1. Alert Generation Subsystem

This diagram models how the system evaluates patient vitals against personalized thresholds and dispatches alerts.

- **Design Rationale**
    - **Separation of Concerns**: Detection, data representation and notification are decoupled.
    - **Extensibility**: New threshold rules or output channels plug in via strategy interfaces without altering core logic.
    - **Safety-Critical Clarity**: Each class has a single responsibility and clear boundaries, vital in a hospital setting.

2. Data Storage Subsystem

This diagram depicts how incoming vitals are persisted, retrieved and purged securely.

- **Design Rationale**
    - **Modularity**: Storage interface abstracts the underlying mechanism, allowing future in-memory or cloud implementations.
    - **Security & Privacy**: AccessControl ensures only authorized users retrieve or delete data.
    - **Data Lifecycle**: Retention policy separated into its own class, simplifying cleanup logic and configurability.

3. Patient Identification System

This diagram shows how raw simulator IDs are matched to real patient records and handling mismatches gracefully.

- **Design Rationale**

    - **Error Handling**: `PatientMatchResult` cleanly separates success vs. failure paths without exceptions.
    - **Extensibility**: Swapping `PatientIdentifier` for a mock or alternative lookup is trivial.
    - **Traceability & Privacy**: Only successful matches expose patient data; failures can be audited without leaking PII.

4. Data Access Layer

This diagram covers ingestion of raw streams (TCP, WebSocket, file), parsing, and handoff to storage.

- **Design Rationale**
    - **Flexibility**: New transports or formats plug in via listener/handler interfaces without adapter changes.
    - **Separation of Concerns**: Transport, parsing, and storage are kept in distinct layers.
    - **Maintainability**: Central adapter orchestrates components, simplifying lifecycle management in an application entry point.