MedicationTrackingSystem

Software Manual

Author

Kepich, Eugene

Keyin College

Software Development

Group 12

Table of Contents

1 Introduction

1.1 General Overview

The Medication Tracking System is a flexible and scalable Java-based system designed to manage patients, doctors, medications, and prescriptions efficiently. It is built in a modular and object-oriented manner, ensuring that it can be easily adapted for different use cases, from small pharmacies to large hospital systems.

At its core, the system revolves around the MedicationTrackingSystem class, which provides all the necessary methods for managing entities such as:

1. Patients (storing names, ages, contact details, and assigned medications and prescriptions)
2. Doctors (storing names, specializations, and assigned patients)
3. Medications (tracking stock levels, expiry dates, and dosages)
4. Prescriptions (linking doctors, patients, and medications)

1.2 Key Features

- Complete Management System: Supports adding, editing, searching, and deleting patients, doctors, and medications.
- Prescription Tracking: Allows assigning medications to patients via prescriptions while keeping track of expiration dates.
- Data Storage Flexibility: The system is independent of input and output methods, meaning it can work with:
- CSV Files (as seen in MedicationTrackingSystemTest)
- Databases (easily adaptable)
- API Interfaces (could be extended for web-based applications)
- Search and Reporting: Allows fuzzy searches for patients, doctors, and medications while generating structured reports.
- Separation of Concerns: Business logic is completely decoupled from user interface logic, allowing integration into any front-end or terminal-based system.

2 System Structure

2.1 Core Class: MedicationTrackingSystem

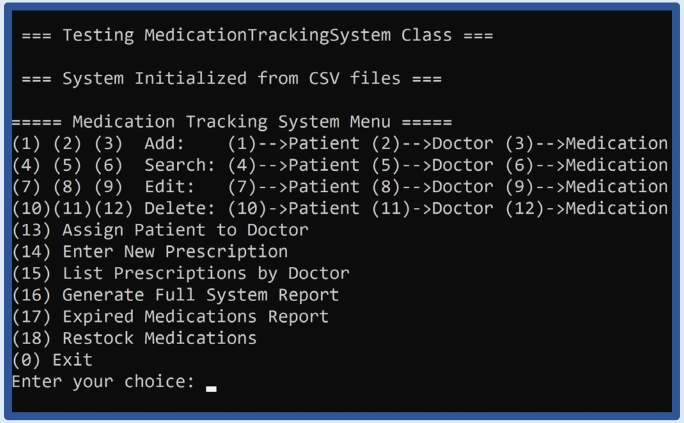
The MedicationTrackingSystem class is the foundation of the system. It acts as a data manager, containing all the necessary methods to manipulate and retrieve information about:

- Patients
- Doctors
- Medications
- Prescriptions

Crucially, this class is designed to be completely independent of any specific input or output format. It does not rely on terminal-based interactions, allowing it to be embedded into any graphical user interface (GUI), web application, or API-driven service.

2.2 Example Wrapper

To demonstrate how a menu-based interface can be wrapped around MedicationTrackingSystem, we created the MedicationTrackingSystemExampleMenu class.



```
 === Testing MedicationTrackingSystem Class ===

 === System Initialized from CSV files ===

===== Medication Tracking System Menu =====
(1) (2) (3)  Add:    (1)-->Patient (2)-->Doctor (3)-->Medication
(4) (5) (6)  Search: (4)-->Patient (5)-->Doctor (6)-->Medication
(7) (8) (9)  Edit:   (7)-->Patient (8)-->Doctor (9)-->Medication
(10)(11)(12) Delete: (10)->Patient (11)->Doctor (12)->Medication
(13) Assign Patient to Doctor
(14) Enter New Prescription
(15) List Prescriptions by Doctor
(16) Generate Full System Report
(17) Expired Medications Report
(18) Restock Medications
(0) Exit
Enter your choice: _
```

This is a simple console-based wrapper that:

- Provides a menu-driven interface for user interaction.
- Accepts user input through the terminal.
- Calls the appropriate methods in MedicationTrackingSystem.

This example wrapper is just one of many possible implementations. The system can be easily extended into a GUI-based pharmacy application, an automated API, or even a mobile app.

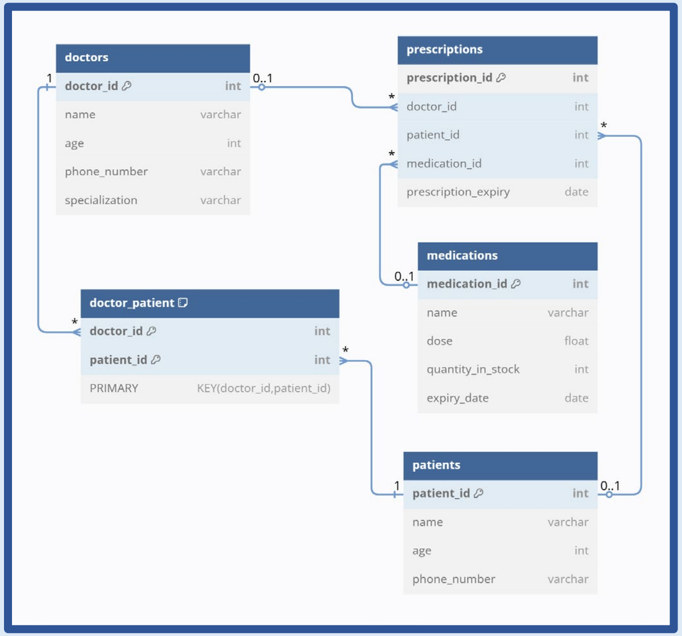
2.3 Initialization and Data Loading

The MedicationTrackingSystemTest class is used to initialize the system with data from CSV files. However, the data source is not limited to CSV—this setup is purely for demonstration.

In a real-world application, the system could be initialized with:

- Database connections (MySQL, PostgreSQL, MongoDB, see figure below for suggested database structure, prescriptions table should contain snapshots of prescriptions, data cannot change)
- REST APIs (to retrieve patient or prescription data from external services)
- Hardcoded JSON/XML data (for embedded systems)

This approach ensures that the Medication Tracking System can be integrated into any medical or pharmacy software, regardless of storage method.



2.4 How to Use the Medication Tracking System

To use the Medication Tracking System, all modules must be compiled together since they interact with each other. The system is implemented in Java, and compilation can be done using the javac command.

```
javac Person.java Patient.java Medication.java
Doctor.java FileUtils.java MedUtils.java
Prescription.java MedicationTrackingSystem.java
MedicationTrackingSystemTest.java
MedicationTrackingSystemExampleMenu.java
```

Once compiled, the system can be executed by running:

```
java MedicationTrackingSystemTest
```

This will initialize the system using data from CSV files and launch the example menu wrapper (MedicationTrackingSystemExampleMenu) for user interaction.

Since MedicationTrackingSystem is designed to be independent of terminal input/output, it can be integrated into graphical user interfaces (GUIs), connected to databases instead of CSV files, used within web services or REST APIs.

By following this compilation and execution process, developers can easily modify, extend, and integrate the system into larger pharmacy or healthcare applications.

2.5 Summary

The Medication Tracking System is designed to be a robust, independent, and flexible system that can manage a wide range of medical tracking needs. While we have implemented a terminal-based menu (MedicationTrackingSystemExampleMenu) and CSV-based initialization (MedicationTrackingSystemTest) as examples, the core class MedicationTrackingSystem is completely modular, making it suitable for applications of any complexity, including database-driven hospital systems, mobile pharmacy applications, or cloud-based healthcare platforms.

3 Class Documentation (Javadocs)

3.1 Person Superclass

```
Person
-----------------------------------------------
# id: int
- name: string
- age: int
- phone_number: String
-----------------------------------------------
+ Person(id: int, name: String, age: int,
phone_number: String)
-----------------------------------------------
+ getId(): int
+ getName(): String
+ getAge(): int
+ getPhoneNumber(): String
+ setName(name: String): boolean
+ setAge(age: int): boolean
+ setPhoneNumber(phone_number: String): boolean
+ updateCommonFields(name: String, age: int,
phoneNumber: String): boolean
+ toString(): String
```

- # id: int (final) - Unique identifier (final, cannot be changed).
- - name: string - Stores the person's name (validated).
- - age: int - Stores the person's age (validated between 0-150).
- - phone_number: String - Stores the person's phone number (validated for correct format 10-digit format).
- + Person(id: int, name: String, age: int, phone_number: String) - Creates a Person object with validated attributes (defaults to preset values if invalid).
- + getId(): int - Returns the unique ID of the person.
- + getName(): String - Returns the name of the person.
- + getAge(): int - Returns the age of the person.
- + getPhoneNumber(): String - Returns the phone number of the person.
- + setName(name: String): boolean - Updates the name if validated (returns true if valid and updated, false otherwise).
- + setAge(age: int): boolean - Updates the age (returns true if valid, false otherwise).
- + setPhoneNumber(phone_number: String): boolean - Updates the phone number (returns true if valid, false otherwise).

- + updateCommonFields(name: String, age: int, phoneNumber: String): boolean - Updates multiple fields at once (keeps old values if null or -1).
- + toString(): String - Returns a formatted string representation of the Person object.

3.2 Doctor Class

```
Doctor
extends Person
---------------------------------------------------
- specialization: String
- patients: List<Patient>
---------------------------------------------------
+ Doctor(id: int, name: String, age: int,
phone_number: String, specialization: String)
---------------------------------------------------
+ getSpecialization(): String
+ getPatients(): List<Patient>
+ getPatientIds(): List<Integer>
+ setSpecialization(specialization: String):
boolean
+ updateFields(name: String, age: int,
phoneNumber: String, specialization: String):
boolean
+ addPatient(patient: Patient): boolean
+ removePatient(patient: Patient): boolean
+ removePatientById(patient_id: int): boolean
+ removeAllPatients(): void
```

```
+ toString(): String
```

- - specialization: String - Stores the doctor's medical specialty.
- - patients: List<Patient> - A list of assigned patients.
- + Doctor(id: int, name: String, age: int, phone_number: String, specialization: String) - Creates a doctor with validated specialization and an empty patient list.
- + getSpecialization(): String - Returns the doctor's specialization.
- + getPatients(): List<Patient> - Returns a list of assigned Patient objects.
- + getPatientIds(): List<Integer> - Returns a list of assigned patient IDs.
- + setSpecialization(specialization: String): boolean - Updates specialization (if valid).
- + updateFields(name: String, age: int, phoneNumber: String, specialization: String): boolean - Updates name, age, phone, and specialization.
- + addPatient(patient: Patient): boolean - Assigns a patient to this doctor.
- + removePatient(patient: Patient): boolean - Removes a patient by object reference.
- + removePatientById(patient_id: int): boolean - removePatientById():
- + removeAllPatients(): void - Clears the patient list.
- + toString(): String - Returns a formatted string representation of the doctor

3.3 Patient Class

```
Patient
extends Person
---
- medications: List<Medication>
- prescriptions: List<Prescription>
---
+ Patient(id: int, name: String, age: int,
phone_number: String)
---
+ getMedications(): List<Medication>
+ getPrescriptions(): List<Prescription>
+ getMedicationIds(): List<Integer>
+ getPrescriptionIds(): List<Integer>
+ addPrescription(prescription: Prescription):
boolean
+ addMedication(medication: Medication): boolean
+ removePrescription(prescription:
Prescription): boolean
+ removeMedication(medication: Medication): void
+ toString(): String
```

- medications: List<Medication> - Stores a list of medications assigned to the patient.
- prescriptions: List<Prescription> - Stores a list of prescriptions linked to the patient.

- + Patient(id: int, name: String, age: int, phone_number: String) - Initializes a patient with empty lists of medications and prescriptions.
- + getMedications(): List<Medication> - Returns the list of Medication objects assigned to the patient.
- + getPrescriptions(): List<Prescription> - Returns the list of Prescription objects for the patient.
- + getMedicationIds(): List<Integer> - Returns IDs of assigned medications.
- + getPrescriptionIds(): List<Integer> - Returns IDs of assigned prescriptions.
- + addPrescription(prescription: Prescription): boolean - Assigns a prescription to the patient.
- + addMedication(medication: Medication): boolean - Private method that adds a medication if it does not already exist (used from within addPrescription)
- + removePrescription(prescription: Prescription): boolean - Removes a prescription (also removes the associated medication if applicable).
- + removeMedication(medication: Medication): void - Private method that removes a medication if its associated prescription is removed.
- + toString(): String - Returns a formatted string representation of the patient.

3.4 Medication Class

```
Medication
--------------------------------------------------
- id: int
- name: String
- dose: double
- quantity_in_stock: int
- expiry_date: LocalDate
--------------------------------------------------
+ Medication(id: int, name: String, dose:
double, quantity_in_stock: int, expiry_date:
String)
--------------------------------------------------
+ getId(): int
+ getName(): String
+ getDose(): double
+ getQuantityInStock(): int
+ getExpiryDateString(): String
+ getExpiryDate(): LocalDate
+ setName(name: String): boolean
+ setDose(dose: double): boolean
+ setQuantityInStock(quantity_in_stock: int):
boolean
+ setExpiryDateFromString(expiry_date: String):
boolean
+ updateFields(name: String, dose: double,
quantity_in_stock: int, expiry_date: String):
boolean
+ toString(): String
```

- - id: int - Unique identifier (final, cannot be changed).
- - name: String - Name of the medication (validated).
- - dose: double - Dosage amount (validated).
- - quantity_in_stock: int - Current stock level (validated).
- - expiry_date: LocalDate - Expiry date stored as a LocalDate.
- + Medication(id: int, name: String, dose: double, quantity_in_stock: int, expiry_date: String) - Initializes a medication object with validated values (defaults used if invalid).
- + getId(): int - Returns the ID of the medication.
- + getName(): String - Returns the name of the medication.
- + getDose(): double - Returns the dose of the medication.
- + getQuantityInStock(): int - Returns the stock quantity.
- + getExpiryDateString(): String - Returns the expiry date formatted as "YYYY-MM".
- + getExpiryDate(): LocalDate - Returns the expiry date as a LocalDate object.
- + setName(name: String): boolean - Updates the name (if valid)
- + setDose(dose: double): boolean - Updates the dose (if valid).
- + setQuantityInStock(quantity_in_stock: int): boolean - Updates the stock quantity (if valid).
- + setExpiryDateFromString(expiry_date: String): boolean - Updates the expiry date (if valid).
- + updateFields(name: String, dose: double, quantity_in_stock: int, expiry_date: String): boolean - Updates multiple fields (keeps old values if null or -1).
- + toString(): String - Returns a formatted string representation of the medication.

3.5 Prescription Class

```
Prescription
---------------------------------------------------------
- id: int
- doctor: Doctor
- patient: Patient
- medication: Medication
- prescription_expiry: LocalDate
---------------------------------------------------------
+ Prescription(id: int, doctor: Doctor, patient:
Patient, medication: Medication,
prescription_expiry: LocalDate)
---------------------------------------------------------
+ getId(): int
+ getDoctor(): Doctor
+ getPatient(): Patient
+ getMedication(): Medication
+ getPrescriptionExpiry(): LocalDate
+ toString(): String
```

- ❖ - id: int - Unique identifier for the prescription (final, cannot change).
- ❖ - doctor: Doctor - Doctor who issued the prescription).
- ❖ - patient: Patient - Patient to whom the prescription is assigned.
- ❖ - medication: Medication - Medication prescribed.
- ❖ - prescription_expiry: LocalDate - Expiration date of the prescription.
- ❖ + Prescription(id: int, doctor: Doctor, patient: Patient, medication: Medication, prescription_expiry: LocalDate) - Initializes a prescription with all required attributes, ensures the patient is assigned to the doctor, automatically adds the prescription to the patient's record.
- ❖ + getId(): int - Returns the prescription ID.
- ❖ + getDoctor(): Doctor - Returns the Doctor associated with this prescription.
- ❖ + getPatient(): Patient - Returns the Patient receiving this prescription.
- ❖ + getMedication(): Medication - Returns the Medication prescribed.
- ❖ + getPrescriptionExpiry(): LocalDate - Returns the expiry date of the prescription.
- ❖ + toString(): String - Returns a formatted string representation of the prescription.

3.6 MedicationTrackingSystem Class

```
MedicationTrackingSystem
---------------------------------------------------------
- patients: List<Patient>
- doctors: List<Doctor>
- medications: List<Medication>
- prescriptions: List<Prescription>
---------------------------------------------------------
+ MedicationTrackingSystem()
+ getNumPatients(): int
+ getNumDoctors(): int
+ getNumMedications(): int
+ getNumPrescriptions(): int
+ getPatients(): List<Patient>
+ getDoctors(): List<Doctor>
+ getMedications(): List<Medication>
+ getPrescriptions(): List<Prescription>
+ getPatientById(patient_id: int): Patient
+ getDoctorById(doctor_id: int): Doctor
+ getMedicationById(medication_id: int):
Medication
+ getPrescriptionById(prescription_id: int):
Prescription
+ searchPatientsByName(name: String,
exact_match: boolean): List<Patient>
+ searchPatientsByName(name: String):
List<Patient>
+ searchPatientReport(name: String):
List<String>
+ addPatient(name: String, age: int,
phone_number: String): boolean
+ editPatientById(patient_id: int, name: String,
age: int, phone_number: String): boolean
+ editPatient(patient: Patient, name: String,
age: int, phone_number: String): boolean
+ removePatient(patient: Patient): boolean
+ removePatientById(patient_id: int): boolean
+ searchDoctorsByName(name: String, exact_match:
boolean): List<Doctor>
```

```
+ searchDoctorsByName(name: String):
List<Doctor>
+ searchDoctorReport(name: String): List<String>
+ addDoctor(name: String, age: int,
phone_number: String, specialization: String):
boolean
+ editDoctorById(doctor_id: int, name: String,
age: int, phone_number: String, specialization:
String): boolean
+ editDoctor(doctor: Doctor, name: String, age:
int, phone_number: String, specialization:
String): boolean
+ removeDoctor(doctor: Doctor): boolean
+ removeDoctorById(doctor_id: int): boolean
+ assignPatientToDoctorById(doctor_id: int,
patient_id: int): boolean
+ assignPatientToDoctor(patient: Patient,
doctor: Doctor): boolean
+ searchMedicationsByName(name: String,
exact_match: boolean): List<Medication>
+ searchMedicationsByName(name: String):
List<Medication>
+ searchMedicationReport(name: String):
List<String>
+ addMedication(name: String, dose: double,
quantity_in_stock: int, expiry_date: String):
boolean
+ editMedicationById(medication_id: int, name:
String, dose: double, quantity_in_stock: int,
expiry_date: String): boolean
```

```
+ editMedication(medication: Medication, name:
String, dose: double, quantity_in_stock: int,
expiry_date: String): boolean
+ removeMedication(medication: Medication):
boolean
+ removeMedicationById(medication_id: int):
boolean
+ checkForExpiredMeds(): List<Medication>
+ checkLowStockMeds(threshold: int):
List<Medication>
+ restockMedicationById(medication_id: int,
amount: int): boolean
+ restockMedication(medication: Medication,
amount: int): boolean
+ restockAllMedications(amount: int): void
+ checkForExpiredMedsReport(): List<String>
+ addPrescription(doctor: Doctor, patient:
Patient, medication: Medication): boolean
+ addPrescription(doctor: Doctor, patient:
Patient, medication: Medication,
expiry_full_date: String): boolean
+ addPrescription(doctor: Doctor, patient:
Patient, medication: Medication, expiry_date:
LocalDate): boolean
+ getPrescriptionsByDoctor(doctor: Doctor):
List<Prescription>
+ getPrescriptionsByDoctor(doctor_id: int):
List<Prescription>
+ checkPrescriptionsByDoctorReport(doctor_id:
int): List<String>
```

```
+ checkPrescriptionsByDoctorReport(doctor:
Doctor): List<String>
+ medicationReport(): List<String>
+ patientReport(): List<String>
+ doctorReport(): List<String>
+ prescriptionReport(): List<String>
+ fullSystemReport(): List<String>
+ toString(): String
```

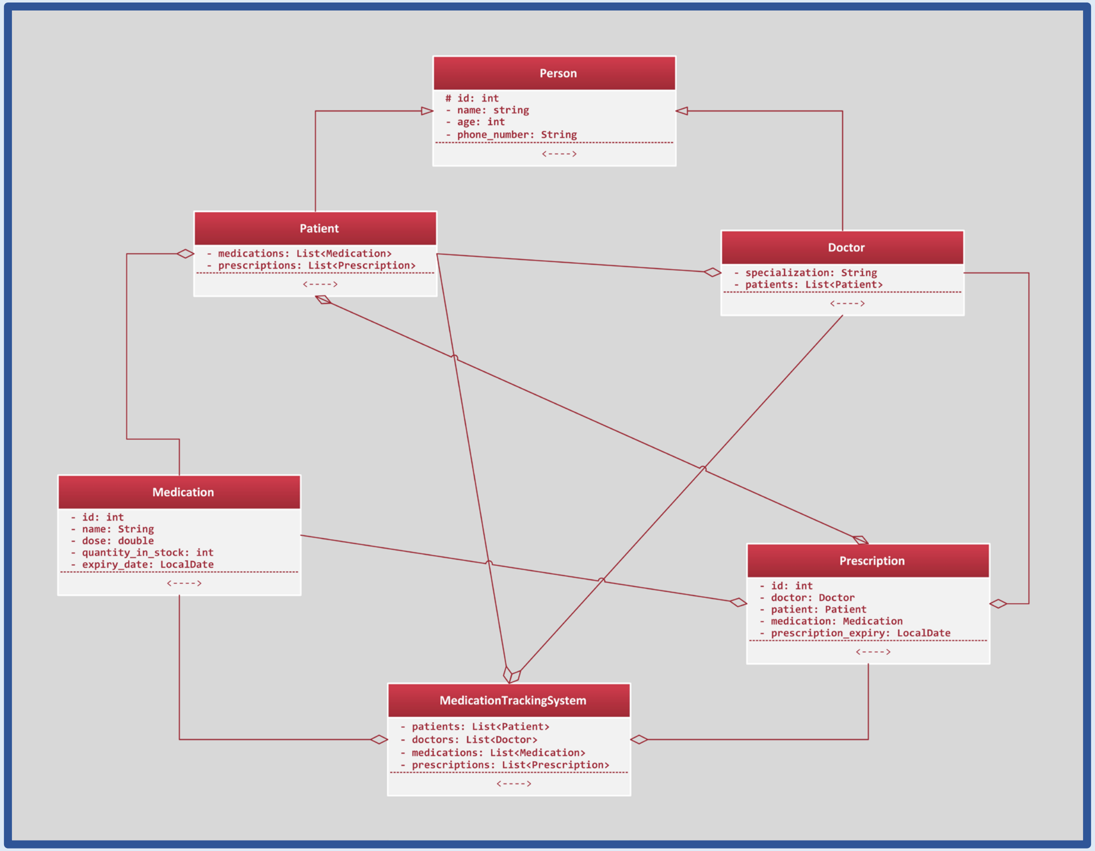
- - patients: List<Patient> - Stores a list of all patients in the system.
- - doctors: List<Doctor> - Stores a list of all doctors in the system.
- - medications: List<Medication> - Stores a list of all medications.
- - prescriptions: List<Prescription> - Stores a list of all prescriptions.
- + MedicationTrackingSystem() - Initializes empty lists for patients, doctors, medications, and prescriptions.
- + getNumPatients(): int – Returns the number of registered patients.
- + getNumDoctors(): int - Returns the number of registered doctors.
- + getNumMedications(): int - Returns the number of medications available.
- + getNumPrescriptions(): int - Returns the number of prescriptions issued.
- + getPatients(): List<Patient> - Returns the list of all patients registered in the system.

- + getDoctors(): List<Doctor> - Returns the list of all doctors registered in the system.
- + getMedications(): List<Medication> - Returns the list of all medications registered in the system.
- + getPrescriptions(): List<Prescription> - Returns the list of all prescriptions registered in the system.
- + getPatientById(patient_id: int): Patient – Looks up for the Patient by it's ID, returns Patient reference, null if not found.
- + getDoctorById(doctor_id: int): Doctor - Looks up for the Doctor by it's ID, returns Doctor reference, null if not found
- + getMedicationById(medication_id: int): Medication – same for Medication.
- + getPrescriptionById(prescription_id: int): Prescription – same for Prescription.
- + searchPatientsByName(name: String, exact_match: boolean): List<Patient> - Finds patients by name, returns list of results, list is empty if not found.
- + searchPatientsByName(name: String): List<Patient> - same, but exact_match = false (contains-search).
- + searchPatientReport(name: String): List<String> - Generates a report with search results.
- + addPatient(name: String, age: int, phone_number: String): boolean – Add patient to the system, returns true if successful, false if not (Patient already exists).
- + editPatientById(patient_id: int, name: String, age: int, phone_number: String): boolean – Edits patient record, looking up Patient by its ID. Returns false if unsuccessful (fields not validated, ID not found).
- + editPatient(patient: Patient, name: String, age: int, phone_number: String): boolean – same, but looks for Patient object reference.
- + removePatient(patient: Patient): boolean – Removes Patient from the system, returns false if Patient not found.
- + removePatientById(patient_id: int): boolean – same, but finds Patient to remove by its ID.
- + searchDoctorsByName(name: String, exact_match: boolean): List<Doctor> - Finds Doctors by name, returns list of results, list is empty if not found.
- + searchDoctorsByName(name: String): List<Doctor> - same, but exact_match = false (contains-search).
- + searchDoctorReport(name: String): List<String> - Generates a report with search results.
- + addDoctor(name: String, age: int, phone_number: String, specialization: String): boolean – Adds Doctor to the system, returns false if failed (Doctor already exists).
- + editDoctorById(doctor_id: int, name: String, age: int, phone_number: String, specialization: String): boolean – edits Doctor's information, returns true if OK, false if fields not validated, of ID not found.
- + editDoctor(doctor: Doctor, name: String, age: int, phone_number: String, specialization: String): boolean – same, but looks for Doctor by reference.
- + removeDoctor(doctor: Doctor): boolean – removes Doctoc from the system by its reference, returns true if OK, false if not found.

- + removeDoctorById(doctor_id: int): boolean – same, but looks by Doctro's ID.
- + assignPatientToDoctorById(doctor_id: int, patient_id: int): boolean – assigns Patient to a Doctors by IDs, true if OK, false if not successful (already assigned, or IDs not found).
- + assignPatientToDoctor(patient: Patient, doctor: Doctor): boolean – same, but assigns by references.
- + searchMedicationsByName(name: String, exact_match: boolean): List<Medication> - searches for Medication in the system by name, returns empty list if not found.
- + searchMedicationsByName(name: String): List<Medication> - same, but always contains-search (exact_match=false).
- + searchMedicationReport(name: String): List<String> - generates a report of Medication search results.
- + addMedication(name: String, dose: double, quantity_in_stock: int, expiry_date: String): boolean – add Medication to the system, returns true if all OK, false if not (fields do not validate).
- + editMedicationById(medication_id: int, name: String, dose: double, quantity_in_stock: int, expiry_date: String): boolean – edits Medications details by its ID, returns true if all OK, false if edit failed (fields not validated, ID not found).
- + editMedication(medication: Medication, name: String, dose: double, quantity_in_stock: int, expiry_date: String): boolean – same, but by reference, returns boolean result.
- + removeMedication(medication: Medication): boolean – removes Medication by reference, false if not successful (not found).
- + removeMedicationById(medication_id: int): boolean – same, but by ID.
- + checkForExpiredMeds(): List<Medication> - Generates a list of expired Medications in the system. List is empty if none found.
- + checkLowStockMeds(threshold: int): List<Medication> - Finds medications below a given stock level.
- + restockMedicationById(medication_id: int, amount: int): boolean - Restocks a specific medication by a given amount, result is true if ok, false if amount not validates or ID not found.
- + restockMedication(medication: Medication, amount: int): boolean – same, but looks for Medication by reference.
- + restockAllMedications(amount: int): void – restock all Medications in the system by specified amount, if amount <=0 doesn't do anything.
- + checkForExpiredMedsReport(): List<String> - Generates text report of expired Medications, calls checkForExpiredMeds() from within. Returns list of strings.
- + addPrescription(doctor: Doctor, patient: Patient, medication: Medication): boolean – adds Prescription in the system, looks by reference, Patient is automatically added to the Doctor if wasn't added before. Returns true if all OK, false if unsuccessful (one of the references not found). Expiry date set automatically +1 year from current date.
- + addPrescription(doctor: Doctor, patient: Patient, medication: Medication, expiry_full_date: String): boolean – same, but expiry date must be provided "YYYY-MM".

- + addPrescription(doctor: Doctor, patient: Patient, medication: Medication, expiry_date: LocalDate): boolean – same, but expiry date is in LocalDate format.
- + getPrescriptionsByDoctor(doctor: Doctor): List<Prescription> - Gets a list of presctiptions by specific Doctor reference, returns empty list if none.
- + getPrescriptionsByDoctor(doctor_id: int): List<Prescription> - same, but looks for Doctor by its ID.
- + checkPrescriptionsByDoctorReport(doctor_id: int): List<String> - Generates text report (list of strings), calls getPrescriptionsByDoctor(ID) from within.
- + checkPrescriptionsByDoctorReport(doctor: Doctor): List<String> - same, but by Doctor reference (overloaded methods).
- + medicationReport(): List<String> - Full text report (list of strings) about Medications, this method is used during full report generation.
- + patientReport(): List<String> - Full text report (list of strings) about Patients, this method is used during full report generation.
- + doctorReport(): List<String> - Full text report (list of strings) about Doctors, this method is used during full report generation.
- + prescriptionReport(): List<String> - Full text report (list of strings) about Prescriptions, this method is used during full report generation.
- + fullSystemReport(): List<String> - Generates full text report (list of strings), includes all Doctors, Patients, Medications, Prescriptions, including all details for each Patient (Meds and Prescriptions), and Doctors (assigned Patients).
- + toString(): String - Returns a summary of key system statistics (number of Patients/Doctors/Medications/Prescriptions).

4 Appendix: UML Diagram


Person
# id: int
- name: string
- age: int
- phone_number: String
<---->
Patient
- medications: List<Medication>
- prescriptions: List<Prescription>
<---->
Doctor
- specialization: String
- patients: List<Patient>
<---->
Medication
- id: int
- name: String
- dose: double
- quantity_in_stock: int
- expiry_date: LocalDate
<---->
Prescription
- id: int
- doctor: Doctor
- patient: Patient
- medication: Medication
- prescription_expiry: LocalDate
<---->
MedicationTrackingSystem
- patients: List<Patient>
- doctors: List<Doctor>
- medications: List<Medication>
- prescriptions: List<Prescription>
<---->