Observer Design Pattern (Java) - Cheat Sheet

What is the Observer Design Pattern?

A behavioral pattern that creates a one-to-many dependency between objects: when the subject (observable) changes state, all registered observers are notified and updated automatically.

Why it's Useful

- Decouples subject from observers
- Observers can be added/removed dynamically
- Perfect for real-time updates & event-driven systems
- Supports clean, maintainable code

Key Participants

1. Subject / Observable: keeps list of observers, has attach(), detach(), notifyObservers()
2. Observer: interface with update()
3. ConcreteObserver: implements Observer, updates itself
4. Client: wires observers and subject together

Typical Flow

Observer subscribes -> Subject changes state -> Subject calls notifyObservers() -> Observers' update() methods are called

Real-world Use Cases

- UI toolkits & event listeners
- Chat apps (new message notifications)
- News publisher/subscriber systems
- Stock ticker apps
- File system watchers

Summary

Lets multiple objects observe another object and automatically get notified of changes, without tight coupling.