HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

Blatt 9

Software Engineering 2

Luis Staudt

Aufgabe 1

Klassendiagramm

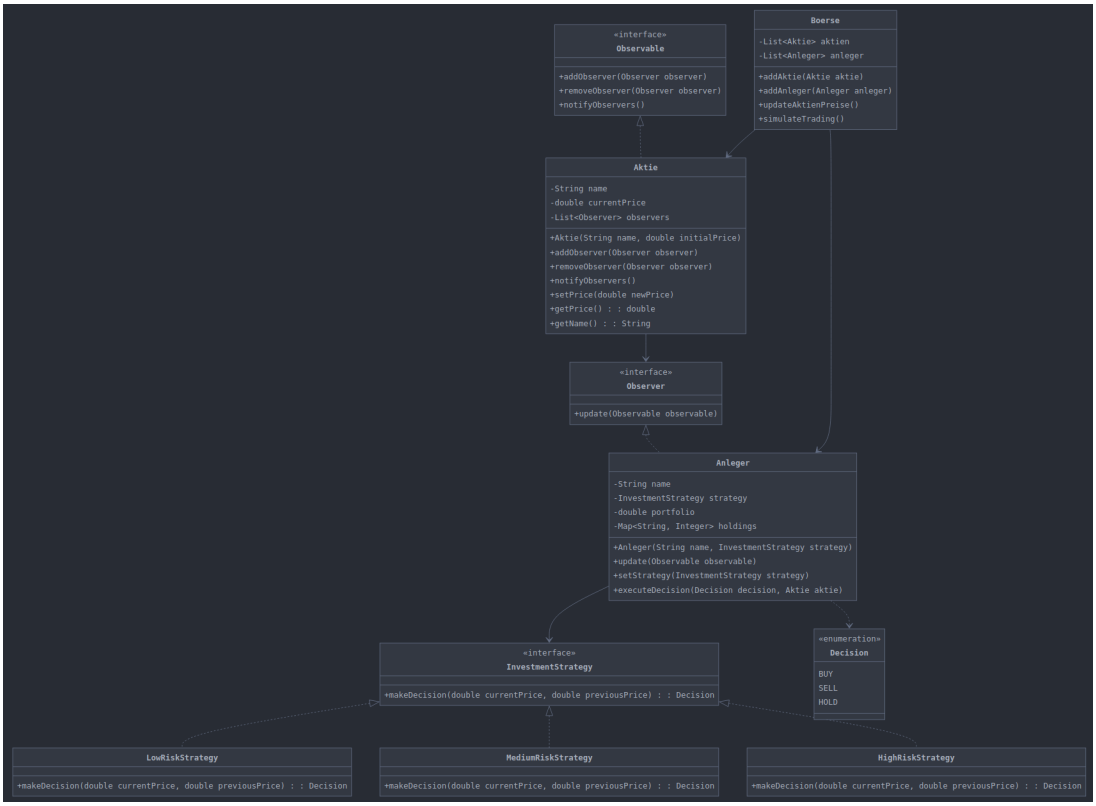

Abbildung 1: Klassendiagramm für Börsen-Aktienhandel mit Observer- und Strategy-Pattern

Sequenzdiagramm

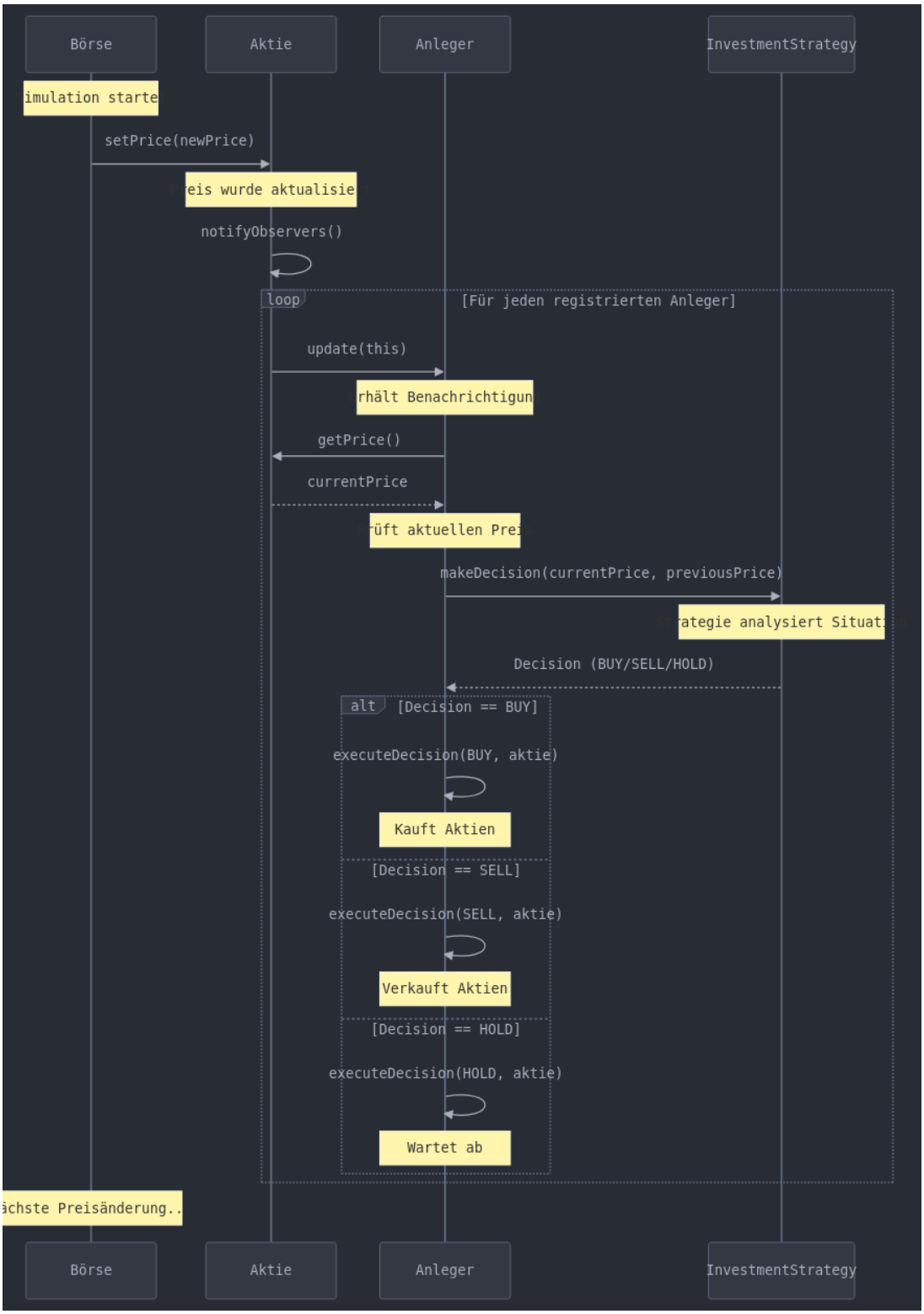

Abbildung 2: Sequenzdiagramm für Aktienhandel-Simulation

Erläuterung der Entwurfsmuster

0.0.1 Observer Pattern

Das Observer Pattern wird verwendet, um die Anleger automatisch über Änderungen der Aktienkurse zu benachrichtigen:

- Die `Aktie` fungiert als Subject/Observable
- `Anleger` sind Observer, die über Preisänderungen benachrichtigt werden
- Bei Preisänderungen werden alle registrierten Anleger automatisch informiert
- Dies ermöglicht eine lose Kopplung zwischen Aktien und Anlegern

Strategy Pattern

Das Strategy Pattern ermöglicht es, verschiedene Anlagestrategien zur Laufzeit auszuwählen:

- `InvestmentStrategy` definiert das Interface für verschiedene Anlagestrategien
- Konkrete Strategien implementieren unterschiedliche Entscheidungslogiken:
  - `LowRiskStrategy`: Konservative Anlagestrategie
  - `MediumRiskStrategy`: Ausgewogene Anlagestrategie
  - `HighRiskStrategy`: Risikoreiche Anlagestrategie
- Anleger können ihre Strategie zur Laufzeit wechseln

Aufgabe 2

Klassendiagramm

Abbildung 3: Klassendiagramm für Kaffeemaschine mit State-Pattern

Zustandsdiagramm

Das gegebene Zustandsdiagramm zeigt die verschiedenen Zustände der Kaffeemaschine und die Übergänge zwischen ihnen. Die Implementierung folgt diesem Diagramm exakt.

Erläuterung des State Patterns

Das State Pattern ermöglicht es der Kaffeemaschine, ihr Verhalten abhängig vom aktuellen Zustand zu ändern:

- **Context:** `CoffeeMachine` verwaltet den aktuellen Zustand und delegiert Aktionen an das State-Objekt
- **State Interface:** `CoffeeMachineState` definiert die gemeinsame Schnittstelle für alle Zustände
- **Concrete States:** Jeder Zustand implementiert das Verhalten für alle möglichen Aktionen
- **Zustandsübergänge:** Werden durch die State-Objekte selbst gesteuert

Vorteile des State Patterns

- Saubere Trennung der zustandsspezifischen Logik
- Einfache Erweiterbarkeit um neue Zustände
- Vermeidung von komplexen if-else Strukturen
- Klare Zuständigkeiten für jeden Zustand
- Zustandsübergänge sind explizit und nachvollziehbar