MAYFLOWER

Observer Pattern

Pattern DoJo, 09. März 2021

Definition

Das Pattern dient der Weitergabe von Änderungen an einem Objekt an von diesem Objekt abhängige Strukturen.

Cassandra liebt klare Codestrukturen!

Cassandra und ihr Team haben eine Anwendung für eine Nachrichtenagentur entwickelt. Werden im Haus neue Nachrichten generiert, so werden diese unverzüglich an die kooperierende Agentur Bloomberg News weitergeleitet.

Die Anwendung läuft stabil und ist klar strukturiert. Schauen wir uns die Weitergabe der Nachrichten mal im Code an..

Alle handelnden Personen sind frei erfunden

Problem

Die Methode NewsAgency.updateNews(news: String) wird vom Agentursystem aufgerufen, wenn die bestehenden Nachrichten aktualisiert werden sollen.

Ist dies der Fall, so werden diese Änderungen an die Klasse NewsChannelBloomberg weitergegeben über einen Aufruf der Methode NewsChannelBloomberg.update(news: String).

NewsAgency1.kt

Neue Anforderungen

Nun wurde eine Kooperation mit den Nachrichtenagenturen CNS und Thomas Reuters geschlossen. Das Programm soll die generierten Nachrichten nun auch an diese beiden Agenturen übermitteln.

Problem

Analog zu der bestehenden Struktur im Code gibt die Methode NewsAgency.updateNews(news: String) beim Eintreffen neuer Nachrichten diese Änderungen nun auch an die Instanzen der Klassen NewsChannelCNS und NewsChannelReuters weiter.

NewsAgency2.kt

Technische Schulden

Der Code gibt die Nachrichten nun korrekt an alle drei kooperierenden Agenturen weiter, aber Cassandra hat trotzdem Technische Schulden produziert.

Die Klasse NewsAgency koppelt nun drei andere Klassen hart an den eigenen Kontext, obwohl diese drei Klassen ihren jeweils eigenen Kontext haben.

Dies macht die Klassen weniger flexibler, weniger wiederverwendbar und die Struktur im Code spröder.

Anwendbarkeit

- Das Observer Pattern findet Anwendung, wenn eine Abstraktion mehrere Aspekte hat, die von einem anderen Aspekt derselben Abstraktion abhängen.
- Änderung eines Objekts sollen Änderungen an anderen Objekten nach sich ziehen.
- Ein Objekt soll andere Objekte benachrichtigen können, ohne diese im Detail zu kennen.

Implementierung

1. Das beobachtete Objekt (Subjekt, Observable) bietet einen Mechanismus, um Beobachter (Observer) an- und abzumelden und diese über Änderungen zu informieren.

2. Das Subjekt kennt alle seine Beobachter nur über die minimale Schnittstelle Observer. Über diese werden die Änderungen völlig unspezifisch weitergegeben.

3. Die Beobachter implementieren ihrerseits eine spezifische Methode, um auf die Änderung zu reagieren.

Lösung

Die Kontextklasse NewsAgency kommuniziert mit den Kanälen ausschließlich über die neu eingeführte Schnittstelle NewsChannel. Änderungen werden über die Funktion NewsChannel.update(news: String) weitergegeben.

Damit sich Beobachter am Subjekt NewsAgency an- und abmelden können, werden entsprechende Funktionen zur Verfügung gestellt.

NewsAgency3.kt

Lösung

Viele Sprachen bringen für das Observer Pattern eigene API Klassen mit. Beispielsweise sind die Klasse java.util.Observable und die Schnittstelle java.util.Observer von Beginn an fester Bestandteil der Java API.

Durch den Einsatz dieser standardisierten Elemente aus der API reduzieren wir die Anzahl eigener Klassen reduzieren.

NewsAgency4.kt

Cassandra hat das Observator Pattern richtig angewandt!

Cassandra hat mit dem Einsatz des Observer Patterns die Grundlage für eine gute Entkopplung aller Kontextklassen geschaffen, mit der die Anwendung auch in Zukunft gesund wachsen kann.

Vorteile

- Subjekte und Beobachter können unabhängig variiert werden. Die beiden sind auf abstrakte und minimale Art lose gekoppelt.

- Das beobachtete Objekt braucht keine Kenntnis über die Struktur seiner Beobachter zu besitzen, sondern kennt diese nur über die Beobachter-Schnittstelle.

- Das Pattern ermögloicht eine hohe Unabhängigkeit der beteiligten Komponenten.

Nachteile

- Die Komplexität der Anwendung wird erhöht.
- Änderungen am Subjekt führen bei einer großen Anzahl an Beobachtern zu hohen Änderungskosten. Zusätzlich können die Änderungen weitere Änderungen nach sich ziehen und so einen unerwartet hohen Aufwand haben.
- Bei Änderungen am Subjekt informiert das Observable alle Beobachter, auch wenn diese die Änderungsinformationen gar nicht benötigen.
- Der Mechanismus liefert keine Information darüber, was sich geändert hat sondern liefert lediglich die aktuellste Version eines Zustands.

Ja zum Observer Pattern!

Das Pattern ermöglicht eine Entkopplung der beteiligter Komponenten da die Weitergabe von Änderungen über eine kontextunabhängige Schnittstelle erfolgt.

Developer, Würzburg

Es macht Sinn, die standard Klassen Observer und Observable aus der API zu verwenden, da es die Anzahl an eigenen Klassen reduziert.

Developer, München

Zitate dienen nur Veranschaulichungszwecken

Nein zum Observer Pattern!

Die Erhöhung der Komplexität im Code rechnet sich erst ab einer gewissen Anzahl an Observern.

Developer, Berlin

Im Programmcode des Subjekts ist nicht mehr genau erkennbar, welche Beobachter bei Änderungen eigentlich informiert werden.

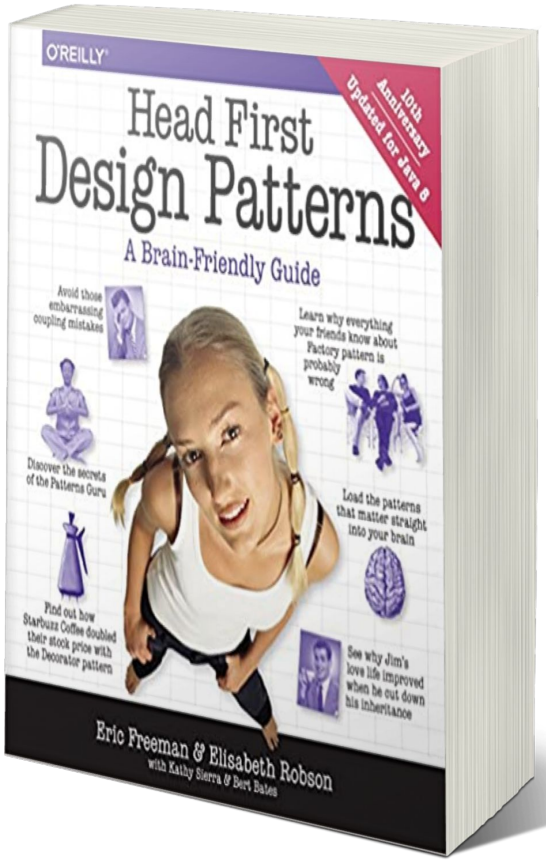
Developer, Würzburg

Zitate dienen nur Veranschaulichungszwecken

Vielen Dank für Ihre Aufmerksamkeit!

Die wichtigsten Design Patterns werden in dem modernen Klassiker Head First Design Patterns ausführlich behandelt.

Mit dem einzigartigen Head First Lernkonzept gelingt es den Autoren, die Materie witzig, leicht verständlich und dennoch gründlich darzustellen. Das ist nicht nur unterhaltsam, sondern auch effektiv.



Quellen und mehr Info

O’Reilly - Head First Design Patterns
Observer Pattern - Refactoring Guru
Wikipedia - Beobachter (Entwurfsmuster)
Google Slides Template “Zündende Idee”
Try the Kotlin Programming Language!
Code Beispiele auf GitHub


Du kannst mithelfen, das Wissen über Modernisierungs- und Design-Patterns weiter zu verbessern, indem Du Dein Wissen in der Gruppe Dev teilst.