Java Sprint 1 SD12 - Evan, Nick, & Vanessa

User Documentation

The Medication Tracking System is a Java-based application that helps medical professionals manage patients, doctors, medications, and prescriptions.

Main.java

The entry point of the application; creates and starts the medication system

MedicationSystem.java

The core system class that manages all operations. Maintains lists of patients, doctors, medications, and prescriptions Provides functionality for:

- ➔ Searching records
- ➔ Managing doctors/patients/medications
- ➔ Handling prescriptions
- ➔ Generating reports
- ➔ Checking expired medications
- ➔ Restocking inventory

MenuSystem.java

Handles the user interface for the management system. Provides menus for:

- ➔ Main system navigation
- ➔ Search operations
- ➔ Record modification
- ➔ Report printing

Person.java

Super class for Doctor and Patient. Contains the following attributes:

- ➔ ID
- ➔ Name
- ➔ Age

- ➔ Phone number

Doctor.java

An extension of Person.java, representing a doctor in the management system. It handles patient assignment/removal, and manages the following doctor information:

- ➔ ID
- ➔ Name
- ➔ Specialization
- ➔ List of assigned patients

Patient.java

An extension of Person.java, representing a patient in the management system. It manages the following patient information:

- ➔ ID
- ➔ First/last name
- ➔ List of medications
- ➔ List of prescriptions

Medication.java

Handles medication information in the management system while managing inventory. It manages the following medicine information:

- ➔ ID
- ➔ Name
- ➔ Dose
- ➔ Stock quantity
- ➔ Expiry date

Prescription.java

Links doctors, patients, and medications. Handles prescription information in the management system and tracks the following details:

- ➔ Doctor who prescribed
- ➔ Patient receiving medication
- ➔ Prescribed medication
- ➔ Expiry date

Class Diagram

```
classDiagram
    Person <|-- Doctor
    Person <|-- Patient
    MedicationSystem -- MenuSystem
```

```
MedicationSystem -- Doctor
MedicationSystem -- Patient
MedicationSystem -- Medication
MedicationSystem -- Prescription
Doctor -- Patient : manages
Patient -- Medication : takes
Patient -- Prescription : has
Doctor -- Prescription : issues
Medication -- Prescription : prescribed in

class Person {
    -int id
    -String name
    -int age
    -String phoneNumber
}

class Doctor {
    -int id
    -String name
    -String specialization
    -List~Patient~ patients
}

class Patient {
    -int id
    -String firstName
    -String lastName
    -List~Medication~ medications
    -List~Prescription~ prescriptions
}

class Medication {
    -int id
    -String name
    -String dose
    -int inStock
    -Date expiryDate
}

class Prescription {
    -int id
    -String doctor
    -String patient
    -String medication
    -Date expiryDate
}

class MedicationSystem {
    -List~Patient~ patients
    -List~Medication~ medications
    -List~Doctor~ doctors
    -List~Prescription~ prescriptions
}

class MenuSystem {
    +showMainMenu()
    +showSearchMenu()
    +showModifyMenu()
}
```

Development Documentation

Directory Structure

```
s3_java_sprint_1_evan_nick_vanessa/
    ├── Doctor.java
    ├── Medication.java
```

```
    ├── MedicationSystem.java
    ├── MenuSystem.java
    ├── Patient.java
    ├── Person.java
    ├── Prescription.java
    ├── TestMedicationSystem.java
    └── README.md
```

Build Process

Prerequisites

- ➔ JDK 17 or higher
- ➔ java.util.List
- ➔ java.util.ArrayList

Compilation

The project can be run directly in VS Code using the 'run' button in VS Code.

VS Code's Java extension handles compilation automatically.

For manual compilation in terminal:

1. Navigate to the project root directory
2. Compile the source files:

```
javac -d ./bin src/main/java/*.java
```

3. Run the application:

```
java -cp ./bin MedicationSystem
```

Theoretical Database Design

Entity Relationship Diagram

```
erDiagram
    DOCTOR ||--o{ PATIENT : treats
    DOCTOR ||--o{ PRESCRIPTION : writes
    PATIENT ||--o{ PRESCRIPTION : receives
    MEDICATION ||--o{ PRESCRIPTION : included_in

    DOCTOR {
        int doctor_id PK
        string first_name
        string last_name
        string specialization
        string phone_number
    }

    PATIENT {
        int patient_id PK
        string first_name
        string last_name
        string phone_number
    }

    MEDICATION {
```

```
    int medication_id PK
    string name
    string dosage
    int stock_quantity
    date expiry_date
    decimal price
}

PRESCRIPTION {
    int prescription_id PK
    int doctor_id FK
    int patient_id FK
    int medication_id FK
    date prescribed_date
    date expiry_date
    string dosage_instructions
}
```

Source Code Access

1. Clone the repo:

```
https://github.com/npkav/s3_java_sprint_1_evan_nick_vanessa.git
```

2. Create a new branch for development and switch:

```
git branch your-feature-name
git checkout your-feature-name
```

3. After making changes:

```
git add .
git commit -m "description of changes"
git push origin feature/your-feature-name
```

Deployment Documentation

1) Search for Drugs, Patients, and Doctors

Upon inputting '1' into the console, you will be given the choice between 3 options:

1. Search for Drugs
2. Search for Patients
3. Search for Doctors

Depending on your choice, you will either be asked to provide a medication ID, patient ID, or doctor ID respectively. Upon inputting this respective ID, the program will search within its database to track down what is associated with that ID.

2) Add/Remove Doctor

Upon inputting '2' into the console, you will be given the choice between 5 options:

1. Add Doctor
2. Remove Doctor
3. Edit Doctor

4. Add Patient to Doctor
5. Remove Patient from Doctor

When adding a doctor, you will be asked for the doctor's ID, name, and specialization. When removing a doctor, you will only be asked for a pre-existing doctor's ID, and the program will remove any data associated with that doctor's ID. When editing a doctor, you will be asked for a pre-existing doctor's ID - once provided, you will be given the ability to change the doctor's name and specialization. When adding a patient to a doctor, you will be asked for the ID of a doctor first, and then the ID of a patient. Once both are given, said doctor will be assigned to said patient. If you wish to remove a patient from a doctor, you will be asked once again for the ID of the doctor first, and then the ID of the patient you wish to remove. Bear in mind that removing a doctor will also remove any associated prescriptions as well.

3) Add/Remove Patient

Upon inputting '3' into the console, you will be given the choice between 3 options:

1. Add Patient
2. Remove Patient
3. Edit Patient

When adding a patient, you will be asked for the patient's ID, first name, and last name. When removing a patient, you will only be asked for a pre-existing patient's ID, and the program will remove any data associated with that patient's ID. When editing a patient, you will be asked for a pre-existing patient's ID - once provided, you will be given the ability to change the patient's first name, and last name. Bear in mind that removing a patient will also remove any associated prescriptions as well.

4) Add/Remove Medication

Upon inputting '4' into the console, you will be given the choice between 3 options:

1. Add Medication
2. Remove Medication
3. Edit Medication

When adding a medication, you will be asked for the medication's ID, name, dosage, and the amount in stock. Then, an expiry date will be generated to coincide with this medication. When removing a medication, you will only be asked for a pre-existing medication's ID, and the program will remove any data associated with that medication's ID. When editing a medication, you will be asked for a pre-existing medication's ID - once provided, you will be given the ability to change the name and dosage of the medication in question. Bear in mind that removing a medication will also remove any associated prescriptions as well.

5) Create Prescription

Upon inputting ‘5’ into the console, you will be asked to create a prescription’s ID. Then, one by one, you will be given a list of every available doctor, patient, and medication. Upon entering a valid ID for each of the three options, a prescription will be successfully created and you will be provided with the prescription’s ID, the doctor’s name, the patient’s name, the medication’s name, and the expiry date of the prescription, which is one year after the prescription’s creation.

6) Print All Prescriptions by a Specific Doctor

Upon inputting ‘6’ into the console, you will be given a list of every available doctor. Upon entering a valid ID, you will be given a list of all the prescriptions associated with said doctor.

7) Restock Medications

Upon inputting ‘7’ into the console, the program will look over every single medication that has been registered. Then, it will restock all registered medications, and inform you of the current stock of each medication.

8) Check for Expired Medications

Upon inputting ‘8’ into the console, the program will look over each and every medication that has been registered, in order to check and see which ones are expired. If it notices an expired medication, it will inform you which medication has been expired and when it expired.

9) Generate Report

Upon inputting ‘9’ into the console, the program will generate a list of all the doctors, patients, and medications that have been registered, and provide said list to you. This will only keep track of doctors, patients, and medications that have been added and remain added by the user, it will not keep track of any of the three that have been removed by the user. The program will also provide the following information, depending on which of the three it has listed:

1. Doctor: Doctor ID, Doctor Name, Specialization
2. Patient: Patient ID, Patient First Name + Last Name, Patient Medications, Patient Prescriptions.

3. Medication: Medication ID, Medication Name, Medication Dose, Medication Quantity in Stock, Medication Expiry Date

0) Exit

Upon inputting ‘9’ into the console, the program will close.