🔔 Build Your Own Notification Engine

Problem statement:

- 🔌 *Plug and play model:* we can integrate any where with minimal code changes.
- 📦 *Highly extendable:* sms, email, pop-up, …
- 🧩 *Notification modifiable:* Modify notification dynamically. (adding headers or footers)
- 🗂️ *Store notifications / logging:* save all notifications.

Design Approach:

To build a ***scalable*** and ***modular*** **notification engine**, the following **design patterns** are implemented:

1. 🧠 Strategy Design Pattern:
    - Allows ***dynamic selection*** of ***notification channels*** (e.g., Email, SMS, Pop-up).
    - Promotes ***flexibility*** and ***scalability***.
2. 🎨 Decorator Design Pattern:
    - Enables ***modification*** of ***notification*** content ***dynamically*** (e.g., adding headers, footers).
    - Promotes ***code reusability*** and separation of concerns.
3. 📢 Observer Design Pattern:
    - Provides a ***loosely coupled mechanism*** to ***notify*** multiple components (e.g., delivery engines, loggers).
    - Facilitates ***real-time updates*** and ***event-driven*** **architecture**.
4. 🧿 Singleton Design Pattern (NotificationService):
    - ***Manages*** and ***coordinates*** the entire **notification flow**.
    - Acts as the ***central hub*** for ***storing*** and ***dispatching*** **notifications**.
    - Ensures ***consistent access*** and ***state management*** across the application.

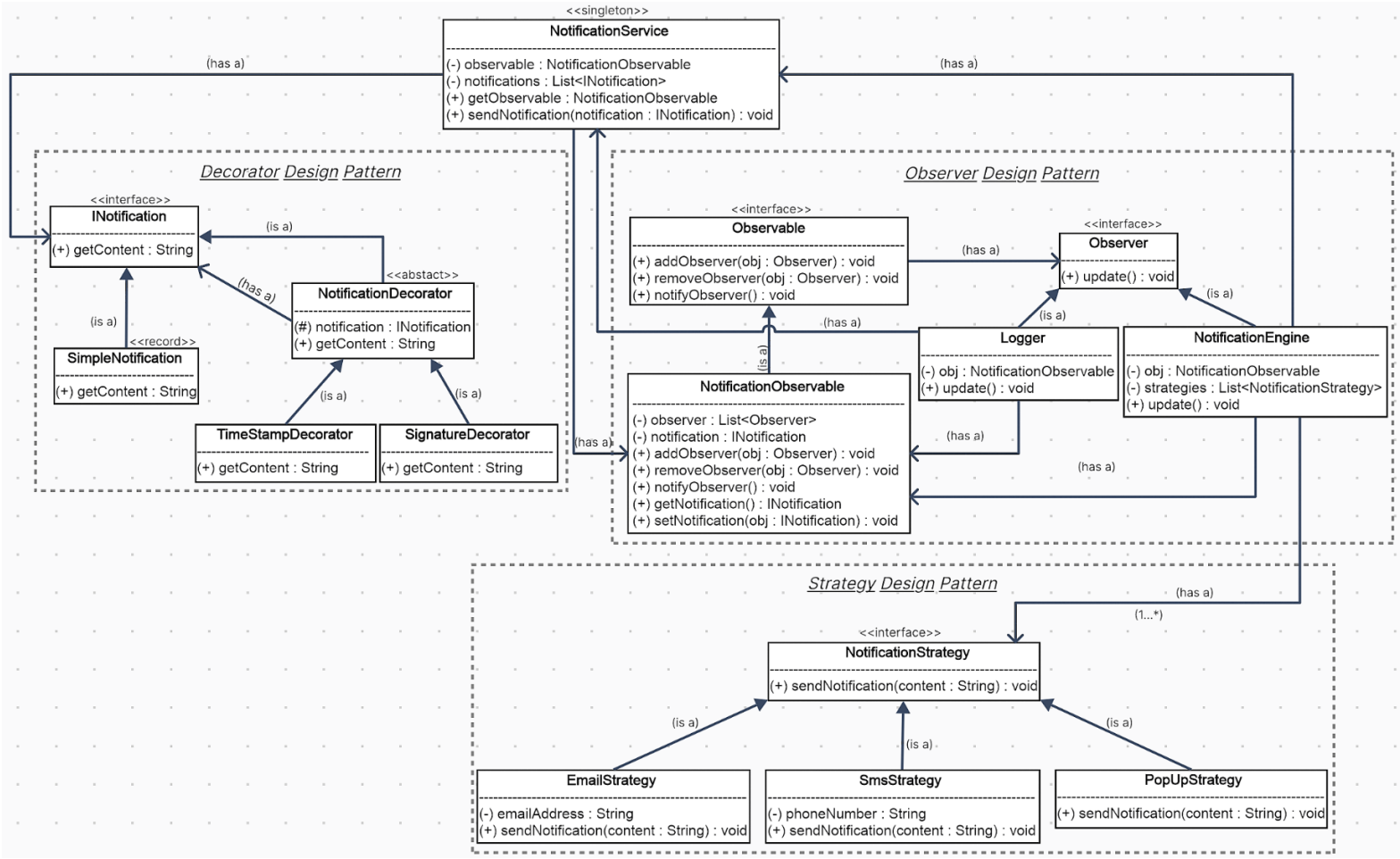
Conclusion: By combining ***Strategy***, ***Decorator***, and ***Observer*** **design patterns** with a ***Singleton*** **NotificationService**, the notification engine is:

- 🧱 **Modular**
- 📈 **Scalable**
- 🛠️ **Easily maintainable**
- 🔄 **Adaptable to changing business requirements**

This architecture supports ***plug-and-play*** **integration**, ***dynamic*** **content modification**, and ***reliable*** **notification tracking**.

@Sibasundarj8

UML design for notification system:



Code Link: https://github.com/sibasundarj8/-System-Design-/tree/main/Projects/NotificationEngine