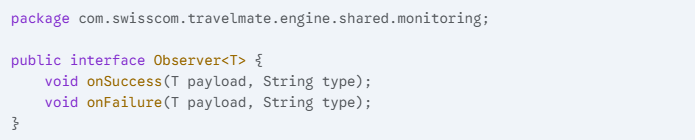
**Prometheus Logging Implementation**

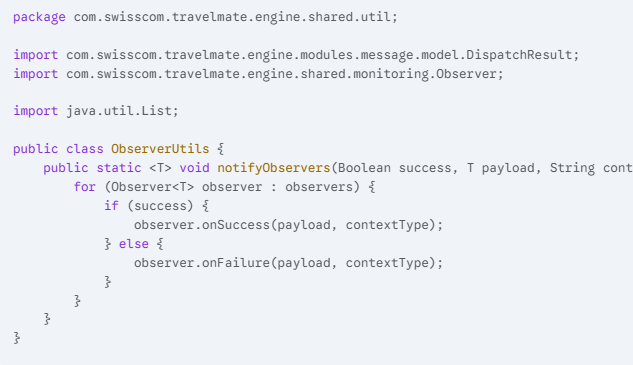
This implementation utilizes an Observer pattern to provide a generic and flexible way to record success and failure metrics in Prometheus. It consists of three key components: Observer.java, ObserverUtils.java, and PrometheusObserver.java.

**1. Observer.java (Interface)**

The Observer interface defines the contract for any class that wishes to "observe" the outcome of an operation.



**2. ObserverUtils.java (Utility Class)**

The ObserverUtils class provides a static helper method to notify a list of observers about the outcome of an operation.

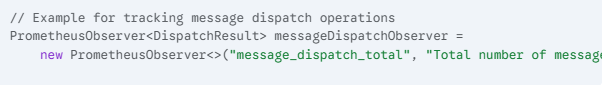
* **notifyObservers(Boolean success, T payload, String contextType, List<Observer<T>> observers)**:
  + success: A Boolean indicating whether the operation was successful (true) or failed (false).
  + payload: The data payload to be passed to the observers.
  + contextType: A String defining the context or type of the operation. This will be used by observers for categorization (e.g., as a label in Prometheus).
  + observers: A List of Observer instances that need to be notified.

**3. PrometheusObserver.java (Prometheus Implementation of Observer)**

The PrometheusObserver class is a concrete implementation of the Observer interface specifically designed to integrate with Prometheus. It increments a Prometheus Counter metric based on the success or failure of an observed operation.

* **Constructor PrometheusObserver(String metricName, String help)**:
  + metricName: The name of the Prometheus counter metric (e.g., "application\_operation\_total").
  + help: A helpful description for the Prometheus metric, explaining its purpose.
  + Upon instantiation, a Counter is built with the provided metricName and help. It also defines two fixed labels: "type" and "status".
* **onSuccess(T payload, String type)**: When an operation succeeds, this method increments the Prometheus counter. The "type" label is set to the provided type parameter, and the "status" label is set to "success".
* **onFailure(T payload, String type)**: When an operation fails, this method increments the Prometheus counter. The "type" label is set to the provided type parameter, and the "status" label is set to "failure".

**How to Use**

1. **Instantiate PrometheusObserver**: Create an instance of PrometheusObserver for each metric you want to track.
2. **Maintain a List of Observers**: In the class or component where the operation occurs, maintain a list of Observer instances.
3. **Notify Observers**: After an operation completes (either successfully or with a failure), call ObserverUtils.notifyObservers.

**Benefits**

* **Genericity**: The solution is generic (<T>) and can be used for any type of payload, making it reusable across different parts of the application.
* **Decoupling**: The observer pattern decouples the metric logging logic from the core business logic, improving modularity.
* **Centralized Notification**: ObserverUtils provides a single point of entry for notifying all interested observers.
* **Extensibility**: New observers (e.g., for different monitoring systems like Grafana Loki, or for logging to a file) can be easily added without modifying existing code.