ENTWURFSMUSTER

Studientag 01853

Moderne Programmiertechniken und -methoden

GRUNDPRINZIPIEN DER OOP

- Program to an interface, not an implementation.
  - Interface-als-Typ

GRUNDPRINZIPIEN DER OOP

- Favour object composition over inheritance.
  - Umgehung der Kopplung/Aufrufabhängigkeit von Sub- zu Superklasse
- Komposition
  - Forwarding (bindet this auf das aufgerufene Objekt)
  - Delegation (bindet this auf das aufrufende Objekt)

ENTWURFSMUSTER

- schemenhafte Beschreibung einer Lösung für ein Standard-Problem

- als solches zählt, wenn sich mind. drei konkrete Beispiele dafür finden lassen

- ist also zu komplex, um sich mit parametrischem Polymorphismus (Generics) umsetzen zu lassen, aber dennoch wiederkehrend

COMPOSITE PATTERN

- Teil-Ganzes Beziehungen
  - Grafik (Ebenen und Gruppen)
  - Dateisystem
- zwei Arten
  - klassisch: gemeinsames Interface für Teile und Ganzes
  - erweitert: zwei Interfaces für Teile und Ganzes

OBSERVER PATTERN

- An- und Abmeldung von Observern
- publish-subscribe Verfahren, Benachrichtigung als Multicast
- Subjekt muss nicht dauerhaft vom Observer referenziert werden (kann Teil der Nachricht sein)
- Starke Kopplung vom Observer an das Subjekt (aber nicht umgekehrt)
- Java: z.B. Event-Listener

TEMPLATE METHOD PATTERN

- Auslagerung eines Teils eines Algorithmus mittels offener Rekursion
  - offene Rekursion auf abstrakte Methode
  - offene Rekursion auf Methode mit Default-Implementierung (Hook)

STRATEGY PATTERN

- Auslagerung eines Teils eines Algorithmus mittels Komposition
- Teil ist in eigener Klasse gekapselt, mehrere Implementierungen über ein Interface möglich
- kein Zugriff auf Daten des Kontexts (da keine Vererbung)
- dafür flexible Auswahl (und dynamische Änderung) der Implementierung

ROLE OBJECT PATTERN

- Rollen bieten verschiedene Sichten auf ein Subjekt (Objekt)
- können zur Laufzeit hinzugefügt und gelöscht werden
- Rollen können Anfragen auf eigentliches Subjekt forwarden oder delegieren
- i.d.R. muss Identitätstest zweier Rollen positiv verlaufen
  - sonst: Objektschizophrenie (eine logische Instanz, mehrere Identitäten)

FACTORIES

- Problem: Name des Konstruktors muss zur Objekterzeugung bekannt sein

- Dependency Injection

- Factory-Methode (als Klassen- oder Instanzmethode)

FACTORIES II

- Abhängigkeit des Clients gegen die Factory
- Unterbringung der Factory in
  - abstrakter Basisklasse
  - innerer Klasse eines Interfaces

FACTORY METHOD PATTERN

- auch: virtueller Konstruktor, da Aufruf dynamisch gebunden wird

- abstrakte Basisklasse gibt abstrakte Factory-Methode vor

  - Rückgabetyp ist mit dem der Subklasse nicht per Subtyping verwandt

  - durch implizite Fallunterscheidung beim dynamischen Binden wird ein Parameter der Methode erspart

ADAPTER PATTERN

- Adapter stellen für aufrufende Klassen das benötigte Interface zu einer anderen, nicht zuweisungskompatiblen Klasse bereit
- letzteres wird aber nicht durch den Adapter implementiert, sondern an eine andere Klasse delegiert, per
  - Vererbung
  - Forwarding

FASCADE PATTERN

- Fassade kapselt ein Subsystem (mehrere Teile) hinter einer einfachen Schnittstelle
- vermindert so die Kopplung
- erhöht Wartbarkeit - Subsystem kann komplett ausgetauscht werden

VISITOR PATTERN

- Problem: Klassenhierarchie implementiert gleiche Methoden, aber jede Implementierung ist von jeder anderen so verschieden, dass Vererbung nicht genutzt werden kann
- Bsp: AST eines Editors oder Compilers
- typischer crosscutting concern: zwei unterschiedliche Ordnungen, keine kann der anderen untergeordnet werden
- Einführung einer neuen Methode erfordert Änderung aller Klassen der Hierarchie

VISITOR PATTERN II

- Lösung: Methoden werden als Klasse eingeführt, die besonderen Zugriff auf die eigentlichen Klassen und deren Interna haben
- Name der Funktionalität im Klassennamen
- Name der bearbeiteten Klasse im Methodennamen (oder gleicher Name bei Überladung)
- erbrachtes Opfer: besuchte Klasse macht Teil ihres Implementationsgeheimnisses öffentlich

VISITOR PATTERN III

- Umsetzung
- dynamisches Binden anhand der Empfänger- und Parametertypen
- Double Dispatch: Simulation des o.g. Falls in z.B. Java

VORTEILE VON PATTERNS

- gemeinsames Vokabular für Design-Entscheidungen
- Erhöhung der Code-Qualität