Abgabe Übungsblatt 7

Ben Zöttl 457736 Deni Ismailov 454935 Ahmed Assahub 442562

Dezember 2025

Aufgabe 7.1) Entwurfsmuster

a) Verfeinerungen mit Analysemustern

Die folgenden Verfeinerungen des Analysemodells werden unter Anwendung geeigneter Analysemuster durchgeführt.

i) Quests: Individuelle Aufgaben oder komplexe Quests

- **Muster: Composite** (Kompositum)
- **Begründung:** Das Composite-Muster erlaubt es, hierarchische Strukturen abzubilden, indem es eine uniforme Schnittstelle für Einzelobjekte (`IndividuelleAufgabe`) und Containerobjekte (`KomplexeQuest`) bereitstellt. Dies ermöglicht die Zusammensetzung von Quests aus Sub-Quests.
- **Anwendung:** Die Klasse `Quest` wird zur abstrakten Komponente.
  - `IndividuelleAufgabe` (Blatt): Erbt von `Quest`, enthält Aufgabenbeschreibung.
  - `KomplexeQuest` (Container): Erbt von `Quest`, besitzt eine Assoziation zu einer Sammlung von `Quest`-Komponenten (Sub-Quests).

ii) Abschluss-Status und Fortschritt der Quest pro Player-Charakter

- **Muster: Association Class** (Assoziationsklasse)
- **Begründung:** Die Informationen (Fortschritt, Status, Zeitpunkte) gehören nicht nur zur `Quest` oder zum `PlayerCharakter`, sondern zur **spezifischen Beziehung** zwischen beiden. Eine Assoziationsklasse ermöglicht die Kapselung dieser relationalen Attribute.
- **Anwendung:** Die Assoziation zwischen `PlayerCharakter` und `Quest` wird durch die Assoziationsklasse `QuestFortschritt` qualifiziert.
  - `QuestFortschritt` enthält Attribute wie `String status`, `int fortschritt`, `Date startZeitpunkt`, `Date endZeitpunkt`.

iii) NPC als Verkäufer, Gegner oder keines

- **Muster: Role-Muster** (abgeleitet vom Strategy-Muster)
- **Begründung:** Da die Rollen optional sind, wechselbar und nicht exklusiv, ist das Role-Muster ideal, um die flexiblen Verhaltensweisen (`Verkäufer`, `Gegner`) von der Hauptklasse `NPC` zu entkoppeln.
- **Anwendung:**
  - Interface/Abstrakte Klasse `NPCRolle` wird definiert.
  - Konkrete Klassen `VerkäuferRolle` und `GegnerRolle` implementieren `NPCRolle`.
  - `NPC` erhält eine optionale Assoziation (0..1) zur `NPCRolle`.

iv) Unterscheidung zwischen Item-Exemplar und Item-Art

- **Muster: Type-Object** (Typobjekt)
- **Begründung:** Gemeinsame, statische Eigenschaften (`name`, `beschreibung`) werden in einer separaten **Typklasse** (`ItemArt`) zusammengefasst. Die individuelle Klasse (`Item`) behält nur die variablen, exemplarspezifischen Eigenschaften (`zustand`).
- **Anwendung:**
  - Neue Klasse `ItemArt` enthält `String name` und `String beschreibung`.
  - `Item` (Exemplar) behält `Zustand zustand` und erhält eine 1:1-Assoziation zu `ItemArt`.

v) Wert jeder Item-Art für einen NPC

- **Muster: Association Class** (Assoziationsklasse)
- **Begründung:** Der `wert` ist ein Attribut der **Beziehung** zwischen einem `NPC` (der bewertet) und einer `ItemArt` (die bewertet wird), nicht nur Attribut einer einzelnen Klasse.
- **Anwendung:** Die Assoziation zwischen `NPC` und `ItemArt` wird durch die Assoziationsklasse `Handelswert` qualifiziert.
  - `Handelswert` enthält das Attribut `int wert`.

b) Verwaltungsklasse und Entwurfsmuster

- **Verwaltungsklasse:** Eine Klasse `Spielverwaltung` wird eingefügt.
- **Zuständigkeiten:** Die Klasse verwaltet die Erzeugung, das Löschen, Laden und Speichern von `Account`- und `PlayerCharakter`-Objekten.
- **Entwurfsmuster: Singleton**
- **Begründung:** Das Singleton-Muster stellt sicher, dass die `Spielverwaltung` als zentrale Steuereinheit nur einmal im System existiert, was für globale Verwaltungsaufgaben sinnvoll ist.

c) Assoziation qualifizieren

Die Assoziation zwischen `Account` und `PlayerCharakter` wird qualifiziert, um den direkten Zugriff auf Charaktere über ihren Namen zu ermöglichen.

- **Assoziation:** `Account` $\xrightarrow{\text{charaktere } *}$ `PlayerCharakter`
- **Qualifizierer:** `String charakterName`
- **Anwendung:**

$$\text{Account} \xrightarrow{\text{charaktere [charakterName] : 1}} \text{PlayerCharakter}$$

  Die qualifizierte Assoziation ermöglicht es, dass ein `Account` einen bestimmten `PlayerCharakter` über den `charakterName` eindeutig identifiziert.