Overview of the Alert Class

Package: com.alerts

Purpose

The Alert class represents an alert event related to a patient in the cardiovascular data simulator. It encapsulates critical data such as the patient identifier, the triggering health condition (e.g., abnormal blood pressure or heart rate), and the timestamp when the alert occurred. It acts as a core data structure enabling the system to store, process, and transmit alerts through different output strategies (e.g., console, file, WebSocket, or TCP).

Role in the Project

The Alert class is a key component of the simulator's alert system. It is utilized by classes like AlertGenerator to represent triggered or resolved alerts and by alert factories to create specific alert types. The class provides a consistent structure for alert data, facilitating integration with storage and output mechanisms.

Technical Characteristics

- Plain Old Java Object (POJO) with private fields and public getter methods.
- Immutable; no setter methods, ensuring data integrity.
- Lightweight and inherently serializable (only contains String and long fields).
- Suitable for transmission over networks and concurrent environments.

Internal Components and Their Purposes

Fields

private String patientId

- **Purpose:** Stores the patient's unique identifier.
- **Technical Details:**
  - Type: String
  - Private access ensures encapsulation.

    - Expected to be non-null and valid (no enforced validation).
- **Role:** Links the alert to a specific patient.

private String condition

- **Purpose:** Describes the condition triggering the alert (e.g., HighBloodPressure, triggered, resolved).
- **Technical Details:**
    - Type: String
    - Private access for encapsulation.
    - Expected to be a meaningful descriptor (no enforced format).
- **Role:** Provides context for categorizing and prioritizing the alert.

private long timestamp

- **Purpose:** Records the time of the alert in milliseconds since the Unix epoch.
- **Technical Details:**
    - Type: long
    - Private access.
    - Expected to be a non-negative value (no validation enforced).
- **Role:** Enables time-based tracking and sequencing of alerts.

Constructor

public Alert(String patientId, String condition, long timestamp)

- **Purpose:** Initializes an Alert object with the specified details.
- **Technical Details:**
    - Parameters: patientId, condition, timestamp.
    - No input validation is performed.
    - Public access for use by other classes.
- **Role:** Creates an immutable instance with all fields initialized.

Methods

public String getPatientId()

- **Purpose:** Returns the patient identifier.
- **Technical Details:**
    - Return type: String.
    - No side effects.
- **Role:** Allows other components to access the patient ID.

public String getCondition()

- **Purpose:** Returns the alert condition.
- **Technical Details:**
  - Return type: String.
  - No side effects.
- **Role:** Enables categorization and processing of the alert.

public long getTimestamp()

- **Purpose:** Returns the timestamp of the alert.
- **Technical Details:**
  - Return type: long.
  - No side effects.
- **Role:** Supports time-based analysis and logging of alerts.

Technical Points and Design Considerations

Encapsulation and Immutability

- Uses private fields with public getters.
- No setters; immutable and thread-safe.

Simplicity and Extensibility

- Simple data container.
- Supports subclassing (e.g., BloodPressureAlert, ECGAlert) in future tasks.

Lack of Validation

- No validation in the constructor; responsibility lies with the caller.
- Potential risk if invalid data is provided.

Integration with Design Patterns

- **Factory Method Pattern:** Factories create specific Alert instances.
- **Decorator Pattern:** Alert decorators add functionality (e.g., priority levels).
- **Strategy Pattern:** Works with strategies determining alert triggers.

Thread Safety

- Immutable, thread-safe for concurrent use.
- Fields are safe as they are primitive (`long`) or immutable (`String`).

Potential Improvements

- Add validation in the constructor.
- Implement `toString()` for easier debugging.
- Use `enum` for condition types.
- Override `equals()` and `hashCode()` for collection support.

Interaction with Other Components

With `AlertGenerator`

- Used to represent generated or resolved alerts.
- Provides structured alert objects for output strategies.

With `HealthDataSimulator`

- Periodically generates alerts per patient.
- `Alert` encapsulates data for output strategies.

With Design Patterns (Task 7)

- Created by `AlertFactory` subclasses (e.g., `ECGAlertFactory`).
- Decorated by `AlertDecorator` classes.
- Used by `AlertStrategy` implementations to determine triggers.

Summary of Functionality

The `Alert` class is a lightweight, immutable data structure containing a patient ID, condition, and timestamp. Its simple yet robust design makes it a reliable component for the alert system, supporting concurrent environments and integrating seamlessly with design patterns for alert generation, decoration, and output.
