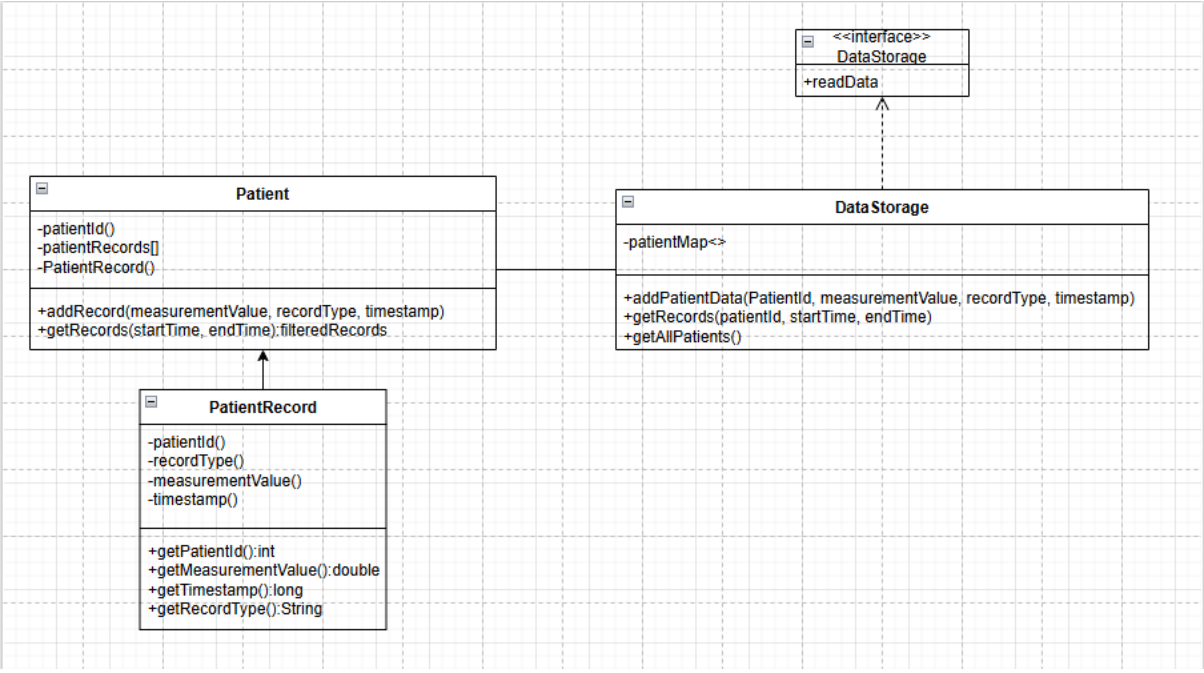

This document outlines the design of a patient data storage system represented using a UML class diagram. The system focuses on storing, retrieving, and managing patient health records such as measurements over time. It supports functionalities such as adding new patient data, filtering records by time, and retrieving patient information.

The system is centered around the management of patients and their corresponding health measurement records. It includes the following key components:

- Patient
- PatientRecord
- DataStorage (and its interface)

Patient

Represents an individual patient.

Attributes:

- patientId – A unique identifier for each patient.
- patientRecords[] – A collection of PatientRecord objects for storing the patient's data.

Methods:

- addRecord(measurementValue, recordType, timestamp) – Adds a new record to the patient's record list.
- getRecords(startTime, endTime): filteredRecords – Retrieves records within a specified time range.

PatientRecord

Encapsulates a single medical record entry.

Attributes:

- patientId – Identifies the patient the record belongs to.
- recordType – Type of medical measurement (e.g., heart rate, temperature).
- measurementValue – The value of the measurement.
- timestamp – Time at which the measurement was taken.

Methods:

- getPatientId(): int
- getMeasurementValue(): double
- getTimestamp(): long
- getRecordType(): String

DataStorage

Acts as a centralized storage system for all patients.

Attributes:

- patientMap – A collection (likely a map or dictionary) associating patient IDs with Patient objects.

Methods:

- addPatientData(patientId, measurementValue, recordType, timestamp) – Adds a record for the specified patient, creating the patient entry if necessary.
- getRecords(patientId, startTime, endTime) – Retrieves a filtered list of patient records based on time range.
- getAllPatients() – Returns all patients stored in the system.

DataStorage Interface

An interface defining the contract for reading data.

Methods:

- readData() – Abstract method for data retrieval, to be implemented by concrete storage classes.
-

- **Data Entry**: Data can be added dynamically per patient.
- **Filtering**: Historical records can be retrieved using time filters.
- **Scalability**: Centralized `DataStorage` ensures extensibility for managing numerous patients.

Strengths:

- Modular: Clean separation between patient logic and storage logic.
- Reusable: DataStorage interface supports multiple implementations (e.g., in-memory, database).
- Scalable: Efficient use of maps allows fast lookups for patient data.

Areas for Improvement:

- Error Handling: UML does not indicate how missing patients or invalid inputs are handled.
- Record Type Standardization: Could benefit from an enumeration for recordType.

This UML diagram demonstrates a well-structured and scalable design for managing patient health records. The system's object-oriented nature makes it suitable for extension and integration with larger health informatics solutions.