OBSERVER

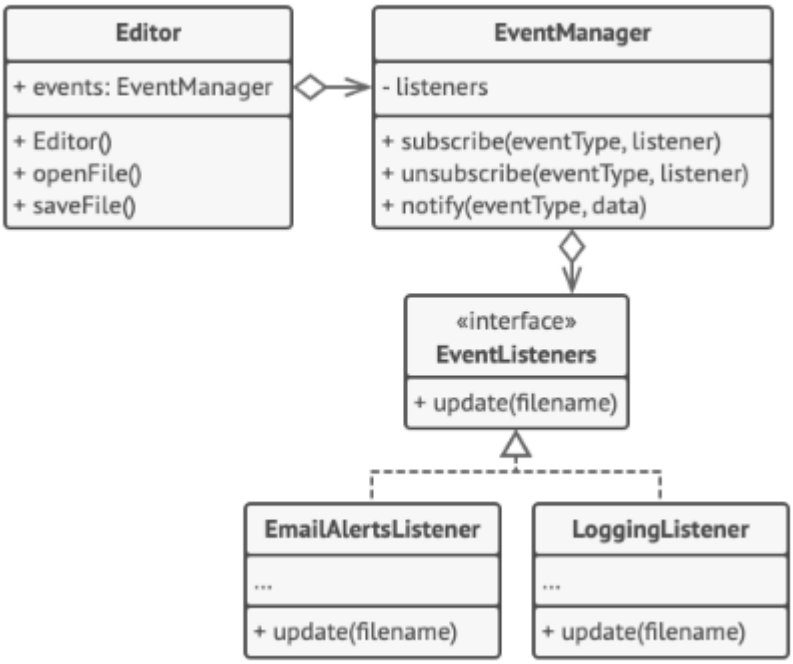
MIGUEL ANGEL RAMIREZ JUAREZ

JAVA ACADEMY

AUGUST/ 2024

Observer Pattern in Java?

The **Observer Pattern** is a behavioral design pattern used in software development to define a one-to-many dependency between objects. When one object (the subject) changes its state, all dependent objects (observers) are automatically notified and updated. This pattern is particularly useful in scenarios where a change in one part of an application requires updates to other parts, and you want to keep the components loosely coupled.



Problem Statement

In complex software systems, it is often necessary for certain components (observers) to stay in sync with the state of other components (subjects) and receive real-time updates. For instance, in a notification application, it is crucial for multiple modules to receive updates when the state of the system changes. The challenge is designing a system where components are decoupled but can still receive updates efficiently.

Problem Description

The **Observer** design pattern was used to solve this problem. This pattern allows a subject (or "observable") to notify a list of observers (or "subscribers") when its state changes, without the subject needing to know specific details about the observers.

The Observer pattern addresses the problem of synchronization between the subject and observers by providing a decoupled way of notification. The subject does not need to know the specifics of the observers; it simply notifies them of changes in its state. This approach allows for greater flexibility and scalability in the system design:

- **Decoupling:** Observers can be added or removed at runtime without modifying the subject's code.
- **Flexibility:** Observers can implement different behaviors upon receiving notifications, without the subject needing to know how they will react.

This approach improves the maintainability of the system and makes future extensions easier without significant changes to existing classes.