Documentation UML diagrams

Sequence diagram:

The sequence diagram depicts the process of generating alerts in the Cardiovascular Health Monitoring System (CHMS). It involves interactions between the DataListener, AlertGenerator, DataStorage, AlertManager and nurse, highlighting how alerts are created, validated, and communicated.

Key components:

- DataListener: Continuously receives the latest patient data.
- AlertGenerator: Evaluates incoming data for abnormalities and checks against historical data.
- DataStorage: Stores and retrieves historical patient data.
- AlertManager: Manages and transmits alerts to medical staff.
- Nurse: Receives and responds to alerts (acknowledgement).

Process flow:

- Receiving data: DataListener receives new patient data.
- Evaluating data: If data is abnormal, it's sent to AlertGenerator.
- Validating data: AlertGenerator checks the data against historical records in DataStorage.
    - If a trend is detected, an alert is generated.
    - If no trend, the data is updated in DataStorage.
- Transmitting alerts: AlertManager sends the alert to the nurse.
- Nurse action: Nurse receives and addresses the alert.

This diagram ensures a clear separation of responsibilities among components for easier maintenance and scalability. Moreover, it reduces false positives by checking current data against historical trends(DataStorage). Lastly, it makes sure that the alert gets timely delivered to medical staff for prompt intervention.

State diagram:

The state diagram illustrates the lifecycle of an alert within the Cardiovascular Health Monitoring System (CHMS). It shows the process from checking patient data to updating patient records after an alert is generated and resolved.

Description:

1. Check patient's data: The system continuously monitors incoming patient data.
2. Check past data against DataStorage: The system verifies the new data against historical data in DataStorage to identify trends.
3. If threshold exceeded: If the data exceeds predefined thresholds, the system creates an alert.
4. If continuing trend: The system checks if the abnormal data is part of a continuing trend.
5. Create alert: An alert is generated if a critical trend is detected.

6. Send alert to medical staff: The alert is sent to the medical staff for immediate attention.
7. Medical staff (acknowledged): The medical staff acknowledges the alert.
8. Manual fix by nurse: The nurse addresses the alert condition through medical intervention.
9. Update patient's data: Patient data is updated in the system.
10. Preparing for next data: The system prepares to receive the next set of patient data.

Class diagrams:

- **AlertGenerationSystem**

The UML class diagram for the Alert Generation System in the Cardiovascular Health Monitoring System (CHMS) defines the key components and their relationships involved in generating and managing alerts based on patient data.

Key classes and attributes:

1. **HealthDataSimulator**:
    - **Attributes**:
        - `patientCount`: Number of patients being simulated.
        - `scheduler`: Schedules tasks for patient data generation.
        - `outputStrategy`: Strategy for outputting data (e.g., console).
        - `random`: Random number generator for data simulation.
    - **Methods**:
        - `main()`: Entry point for the simulator.
        - `parseArguments()`: Parses command-line arguments.
        - `printHelp()`: Prints help information.
        - `initializePatientids()`: Initializes patient IDs.
        - `scheduleTaskForPatients()`: Schedules data generation tasks for patients.
        - `scheduleTask()`: Schedules individual tasks.
2. **AlertGenerator**:
    - **Attributes**:
        - `personalizedThresholds`: Array of thresholds specific to each patient.
        - `realDataStream`: Array of real-time data streams.
    - **Methods**:
        - `scanForAbnormality()`: Scans data for abnormalities.
        - `triggerAlert()`: Triggers an alert if abnormalities are detected.
        - `sendToManager()`: Sends the alert to the AlertManager.
3. **AlertManager**:
    - **Attributes**:
        - `alert`: Stores the alert details.

        - `hospitalDepartments`: Map of hospital departments responsible for specific alerts.
    - **Methods**:
        - `sendAlert()`: Sends the alert to the appropriate department.
4. **Alert**:
    - **Attributes**:
        - `alert`: Details of the alert.
        - `patientId`: ID of the patient associated with the alert.

Relationships:

- HealthDataSimulator generates data and sends it to AlertGenerator.
- AlertGenerator creates and sends alerts to AlertManager.
- AlertManager determines the appropriate hospital department and forwards the alert.

- **DataStorageSystem**

The UML class diagram for the Data Storage System in the Cardiovascular Health Monitoring System (CHMS) outlines the key components and their interactions involved in storing, retrieving, and managing patient data.

Key classes and attributes:

1. **DataStorage**:
    - **Attributes**:
        - `patientData`: A `HashMap` storing `PersonData` objects indexed by patient ID.
        - `logIn`: A boolean indicating if the system is in a logged-in state.
    - **Methods**:
        - `getECGData()`: Retrieves ECG data for a patient.
        - `getBloodSaturationData()`: Retrieves blood saturation data for a patient.
        - `getBloodLevelsData()`: Retrieves blood levels data for a patient.
        - `getBloodPressureData()`: Retrieves blood pressure data for a patient.
        - `setECGData()`: Sets ECG data for a patient.
        - `setBloodSaturationData()`: Sets blood saturation data for a patient.
        - `setBloodLevelsData()`: Sets blood levels data for a patient.
        - `setBloodPressureData()`: Sets blood pressure data for a patient.
2. **PatientData**:
    - **Attributes**:
        - `ECGData`: Double representing ECG data.
        - `Saturation`: Double representing blood saturation.

        - `Levels`: Double representing blood levels.
        - `Pressure`: Integer representing blood pressure.
        - `personID`: Integer representing the patient ID.
    - **Methods**:
        - `getECGData()`: Retrieves ECG data.
        - `getBloodSaturationData()`: Retrieves blood saturation data.
        - `getBloodLevelsData()`: Retrieves blood levels data.
        - `getBloodPressureData()`: Retrieves blood pressure data.
        - `setECGData()`: Sets ECG data.
        - `setBloodSaturationData()`: Sets blood saturation data.
        - `setBloodLevelsData()`: Sets blood levels data.
        - `setBloodPressureData()`: Sets blood pressure data.
3. **DataRetriever**:
    - **Attributes**:
        - `login`: A `HashMap` storing login states indexed by DoctorID.
    - **Methods**:
        - `login()`: Manages user login.
        - `getECGData()`: Retrieves ECG data for a patient.
        - `getBloodSaturationData()`: Retrieves blood saturation data for a patient.
        - `getBloodLevelsData()`: Retrieves blood levels data for a patient.
        - `getBloodPressureData()`: Retrieves blood pressure data for a patient.

Relationships:

- DataStorage uses `PatientData` objects to store and manage patient data.
- DataRetriever accesses data from `DataStorage` and ensures only authorized personnel can retrieve sensitive patient information.

• DataAccessLayer

The UML class diagram for the Data Access Layer in the Cardiovascular Health Monitoring System (CHMS) describes the key components and their interactions for receiving and processing data from various sources. This layer serves as a bridge between external data sources and the internal data processing systems.

Key classes and attributes:

1. **DataListener**:
    - **Attributes**:
        - `sourceInflow`: Represents the data source (unspecified type).
    - **Methods**:
        - `listen()`: Listens for incoming data.

        - `receiveData()`: Receives data from the source.
2. **TCPDataListener** (inherits from DataListener):
    - **Methods**:
        - `processData()`: Processes the received data.
        - `connect()`: Connects to the TCP data source.
3. **FileDataListener** (inherits from DataListener):
    - **Methods**:
        - `processData()`: Processes the received data.
        - `connect()`: Connects to the file data source.
        - `storeData()`: Stores the processed data.
4. **WebSocketDataListener** (inherits from DataListener):
    - **Methods**:
        - `processData()`: Processes the received data.
        - `connect()`: Connects to the WebSocket data source.
5. **DataParser**:
    - **Methods**:
        - `parseData(String rawData)`: Parses raw data into `PatientData`.

Relationships:

- DataListener is the base class for TCPDataListener, FileDataListener, and WebSocketDataListener.
- DataParser is utilized by data listeners to convert raw data into a usable format.

- **PatientIdentificationSystem**

The UML class diagram for the Patient Identification System in the Cardiovascular Health Monitoring System (CHMS) describes the key components and their interactions for accurately identifying patients and managing their records. This system ensures that patient data is correctly linked to the appropriate patient profile.

Key classes and attributes:

1. **PatientIdentifier**:
    - **Attributes**:
        - `recordID`: String representing the unique identifier for each patient record.
    - **Methods**:
        - `matchPatient()`: Matches incoming patient data with existing patient records.
        - `getPatientRecord()`: Retrieves the patient record based on the `recordID`.
2. **IdentityManager**:
    - **Attributes**:

        - `discrepancy`: String representing any discrepancies found during the matching process.
    - **Methods**:
        - `resolveDiscrepancy()`: Resolves discrepancies between incoming data and existing records.
3. **PatientRecord**:
    - **Attributes**:
        - `patientID`: Integer representing the unique identifier for the patient.
        - `medicalHistory`: Method that returns the patient's medical history as a String.
        - `personalDetails`: Method that returns the patient's personal details as a String.
    - **Methods**:
        - `showRecord()`: Displays the patient’s record.
        - `updateRecord()`: Updates the patient’s record with new data.

Relationships:

- PatientIdentifier uses `PatientRecord` to access and match patient data.
- IdentityManager interacts with `PatientIdentifier` to handle and resolve discrepancies.