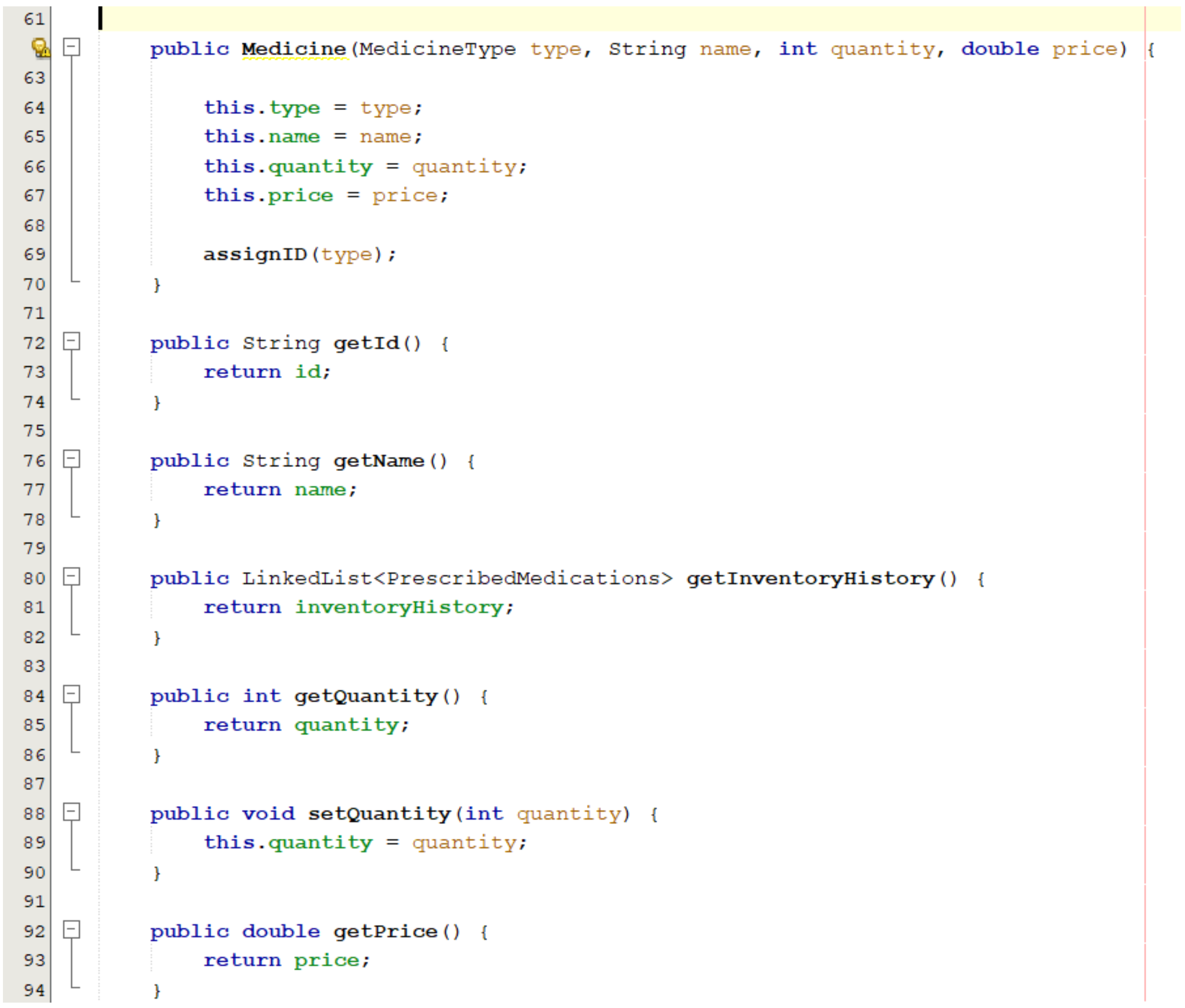
Medicine.java



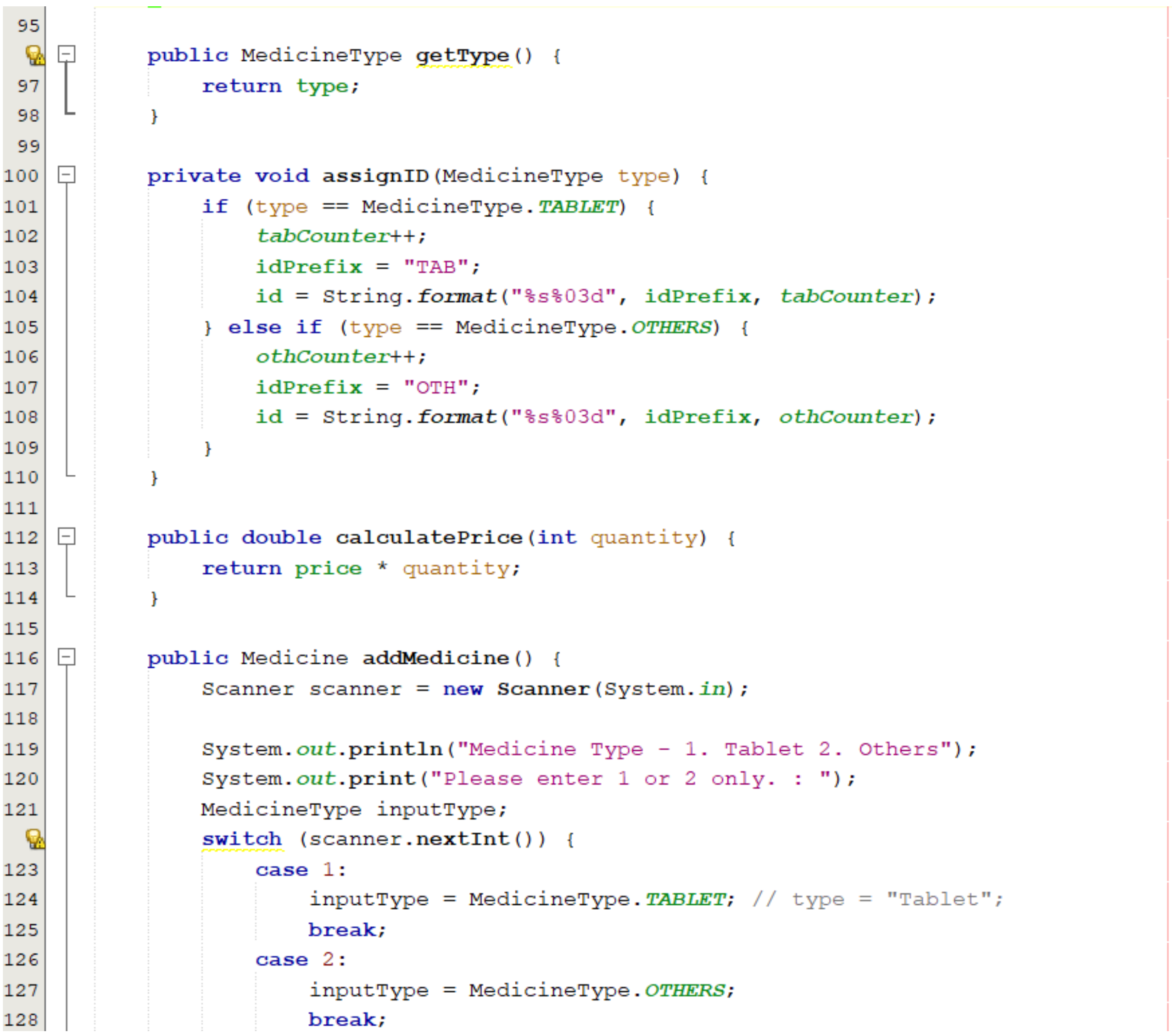
```java
package entity;

import adt.LinkedList;
import adt.SortedArrayList;
import static entity.Patient.formattedDateTime;
import java.util.Scanner;


/**
 *
 * @author Wong Kah Ming
 */
enum MedicineType {
    TABLET,
    OTHERS
}

public class Medicine implements Comparable<Medicine> {

    private String idPrefix;
    public static int othCounter = 0;
    public static int tabCounter = 0;

    private String id;
    private MedicineType type;
    private String name;
    private int quantity;
    private double price;

    // history
    private LinkedList<PrescribedMedications> inventoryHistory = new LinkedList<>();
```



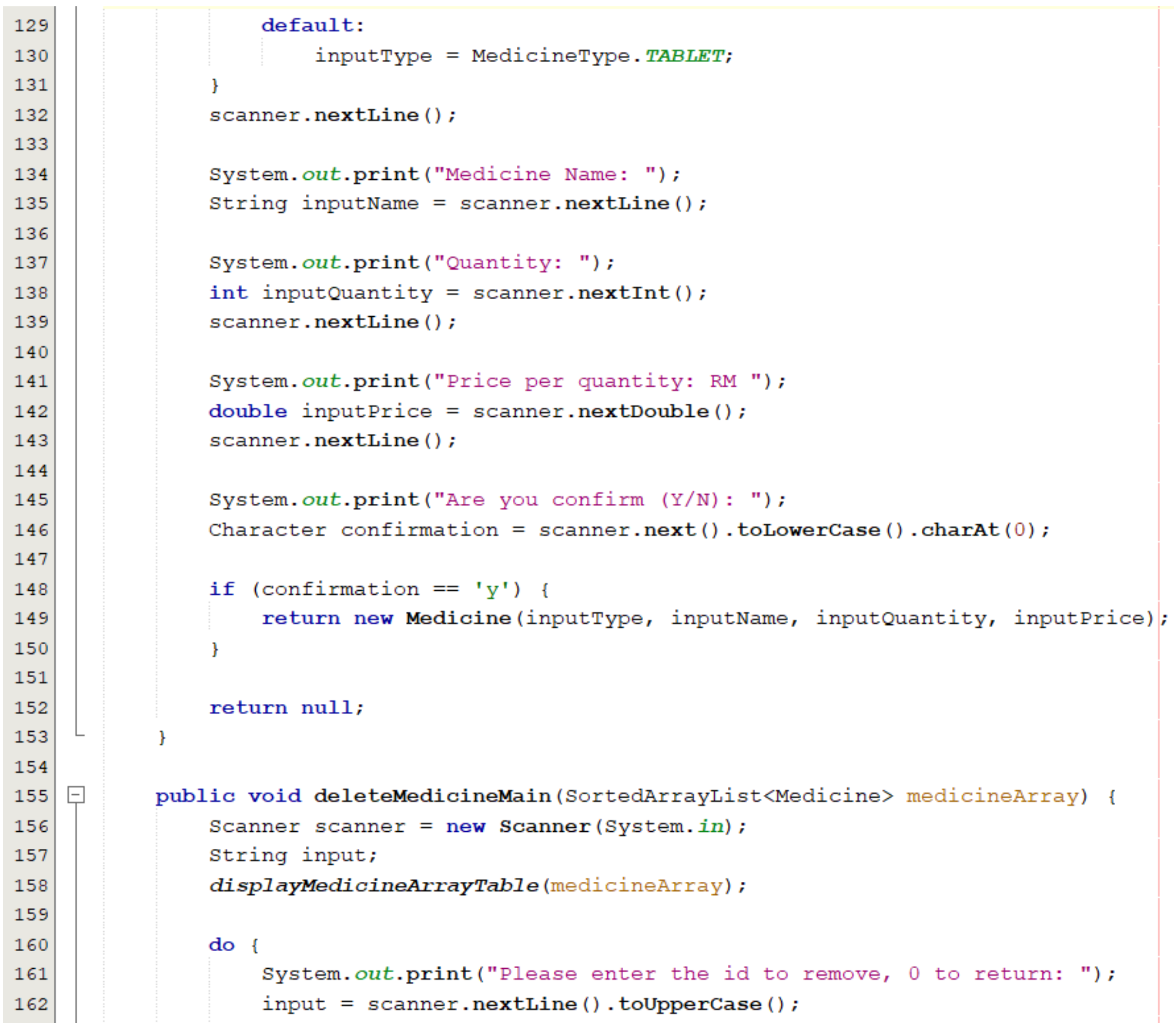
```java

    public Medicine() {
        id = null;
        type = null;
        name = null;
        quantity = 0;
        price = 0;
    }

    public Medicine(Medicine medicine) {
        this.type = medicine.type;
        this.name = medicine.name;
        this.quantity = medicine.quantity;
        this.price = medicine.price;

        if (type == MedicineType.TABLET) {
            tabCounter++;
            idPrefix = "TAB";
            id = String.format("%s%03d", idPrefix, tabCounter);
        } else if (type == MedicineType.OTHERS) {
            othCounter++;
            idPrefix = "OTH";
            id = String.format("%s%03d", idPrefix, othCounter);
        }
    }

    public Medicine(String id) {
        this.id = id;
    }
```



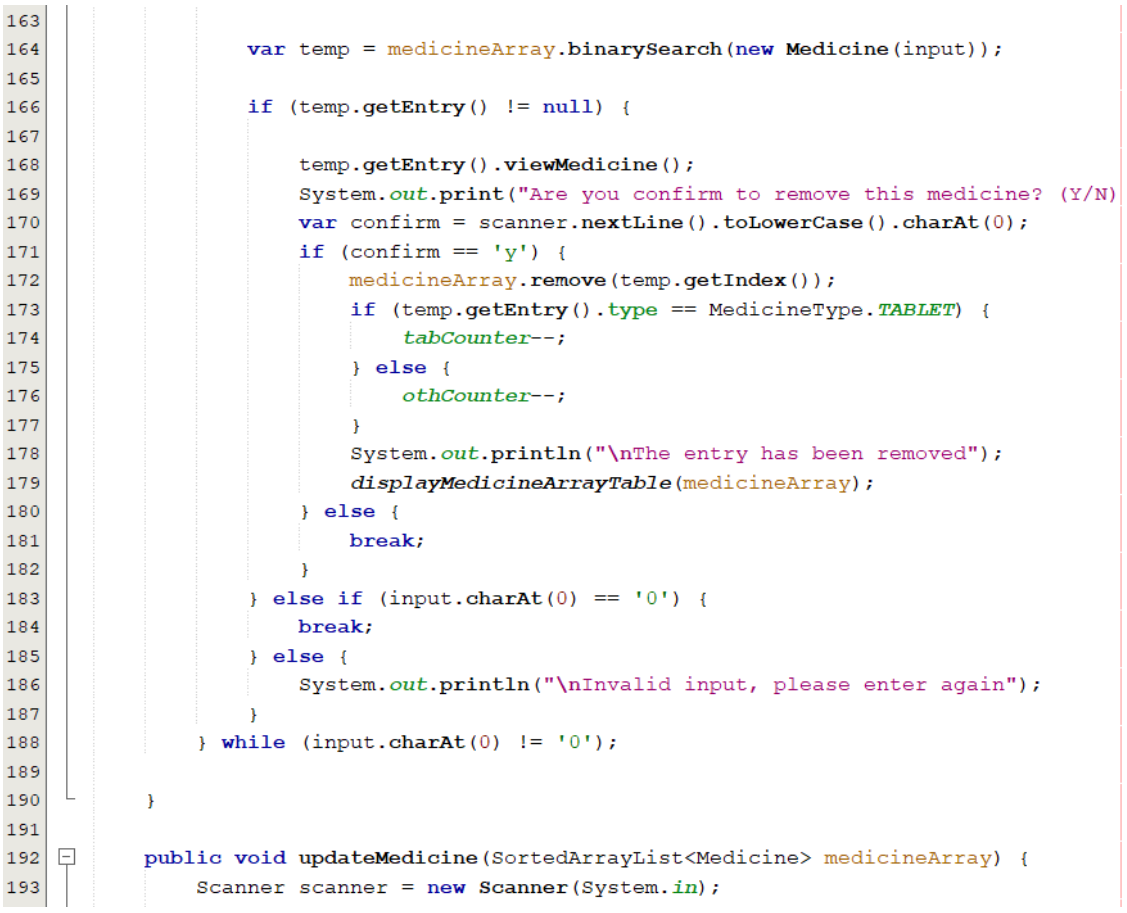
```

    public Medicine(MedicineType type, String name, int quantity, double price) {

        this.type = type;
        this.name = name;
        this.quantity = quantity;
        this.price = price;

        assignID(type);
    }

    public String getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public LinkedList<PrescribedMedications> getInventoryHistory() {
        return inventoryHistory;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public double getPrice() {
        return price;
    }
```



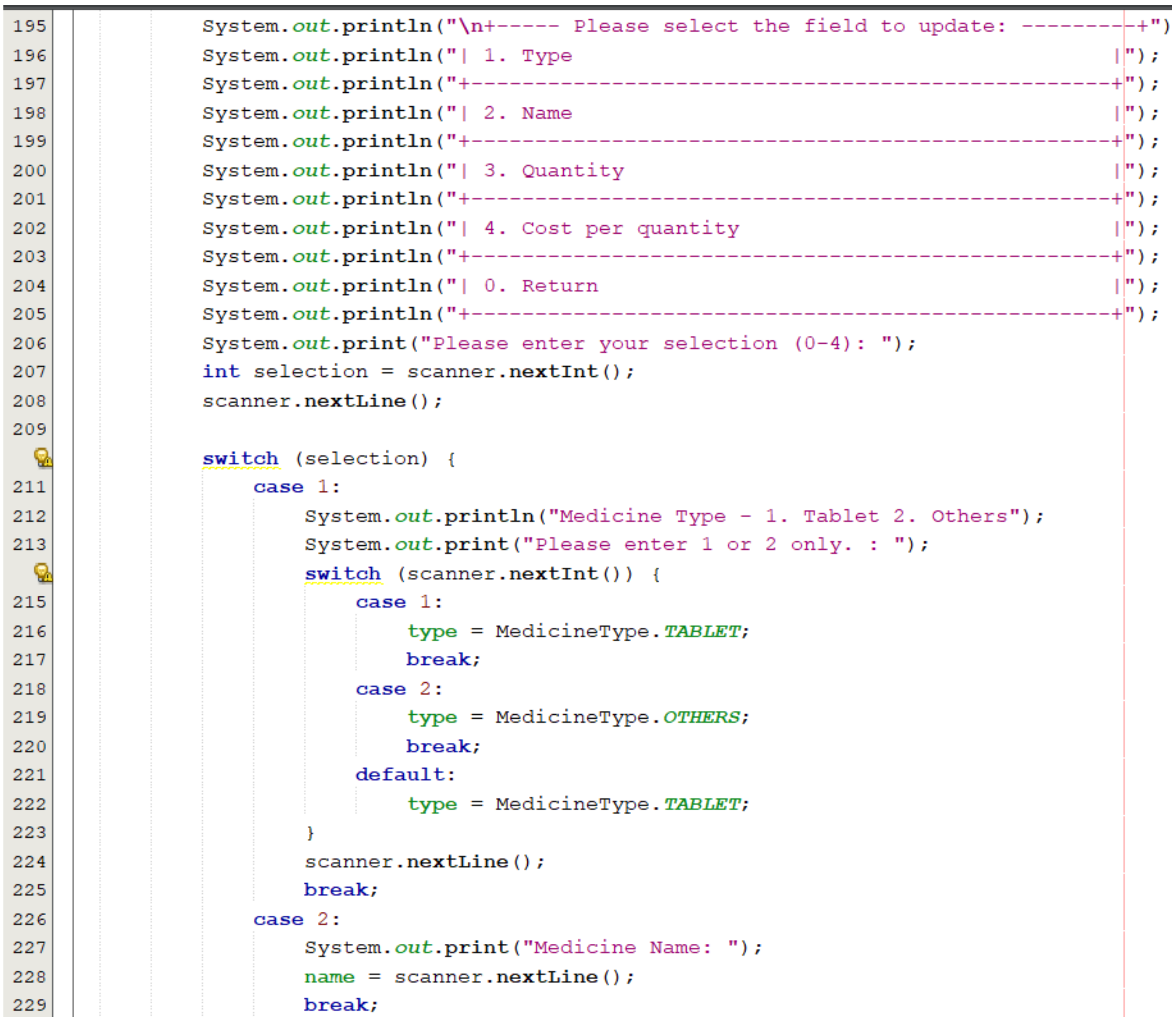
```java

    public MedicineType getType() {
        return type;
    }

    private void assignID(MedicineType type) {
        if (type == MedicineType.TABLET) {
            tabCounter++;
            idPrefix = "TAB";
            id = String.format("%s%03d", idPrefix, tabCounter);
        } else if (type == MedicineType.OTHERS) {
            othCounter++;
            idPrefix = "OTH";
            id = String.format("%s%03d", idPrefix, othCounter);
        }
    }

    public double calculatePrice(int quantity) {
        return price * quantity;
    }

    public Medicine addMedicine() {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Medicine Type - 1. Tablet 2. Others");
        System.out.print("Please enter 1 or 2 only. : ");
        MedicineType inputType;
        switch (scanner.nextInt()) {
            case 1:
                inputType = MedicineType.TABLET; // type = "Tablet";
                break;
            case 2:
                inputType = MedicineType.OTHERS;
                break;
```



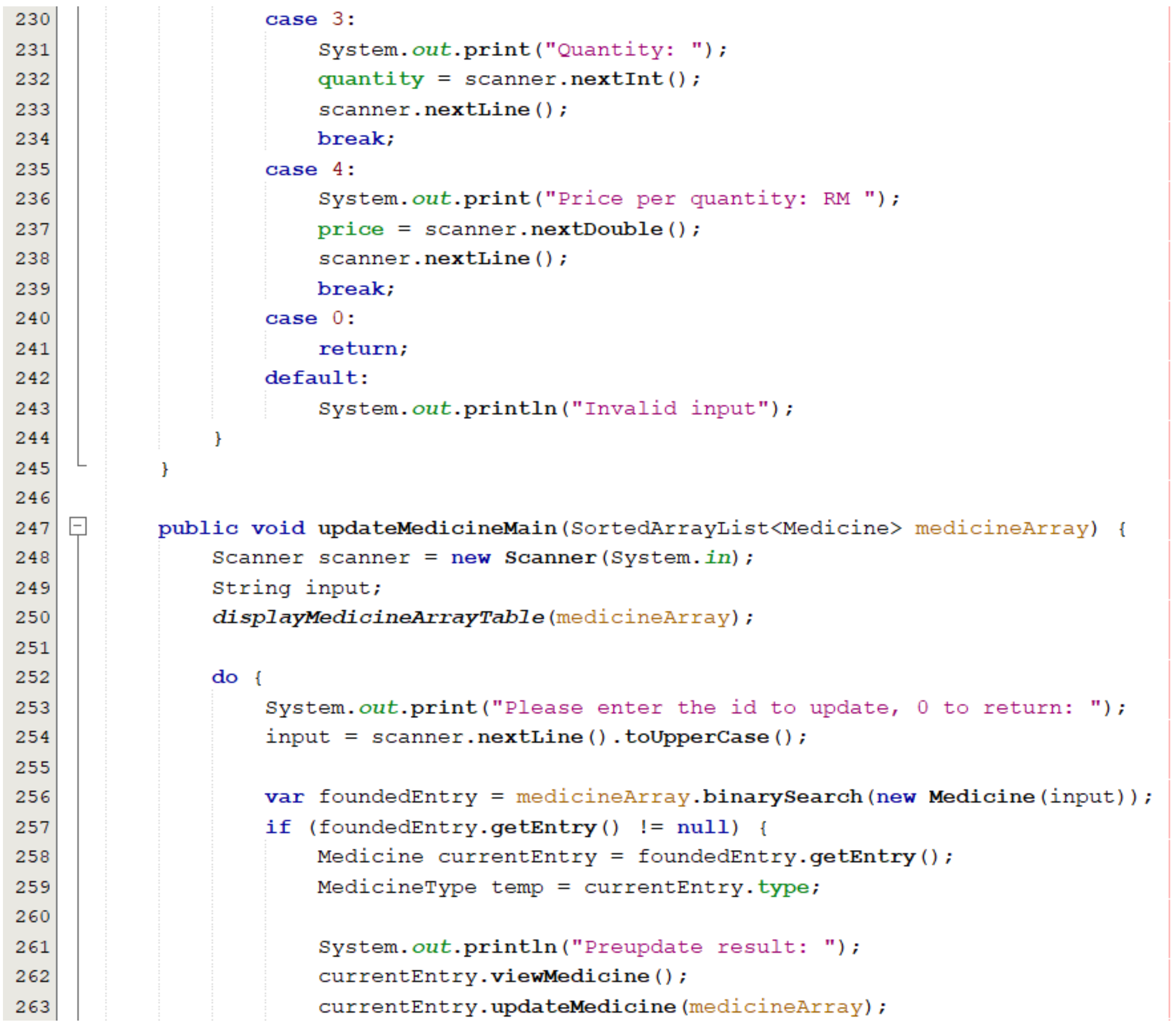
```java
            default:
                inputType = MedicineType.TABLET;
        }
        scanner.nextLine();

        System.out.print("Medicine Name: ");
        String inputName = scanner.nextLine();

        System.out.print("Quantity: ");
        int inputQuantity = scanner.nextInt();
        scanner.nextLine();

        System.out.print("Price per quantity: RM ");
        double inputPrice = scanner.nextDouble();
        scanner.nextLine();

        System.out.print("Are you confirm (Y/N): ");
        Character confirmation = scanner.next().toLowerCase().charAt(0);

        if (confirmation == 'y') {
            return new Medicine(inputType, inputName, inputQuantity, inputPrice);
        }

        return null;
    }

    public void deleteMedicineMain(SortedArrayList<Medicine> medicineArray) {
        Scanner scanner = new Scanner(System.in);
        String input;
        displayMedicineArrayTable(medicineArray);

        do {
            System.out.print("Please enter the id to remove, 0 to return: ");
            input = scanner.nextLine().toUpperCase();
```



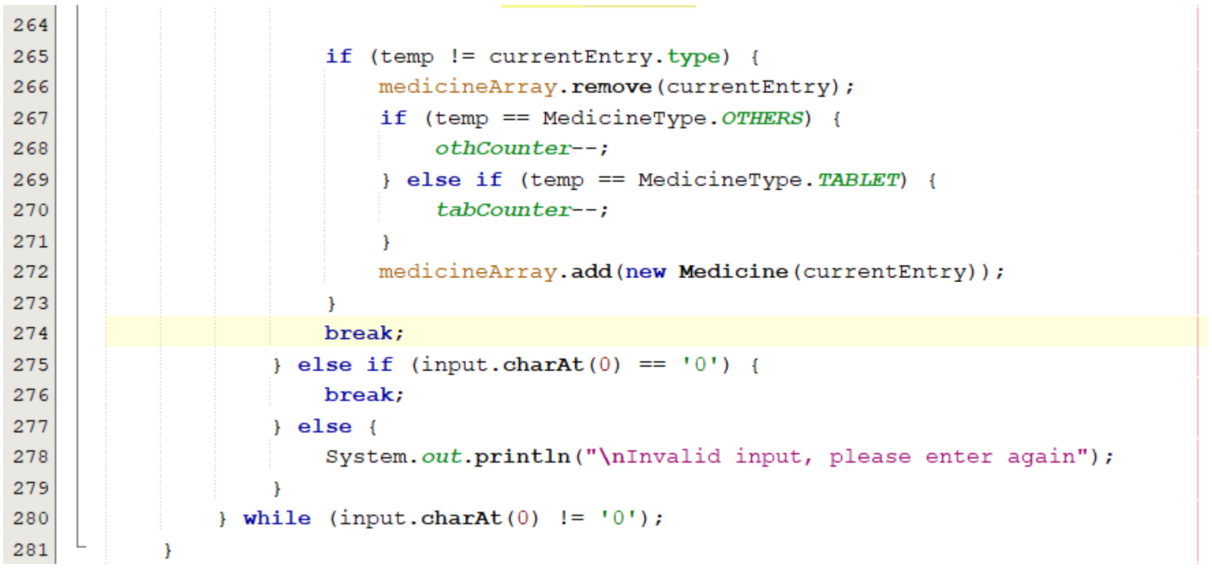
```java

            var temp = medicineArray.binarySearch(new Medicine(input));

            if (temp.getEntry() != null) {

                temp.getEntry().viewMedicine();
                System.out.print("Are you confirm to remove this medicine? (Y/N) : ");
                var confirm = scanner.nextLine().toLowerCase().charAt(0);
                if (confirm == 'y') {
                    medicineArray.remove(temp.getIndex());
                    if (temp.getEntry().type == MedicineType.TABLET) {
                        tabCounter--;
                    } else {
                        othCounter--;
                    }
                    System.out.println("\nThe entry has been removed");
                    displayMedicineArrayTable(medicineArray);
                } else {
                    break;
                }
            } else if (input.charAt(0) == '0') {
                break;
            } else {
                System.out.println("\nInvalid input, please enter again");
            }
        } while (input.charAt(0) != '0');

    }

    public void updateMedicine(SortedArrayList<Medicine> medicineArray) {
        Scanner scanner = new Scanner(System.in);
```



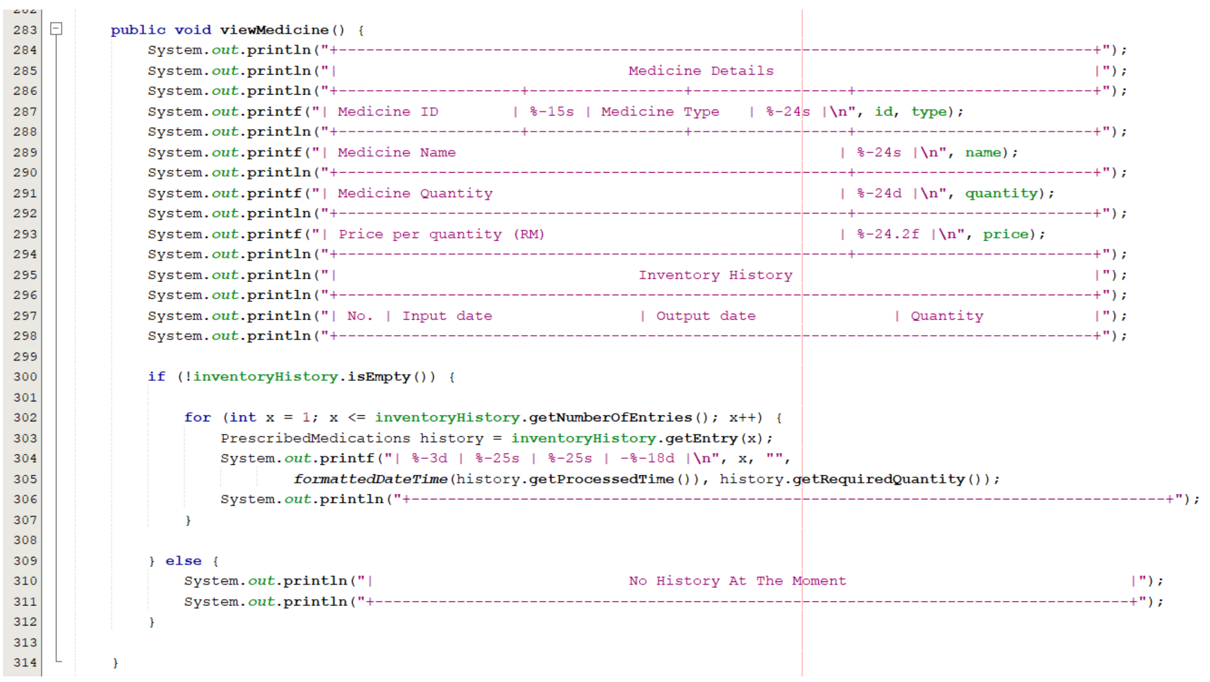
```
        System.out.println("\n+----- Please select the field to update: --------+")
        System.out.println("| 1. Type                                         |");
        System.out.println("+-------------------------------------------------+");
        System.out.println("| 2. Name                                         |");
        System.out.println("+-------------------------------------------------+");
        System.out.println("| 3. Quantity                                     |");
        System.out.println("+-------------------------------------------------+");
        System.out.println("| 4. Cost per quantity                            |");
        System.out.println("+-------------------------------------------------+");
        System.out.println("| 0. Return                                       |");
        System.out.println("+-------------------------------------------------+");
        System.out.print("Please enter your selection (0-4): ");
        int selection = scanner.nextInt();
        scanner.nextLine();

        switch (selection) {
            case 1:
                System.out.println("Medicine Type - 1. Tablet 2. Others");
                System.out.print("Please enter 1 or 2 only. : ");
                switch (scanner.nextInt()) {
                    case 1:
                        type = MedicineType.TABLET;
                        break;
                    case 2:
                        type = MedicineType.OTHERS;
                        break;
                    default:
                        type = MedicineType.TABLET;
                }
                scanner.nextLine();
                break;
            case 2:
                System.out.print("Medicine Name: ");
                name = scanner.nextLine();
                break;
```



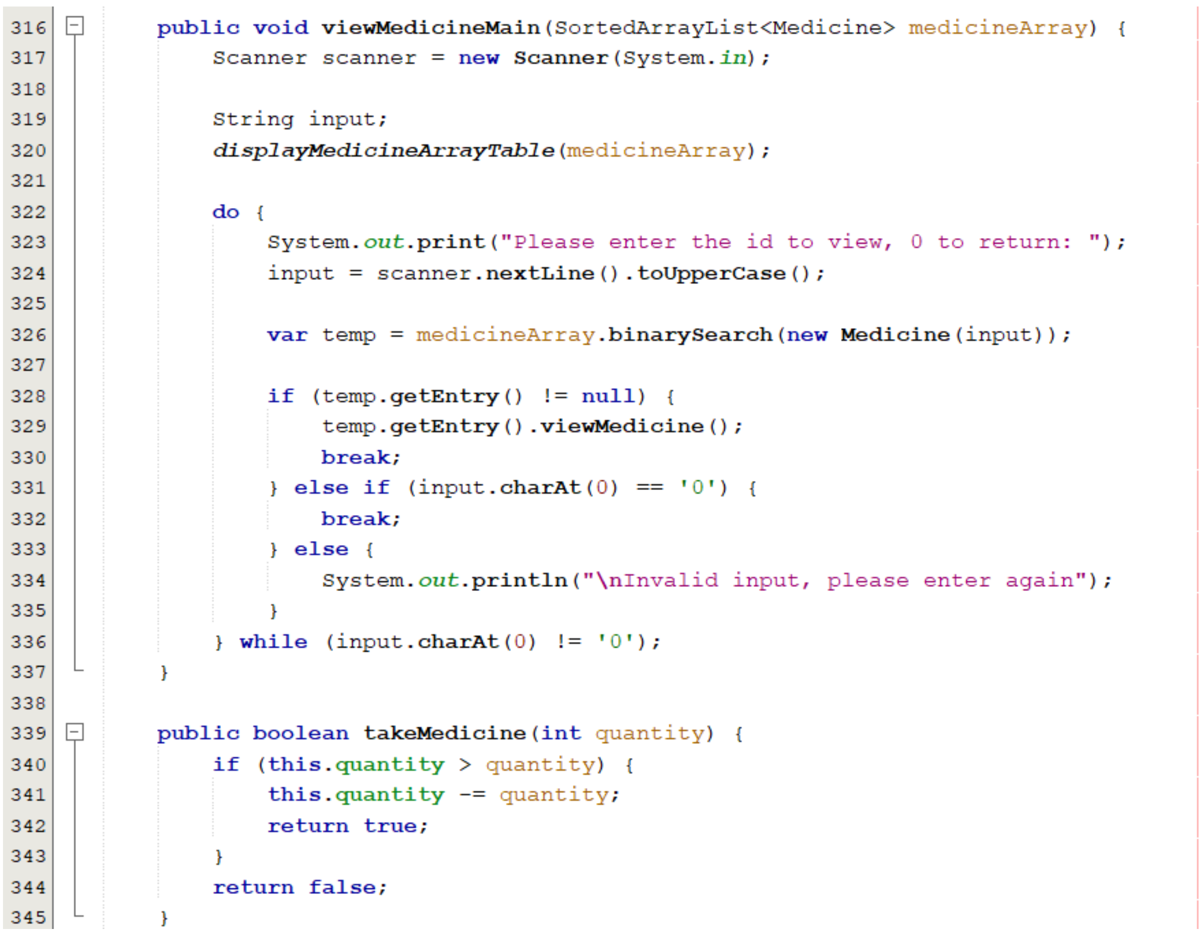
```
            case 3:
                System.out.print("Quantity: ");
                quantity = scanner.nextInt();
                scanner.nextLine();
                break;
            case 4:
                System.out.print("Price per quantity: RM ");
                price = scanner.nextDouble();
                scanner.nextLine();
                break;
            case 0:
                return;
            default:
                System.out.println("Invalid input");
        }
    }

    public void updateMedicineMain(SortedArrayList<Medicine> medicineArray) {
        Scanner scanner = new Scanner(System.in);
        String input;
        displayMedicineArrayTable(medicineArray);

        do {
            System.out.print("Please enter the id to update, 0 to return: ");
            input = scanner.nextLine().toUpperCase();

            var foundedEntry = medicineArray.binarySearch(new Medicine(input));
            if (foundedEntry.getEntry() != null) {
                Medicine currentEntry = foundedEntry.getEntry();
                MedicineType temp = currentEntry.type;

                System.out.println("Preupdate result: ");
                currentEntry.viewMedicine();
                currentEntry.updateMedicine(medicineArray);
```



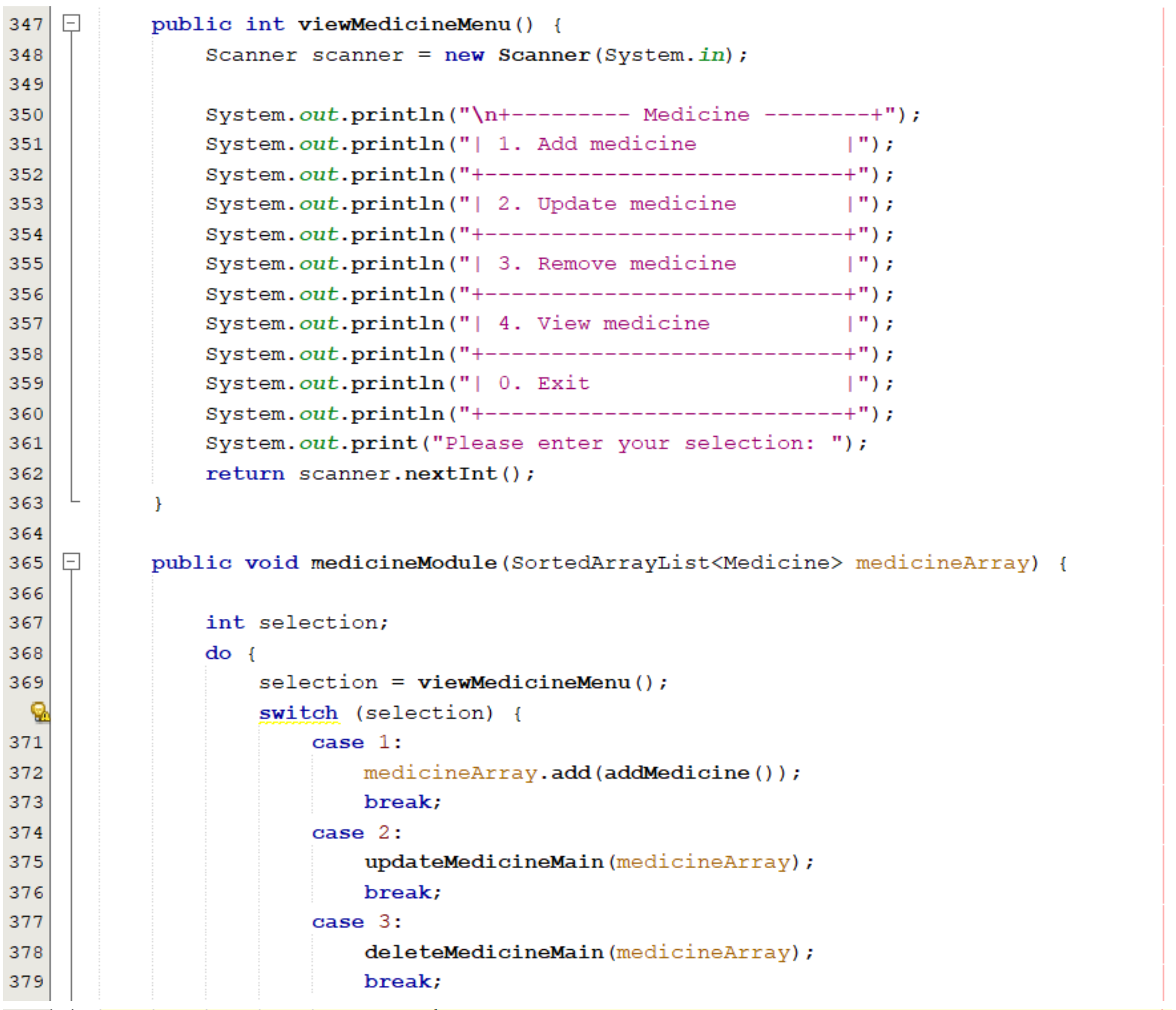
```

                if (temp != currentEntry.type) {
                    medicineArray.remove(currentEntry);
                    if (temp == MedicineType.OTHERS) {
                        othCounter--;
                    } else if (temp == MedicineType.TABLET) {
                        tabCounter--;
                    }
                    medicineArray.add(new Medicine(currentEntry));
                }
                break;
            } else if (input.charAt(0) == '0') {
                break;
            } else {
                System.out.println("\nInvalid input, please enter again");
            }
        } while (input.charAt(0) != '0');
    }
```



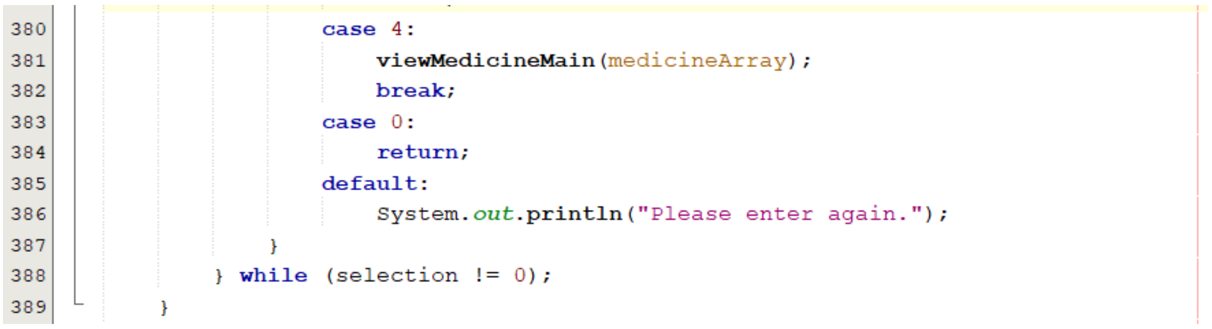
```java
    public void viewMedicine() {
        System.out.println("+--------------------------------------------------------------------------------+");
        System.out.println("|                                Medicine Details                                |");
        System.out.println("+---------------------+------------------+--------------------------------------+");
        System.out.printf("| Medicine ID         | %-15s | Medicine Type   | %-24s |\n", id, type);
        System.out.println("+---------------------+------------------+--------------------------------------+");
        System.out.printf("| Medicine Name                                        | %-24s |\n", name);
        System.out.println("+-------------------------------------------------------+--------------------------+");
        System.out.printf("| Medicine Quantity                                    | %-24d |\n", quantity);
        System.out.println("+-------------------------------------------------------+--------------------------+");
        System.out.printf("| Price per quantity (RM)                              | %-24.2f |\n", price);
        System.out.println("+-------------------------------------------------------+--------------------------+");
        System.out.println("|                               Inventory History                                |");
        System.out.println("+--------------------------------------------------------------------------------+");
        System.out.println("| No. | Input date                | Output date                | Quantity          |");
        System.out.println("+--------------------------------------------------------------------------------+");

        if (!inventoryHistory.isEmpty()) {

            for (int x = 1; x <= inventoryHistory.getNumberOfEntries(); x++) {
                PrescribedMedications history = inventoryHistory.getEntry(x);
                System.out.printf("| %-3d | %-25s | %-25s | -%-18d |\n", x, "",
                        formattedDateTime(history.getProcessedTime()), history.getRequiredQuantity());
                System.out.println("+--------------------------------------------------------------------------------+");
            }

        } else {
            System.out.println("|                             No History At The Moment                             |");
            System.out.println("+--------------------------------------------------------------------------------+");
        }

    }
```



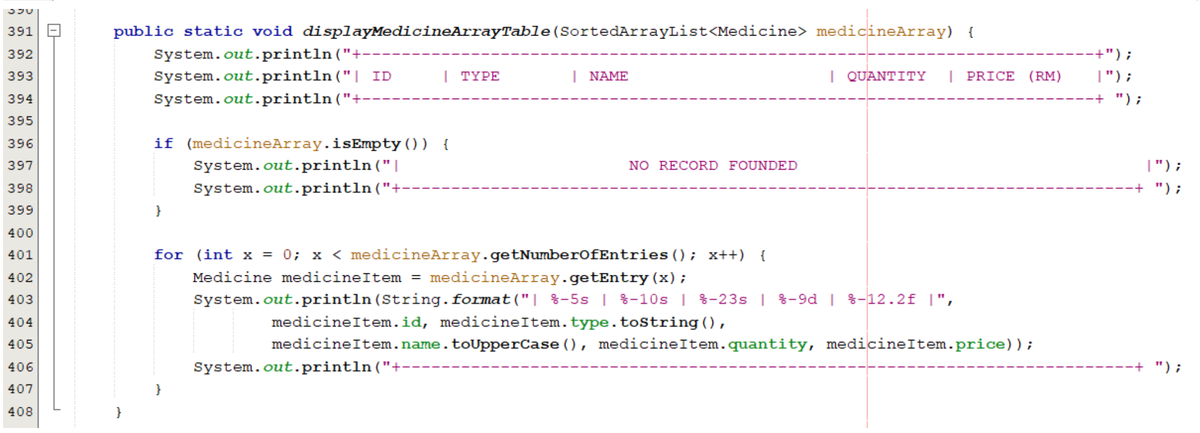
```java
    public void viewMedicineMain(SortedArrayList<Medicine> medicineArray) {
        Scanner scanner = new Scanner(System.in);

        String input;
        displayMedicineArrayTable(medicineArray);

        do {
            System.out.print("Please enter the id to view, 0 to return: ");
            input = scanner.nextLine().toUpperCase();

            var temp = medicineArray.binarySearch(new Medicine(input));

            if (temp.getEntry() != null) {
                temp.getEntry().viewMedicine();
                break;
            } else if (input.charAt(0) == '0') {
                break;
            } else {
                System.out.println("\nInvalid input, please enter again");
            }
        } while (input.charAt(0) != '0');
    }

    public boolean takeMedicine(int quantity) {
        if (this.quantity > quantity) {
            this.quantity -= quantity;
            return true;
        }
        return false;
    }
```



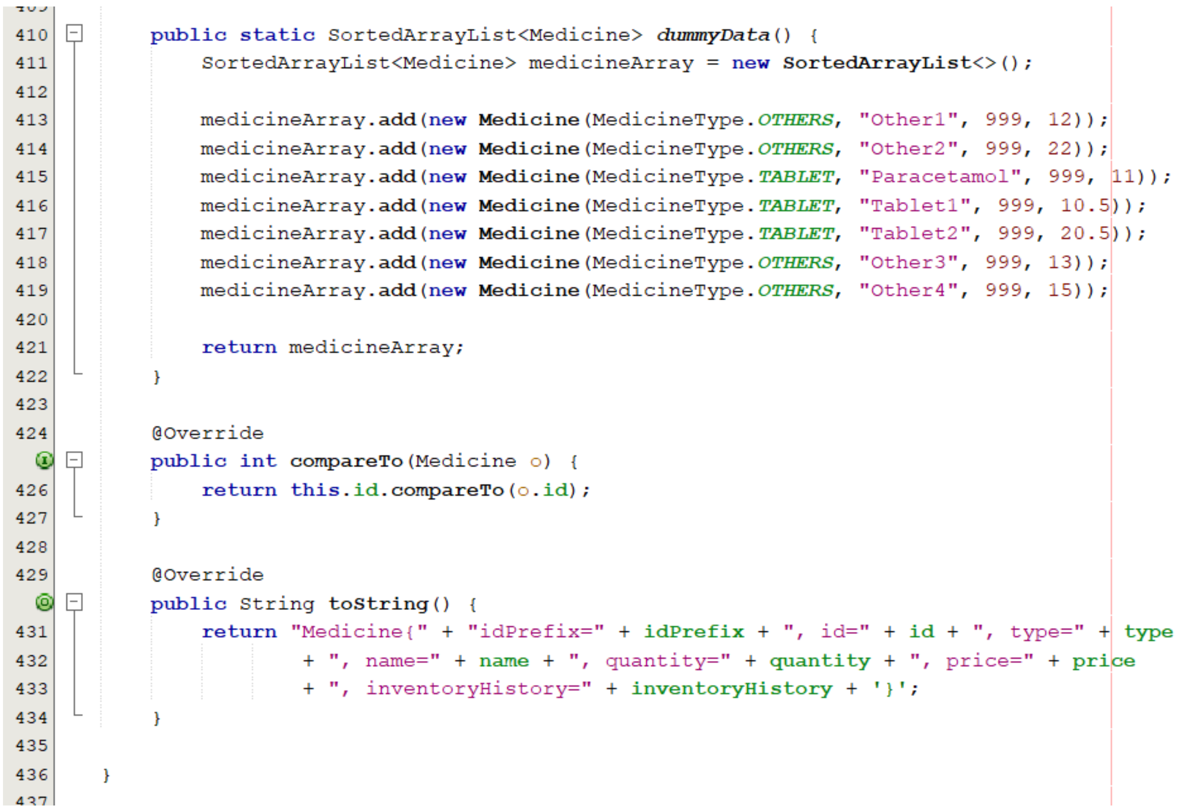
```java
    public int viewMedicineMenu() {
        Scanner scanner = new Scanner(System.in);

        System.out.println("\n+--------- Medicine --------+");
        System.out.println("| 1. Add medicine           |");
        System.out.println("+---------------------------+");
        System.out.println("| 2. Update medicine        |");
        System.out.println("+---------------------------+");
        System.out.println("| 3. Remove medicine        |");
        System.out.println("+---------------------------+");
        System.out.println("| 4. View medicine          |");
        System.out.println("+---------------------------+");
        System.out.println("| 0. Exit                   |");
        System.out.println("+---------------------------+");
        System.out.print("Please enter your selection: ");
        return scanner.nextInt();
    }

    public void medicineModule(SortedArrayList<Medicine> medicineArray) {

        int selection;
        do {
            selection = viewMedicineMenu();
            switch (selection) {
                case 1:
                    medicineArray.add(addMedicine());
                    break;
                case 2:
                    updateMedicineMain(medicineArray);
                    break;
                case 3:
                    deleteMedicineMain(medicineArray);
                    break;
```



```java
                case 4:
                    viewMedicineMain(medicineArray);
                    break;
                case 0:
                    return;
                default:
                    System.out.println("Please enter again.");
            }
        } while (selection != 0);
    }
```

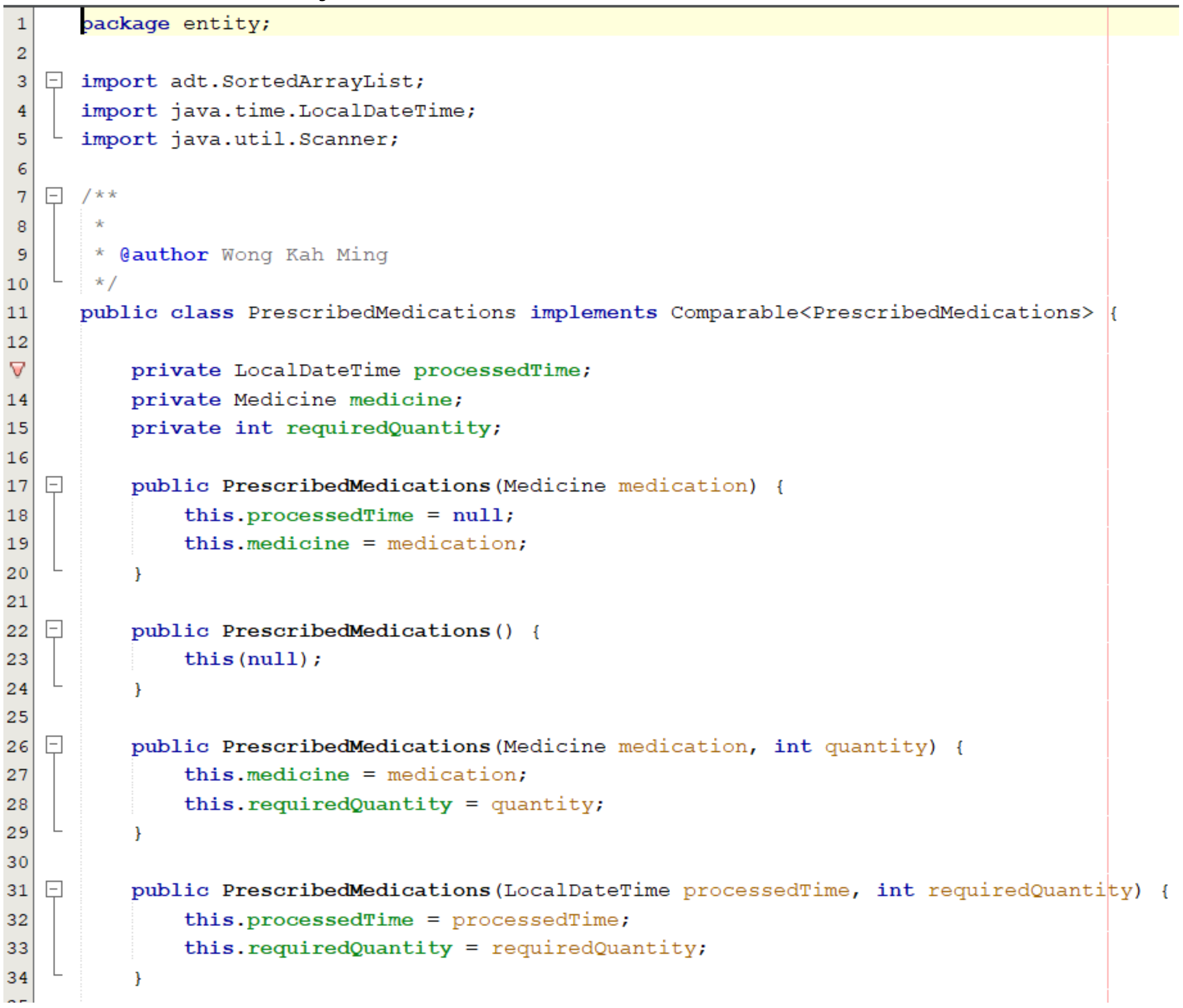


```java
    public static void displayMedicineArrayTable(SortedArrayList<Medicine> medicineArray) {
        System.out.println("+-----------------------------------------------------------------------------+");
        System.out.println("| ID      | TYPE        | NAME                         | QUANTITY  | PRICE (RM)    |");
        System.out.println("+-----------------------------------------------------------------------------+ ");

        if (medicineArray.isEmpty()) {
            System.out.println("|                              NO RECORD FOUNDED                                |");
            System.out.println("+-----------------------------------------------------------------------------+ ");
        }

        for (int x = 0; x < medicineArray.getNumberOfEntries(); x++) {
            Medicine medicineItem = medicineArray.getEntry(x);
            System.out.println(String.format("| %-5s | %-10s | %-23s | %-9d | %-12.2f |",
                    medicineItem.id, medicineItem.type.toString(),
                    medicineItem.name.toUpperCase(), medicineItem.quantity, medicineItem.price));
            System.out.println("+-----------------------------------------------------------------------------+ ");
        }
    }
```

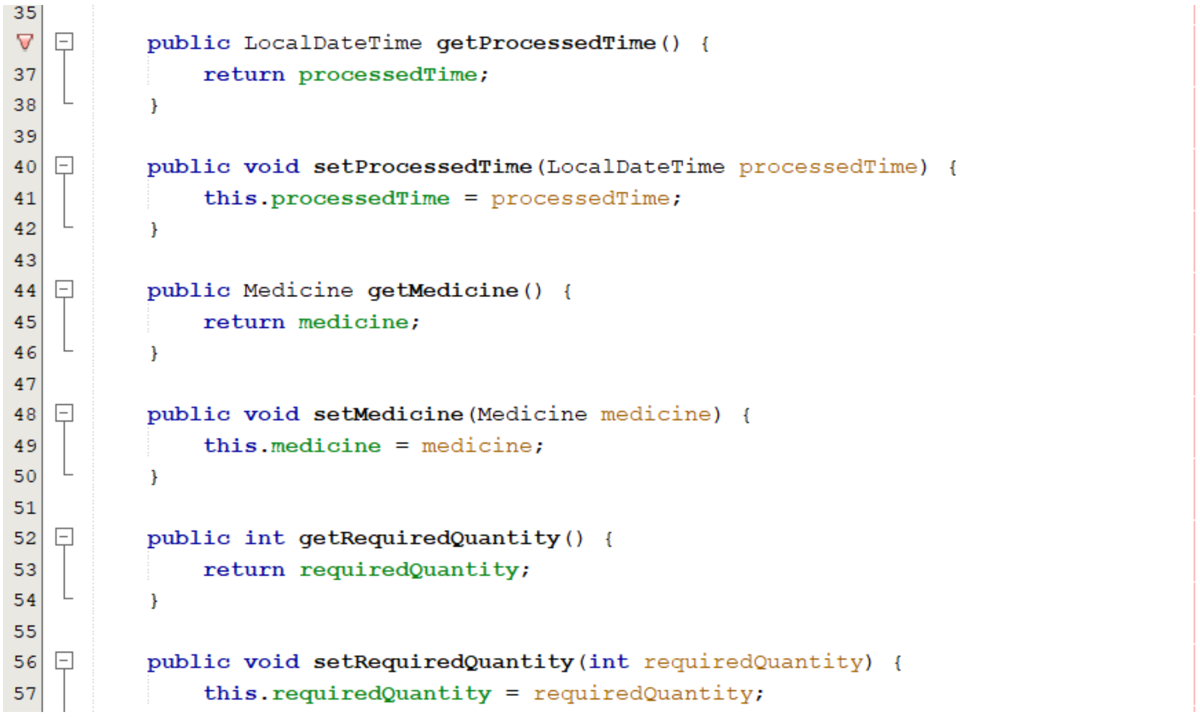


```java
    public static SortedArrayList<Medicine> dummyData() {
        SortedArrayList<Medicine> medicineArray = new SortedArrayList<>();

        medicineArray.add(new Medicine(MedicineType.OTHERS, "Other1", 999, 12));
        medicineArray.add(new Medicine(MedicineType.OTHERS, "Other2", 999, 22));
        medicineArray.add(new Medicine(MedicineType.TABLET, "Paracetamol", 999, 11));
        medicineArray.add(new Medicine(MedicineType.TABLET, "Tablet1", 999, 10.5));
        medicineArray.add(new Medicine(MedicineType.TABLET, "Tablet2", 999, 20.5));
        medicineArray.add(new Medicine(MedicineType.OTHERS, "Other3", 999, 13));
        medicineArray.add(new Medicine(MedicineType.OTHERS, "Other4", 999, 15));

        return medicineArray;
    }

    @Override
    public int compareTo(Medicine o) {
        return this.id.compareTo(o.id);
    }

    @Override
    public String toString() {
        return "Medicine{" + "idPrefix=" + idPrefix + ", id=" + id + ", type=" + type
                + ", name=" + name + ", quantity=" + quantity + ", price=" + price
                + ", inventoryHistory=" + inventoryHistory + '}';
    }

}
```

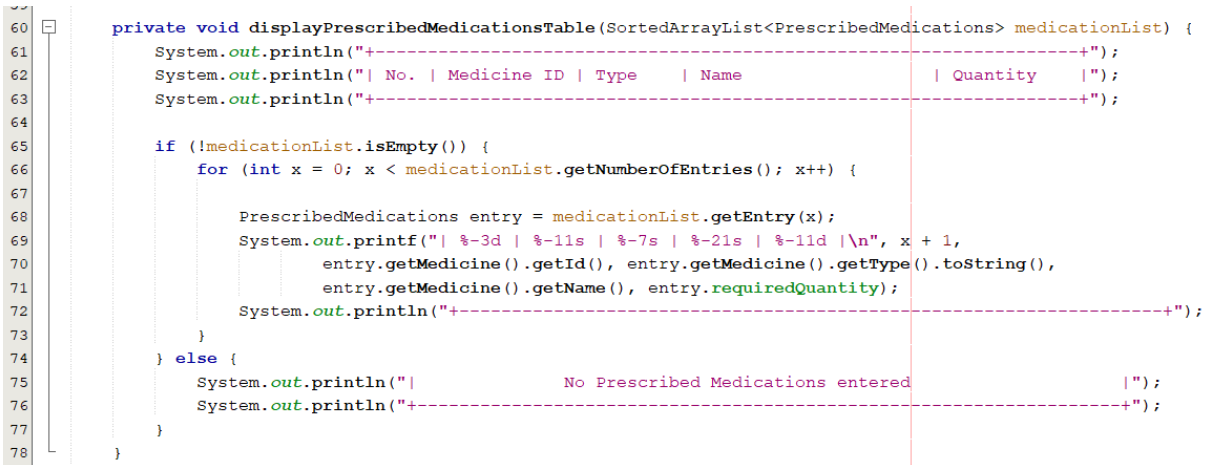
PrescribedMedications.java



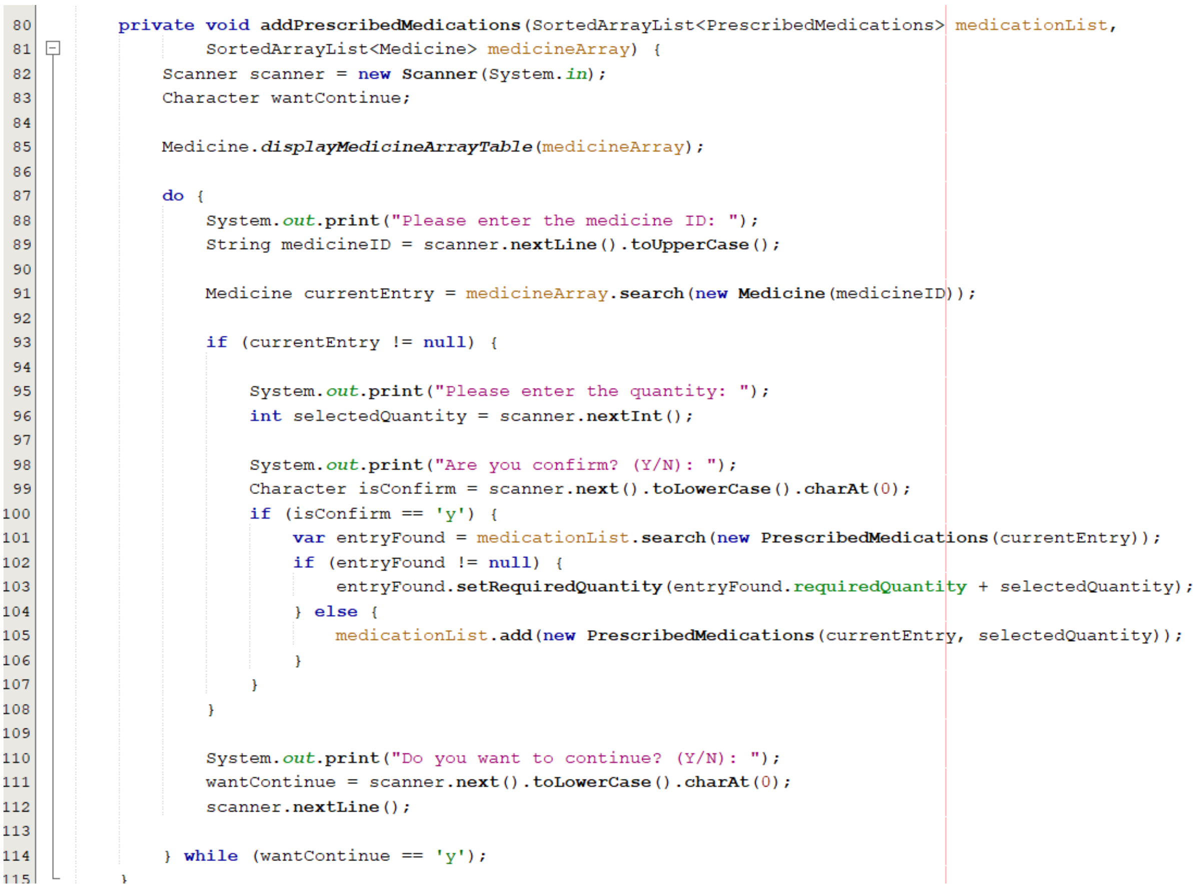
```java
package entity;

import adt.SortedArrayList;
import java.time.LocalDateTime;
import java.util.Scanner;

/**
 *
 * @author Wong Kah Ming
 */
public class PrescribedMedications implements Comparable<PrescribedMedications> {

    private LocalDateTime processedTime;
    private Medicine medicine;
    private int requiredQuantity;

    public PrescribedMedications(Medicine medication) {
        this.processedTime = null;
        this.medicine = medication;
    }

    public PrescribedMedications() {
        this(null);
    }

    public PrescribedMedications(Medicine medication, int quantity) {
        this.medicine = medication;
        this.requiredQuantity = quantity;
    }

    public PrescribedMedications(LocalDateTime processedTime, int requiredQuantity) {
        this.processedTime = processedTime;
        this.requiredQuantity = requiredQuantity;
    }
```



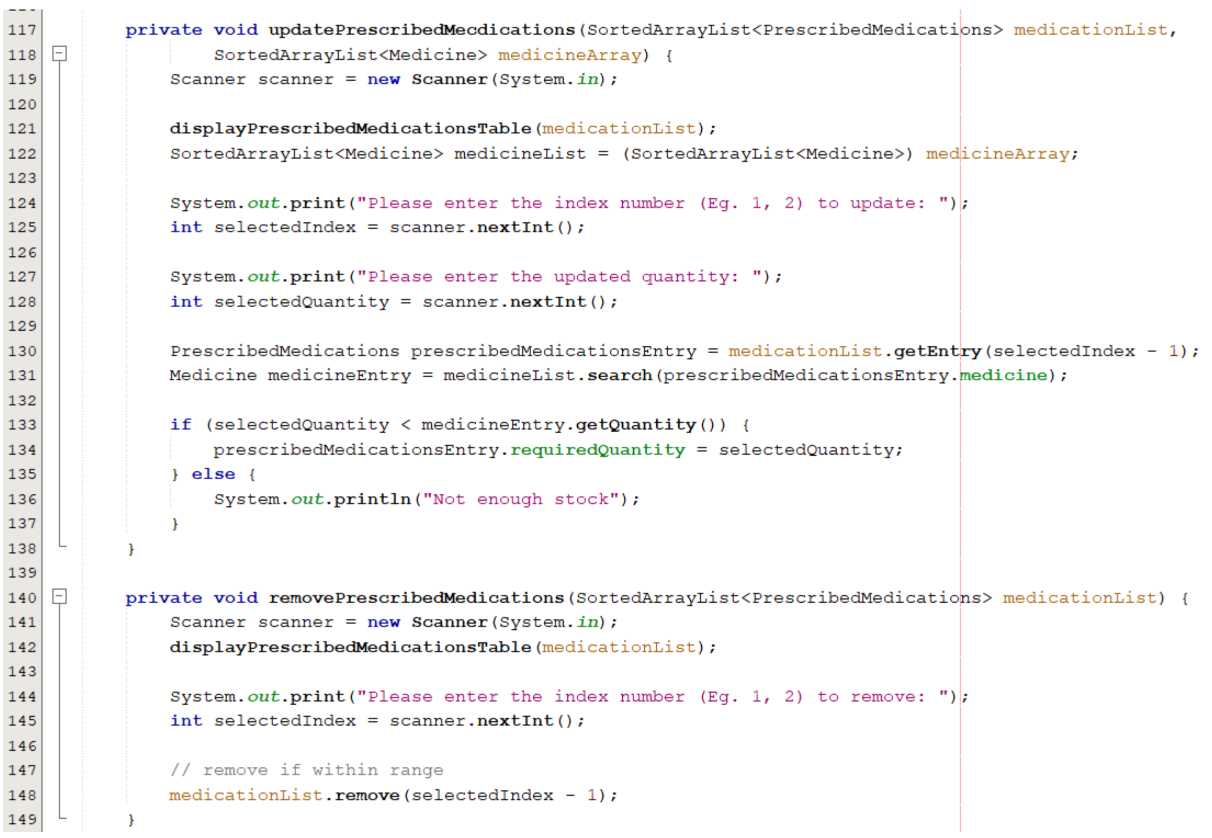
```java

    public LocalDateTime getProcessedTime() {
        return processedTime;
    }

    public void setProcessedTime(LocalDateTime processedTime) {
        this.processedTime = processedTime;
    }

    public Medicine getMedicine() {
        return medicine;
    }

    public void setMedicine(Medicine medicine) {
        this.medicine = medicine;
    }

    public int getRequiredQuantity() {
        return requiredQuantity;
    }

    public void setRequiredQuantity(int requiredQuantity) {
        this.requiredQuantity = requiredQuantity;
```



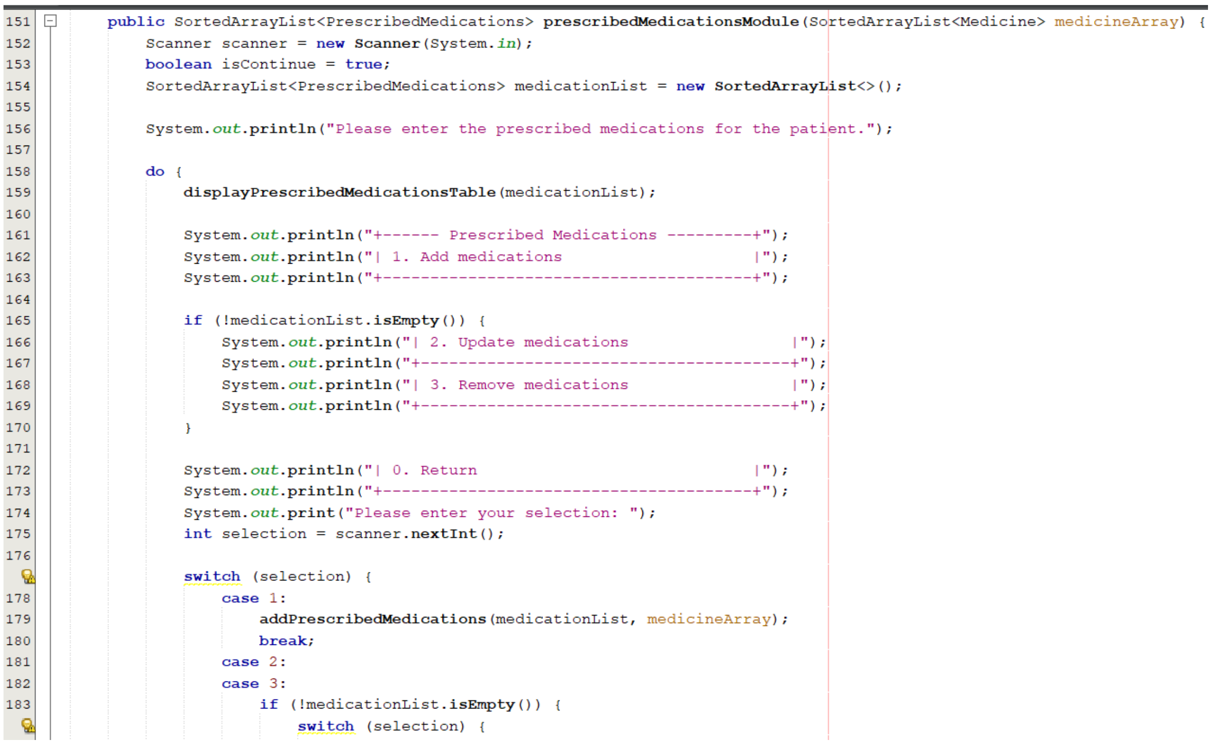
```java
    private void displayPrescribedMedicationsTable(SortedArrayList<PrescribedMedications> medicationList) {
        System.out.println("+-------------------------------------------------------------------+");
        System.out.println("| No. | Medicine ID | Type    | Name                  | Quantity    |");
        System.out.println("+-------------------------------------------------------------------+");

        if (!medicationList.isEmpty()) {
            for (int x = 0; x < medicationList.getNumberOfEntries(); x++) {

                PrescribedMedications entry = medicationList.getEntry(x);
                System.out.printf("| %-3d | %-11s | %-7s | %-21s | %-11d |\n", x + 1,
                        entry.getMedicine().getId(), entry.getMedicine().getType().toString(),
                        entry.getMedicine().getName(), entry.requiredQuantity);
                System.out.println("+-------------------------------------------------------------------+");
            }
        } else {
            System.out.println("|              No Prescribed Medications entered                    |");
            System.out.println("+-------------------------------------------------------------------+");
        }
    }
```



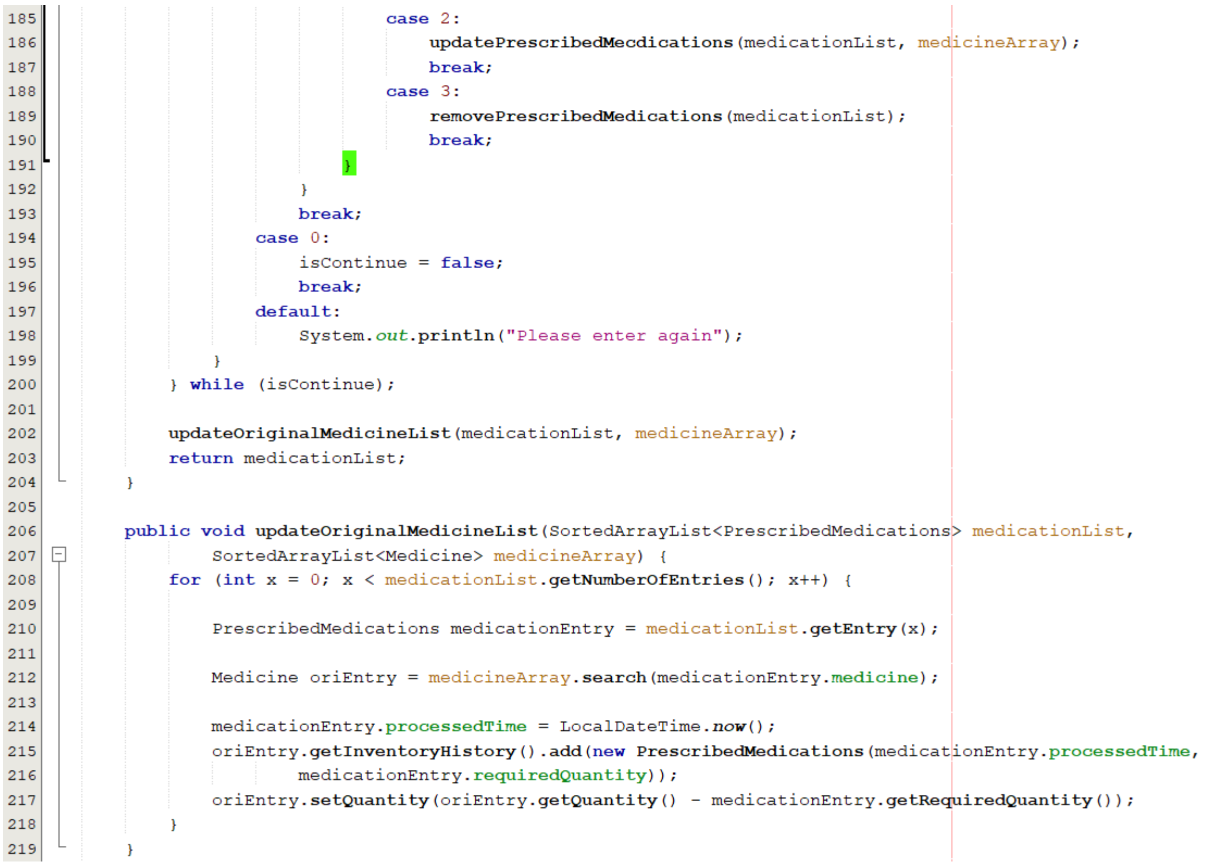
```java
    private void addPrescribedMedications(SortedArrayList<PrescribedMedications> medicationList,
            SortedArrayList<Medicine> medicineArray) {
        Scanner scanner = new Scanner(System.in);
        Character wantContinue;

        Medicine.displayMedicineArrayTable(medicineArray);

        do {
            System.out.print("Please enter the medicine ID: ");
            String medicineID = scanner.nextLine().toUpperCase();

            Medicine currentEntry = medicineArray.search(new Medicine(medicineID));

            if (currentEntry != null) {

                System.out.print("Please enter the quantity: ");
                int selectedQuantity = scanner.nextInt();

                System.out.print("Are you confirm? (Y/N): ");
                Character isConfirm = scanner.next().toLowerCase().charAt(0);
                if (isConfirm == 'y') {
                    var entryFound = medicationList.search(new PrescribedMedications(currentEntry));
                    if (entryFound != null) {
                        entryFound.setRequiredQuantity(entryFound.requiredQuantity + selectedQuantity);
                    } else {
                        medicationList.add(new PrescribedMedications(currentEntry, selectedQuantity));
                    }
                }
            }

            System.out.print("Do you want to continue? (Y/N): ");
            wantContinue = scanner.next().toLowerCase().charAt(0);
            scanner.nextLine();

        } while (wantContinue == 'y');
    }
```



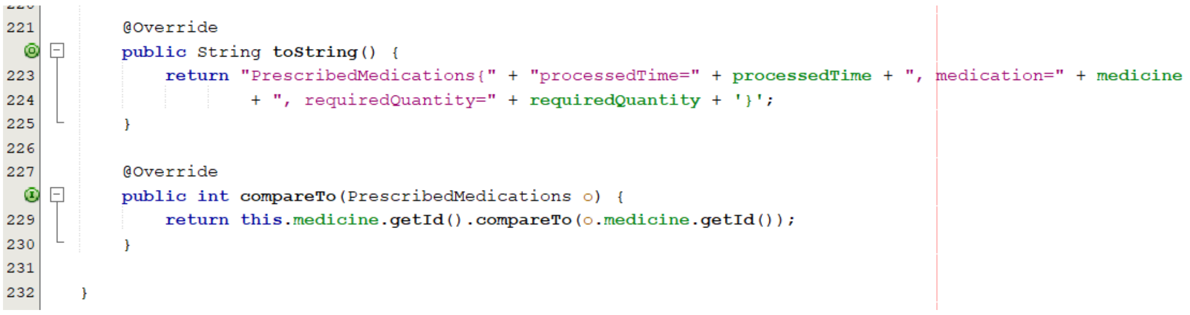
```java
    private void updatePrescribedMecdications(SortedArrayList<PrescribedMedications> medicationList,
            SortedArrayList<Medicine> medicineArray) {
        Scanner scanner = new Scanner(System.in);

        displayPrescribedMedicationsTable(medicationList);
        SortedArrayList<Medicine> medicineList = (SortedArrayList<Medicine>) medicineArray;

        System.out.print("Please enter the index number (Eg. 1, 2) to update: ");
        int selectedIndex = scanner.nextInt();

        System.out.print("Please enter the updated quantity: ");
        int selectedQuantity = scanner.nextInt();

        PrescribedMedications prescribedMedicationsEntry = medicationList.getEntry(selectedIndex - 1);
        Medicine medicineEntry = medicineList.search(prescribedMedicationsEntry.medicine);

        if (selectedQuantity < medicineEntry.getQuantity()) {
            prescribedMedicationsEntry.requiredQuantity = selectedQuantity;
        } else {
            System.out.println("Not enough stock");
        }
    }

    private void removePrescribedMedications(SortedArrayList<PrescribedMedications> medicationList) {
        Scanner scanner = new Scanner(System.in);
        displayPrescribedMedicationsTable(medicationList);

        System.out.print("Please enter the index number (Eg. 1, 2) to remove: ");
        int selectedIndex = scanner.nextInt();

        // remove if within range
        medicationList.remove(selectedIndex - 1);
    }
```



```java
    public SortedArrayList<PrescribedMedications> prescribedMedicationsModule(SortedArrayList<Medicine> medicineArray) {
        Scanner scanner = new Scanner(System.in);
        boolean isContinue = true;
        SortedArrayList<PrescribedMedications> medicationList = new SortedArrayList<>();

        System.out.println("Please enter the prescribed medications for the patient.");

        do {
            displayPrescribedMedicationsTable(medicationList);

            System.out.println("+------ Prescribed Medications ---------+");
            System.out.println("| 1. Add medications                    |");
            System.out.println("+---------------------------------------+");

            if (!medicationList.isEmpty()) {
                System.out.println("| 2. Update medications                 |");
                System.out.println("+---------------------------------------+");
                System.out.println("| 3. Remove medications                 |");
                System.out.println("+---------------------------------------+");
            }

            System.out.println("| 0. Return                             |");
            System.out.println("+---------------------------------------+");
            System.out.print("Please enter your selection: ");
            int selection = scanner.nextInt();

            switch (selection) {
                case 1:
                    addPrescribedMedications(medicationList, medicineArray);
                    break;
                case 2:
                case 3:
                    if (!medicationList.isEmpty()) {
                        switch (selection) {
```



```
                            case 2:
                                updatePrescribedMecdications(medicationList, medicineArray);
                                break;
                            case 3:
                                removePrescribedMedications(medicationList);
                                break;
                        }
                    }
                    break;
                case 0:
                    isContinue = false;
                    break;
                default:
                    System.out.println("Please enter again");
            }
        } while (isContinue);

        updateOriginalMedicineList(medicationList, medicineArray);
        return medicationList;
    }

    public void updateOriginalMedicineList(SortedArrayList<PrescribedMedications> medicationList,
            SortedArrayList<Medicine> medicineArray) {
        for (int x = 0; x < medicationList.getNumberOfEntries(); x++) {

            PrescribedMedications medicationEntry = medicationList.getEntry(x);

            Medicine oriEntry = medicineArray.search(medicationEntry.medicine);

            medicationEntry.processedTime = LocalDateTime.now();
            oriEntry.getInventoryHistory().add(new PrescribedMedications(medicationEntry.processedTime,
                    medicationEntry.requiredQuantity));
            oriEntry.setQuantity(oriEntry.getQuantity() - medicationEntry.getRequiredQuantity());
        }
    }
```



```
    @Override
    public String toString() {
        return "PrescribedMedications{" + "processedTime=" + processedTime + ", medication=" + medicine
                + ", requiredQuantity=" + requiredQuantity + '}';
    }

    @Override
    public int compareTo(PrescribedMedications o) {
        return this.medicine.getId().compareTo(o.medicine.getId());
    }

}
```

5. Client Program

The Sorted Array List has been chosen to be used in my client classes, Medicine and Prescribed Medications, because I would like to sort the entry according to their entry so the staff can have better reference when the medicines or prescribed medications are displayed in order. In addition, since when the medicine or the prescribed medications object is being added into the list is not concerning about the location as it will be sorted as soon as being added into the list, therefore I would not needed to use the unsorted list such as Array List and Linked List that allows the user to add the object at the specific index that needed.

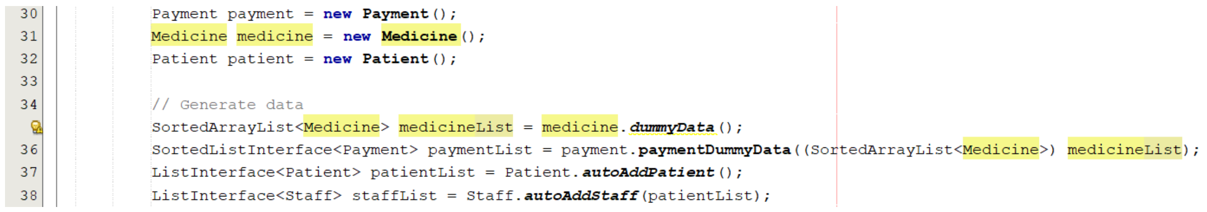
Although queue and stack may perform the same function such as adding the object in sequence, however it does not support for the user to get or obtain the required data in between the queue or stack. For example, when the user wants to view one specific medicine details, he will be entering the id and the system will return an object that matches with the result, and this may be found in between the list. Thus, queue and stack will not be suitable for this case.

To add on, although the Sorted Array List will be having a definite size, which is 20 in this case that been set, it will not bother much as the list for the prescribed medication will only required less size, while as for medicine, it might cause larger size to store the required medicine information, however, the doubleArray() inside the Sorted Array List has solved this issue by doubling the list whenever needed, which is not a big issue compared to using the linked list.

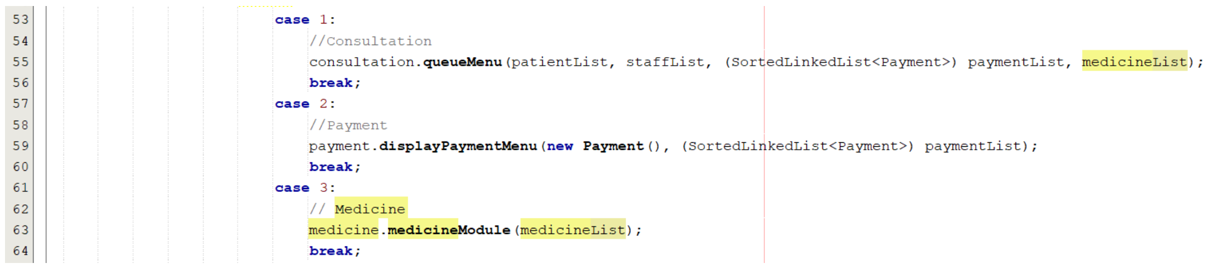
Also, by using Sorted Array List, it allows the binary search searching method to be used. This is because Sorted Array List has fixed size, where the binary search function can directly implemented by using the information available for a sorted array list, compared to linked list or sorted linked list that have undefined size, which will cause longer time of using binary search to search for the required data. Thus, Sorted Array List has to be considered.

However, as for the inventoryHistory in Medicine module, it will be using the linked list as it will have undefined list size, where we can add the history whenever needed without bothering the size. Also, since it belongs to the same medicine object's record, it is not needed to be sorted as inside the client class, it has been declared to use id to compare the object. Thus, a linked list is to be used as it has no fixed size which saves memory, that only allocates memory when needed, and not required to be sorted.

Driver Program:



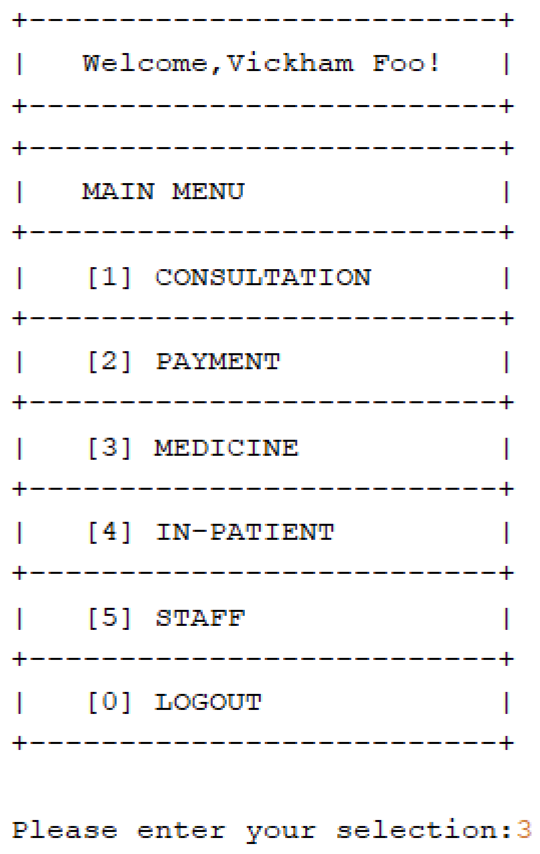
```
Payment payment = new Payment();
Medicine medicine = new Medicine();
Patient patient = new Patient();

// Generate data
SortedArrayList<Medicine> medicineList = medicine.dummyData();
SortedListInterface<Payment> paymentList = payment.paymentDummyData((SortedArrayList<Medicine>) medicineList);
ListInterface<Patient> patientList = Patient.autoAddPatient();
ListInterface<Staff> staffList = Staff.autoAddStaff(patientList);
```



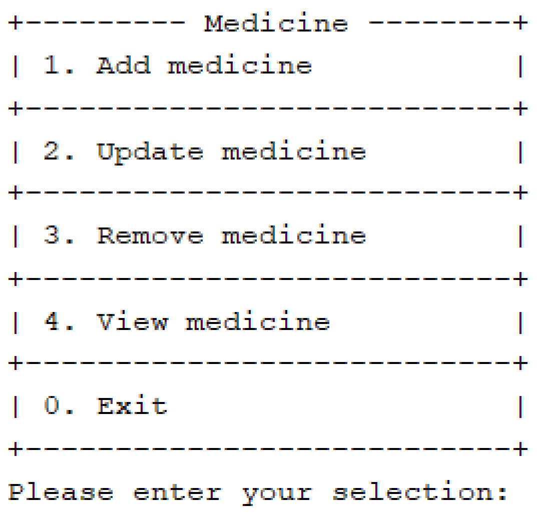
```
case 1:
    //Consultation
    consultation.queueMenu(patientList, staffList, (SortedLinkedList<Payment>) paymentList, medicineList);
    break;
case 2:
    //Payment
    payment.displayPaymentMenu(new Payment(), (SortedLinkedList<Payment>) paymentList);
    break;
case 3:
    // Medicine
    medicine.medicineModule(medicineList);
    break;
```

Inside the driver program, a medicine object and medicineList has been created to provide the medicine data to showcase the program. The medicineList will be passed to the consultation module so the user may obtain and select the available medicine in the clinic or inventory. The medicationList will also be passed to the medicine module as the medicine list data may be changed when the user has selected prescribed medicine and proceed with payment.

Output:
Medicine Module



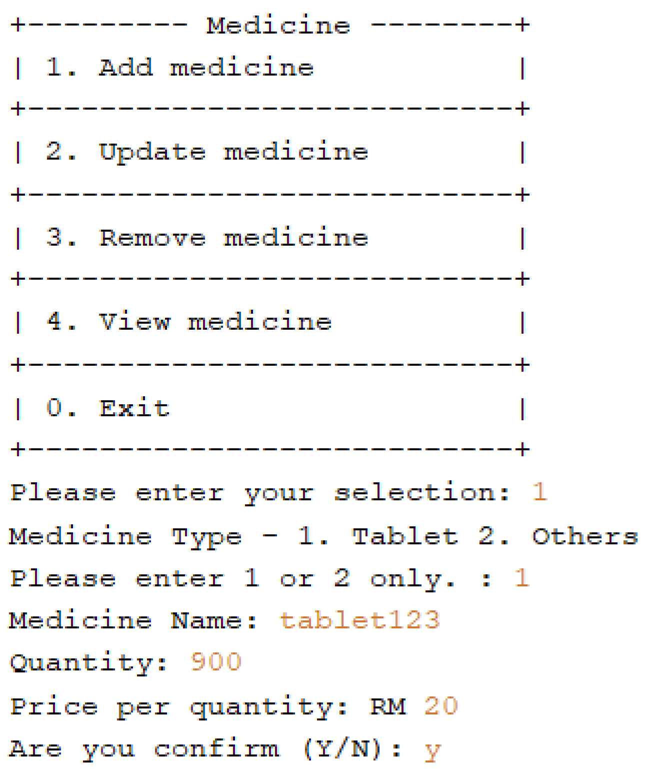
```
+--------------------------+
|   Welcome,Vickham Foo!   |
+--------------------------+
+--------------------------+
|   MAIN MENU              |
+--------------------------+
|    [1] CONSULTATION      |
+--------------------------+
|    [2] PAYMENT           |
+--------------------------+
|    [3] MEDICINE          |
+--------------------------+
|    [4] IN-PATIENT        |
+--------------------------+
|    [5] STAFF             |
+--------------------------+
|    [0] LOGOUT            |
+--------------------------+

Please enter your selection:3
```



```
+--------- Medicine --------+
| 1. Add medicine           |
+---------------------------+
| 2. Update medicine        |
+---------------------------+
| 3. Remove medicine        |
+---------------------------+
| 4. View medicine          |
+---------------------------+
| 0. Exit                   |
+---------------------------+
Please enter your selection:
```

The user may access the Medicine module by entering 3 on the main menu. This medicine module will act as the clinic's inventory module that manipulates the medicine stock on the system such as add, view, update and remove.

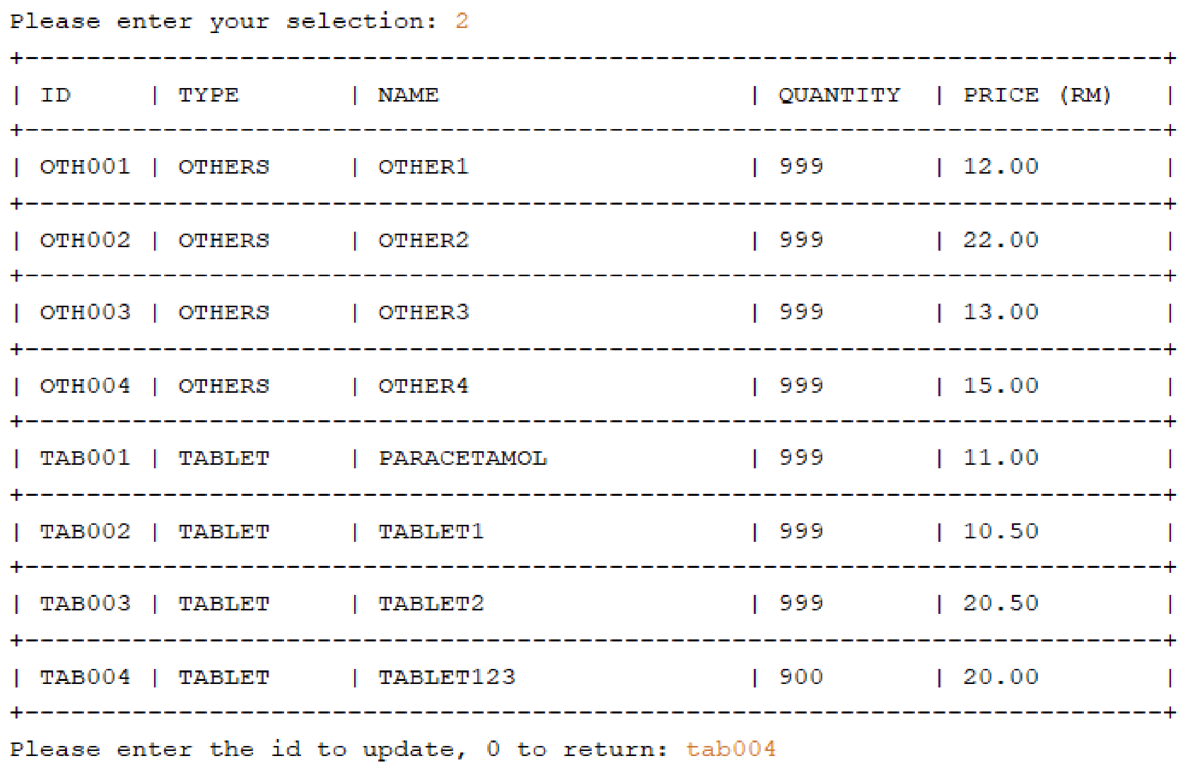
Add medicine



```
+--------- Medicine --------+
| 1. Add medicine           |
+---------------------------+
| 2. Update medicine        |
+---------------------------+
| 3. Remove medicine        |
+---------------------------+
| 4. View medicine          |
+---------------------------+
| 0. Exit                   |
+---------------------------+
Please enter your selection: 1
Medicine Type - 1. Tablet 2. Others
Please enter 1 or 2 only. : 1
Medicine Name: tablet123
Quantity: 900
Price per quantity: RM 20
Are you confirm (Y/N): y
```

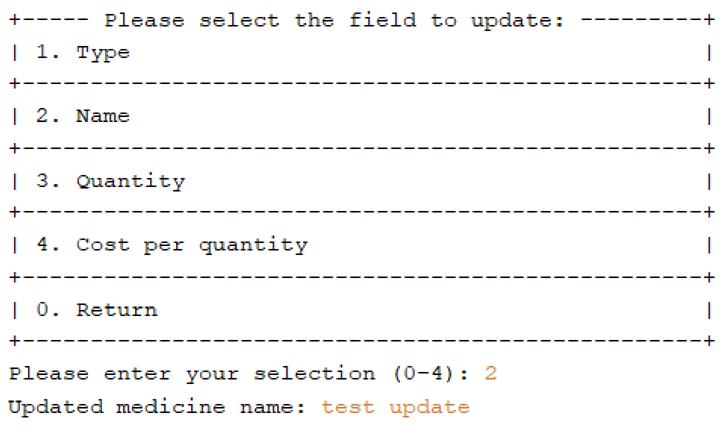
When the user selects 1 from the Medicine module menu, it will be prompted to ask the user to enter the details such as type, name, quantity, and the price. When the user confirms the details, the medicine will be stored in the sorted list and it will be sorted according to the id since different medicine types will have different id prefix such as tablet will be having “TAB” prefix and others will be having “OTH” prefix.

Update medicine



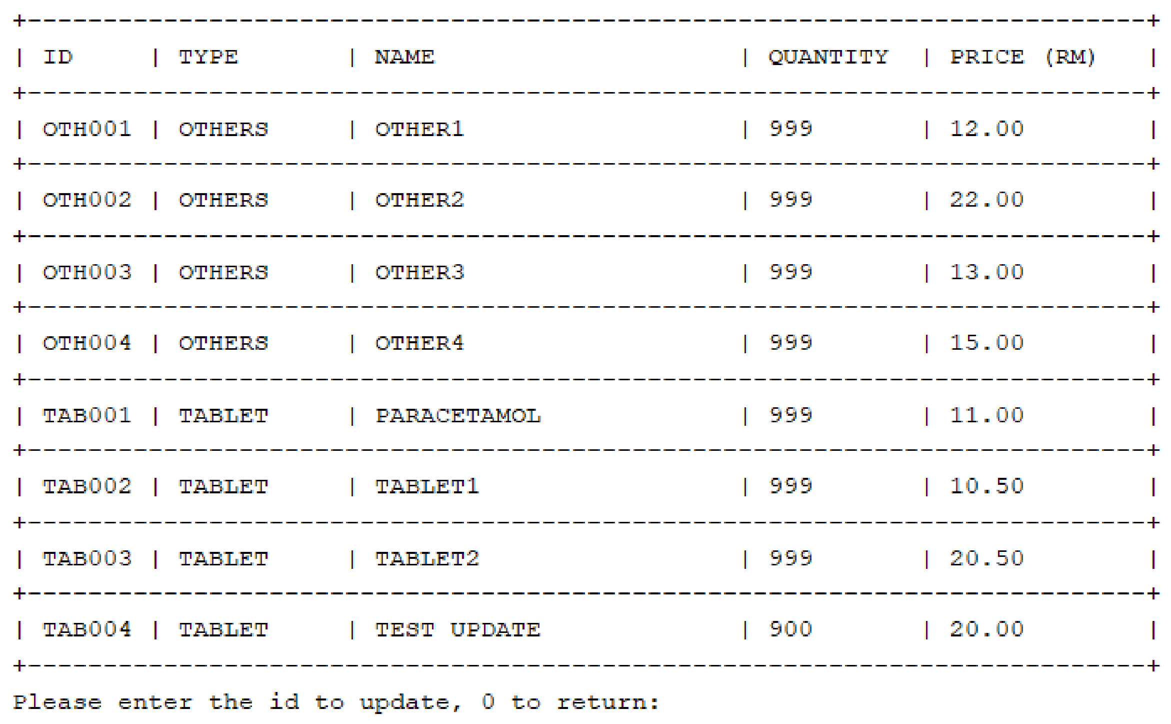
```
Please enter your selection: 2
+-----------------------------------------------------------------------------+
| ID      | TYPE         | NAME                      | QUANTITY  | PRICE (RM)  |
+-----------------------------------------------------------------------------+
| OTH001  | OTHERS       | OTHER1                    | 999       | 12.00       |
+-----------------------------------------------------------------------------+
| OTH002  | OTHERS       | OTHER2                    | 999       | 22.00       |
+-----------------------------------------------------------------------------+
| OTH003  | OTHERS       | OTHER3                    | 999       | 13.00       |
+-----------------------------------------------------------------------------+
| OTH004  | OTHERS       | OTHER4                    | 999       | 15.00       |
+-----------------------------------------------------------------------------+
| TAB001  | TABLET       | PARACETAMOL               | 999       | 11.00       |
+-----------------------------------------------------------------------------+
| TAB002  | TABLET       | TABLET1                   | 999       | 10.50       |
+-----------------------------------------------------------------------------+
| TAB003  | TABLET       | TABLET2                   | 999       | 20.50       |
+-----------------------------------------------------------------------------+
| TAB004  | TABLET       | TABLET123                 | 900       | 20.00       |
+-----------------------------------------------------------------------------+
Please enter the id to update, 0 to return: tab004
```

When the user chooses to update the medicine details, it will prompt a list of medicine data for the user to enter the specific id to edit the required medicine details. The id will be used to find the object that has the same id inside the sorted list using binary search. Once founded, the medicine object will be returned.



```
+----- Please select the field to update: ---------+
| 1. Type                                          |
+--------------------------------------------------+
| 2. Name                                          |
+--------------------------------------------------+
| 3. Quantity                                      |
+--------------------------------------------------+
| 4. Cost per quantity                             |
+--------------------------------------------------+
| 0. Return                                        |
+--------------------------------------------------+
Please enter your selection (0-4): 2
Updated medicine name: test update
```

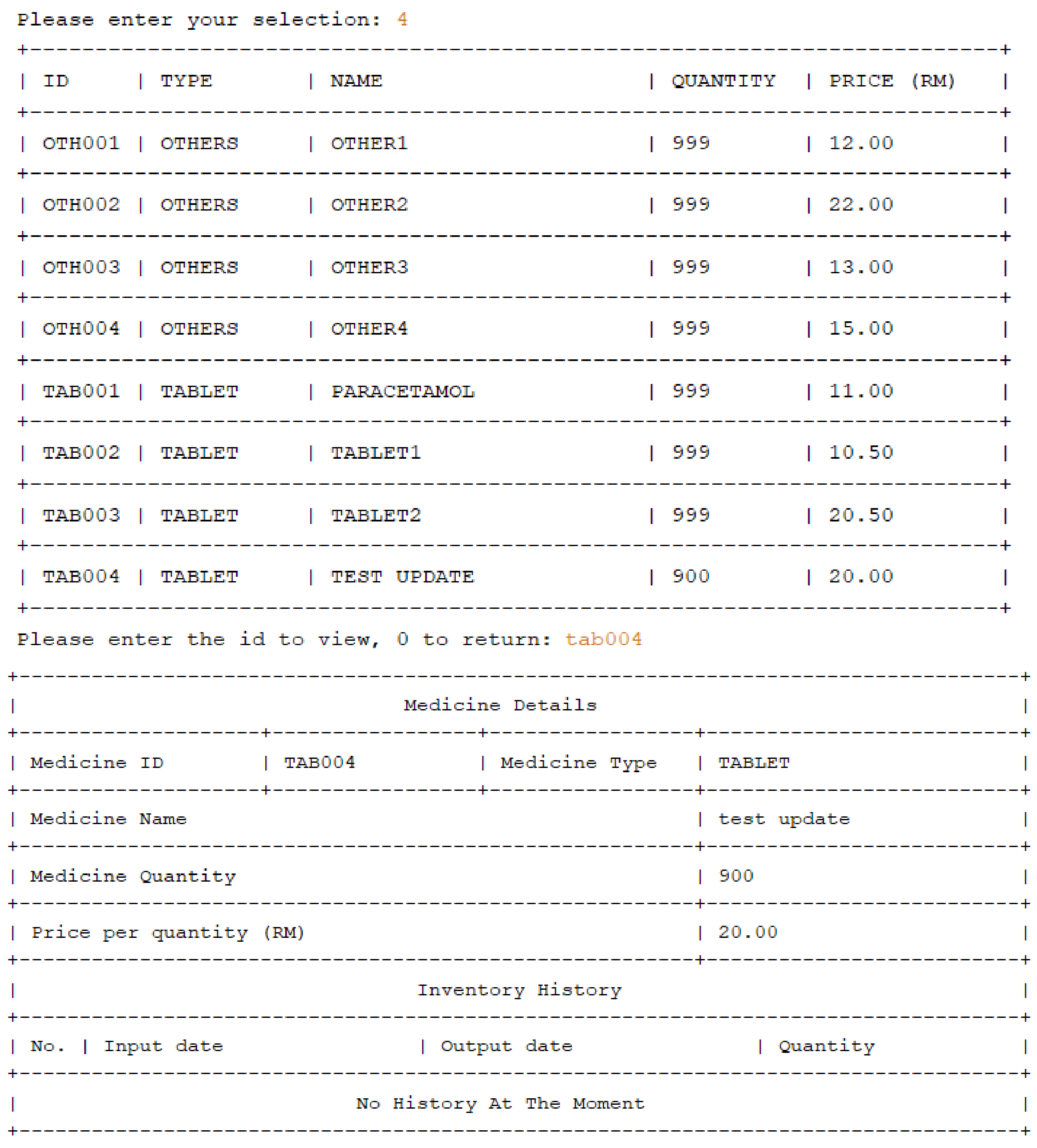
The update function allows the user to make changes to almost all of the attributes such as medicine type, name, quantity, and cost per quantity.



```
+--------------------------------------------------------------------------+
| ID     | TYPE       | NAME                    | QUANTITY  | PRICE (RM)   |
+--------------------------------------------------------------------------+
| OTH001 | OTHERS     | OTHER1                  | 999       | 12.00        |
+--------------------------------------------------------------------------+
| OTH002 | OTHERS     | OTHER2                  | 999       | 22.00        |
+--------------------------------------------------------------------------+
| OTH003 | OTHERS     | OTHER3                  | 999       | 13.00        |
+--------------------------------------------------------------------------+
| OTH004 | OTHERS     | OTHER4                  | 999       | 15.00        |
+--------------------------------------------------------------------------+
| TAB001 | TABLET     | PARACETAMOL             | 999       | 11.00        |
+--------------------------------------------------------------------------+
| TAB002 | TABLET     | TABLET1                 | 999       | 10.50        |
+--------------------------------------------------------------------------+
| TAB003 | TABLET     | TABLET2                 | 999       | 20.50        |
+--------------------------------------------------------------------------+
| TAB004 | TABLET     | TEST UPDATE             | 900       | 20.00        |
+--------------------------------------------------------------------------+
Please enter the id to update, 0 to return:
```

Once the user confirms the updates, the medicine object will be updated. If the type has been changed, the original entry will be removed and stored as a new medicine entry as each medicine will have their corresponding id according to their type.

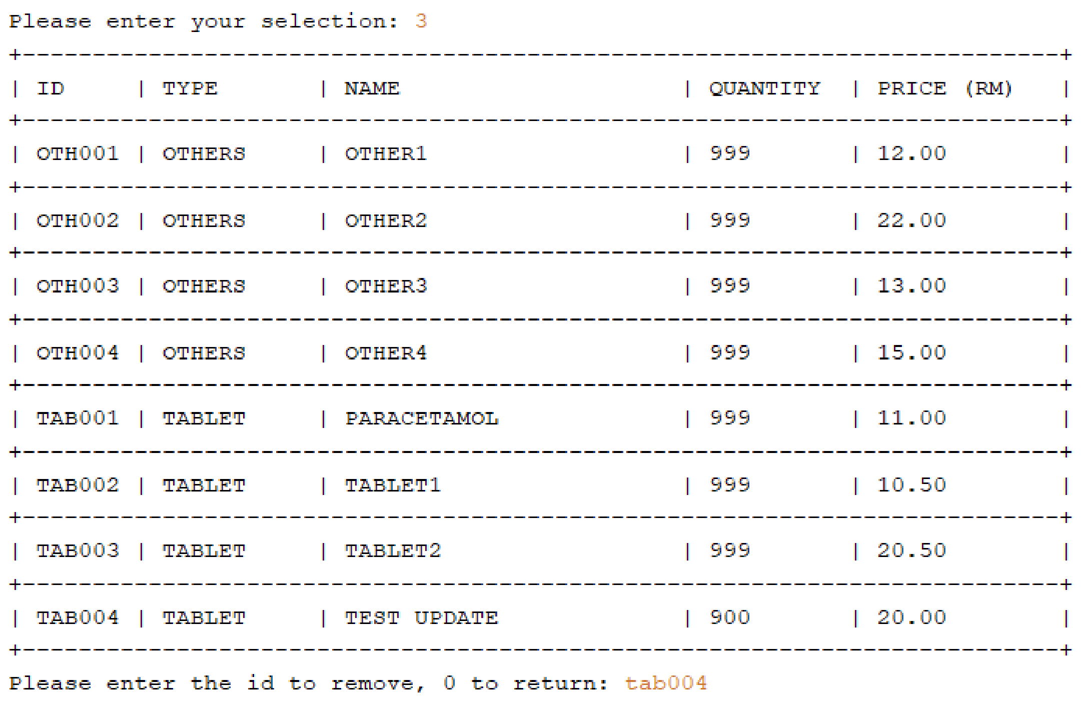
View medicine



```
Please enter your selection: 4
+--------------------------------------------------------------------------------+
| ID     | TYPE         | NAME                           | QUANTITY  | PRICE (RM)   |
+--------------------------------------------------------------------------------+
| OTH001 | OTHERS       | OTHER1                         | 999       | 12.00        |
+--------------------------------------------------------------------------------+
| OTH002 | OTHERS       | OTHER2                         | 999       | 22.00        |
+--------------------------------------------------------------------------------+
| OTH003 | OTHERS       | OTHER3                         | 999       | 13.00        |
+--------------------------------------------------------------------------------+
| OTH004 | OTHERS       | OTHER4                         | 999       | 15.00        |
+--------------------------------------------------------------------------------+
| TAB001 | TABLET       | PARACETAMOL                    | 999       | 11.00        |
+--------------------------------------------------------------------------------+
| TAB002 | TABLET       | TABLET1                        | 999       | 10.50        |
+--------------------------------------------------------------------------------+
| TAB003 | TABLET       | TABLET2                        | 999       | 20.50        |
+--------------------------------------------------------------------------------+
| TAB004 | TABLET       | TEST UPDATE                    | 900       | 20.00        |
+--------------------------------------------------------------------------------+
Please enter the id to view, 0 to return: tab004
+--------------------------------------------------------------------------------------------+
|                                  Medicine Details                                          |
+---------------------+------------------+------------------+------------------------------+
| Medicine ID         | TAB004           | Medicine Type    | TABLET                       |
+---------------------+------------------+------------------+------------------------------+
| Medicine Name                                            | test update                  |
+----------------------------------------------------------+------------------------------+
| Medicine Quantity                                        | 900                          |
+----------------------------------------------------------+------------------------------+
| Price per quantity (RM)                                  | 20.00                        |
+----------------------------------------------------------+------------------------------+
|                                  Inventory History                                         |
+--------------------------------------------------------------------------------------------+
| No. | Input date                | Output date               | Quantity                     |
+--------------------------------------------------------------------------------------------+
|                              No History At The Moment                                      |
+--------------------------------------------------------------------------------------------+
```

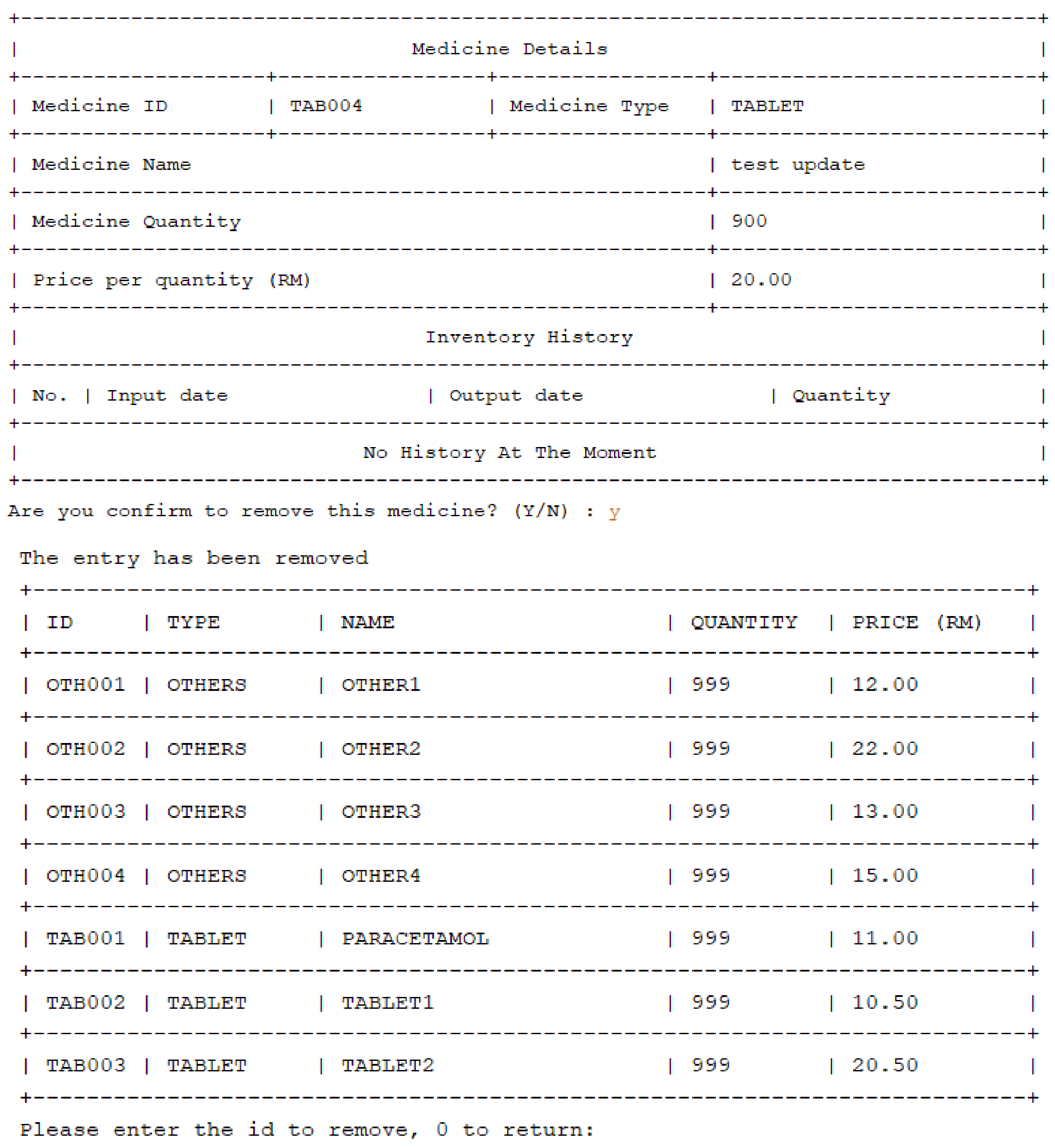
When the user entered 4 to view the medicine, the whole medicine entry in the sorted list will be displayed. It will be displayed with the id's alphabetical order since it has been sorted when being added into the list. Then, the user will be entering the medicine id to view the specific details of the medicine.

Remove medicine



```
Please enter your selection: 3
+--------------------------------------------------------------------------+
| ID     | TYPE       | NAME                    | QUANTITY  | PRICE (RM)   |
+--------------------------------------------------------------------------+
| OTH001 | OTHERS     | OTHER1                  | 999       | 12.00        |
+--------------------------------------------------------------------------+
| OTH002 | OTHERS     | OTHER2                  | 999       | 22.00        |
+--------------------------------------------------------------------------+
| OTH003 | OTHERS     | OTHER3                  | 999       | 13.00        |
+--------------------------------------------------------------------------+
| OTH004 | OTHERS     | OTHER4                  | 999       | 15.00        |
+--------------------------------------------------------------------------+
| TAB001 | TABLET     | PARACETAMOL             | 999       | 11.00        |
+--------------------------------------------------------------------------+
| TAB002 | TABLET     | TABLET1                 | 999       | 10.50        |
+--------------------------------------------------------------------------+
| TAB003 | TABLET     | TABLET2                 | 999       | 20.50        |
+--------------------------------------------------------------------------+
| TAB004 | TABLET     | TEST UPDATE             | 900       | 20.00        |
+--------------------------------------------------------------------------+
Please enter the id to remove, 0 to return: tab004
```

Similar to the view and update function, the whole data will be displayed, and ask for the user to enter the specific id to perform further modification, which is delete in this situation.

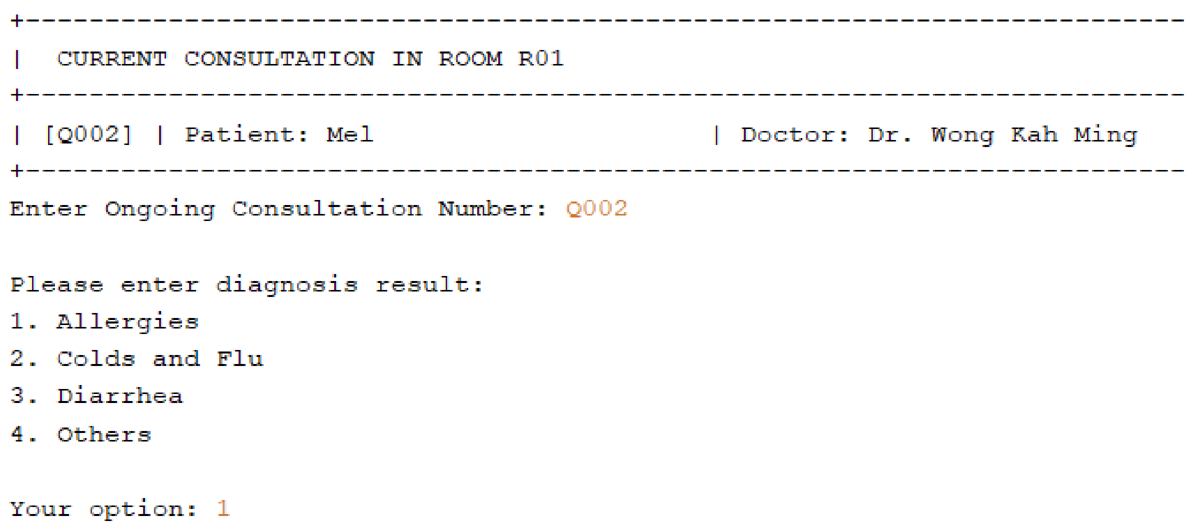


```
+---------------------------------------------------------------------------------+
|                              Medicine Details                                   |
+--------------------+----------------+----------------+-------------------------+
| Medicine ID        | TAB004         | Medicine Type  | TABLET                  |
+--------------------+----------------+----------------+-------------------------+
| Medicine Name                                      | test update             |
+-----------------------------------------------------+-------------------------+
| Medicine Quantity                                  | 900                     |
+-----------------------------------------------------+-------------------------+
| Price per quantity (RM)                            | 20.00                   |
+-----------------------------------------------------+-------------------------+
|                              Inventory History                                  |
+---------------------------------------------------------------------------------+
| No. | Input date                | Output date                | Quantity        |
+---------------------------------------------------------------------------------+
|                         No History At The Moment                                |
+---------------------------------------------------------------------------------+
Are you confirm to remove this medicine? (Y/N) : y

 The entry has been removed
 +-------------------------------------------------------------------------------+
 | ID     | TYPE         | NAME                       | QUANTITY  | PRICE (RM)   |
 +-------------------------------------------------------------------------------+
 | OTH001 | OTHERS       | OTHER1                     | 999       | 12.00        |
 +-------------------------------------------------------------------------------+
 | OTH002 | OTHERS       | OTHER2                     | 999       | 22.00        |
 +-------------------------------------------------------------------------------+
 | OTH003 | OTHERS       | OTHER3                     | 999       | 13.00        |
 +-------------------------------------------------------------------------------+
 | OTH004 | OTHERS       | OTHER4                     | 999       | 15.00        |
 +-------------------------------------------------------------------------------+
 | TAB001 | TABLET       | PARACETAMOL                | 999       | 11.00        |
 +-------------------------------------------------------------------------------+
 | TAB002 | TABLET       | TABLET1                    | 999       | 10.50        |
 +-------------------------------------------------------------------------------+
 | TAB003 | TABLET       | TABLET2                    | 999       | 20.50        |
 +-------------------------------------------------------------------------------+
 Please enter the id to remove, 0 to return:
```

Once the user has confirmed the deletion, the object will be removed from the sorted list and move forwards for each entry that is behind the deleted object.

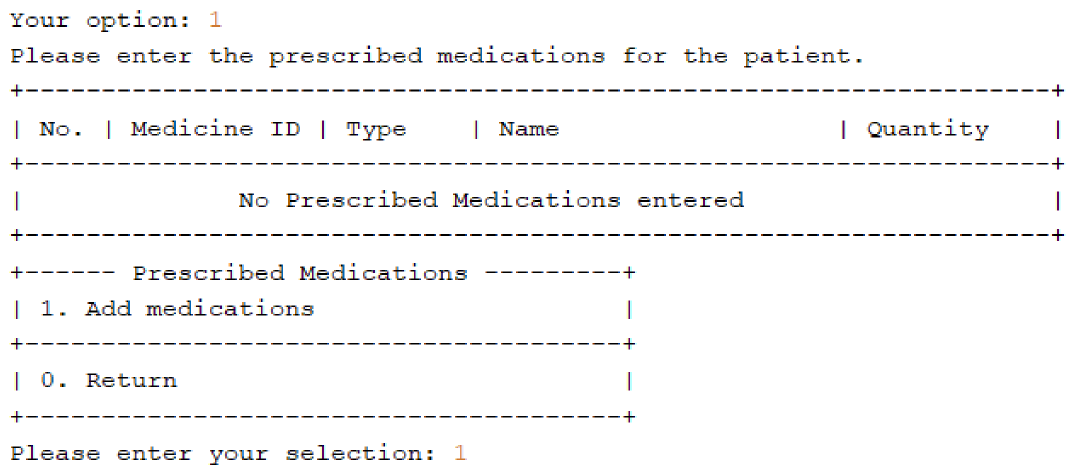
Prescribed Medication Module

This module will be used at the Consultation Module to assign the required medicine according to the doctor's consultation. It will include the add, update and delete function at the prescribed medication.



```
+-----------------------------------------------------------------------
|  CURRENT CONSULTATION IN ROOM R01
+-----------------------------------------------------------------------
| [Q002] | Patient: Mel                     | Doctor: Dr. Wong Kah Ming
+-----------------------------------------------------------------------
Enter Ongoing Consultation Number: Q002

Please enter diagnosis result:
1. Allergies
2. Colds and Flu
3. Diarrhea
4. Others

Your option: 1
```

When the user selected the ongoing consultation, the user may now enter the diagnosis result.



```
Your option: 1
Please enter the prescribed medications for the patient.
+------------------------------------------------------------------+
| No. | Medicine ID | Type     | Name                     | Quantity     |
+------------------------------------------------------------------+
|               No Prescribed Medications entered                  |
+------------------------------------------------------------------+
+------ Prescribed Medications ---------+
| 1. Add medications                    |
+---------------------------------------+
| 0. Return                             |
+---------------------------------------+
Please enter your selection: 1
```

After the user has entered the diagnosis result, the user may now enter the prescribed medicine according to the diagnosis result. The user may choose to add medications, or even exist this page to indicate that the patient does not require any medicine.

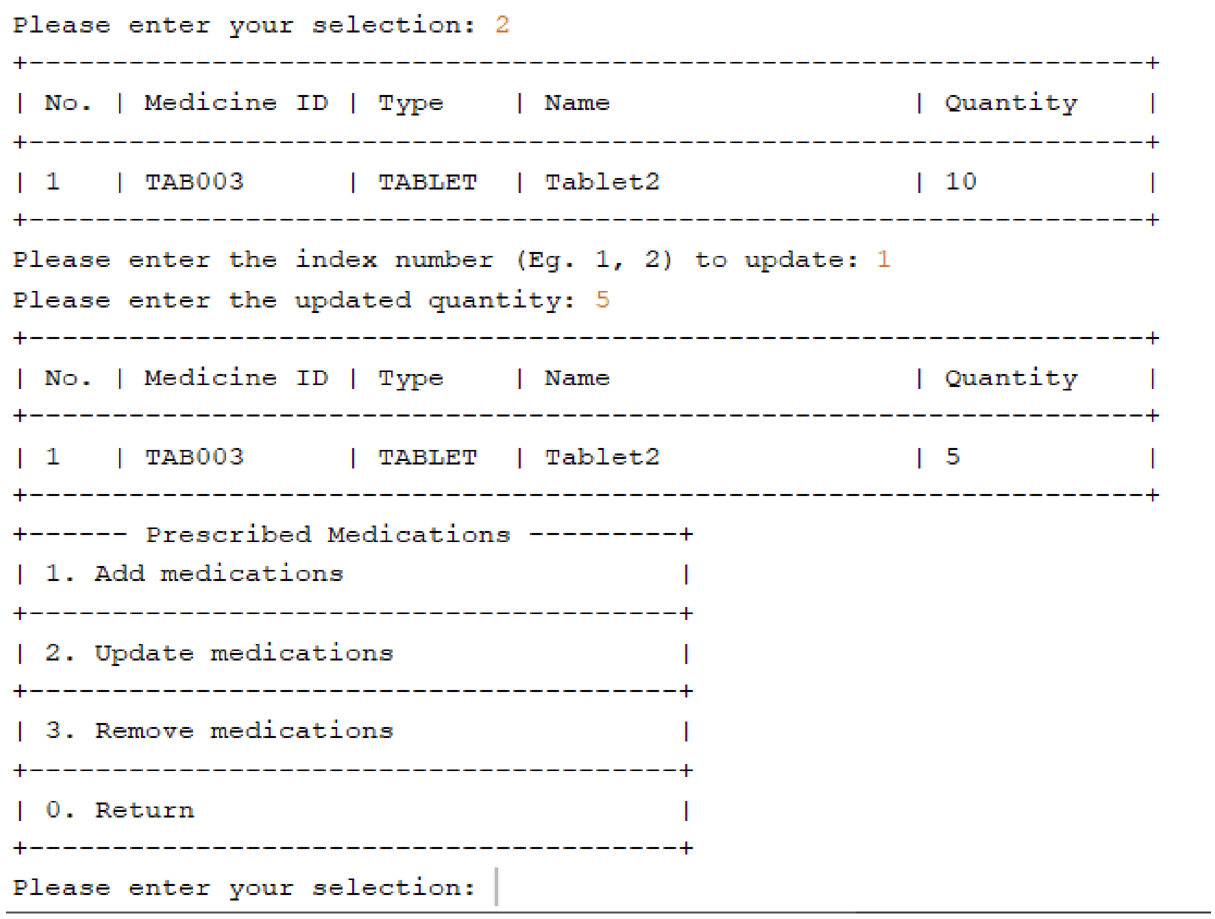
```
+-----------------------------------------------------------------------------+
| ID     | TYPE        | NAME                          | QUANTITY  | PRICE (RM)   |
+-----------------------------------------------------------------------------+
| OTH001 | OTHERS      | OTHER1                        | 999       | 12.00        |
+-----------------------------------------------------------------------------+
| OTH002 | OTHERS      | OTHER2                        | 999       | 22.00        |
+-----------------------------------------------------------------------------+
| OTH003 | OTHERS      | OTHER3                        | 999       | 13.00        |
+-----------------------------------------------------------------------------+
| OTH004 | OTHERS      | OTHER4                        | 999       | 15.00        |
+-----------------------------------------------------------------------------+
| TAB001 | TABLET      | PARACETAMOL                   | 999       | 11.00        |
+-----------------------------------------------------------------------------+
| TAB002 | TABLET      | TABLET1                       | 999       | 10.50        |
+-----------------------------------------------------------------------------+
| TAB003 | TABLET      | TABLET2                       | 999       | 20.50        |
+-----------------------------------------------------------------------------+
Please enter the medicine ID: Tab003
Please enter the quantity: 10
Are you confirm? (Y/N): y
Do you want to continue? (Y/N): n
```

When the user chooses to add the prescribed medication, a list of sorted medicine data will be shown, for the user's reference when entering the required medicine and quantity. The user may continue to add the other prescribed medicine if needed after confirming the current data.

```
+--------------------------------------------------------------------------+
| No. | Medicine ID | Type     | Name                        | Quantity     |
+--------------------------------------------------------------------------+
| 1   | TAB003      | TABLET   | Tablet2                     | 10           |
+--------------------------------------------------------------------------+
+------ Prescribed Medications ---------+
| 1. Add medications                    |
+---------------------------------------+
| 2. Update medications                 |
+---------------------------------------+
| 3. Remove medications                 |
+---------------------------------------+
| 0. Return                             |
+---------------------------------------+
Please enter your selection:
```

Once the user confirms the action, it will be stored in the sorted array list as sorting the data according to the medicine id for better reference. The list of selected medicines or prescribed medications will be shown on the main prescribed medications module page so the user can refer to it easily.

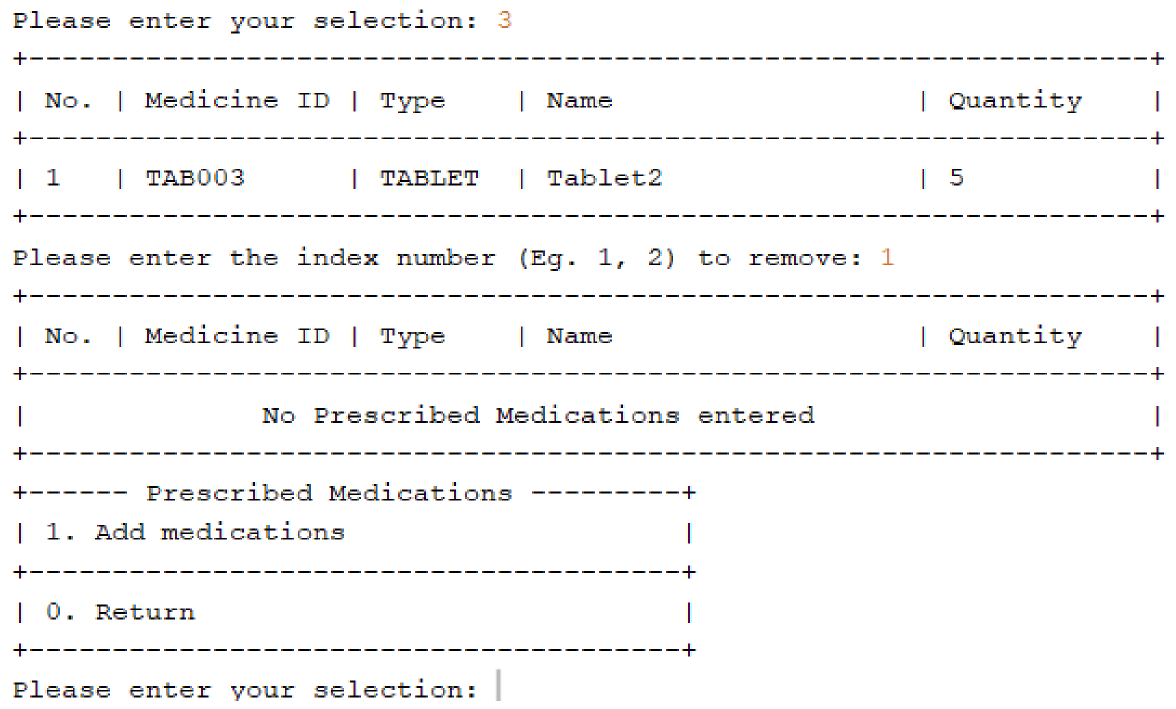
Update Prescribed medication



```
Please enter your selection: 2
+---------------------------------------------------------------------+
| No.  | Medicine ID | Type      | Name                     | Quantity      |
+---------------------------------------------------------------------+
| 1    | TAB003      | TABLET    | Tablet2                  | 10            |
+---------------------------------------------------------------------+
Please enter the index number (Eg. 1, 2) to update: 1
Please enter the updated quantity: 5
+---------------------------------------------------------------------+
| No.  | Medicine ID | Type      | Name                     | Quantity      |
+---------------------------------------------------------------------+
| 1    | TAB003      | TABLET    | Tablet2                  | 5             |
+---------------------------------------------------------------------+
+------ Prescribed Medications ---------+
| 1. Add medications                    |
+---------------------------------------+
| 2. Update medications                 |
+---------------------------------------+
| 3. Remove medications                 |
+---------------------------------------+
| 0. Return                             |
+---------------------------------------+
Please enter your selection:
```

The user may only update the quantity of the prescribed medication.

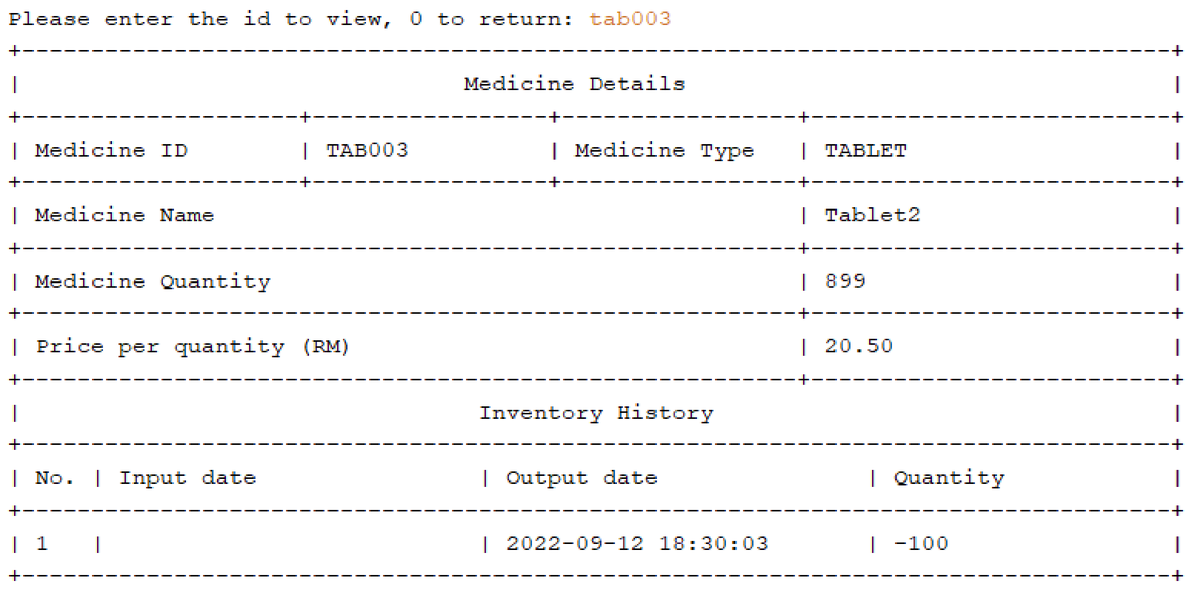
Remove Prescribed medication



```
Please enter your selection: 3
+----------------------------------------------------------------------+
| No. | Medicine ID | Type     | Name                     | Quantity    |
+----------------------------------------------------------------------+
| 1   | TAB003      | TABLET   | Tablet2                  | 5           |
+----------------------------------------------------------------------+
Please enter the index number (Eg. 1, 2) to remove: 1
+----------------------------------------------------------------------+
| No. | Medicine ID | Type     | Name                     | Quantity    |
+----------------------------------------------------------------------+
|              No Prescribed Medications entered                        |
+----------------------------------------------------------------------+
+------ Prescribed Medications ---------+
| 1. Add medications                    |
+---------------------------------------+
| 0. Return                             |
+---------------------------------------+
Please enter your selection:
```

The user may remove the prescribed medication by entering the index number of it. If the index number is valid, the medication entry will be removed from the list. Once the user confirms the prescribed medication, it will be entering 0 to proceed to the payment, and the quantity for the medicine at medication module will be updated accordingly, as well as the medicine inventory history will be stored.

Inventory history



```
Please enter the id to view, 0 to return: tab003
+--------------------------------------------------------------------------------------+
|                                Medicine Details                                      |
+---------------------+-----------------+-----------------+----------------------------+
| Medicine ID         | TAB003          | Medicine Type   | TABLET                     |
+---------------------+-----------------+-----------------+----------------------------+
| Medicine Name                                           | Tablet2                    |
+---------------------------------------------------------+----------------------------+
| Medicine Quantity                                       | 899                        |
+---------------------------------------------------------+----------------------------+
| Price per quantity (RM)                                 | 20.50                      |
+---------------------------------------------------------+----------------------------+
|                                Inventory History                                     |
+--------------------------------------------------------------------------------------+
| No. | Input date                | Output date               | Quantity               |
+--------------------------------------------------------------------------------------+
| 1   |                           | 2022-09-12 18:30:03       | -100                   |
+--------------------------------------------------------------------------------------+
```

Once the user has confirmed with the prescribed medication and made payment on it, the information will be updated and it will be stored into the inventory history for checking reference using a linked list so there is no restriction on the list size.