JAVA DAYS ´99



Muster in Java Geschäftsanwendungen

Tom Krauß
Bereichsleiter Internet
GEBIT Gesellschaft für EDV-Beratung und
Informatik-Technologien mbH

Sun microsystems

Übersicht



- Entwurfsmuster und Geschäftsanwendungen
- Architektur und Rahmen von Geschäftsanwendungen
- Detailbetrachtung einiger Muster

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Entwurfsmuster und Geschäftsanwendungen



- Entwurfsmuster erstmals eingeführt 1978 in der Architektur (Christopher Alexander)
- Populär gemacht in der Informationstechnologie durch die „Gang of Four“ 1995
- Es kommen immer häufiger dokumentierte Entwurfsmuster zum Einsatz
- Geschäftsanwendungen werden nach ähnlichen Mustern entwickelt

Entwurfsmuster sind der revolutionärste Einfluß auf die Informationsverarbeitung in den 90-er Jahren
(Doug Lea)

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Was ist ein Entwurfsmuster?



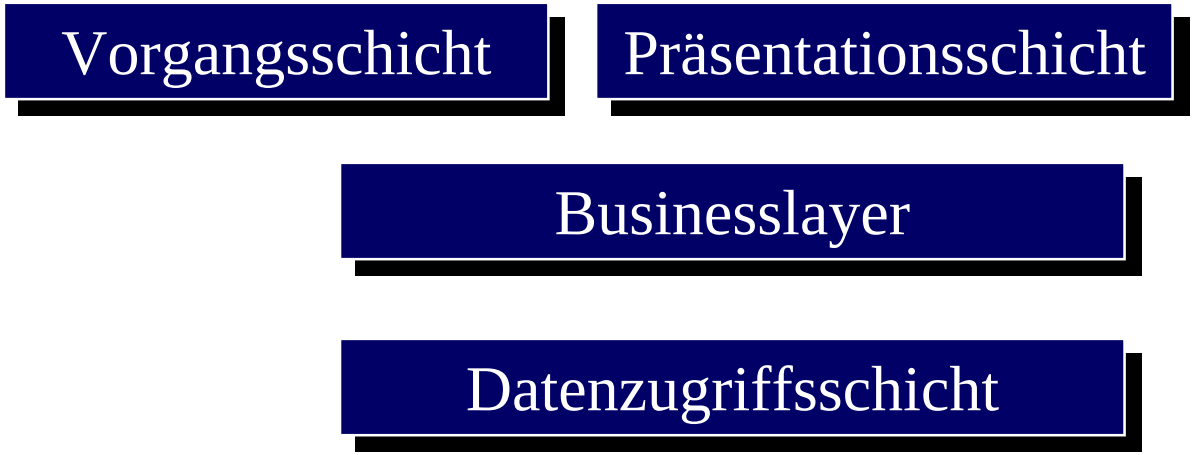
- Ein Entwurfsmuster
  - ... hat einen Namen, der z.B. in Fachgesprächen verwendet werden kann, und die Kommunikation vereinfacht
  - ... hat eine Problembeschreibung, die darauf hinweist, wann es zum Einsatz kommt
  - ... hat einen Vorschlag für eine generelle Problemlösung
  - ... hat positive und unter Umständen auch negative Konsequenzen

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Geschäftsanwendungen
Logisches Schichtenmodell



- Typische **logische** Schichten einer Geschäftsanwendung
- Die Anzahl der Schichten hängt von der Komplexität der Anwendung ab



(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

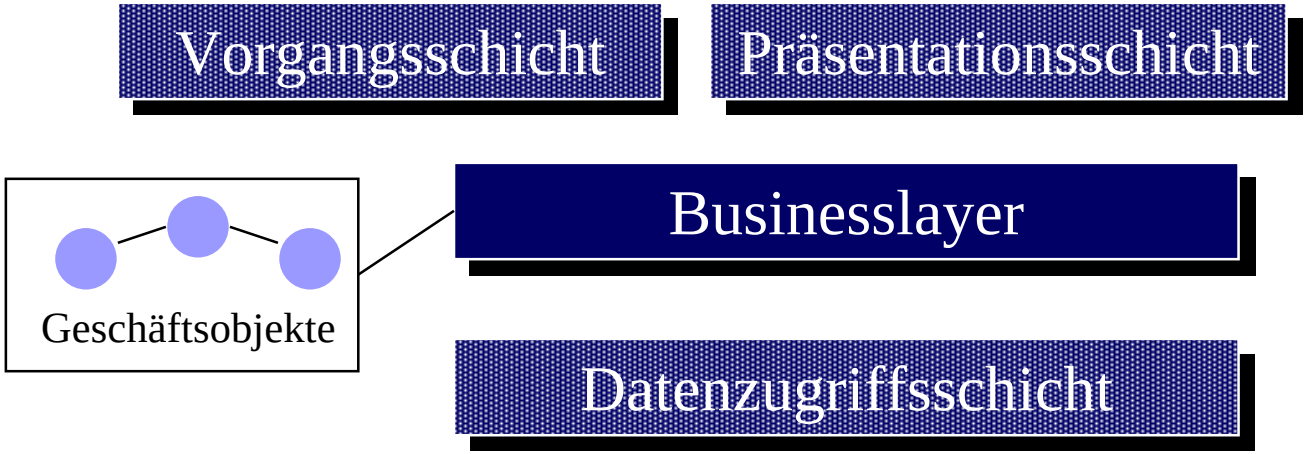
Implementierung des Businesslayers durch Geschäftsobjekte



- Definition

  Fasse die Funktionen, die eine eingegrenzte Geschäftsdomäne betreffen, zu einer Klasse zusammen. Beachte, daß diese Klasse keine Abhängigkeiten zum Anwendungskontext zeigt

- Geschäftsobjekte sind meist persistent

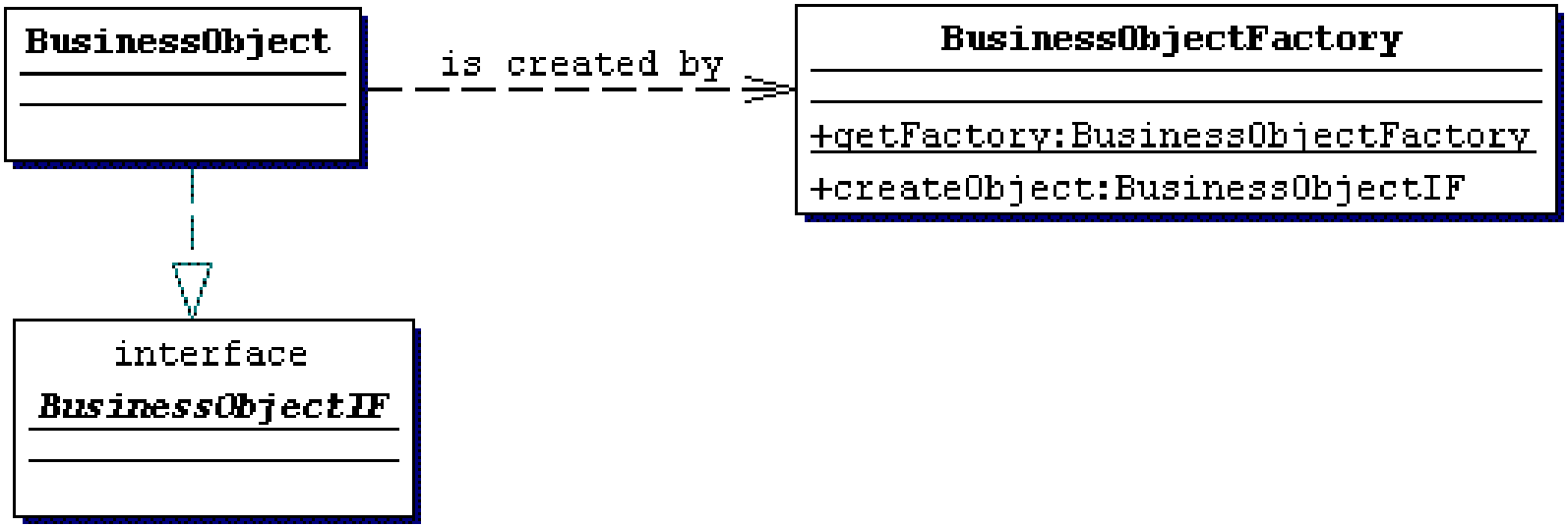


(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Geschäftsobjekt
Entwurfsmuster in der Praxis



- Typische Repräsentation und Erzeugung eines Geschäftsobjekts in Java-Anwendungen
- Entwurfsmuster: Abstrakte Fabrik [GoF1995] und Interface [Grand1998]



(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Geschäftsobjekt
Beispielverwendung




```
ContractIF contract;

// Erzeugt tatsächlich eine Instanz von z.B.
// Contract.class (implementiert ContractIF)
contract = (ContractIF)BusinessObjectFactory.
           getFactory().
           createObject(ContractIF.class);

contract.setContractNumber(new Integer(1));
contract.setPartner(partner);
```


(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Geschäftsobjekt
Begründung für diese Implementierung



- Vermeidung der strengen Kopplung von Anwendung und konkreten Implementierungsklassen
- Konfigurierbarkeit der Anwendung abhängig vom Kontext
  - Beispiel: lokales versus entferntes Objekt

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Entwurfsmuster in Geschäftsanwendungen
Das Muster „Reference Data Objects“



- Reference Data Objects
- Activation Framework
- Domain Adapter
- Lazy Evaluation

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Reference Data Objects



- Definition

  Mache Werte eines eingeschränkten, wenig veränderlichen Wertebereichs zu Geschäftsobjekten. Markiere sie mit einem speziellen *Interface.* Führe ein Objekt ein, das für Verwaltung der Instanzen abhängig vom Typ verantwortlich ist

- Anmerkungen

  - Reference Data Objects sind ebenso wie Geschäftsobjekte meist persistent

- Variante

  - Das Markierungs-*Interface* kann auch Dienste definieren, z.B. ein Standardverfahren für die Repräsentation der Werte

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

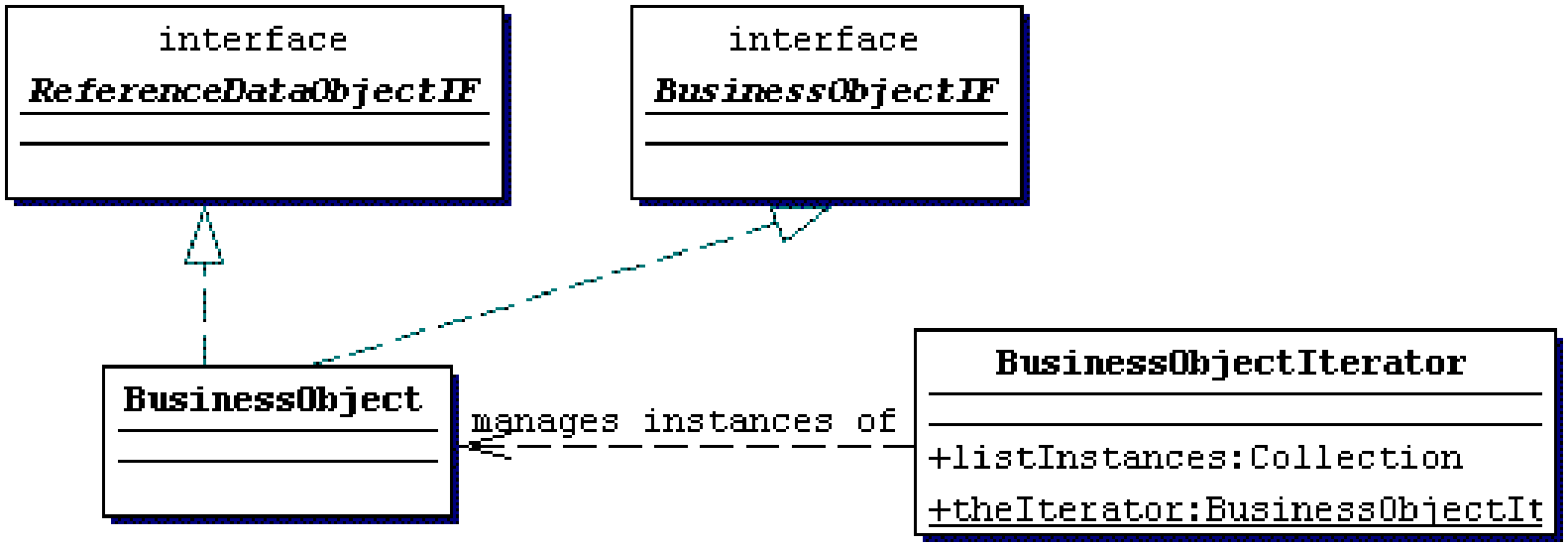
Reference Data Objects
Verwendete Muster



- Iterator (Iterator) [GoF1995]
- Markierungs Interface (Marker Interface) [Grand1998]
- Cache Management [Grand1998]
- Unveränderliches Objekt (Immutable) [Grand1998]

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Reference Data Objects
UML-Modell



(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Reference Data Objects
Konsequenzen



- Verwendungsmöglichkeit der gleichen Persistenzmechanismen wie für Geschäftsobjekte
- Performanz: die Iteratoren können in einem Cache gehalten werden, da alle Daten nur einmal vom Persistenzmedium gelesen werden müssen.
- Die Markierung von Referenzdatenobjekten bietet Rahmenwerken die Möglichkeit zur Sonderbehandlung
- Ein Performanz-Nachteil kann entstehen, wenn die Wertemengen eines Referenzdatentyps sehr groß sind.
- Gegen die Markierung mit einem Interface kann sprechen, wenn die Eigenschaft „ist Referenzdatenobjekt“ sich ändern kann
- Die Schaffung eines eigenen Typs für die Referenzdaten-Werte bringt erhöhte Flexibilität, aber auch einen gewissen Overhead

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Reference Data Objects
Beispiel: Auswahlelement für aufzählbare Werte



- Das Interface `PermanentBOIF` definiert, wie die Werte eines Typs von Referenzdaten als String repräsentiert werden können (siehe Variation des Musters Reference Data Objects)
- Ein Cache verwaltet die für bestimmte Typen verantwortlichen Iteratoren
- Eine einfache Spezialisierung des JFC `JComboBox`-Elements bietet ein einfaches allgemeines Auswahlverfahren für alle Objekte einer Klasse, die dieses Interface implementieren

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Reference Data Objects
Beispiel-Implementierung




```
public class ReferenceDataSelector extends JComboBox {
  public ReferenceDataSelector() {
    setRenderer(new DefaultRenderer() {
      public Component getListCellRendererComponent(value,...
        value =
         ((PermanentBOIF)value).print(getLocale());
          return super.getListCellRendererComponent(value,...
     });
  }
  public void displayAll(Class aReferenceDataType) {
    DefaultComboBoxModel model = new
      DefaultComboBoxModel();
    Collection temp = BusinessObjectIterator().
       theIterator().listInstances(aReferenceDataType);
    model.addAll(temp);
    setModel(model);
  }
}
```


(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Entwurfsmuster in Geschäftsanwendungen
Das Muster „Activation Framework“



- Reference Data Objects
- Activation Framework
- Domain Adapter
- Lazy Evaluation

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Activation Framework



- Definition

  Durch eine einheitliche Veröffentlichung der Schnittstellen von Dienstobjekten können diese in einer Applikation verwendet werden ohne dieser explizit bekannt zu sein. Die Schnittstellenbeschreibung enthält dabei Hinweise für eine visuelle Verwendung

- Anmerkungen

  - Die Veröffentlichung wird in der Regel explizit vom Dienstobjekt angestoßen, kann aber auch passiv erfolgen
  - Dieses Muster kommt bei Dienstobjekten unterschiedlichster Komplexität zum Einsatz
  - Sun‘s Plug & Play-Framework Jini™ ist ein praktisches Beispiel aus der Java-Welt für einen Aktivierungsrahmen

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

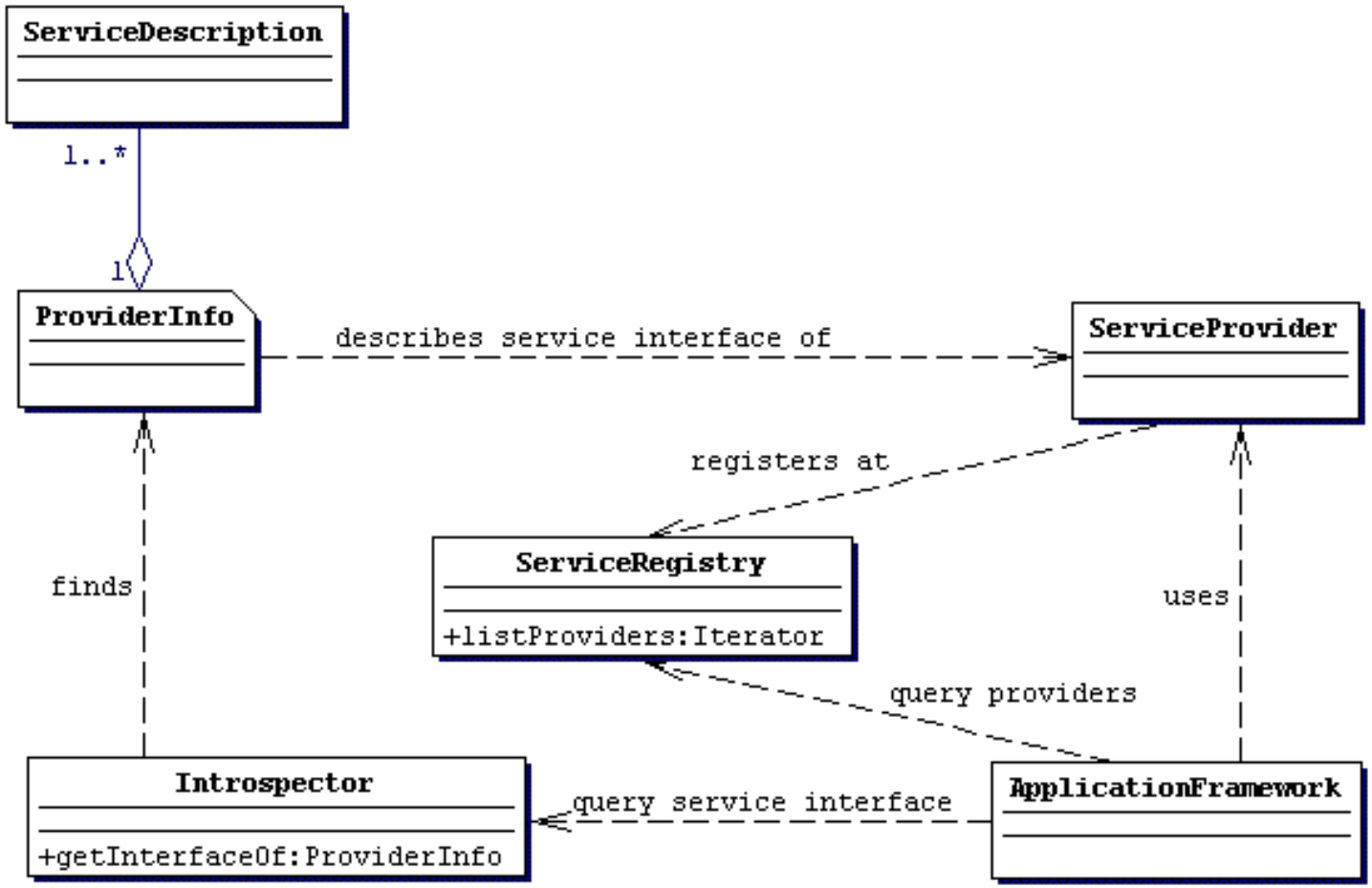
Activation Framework
Verwendete Muster



- Befehl (Command) [GoF19995]
- Objektentkopplung (Decoupling Objects) [ArcusProject]
- Weitere
  - Verwalter (Manager)
  - Lockere Kopplung durch Namenskonvention
    - Eine besonders lockere Kopplung zwischen den Dienstobjekten und den diese beschreibenden Klassen erreicht man durch Konventionen für die Benennung der Beschreibungsklassen

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Activation Framework
UML-Modell



(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Activation Framework
Konsequenzen



- Hohe Flexibilität: Nutzung neuer Funktionalitäten während der Laufzeit möglich
- Einschränkung der Sichtbarkeit von Diensten abhängig vom Anwendungskontext durch explizite Diensteschnittstellen (Black-Box Komponenten).
- Aktivierungsrahmen führen zu einer weiteren Entkopplung der dienstanbietenden Klassen von der Anwendung
  - Einfachere Wartbarkeit
  - Einfacheres Testen von isolierten Komponenten
- Integrationstest unter Umständen schwieriger: viele Varianten, die Konfiguration kann sich zur Laufzeit ändern

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Activation Framework
Beispiel: Installationsabhängige Menüs



- In einer Anwendung wird eine Auswahl von Werkzeugen in einem Menü angeboten
- Je nach Umfang der Installation sind einige verfügbar, einige nicht
- Das Aktivierungs-Framework der Anwendung erfragt die verfügbaren Werkzeuge von der `ServiceRegistry` und ermittelt die in den Werkzeugen implementierten Dienste durch den `Introspector` und erzeugt die Menüs dynamisch

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Activation Framework
Beispiel-Implementierung




```
void createToolMenu(JMenu topLevelMenu) {
  Iterator i = ServiceRegistry.theRegistry().
    listProviders();
  while(i.hasNext()) {
    ProviderInfo info = Introspector.theIntrospector().
      getInfo((ServiceProvider)i.next());
    JMenu menu = new JMenu(info.getProviderName());
    topLevelMenu.add(menu);
    Iterator j = info.getServices();
    while(j.hasNext()) {
      ServiceDescription description =
        (ServiceDescription)j.next();
      menu.add(description.getAction());
    }
  }
}
```


(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Entwurfsmuster in Geschäftsanwendungen
Das Muster „Domain Adapter“



- Reference Data Objects
- Activation Framework
- Domain Adapter
- Lazy Evaluation

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Domain Adapter



- Definition

  Führe die Abbildung von Geschäftsobjekten zu anderen Anwendungsschichten über verbindende Zwischenklassen aus

- Anmerkungen
  - Domain Adapter können unidirektional oder bidirektional sein
  - Die Verbindung zwischen Geschäftsobjekten und ihrer relationalen Datenrepräsentation wird insbesondere als *Objekt-Relationales Mapping* bezeichnet.

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Domain Adapter
Verwandte Muster



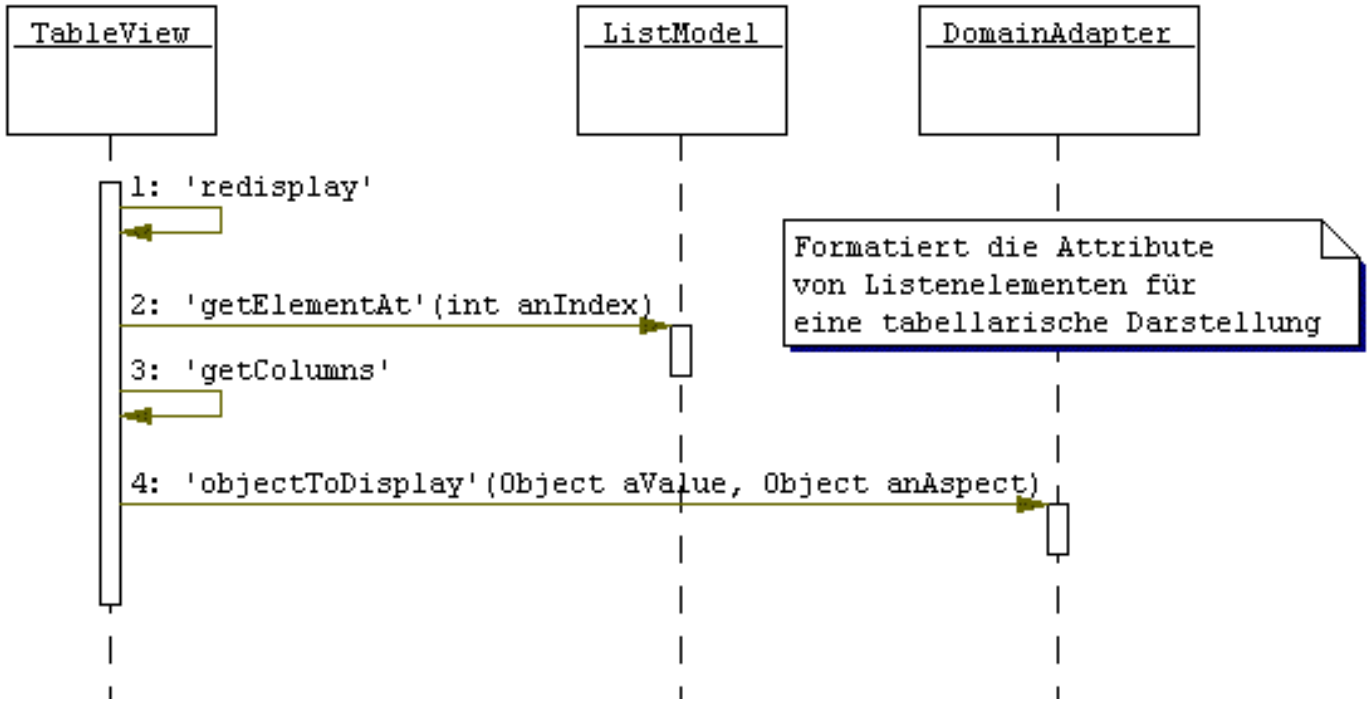
- Zuständigkeitskette (Chain of Responsibility) [GoF1995]
- Brücke (Bridge) [GoF1995]
- Filter [BMRSS1996]

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Domain Adapter
Anwendungsbeispiel 1: Vom Modell zur View



- Darstellung einer Collection von Geschäftsobjekten über Domain Adapter in einer Tabelle



(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

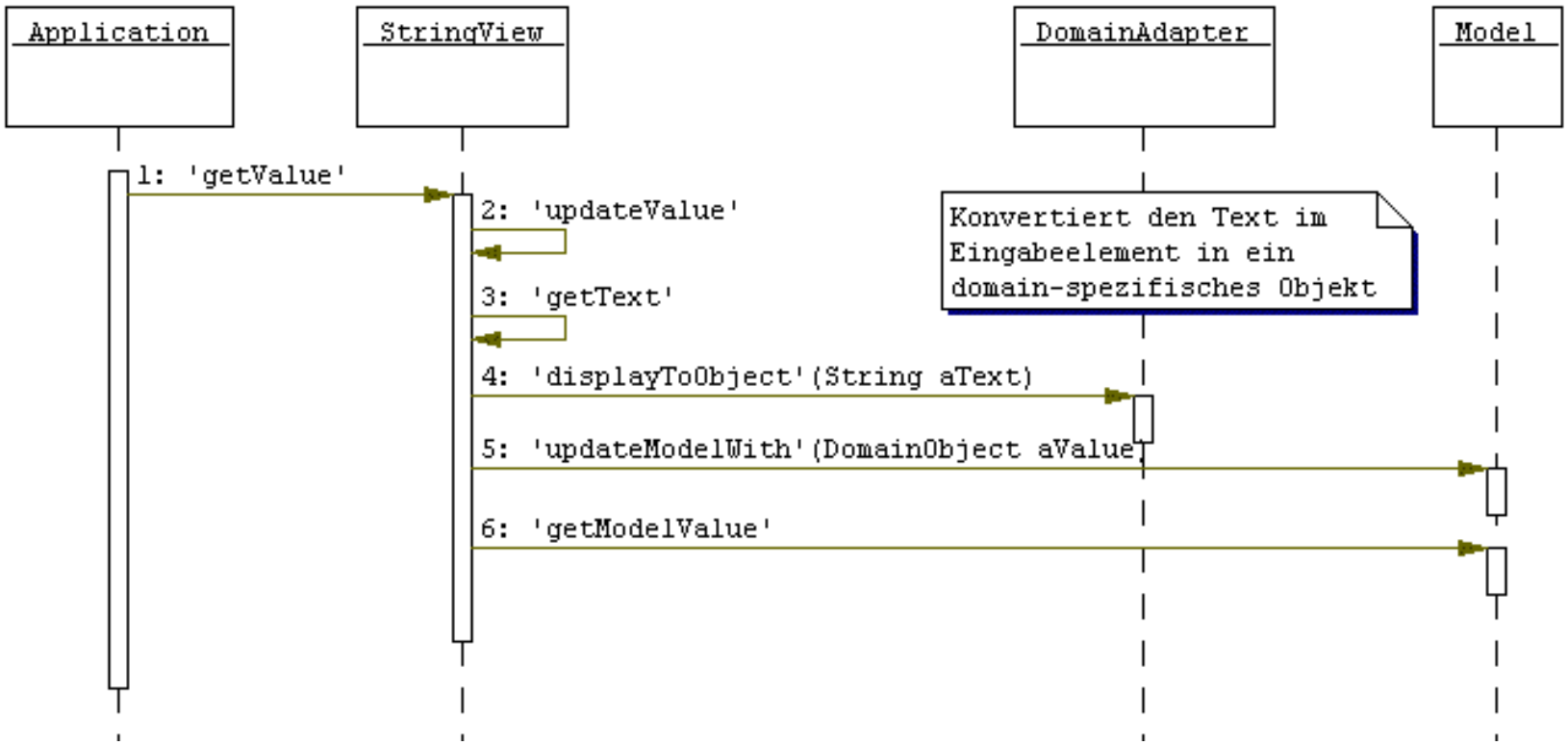
Domain Adapter
Schnittstellenbeschreibung




```
public interface TableDomainAdapter {
        /**
         * Liefert einen Aspekt von aValue dergestalt
         * formatiert, so daß dieser in einer Spalte
         * einer Tabellenkomponente angezeigt werden
         * kann.
         * @param aValue ein Objekt, das in einer
         * Zeile der Tabelle angezeigt werden soll
         * @param aColumnID der logische Bezeichner
         * der Tabellenspalte, der einen
         * entsprechenden Aspekt von aValue
         * kennzeichnet
         */
        public Object objectToDisplay(Object aValue,
                Object aColumnID);
}
```


(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Domain Adapter
Anwendungsbeispiel 2: Domainobjekterzeugung



(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Domain Adapter
Schnittstellenbeschreibung




```
public interface StringDomainAdapter {
        /**
         * Formatiert ein Anwendungsobjekt, das in
         * einem Eingabefeld dargestellt werden soll
         * @param aValue das Anwendungsobjekt
         */
        public Object objectToDisplay(Object aValue);

        /**
         * Parsiert einen Text zur Ermittlung des
         * Anwendungsobjekts
         * @param aText der Text, der geparst wird
         */
        public Object displayToObject(String aText);
}
```


(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Domain Adapter
Konsequenzen



- Beispiel 1: Bessere Wiederverwendung und Austauschbarkeit
  - Die Tabellen-View kann in diesem Beispiel auf einem allgemeinen List-Model arbeiten. Daraus folgt: an Stelle der Tabellen-View kann ohne weitere Änderungen eine beliebiges anderes ListView-Element verwendet werden.
- Beispiel 2: Fokussierung des Anwendungs-entwicklers
  - Für die Programmierung „spezieller Eingabefelder“ reicht die Implementierung eines Domain-Adapters statt eines Subclassing der verwendeten GUI-Komponente

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Entwurfsmuster in Geschäftsanwendungen
Das Muster „Lazy Evaluation“



- Reference Data Objects
- Activation Framework
- Domain Adapter
- Lazy Evaluation

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Lazy Evaluation



- Definition

  Führe Initialisierungen erst dann durch, wenn eine konkrete Objektanfrage durchgeführt wird

- Anmerkungen

  - Für die Implementierung der verzögerten Auflösung existieren verschiedene Varianten

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Lazy Evaluation
Verwandte Muster

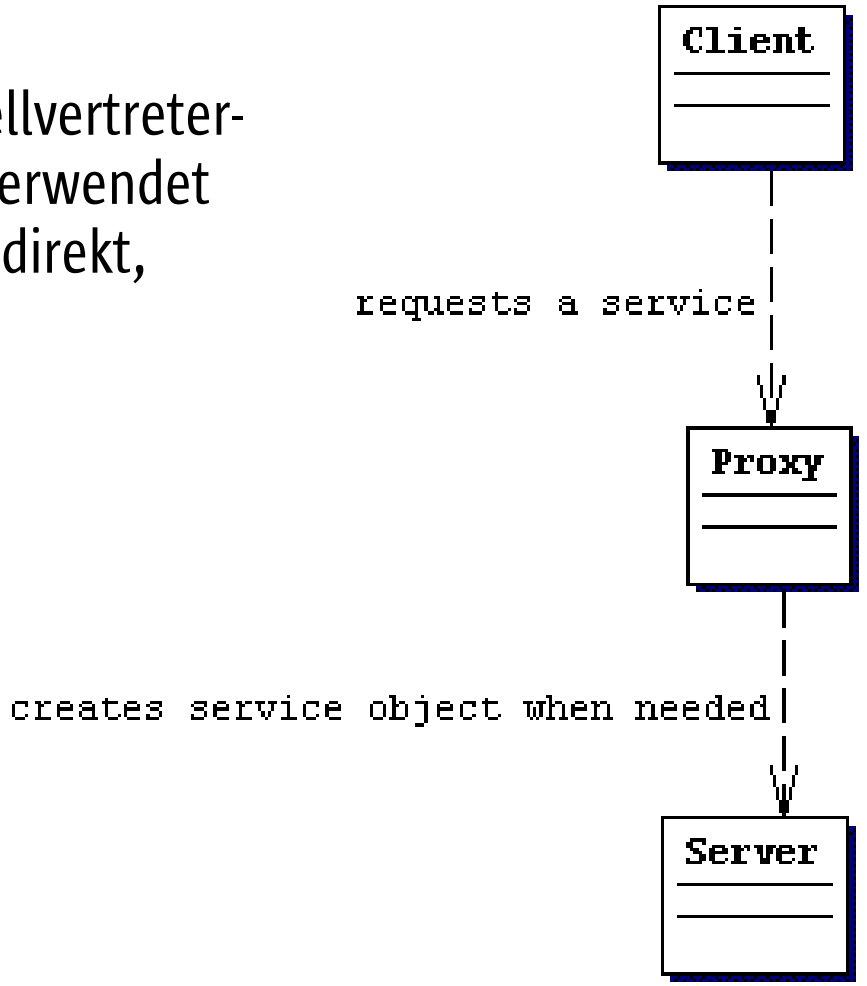


- Virtual Proxy [Larman1998]

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Lazy Evaluation
UML-Modell Variante 1

- Lösung über ein Stellvertreter-Objekt: Der Client verwendet einen Service nicht direkt, sondern über eine Indirektionsstufe

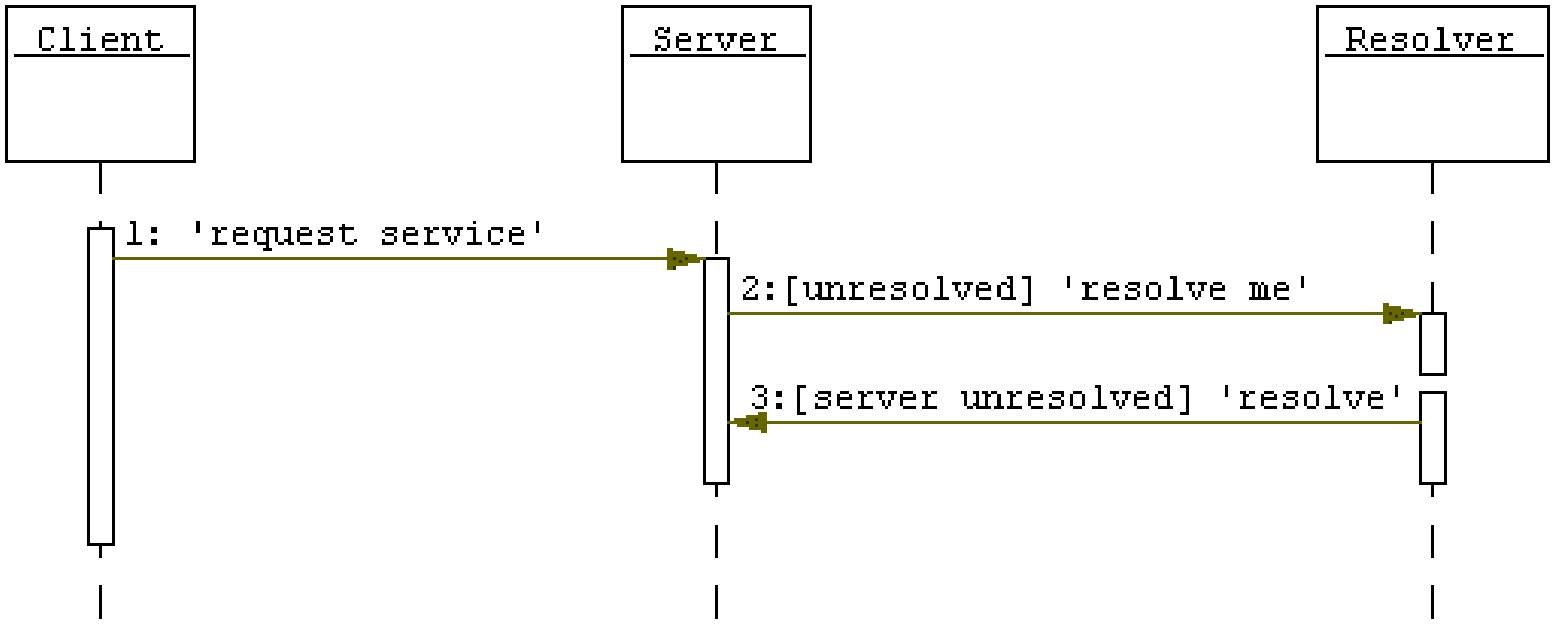


(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Lazy Evaluation
UML-Modell Variante 2



- Lösung über Prüfung vor der Ausführung: Vor der Aus-führung eines Dienstes fordert der Server eine Auflösung an



(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Lazy Evaluation
Konsequenzen



- Vermeidung von zeitintensiven Operationen
  - Dieses Muster wird oft in Geschäftsanwendungen, typischerweise beim Zugriff auf Objekte im Persistenzmedium eingesetzt
- Möglichkeit, unendliche oder quasi-unendliche Datenmengen zu repräsentieren
- Erschwerte Ausnahmebehandlung

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Lazy Evaluation
Beispiel: ein JFC `ListModel`



- Ein `javax.swing.ListModel` bei der Anzeige umfangreicher Abfrage-Ergebnisse in einem JFC-Listenelement
- Das Model hat die Rolle eines virtuellen Stellvertreter-Objekts
  - `getElementAt(int anIndex)` - liefert die Werte erst, wenn sie benötigt werden

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Lazy Evaluation
Beispiel-Implementierung




```
public class QueryListModel extends AbstractListModel {
  private int size;
  private java.util.List cachedElements;
  private ResultSet rs;

  public int getSize() {
    // z.B. durch SELECT COUNT FROM ... ermittelt
    return size;
  }

  public Object getElementAt(int anIndex) {
    while(readElements.size() <= anIndex) {
      cachedElements.add(getNextObjectFrom(rs));
    }
    return cachedElements.get(anIndex);
  }
...
```


(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Ausblick



- Die Verwendung von Entwurfsmustern erlaubt die Modellierung von einer „höheren Warte“ aus
  - Neue Analyse- und Design-Werkzeuge unterstützen die Verwendung von Mustern direkt
- Anwendungs-Frameworks
  - bieten Prototypen und abstrakte Implementierungen von Entwurfsmustern
  - machen die Verwendung von Mustern zum Prinzip für den Anwendungsentwickler

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten

Quellen



- [GoF1995] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. *Entwurfsmuster*.
- [Grand1998] Mark Grand. *Patterns in Java*.
- [BMRSS1996] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal. A *System of Patterns*.
- [Larmann 1998] Craig Larman. *Applying UML and Patterns*.
- [ArcusProject] *http://www.sdm.de/g/arcus/index.htm*

(c)1999, GEBIT Gesellschaft für EDV-Beratung und Informatik-Technologien mbH, alle Rechte vorbehalten