FACULTY OF COMPUTING

SEMESTER 2

2024/2025

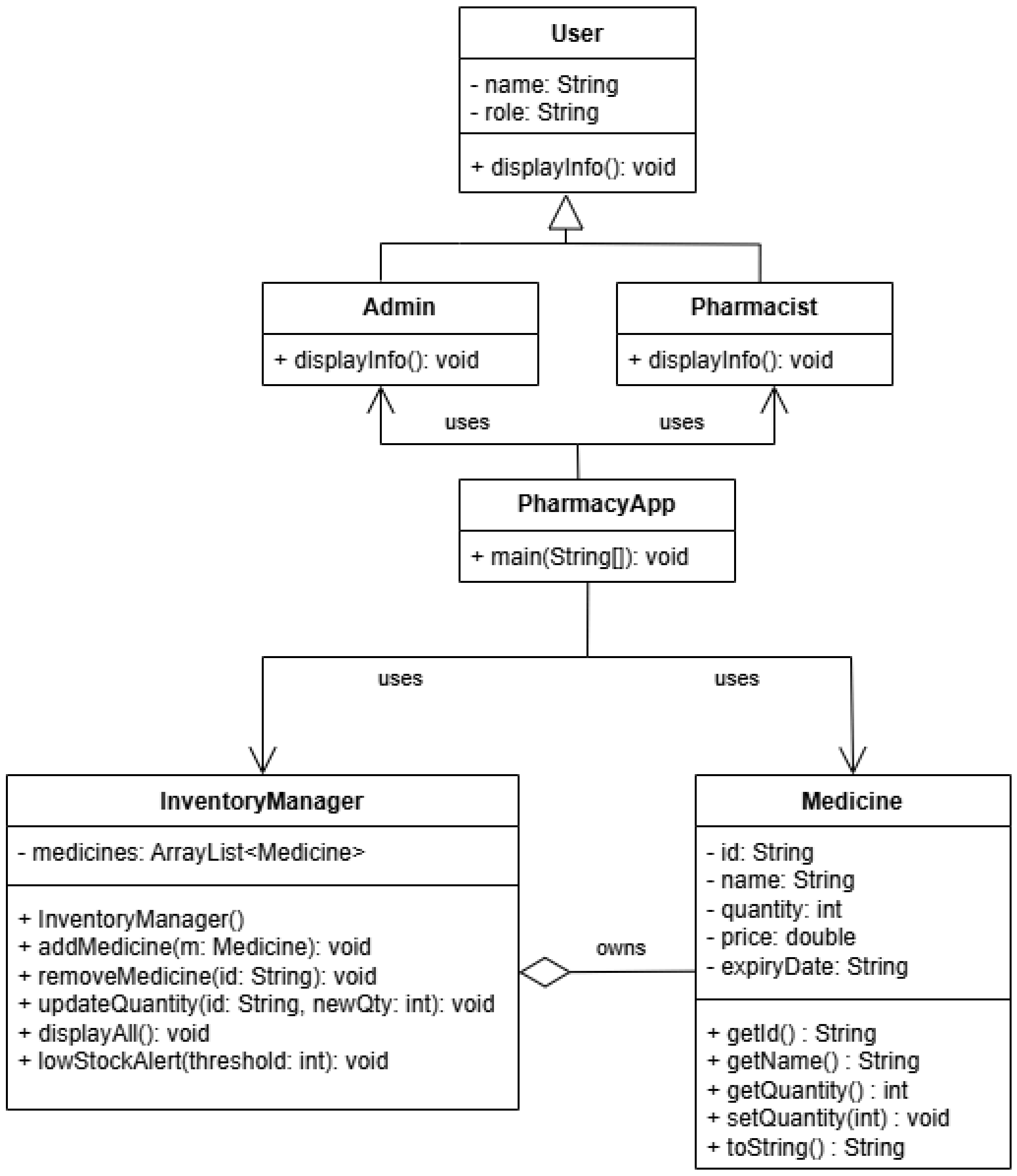
SECJ2154-03 OBJECT ORIENTED PROGRAMMING

LECTURER: DR. MUHAMMAD KHATIBSYARBINI

TOPIC: PHARMACY INVENTORY SYSTEM (MINI PROJECT)

NAME	**MATRIC NUMBER**
DAYANG FARAH FARZANA BINTI ABANG IDHAM	A23CS0071
FARRA NURZAHIN BINTI ZAHARIL ANUAR	A23CS0079
SAFIYA NURSYAHADAH BINTI MASNOOR	A23CS0176

UML DIAGRAM OF PHARMACY INVENTORY SYSTEM



Classes

1. User
   The User class represents anyone who can log into and interact with the pharmacy system. Its responsibility is to hold the basic identity information of who the person is and what general role they have. The class also provides a unified way of presenting that identifies wherever it's needed.

2. Admin
   The Admin class models a high-level system administrator. Its general use is to capture all the special privileges and responsibilities that come with managing the application as a whole such as configuring settings, overseeing user accounts or running reports.

3. Pharmacist
   The Pharmacist class embodies the professional who handles the pharmacy's core operations such as filling prescriptions, dispensing medications and keeping track of stock on the floor.

4. Medicine
   The Medicine class stands for a single product in the pharmacy's catalog. It's responsible for representing every key detail about the product such as its unique identity, how many units are available, what it costs and when it expires.

5. InventoryManager
   The InventoryManager class serves as the central engine for all stock-related activities. Its general use is to maintain the complete list of products, handle adding or removing items, update stock levels and detect low stock medicines.

6. PharmacyApp
   The PharmacyApp class is the glue that holds everything together. As the entry point to the application, its responsibility is to manage the flow of interaction by prompting users for input, decide which operations to invoke and display results.

Relationship between Classes

1. **Inheritance and Polymorphism**
   Admin, Pharmacist → User
   - Both Admin and Pharmacist inherit shared data (name, role) and behavipur (displayInfo()), but each can refine or change how that behaviour works.
   - When PharmacyApp asks a User to "display info," it does not need to know whether that User is an Admin or a Pharmacist, the correct version of displayInfo runs automatically. This makes it easy to add new user types in the future without changing the main program logic

2. **Association**

   PharmacyApp → Admin

   - The application's main driver creates and call on an Admin object to display administrative info and potentially invoke admin-specific workflows

   PharmacyApp → Pharmacist

   - PharmacyApp manifest a Pharmacist to show pharmacist-specific identity and to route dispensing or stock-check operations through that role

   PharmacyApp → InventoryManager

   - PharmacyApp relies on InventoryManager to perform every inventory-related action such as adding, removing, updating and listing medicines

   PharmacyApp → Medicine

   - When gathering input or showing results, PharmacyApp passes Medicine data into and out of InventoryManagerr methods, using Medicine as data carrier

3. **Aggregation**

   InventoryManager → Medicine

   - InventoryManager keeps a list of Medicine objects so it can look them up, update them or show them to users. Those Medicine objects can exist on their own and the manager simply groups them together to make stock control easy

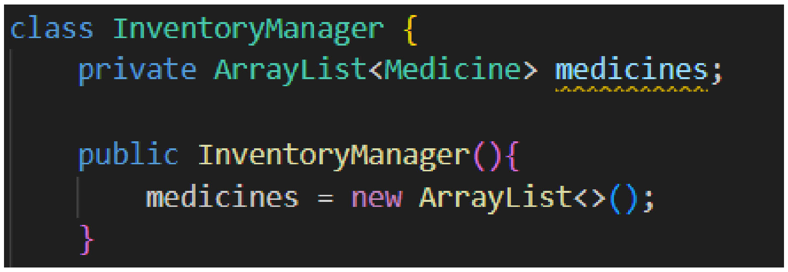
CLASS IMPLEMENTATION IN CODE

This project is about the Pharmacy Inventory System. The system allows pharmacy staff (Admins and Pharmacists) to manage medicine inventory, including adding new medicines, updating quantities, removing expired/unused entries and checking low-stock items. The project includes the implementation of inheritance, polymorphism, class relationships(association, aggregation), ArrayList and exception handling.

Chapter-wise implementation:

CHAPTER 5: Vector and ArrayLists

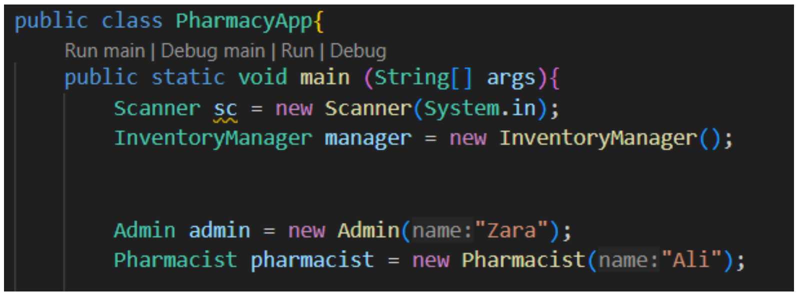
ArrayList<Medicine> is used in the InventoryManager class to dynamically store and manage medicine records.



```
class InventoryManager {
    private ArrayList<Medicine> medicines;

    public InventoryManager(){
        medicines = new ArrayList<>();
    }
```

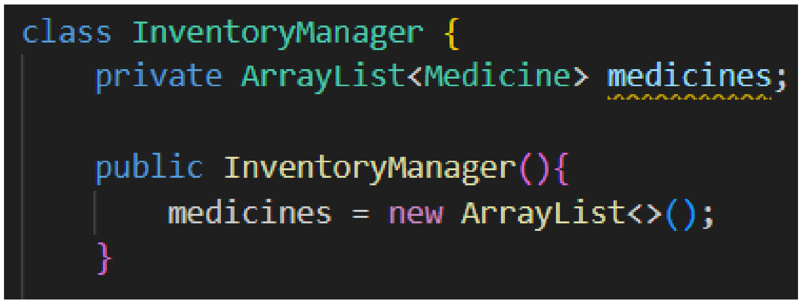
CHAPTER 6: Class Relationships

Association: PharmacyApp has associations with Admin and Pharmacist objects.



```
public class PharmacyApp{
    Run main | Debug main | Run | Debug
    public static void main (String[] args){
        Scanner sc = new Scanner(System.in);
        InventoryManager manager = new InventoryManager();


        Admin admin = new Admin(name:"Zara");
        Pharmacist pharmacist = new Pharmacist(name:"Ali");
```

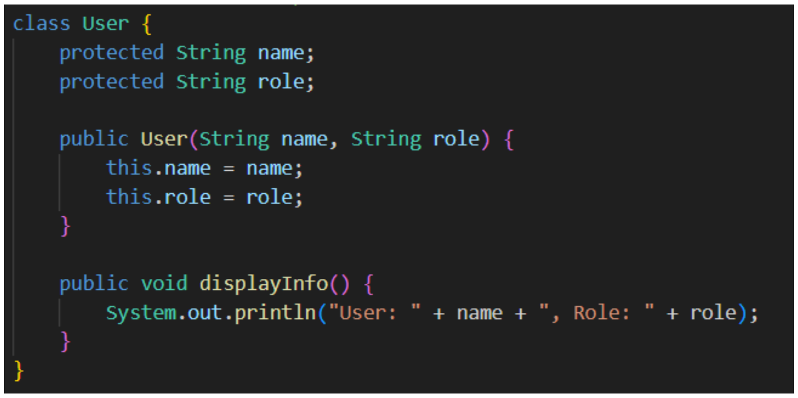
Aggregation: InventoryManager "has-a" list of Medicine objects.



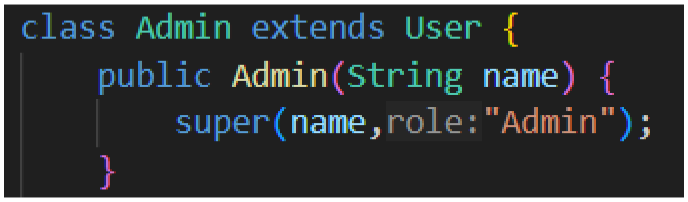
```
class InventoryManager {
    private ArrayList<Medicine> medicines;

    public InventoryManager(){
        medicines = new ArrayList<>();
    }
```

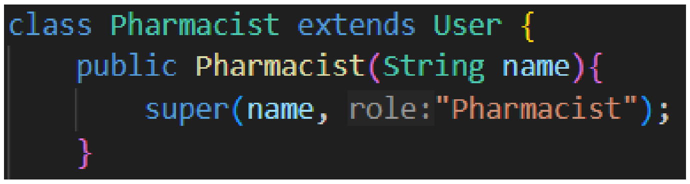
CHAPTER 7: Inheritance

User is a superclass


```java
class User {
    protected String name;
    protected String role;

    public User(String name, String role) {
        this.name = name;
        this.role = role;
    }

    public void displayInfo() {
        System.out.println("User: " + name + ", Role: " + role);
    }
}
```

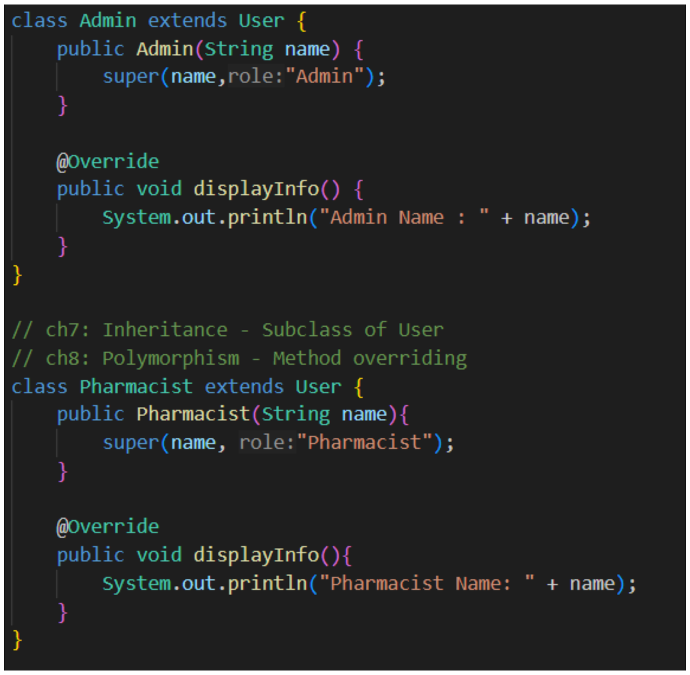
Admin and Pharmacist inherit from User using the extends keyword.


```java
class Admin extends User {
    public Admin(String name) {
        super(name,role:"Admin");
    }
```


```java
class Pharmacist extends User {
    public Pharmacist(String name){
        super(name, role:"Pharmacist");
    }
```

CHAPTER 8: Polymorphism

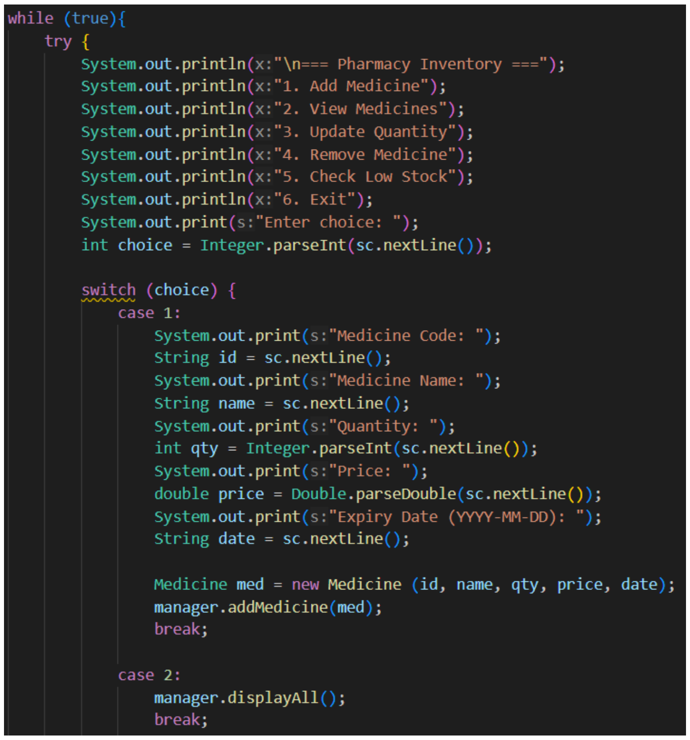
Method Overriding: Admin and Pharmacist override the displayInfo() the displayInfo() method from User class. Polymorphic behavior is shown when calling displayInfo() on User-type references.



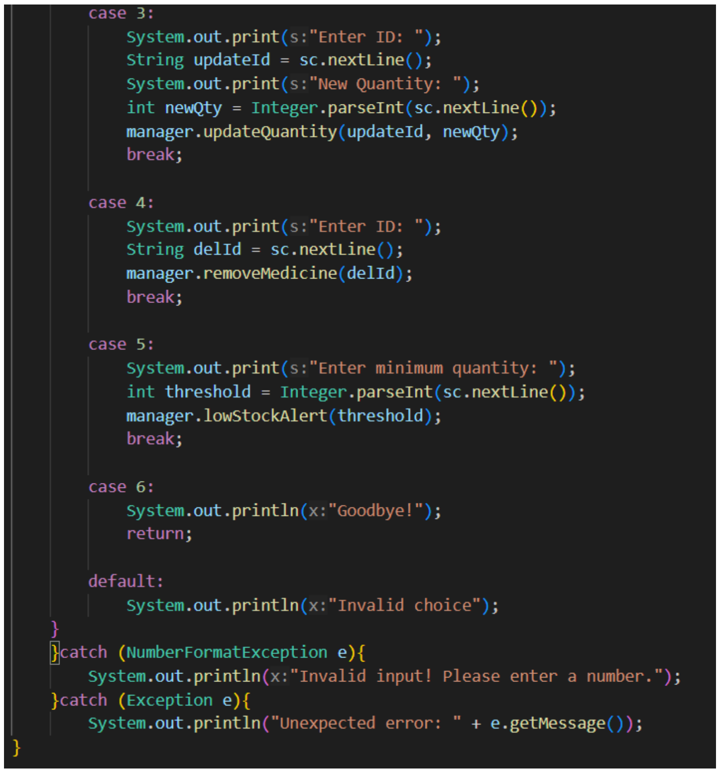
```java
class Admin extends User {
    public Admin(String name) {
        super(name,role:"Admin");
    }

    @Override
    public void displayInfo() {
        System.out.println("Admin Name : " + name);
    }
}

// ch7: Inheritance - Subclass of User
// ch8: Polymorphism - Method overriding
class Pharmacist extends User {
    public Pharmacist(String name){
        super(name, role:"Pharmacist");
    }

    @Override
    public void displayInfo(){
        System.out.println("Pharmacist Name: " + name);
    }
}
```

CHAPTER 9: Exception Handling

try-catch blocks are used in the main application loop to handle invalid input and unexpected runtime errors.

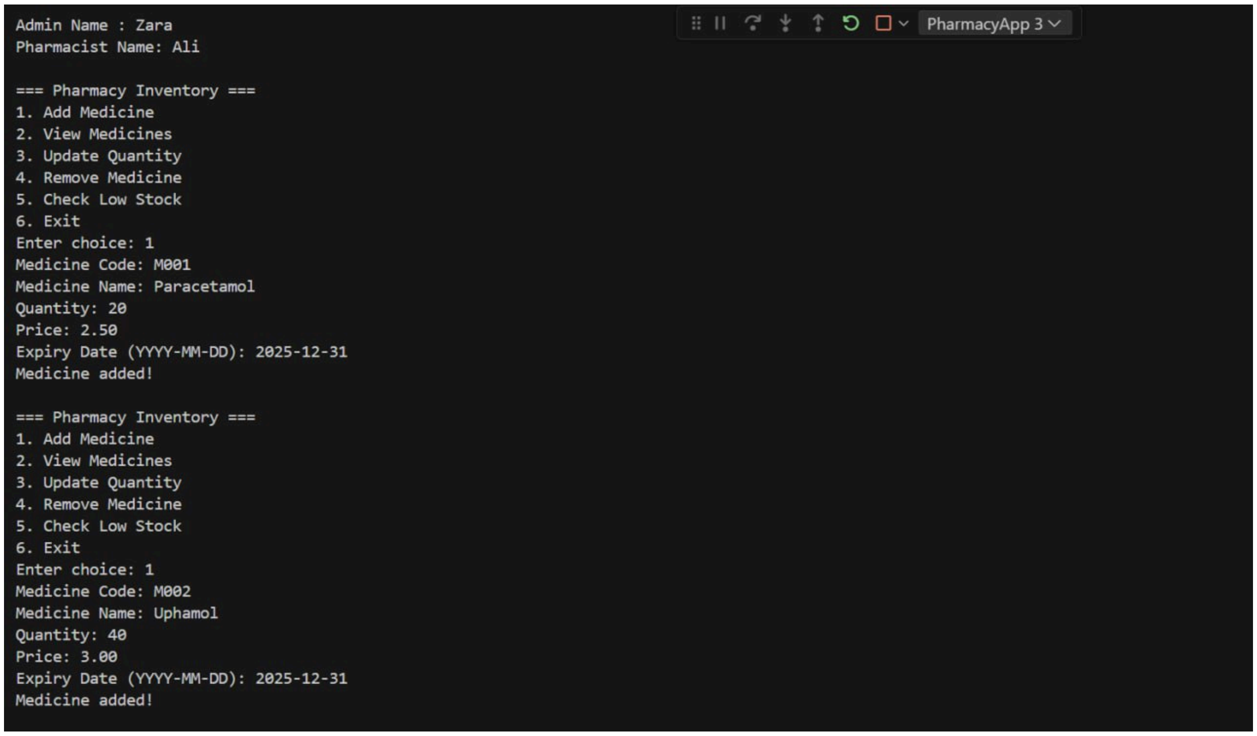


```java
while (true){
    try {
        System.out.println(x:"\n=== Pharmacy Inventory ===");
        System.out.println(x:"1. Add Medicine");
        System.out.println(x:"2. View Medicines");
        System.out.println(x:"3. Update Quantity");
        System.out.println(x:"4. Remove Medicine");
        System.out.println(x:"5. Check Low Stock");
        System.out.println(x:"6. Exit");
        System.out.print(s:"Enter choice: ");
        int choice = Integer.parseInt(sc.nextLine());

        switch (choice) {
            case 1:
                System.out.print(s:"Medicine Code: ");
                String id = sc.nextLine();
                System.out.print(s:"Medicine Name: ");
                String name = sc.nextLine();
                System.out.print(s:"Quantity: ");
                int qty = Integer.parseInt(sc.nextLine());
                System.out.print(s:"Price: ");
                double price = Double.parseDouble(sc.nextLine());
                System.out.print(s:"Expiry Date (YYYY-MM-DD): ");
                String date = sc.nextLine();

                Medicine med = new Medicine (id, name, qty, price, date);
                manager.addMedicine(med);
                break;

            case 2:
                manager.displayAll();
                break;
```

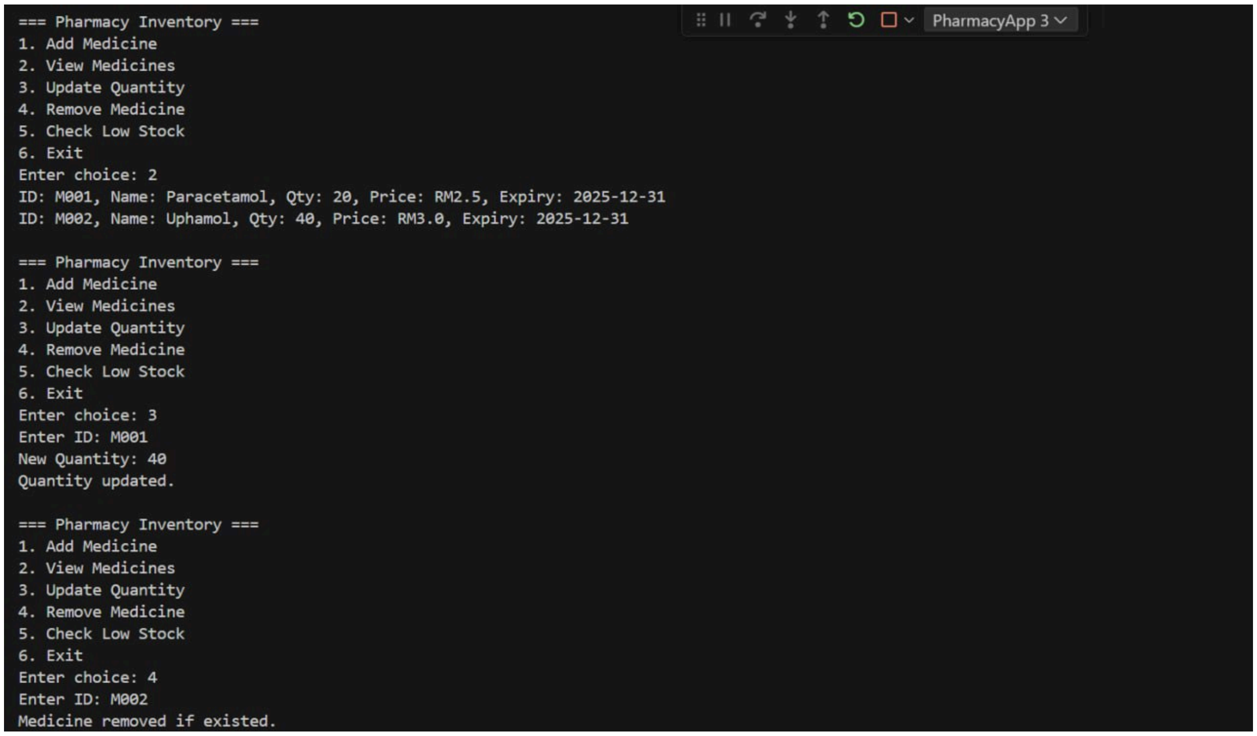


```java
                case 3:
                    System.out.print(s:"Enter ID: ");
                    String updateId = sc.nextLine();
                    System.out.print(s:"New Quantity: ");
                    int newQty = Integer.parseInt(sc.nextLine());
                    manager.updateQuantity(updateId, newQty);
                    break;

                case 4:
                    System.out.print(s:"Enter ID: ");
                    String delId = sc.nextLine();
                    manager.removeMedicine(delId);
                    break;

                case 5:
                    System.out.print(s:"Enter minimum quantity: ");
                    int threshold = Integer.parseInt(sc.nextLine());
                    manager.lowStockAlert(threshold);
                    break;

                case 6:
                    System.out.println(x:"Goodbye!");
                    return;

                default:
                    System.out.println(x:"Invalid choice");
            }
        }catch (NumberFormatException e){
            System.out.println(x:"Invalid input! Please enter a number.");
        }catch (Exception e){
            System.out.println("Unexpected error: " + e.getMessage());
}
```

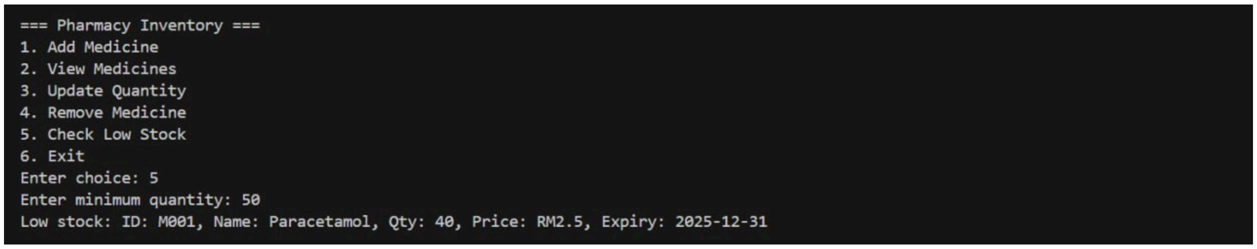
SAMPLE OUTPUT



```
Admin Name : Zara
Pharmacist Name: Ali

=== Pharmacy Inventory ===
1. Add Medicine
2. View Medicines
3. Update Quantity
4. Remove Medicine
5. Check Low Stock
6. Exit
Enter choice: 1
Medicine Code: M001
Medicine Name: Paracetamol
Quantity: 20
Price: 2.50
Expiry Date (YYYY-MM-DD): 2025-12-31
Medicine added!

=== Pharmacy Inventory ===
1. Add Medicine
2. View Medicines
3. Update Quantity
4. Remove Medicine
5. Check Low Stock
6. Exit
Enter choice: 1
Medicine Code: M002
Medicine Name: Uphamol
Quantity: 40
Price: 3.00
Expiry Date (YYYY-MM-DD): 2025-12-31
Medicine added!
```



```
=== Pharmacy Inventory ===
1. Add Medicine
2. View Medicines
3. Update Quantity
4. Remove Medicine
5. Check Low Stock
6. Exit
Enter choice: 2
ID: M001, Name: Paracetamol, Qty: 20, Price: RM2.5, Expiry: 2025-12-31
ID: M002, Name: Uphamol, Qty: 40, Price: RM3.0, Expiry: 2025-12-31

=== Pharmacy Inventory ===
1. Add Medicine
2. View Medicines
3. Update Quantity
4. Remove Medicine
5. Check Low Stock
6. Exit
Enter choice: 3
Enter ID: M001
New Quantity: 40
Quantity updated.

=== Pharmacy Inventory ===
1. Add Medicine
2. View Medicines
3. Update Quantity
4. Remove Medicine
5. Check Low Stock
6. Exit
Enter choice: 4
Enter ID: M002
Medicine removed if existed.
```



```
=== Pharmacy Inventory ===
1. Add Medicine
2. View Medicines
3. Update Quantity
4. Remove Medicine
5. Check Low Stock
6. Exit
Enter choice: 5
Enter minimum quantity: 50
Low stock: ID: M001, Name: Paracetamol, Qty: 40, Price: RM2.5, Expiry: 2025-12-31
```