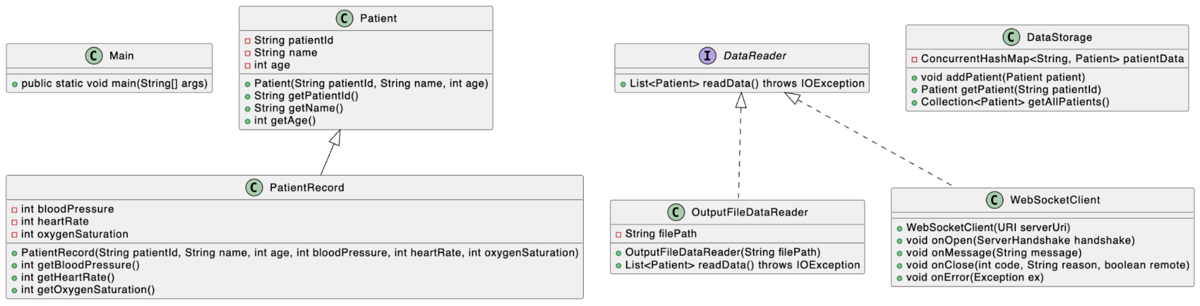
Data Management System UML



Main
public static void main(String[] args)
Patient
String patientId
String name
int age
Patient(String patientId, String name, int age)
String getPatientId()
String getName()
int getAge()
PatientRecord
int bloodPressure
int heartRate
int oxygenSaturation
PatientRecord(String patientId, String name, int age, int bloodPressure, int heartRate, int oxygenSaturation)
int getBloodPressure()
int getHeartRate()
int getOxygenSaturation()
DataReader
List<Patient> readData() throws IOException
DataStorage
ConcurrentHashMap<String, Patient> patientData
void addPatient(Patient patient)
Patient getPatient(String patientId)
Collection<Patient> getAllPatients()
OutputFileDataReader
String filePath
OutputFileDataReader(String filePath)
List<Patient> readData() throws IOException
WebSocketClient
WebSocketClient(URI serverUri)
void onOpen(ServerHandshake handshake)
void onMessage(String message)
void onClose(int code, String reason, boolean remote)
void onError(Exception ex)

The com.data_management package is responsible for managing patient data, reading data from different sources, and storing it for further processing. The package includes classes and interfaces for handling data storage, reading data from files and web sockets, and representing patient records. The DataStorage class is responsible for storing and managing patient data. It uses a ConcurrentHashMap to store patient records by patient ID, ensuring thread-safe access to the data. Key methods include void addPatient(Patient patient), which adds a patient to the data storage, Patient getPatient(String patientId), which retrieves a patient record by ID, and Collection<Patient> getAllPatients(), which returns all patient records.

The DataReader interface defines the contract for reading patient data from different sources. Implementing classes must provide a method to read data and return a list of patient records. The primary method is List<Patient> readData() throws IOException, which reads patient data and returns a list of patient records.

The OutputFileDataReader class implements the DataReader interface to read patient data from a specified output file. It reads lines from the file, parses them, and converts them into Patient objects. The OutputFileDataReader(String filePath) constructor initializes the class with the path to the output file, while the List<Patient> readData() throws IOException method reads the data from the file and returns a list of patient records.

The WebSocketClient class is responsible for connecting to a WebSocket server to receive patient data in real-time. It extends the WebSocketClient class from the org.java_websocket.client package and handles the WebSocket connection lifecycle. Key methods include void onOpen(ServerHandshake handshake), which is called when the connection is opened, void onMessage(String message), which is called when a message is received, void onClose(int code, String reason, boolean remote), which is called when the connection is closed, and void onError(Exception ex), which is called when an error occurs.

The Patient class represents a patient record. It includes fields for patient ID, name, age, and other relevant health data. The class provides constructors for initializing patient records and getter methods for accessing the data. The Patient(String patientId, String name, int age) constructor initializes the patient record with the specified details, while methods like String getPatientId(), String getName(), and int getAge() return the corresponding patient data.

The PatientRecord class extends Patient to include additional health data fields specific to patient records. It adds fields for health metrics such as blood pressure, heart rate, and oxygen saturation. The class provides constructors for initializing patient records with these additional fields and getter methods for accessing the data. The PatientRecord(String patientId, String name, int age, int bloodPressure, int heartRate, int oxygenSaturation) constructor initializes the patient record with the specified details, while methods like int getBloodPressure(), int getHeartRate(), and int getOxygenSaturation() return the corresponding health metrics.

The Main class is the entry point for the data management module. It contains the main method, which initializes the data storage, reads patient data from configured sources, and processes the data as required. The public static void main(String[] args) method initializes the data management components and starts the data processing workflow.