Entwurfsmuster: Abstrakte Fabrik

Seminar Pattern & Antipattern

Robert Sauer
sauer@informatik.hu-berlin.de

Übersicht

- Abstrakte Fabrik (englisch: Abstract Factory)
  - Anderer Name: Bausatz (?) (Kit)
- Erzeugungsmuster
- Siehe auch:
  - Fabrikmethode (Factory Method)
  - Prototyp (Prototype)

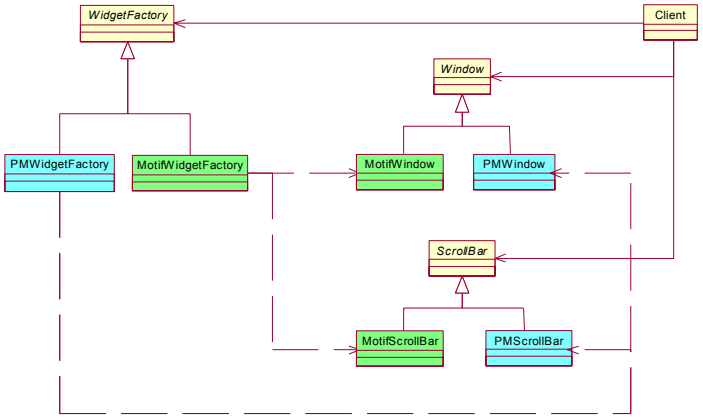
Einschub: Erzeugungsmuster

- Betreffen die Erzeugung von Objekten
  - schnittstellenbasierte Programmierung
- Zwei Grundprinzipien
  - Verstecken der konkreten Implementierung
  - Verstecken der Erzeugungsart

Zweck

- Erzeugung von gewünschte Schnittstellen implementierenden Objekten
- Speziell: Familien von konkreten Objekten
  - Arbeiten zusammen
  - Komponieren einander

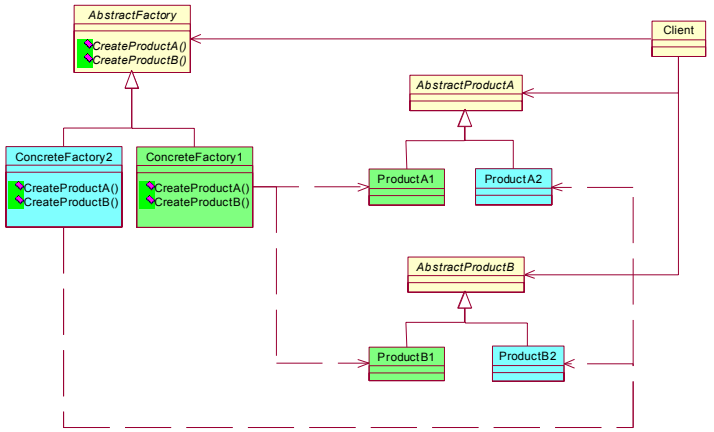
Motivation: GUI Komponenten



Anwendbarkeitskriterien

- Unabhängigkeit von Produkterzeugung
- Konfigurierbarkeit der verwendeten Produktfamilie
- Zwang zur Verwendung von Produkten genau einer Familie
- Bereitstellung von Komponenten ohne Einblick in Implementierung

Struktur



Teilnehmer

- AbstractFactory
  - Schnittstelle zur Produkterzeugung
- ConcreteFactory
  - Implementierung der Produkterzeugung
- AbstractProduct
  - Schnittstelle für Produkt
- ConcreteProduct
  - Implementierung der Produkt-Schnittstelle
- Client

Einsatzfolgen

- Positiv:
  - Isolierung der konkreten Klassen
  - Leichter Austausch ganzer Produktfamilien
  - Förderung einheitlicher Familienzugehörigkeit von Produkten
- Negativ:
  - Unterstützung neuer Produkte aufwendig

Implementierung Fabrikzugriff

- Fabrik als Singleton
- instance()-Methode liest Konfiguration

Implementierung: Fabrik-Methode

- Fabrik-Methode – eine Fabrik pro Produktfamilie
  - Meist streng typisiert: eine (parameterlose) Methode pro Produkt
  - Verwendet Konstruktor des konkreten Produktes

Beispiel: Fabrik-Methode

```
void FactoryMethod( IXMLDOMDocumentPtr & document )
{
  IXMLDOMElementPtr envelope;
  envelope = document->createElement( "envelope" );

  IXMLDOMAttributePtr e_id;
  e_id = document->createAttribute( "id" );
  IXMLDOMAttributePtr e_name;
  e_name = document->createAttribute( "name" );

  envelope->setAttributeNode( e_id );
  envelope->setAttributeNode( e_name );

  document->appendChild( envelope );
}
```

Implementierungsvariante: Prototyp-basiert

- Prototyp-basiert – eine konfigurierbare Fabrik für mehrere Produktfamilien
  - Clonen eines vorkonfigurierten Produkt-Prototyps
  - Für Benutzer an der Fabrik-Schnittstelle nicht ersichtlich

Implementierungsvariante: Erweiterbarkeit

- Lose typisiert: einer parametrisierte Methode für mehrere Produkte
  - Parameter bestimmt Produktart
  - Erforderlich
    - ein Vererbungsbaum für erzeugte Produkte
    - Gleiche Initialisierungsparameter
  - Vorteil: erleichtert Einführung neuer Produkte

Beispiel: parametrisierte Fabrik-Methode

```
void ParametricFactoryMethod( IXMLDOMDocumentPtr & document )
{
  IXMLDOMElementPtr envelope;
  envelope = document->createNode( NODE_ELEMENT, "envelope", "" );

  IXMLDOMAttributePtr e_id;
  e_id = document->createNode( NODE_ATTRIBUTE, "id", "" );
  IXMLDOMAttributePtr e_name;
  e_name = document->createNode( NODE_ATTRIBUTE, "name", "" );

  envelope->setAttributeNode( e_id );
  envelope->setAttributeNode( e_name );

  document->appendChild( envelope );
}
```


Wo verwendet?

- GoF Beispiele:
  - InterViews für GUI Komponenten
  - ET++ für GUI Komponenten
- Andere Beispiele
  - W3C DOM: Document (createXXX Methoden)
  - SourcePro DB: RWDBDatabase

Verwandte Muster

- Fabrik-Methode, Prototype
  - Gängigste Verfahren zur Implementierung
- Singleton
  - Zugriff auf die Fabrik selbst