# Beans

## Definition

1. Öffentlicher parameterloser Konstruktor
2. Serialisierbar
3. Öffentliche Zugriffsmethoden
   1. getFoo(), setFoo(xxx)
   2. Booleans: isFoo()
   3. Indizierte Eigenschaften: je einen Getter/Setter für Gesamtheit und für best. Index

## Verwendung

* einfache Instanziierung von GUI-Klassen (AWT, Swing) zur Übertragung mittel RMI
* Container zur Datenübertragung
* Datenobjekte für Persistenzframeworks

# Design Patterns (GoF)

<http://en.wikipedia.org/wiki/Design_Patterns>

<http://en.wikipedia.org/wiki/Software_design_pattern>

|  |  |  |
| --- | --- | --- |
| **V**orlesung | **G**oF (Design Patterns:  Elements of Reusable Object-Oriented Software) | **C**ode Complete (Steve McConnell) |

## Creational

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Pattern** | **Description** | **V** | **G** | **C** |
| Abstract factory | Provide an interface for creating families of related or dependent objects without specifying their concrete classes. | **X** | **X** | **X** |
| Builder | Separate the construction of a complex object from its representation, allowing the same construction process to create various representations. | **X** | **X** |  |
| Factory method | Define an interface for creating a single object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses. (One common way to implement Dependency Injection.) |  | **X** | **X** |
| Prototype | Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype. |  | **X** |  |
| Singleton | Ensure a class has only one instance, and provide a global point of access to it. |  | **X** | **X** |

## Structural

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Pattern** | **Description** | **V** | **G** | **C** |
| Adapter (Wrapper, Translator) | Allows classes with incompatible interfaces to work together by wrapping its own interface around that of an already existing class. The enterprise integration pattern equivalent is the translator. | **X** | **X** | **X** |
| Bridge | Decouples an abstraction from its implementation so that the two can vary independently. |  | **X** | **X** |
| Composite | Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly. | **X** | **X** | **X** |
| Decorator | Dynamically adds/overrides behaviour in an existing method of an object. Attach additional responsibilities to an object dynamically keeping the same interface. Decorators provide a flexible alternative to subclassing for extending functionality. |  | **X** | **X** |
| Facade | Provide a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use. |  | **X** | **X** |
| Flyweight | Reduces the cost of creating and manipulating a large number of similar objects. |  | **X** |  |
| Proxy | Provides a placeholder for another object to control access, reduce cost, and reduce complexity. |  | **X** |  |

## Behavioral

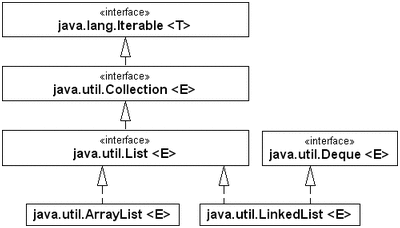
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Pattern** | **Description** | **V** | **G** | **C** |
| Chain of responsibility | Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it. |  | **X** |  |
| Command | Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations. |  | **X** |  |
| Interpreter | Given a language, define a representation for its grammar along with an interpreter that uses the representation to interpret sentences in the language. |  | **X** |  |
| Iterator | Accesses the elements of an object sequentially without exposing its underlying representation. |  | **X** | **X** |
| Mediator | Allows loose coupling between classes by being the only class that has detailed knowledge of their methods. | **X** | **X** |  |
| Memento | Provides the ability to restore an object to its previous state (undo). |  | **X** |  |
| Observer (Publish/  subscribe) | Allows a number of observer objects to see an event. | **X** | **X** | **X** |
| State | Allow an object to alter its behavior when its internal state changes. The object will appear to change its class. |  | **X** |  |
| Strategy | Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it. |  | **X** | **X** |
| Template | Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Template method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure. |  | **X** | **X** |
| Visitor | Separates an algorithm from an object structure by moving the hierarchy of methods into one object. Lets you define a new operation without changing the classes of the elements on which it operates |  | **X** |  |

## Concurrency

15 Patterns

# Datenstrukturen

## Listen



## HashMap

Kollisionen werden über Listen organisiert. Wenn zwei verschiedene Keys den selben Hashcode ergeben (und damit in denselben Bucket fallen), geht HashMap bei einem get(key) die ganze Liste des zugehörigen Buckets durch und vergleicht den Key jeden Elements mit dem gegebenen key per equals.

D.h. jedes Element in einem Bucket ist ein Key/Value-Paar.

## Generics (Java 1.5)

<https://docs.oracle.com/javase/tutorial/java/generics/index.html>

<https://github.com/psicho2000/wiki/blob/master/External/Java-Generics.pdf>

*Note: Generic’s type erasure is necessary because of compatibility with legacy code (not using generics, i.e. raw types).*

## Wildcards (Java 1.5)

|  |  |  |
| --- | --- | --- |
| Unbounded Wildcard | List<?> | Für in Variablen, auf denen Methoden der Object-Klasse angewendet werden sollen. |
| Upper Bounded Wildcard | List<? extends E> | Für in Variablen |
| Lower Bounded Wildcard | List<? super E> | Für out Variablen |

Wenn Wildcards in einer Klasse genutzt werden, darf sich die aufrufende Klasse darum keine Gedanken machen!

## Kovarianz/Kontravarianz/Invarianz

[https://en.wikipedia.org/wiki/Covariance\_and\_contravariance\_(computer\_science)](https://en.wikipedia.org/wiki/Covariance_and_contravariance_%28computer_science%29)

Also see Chapter 5 “Generics” in <https://github.com/psicho2000/wiki/blob/master/External/Java-Generics.pdf>

**Arrays** (d.h. Object[]) sind **kovariant**:

If Sub is a subtype of Super, then the array type Sub[] is a subtype of Super[].

**Generische Typen** sind **invariant** (d.h. nicht kovariant):

For any two distinct types Type1 and Type2, List<Type1> is neither a subtype nor a supertype of List<Type2>.

**Wildcards** bieten dagegen **typensichere Kovarianz**:

For any concrete type A, List<A> is a subtype of List<?>.

### Overview

Let B be a subtype of A (B **≤** A). „Each B is an A“

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Covariance | I<B> **≤** I<A> | read-only (immutable) | source | C#: out | Java: arrays (runtime type safe, not compile time type safe) |
| Contravariance | I<A> **≤** I<B> | write-only | sink | C#: in |  |
| Invariance | Neither I<B> **≤** I<A> nor I<A> **≤** I<B> | read/write (mutable) | source & sink |  | Java: generics (not runtime type safe (type erasure), compile time type safe) |

## Varargs (Java 1.5)

<http://www.straub.as/java/basic/varargs.html>

### Deklaration

Typ... varname

### Beispiel

public static void varargs(Object... x) {

for (Object y : x) {/\* do something with y \*/}

}

### public static void main(String[] args) {

### varargs(a, b, c, d);

### }

### Behandlung durch den Compiler

Im Hintergrund der Funktion wird ein Array verwendet: Object[] x

Den Aufruf übersetzt der Compiler zum anonymen Array: new Object[] {a, b, c, d}

### Die folgenden beiden Funktionen können nicht in einem Scope existieren:

### public static void varargs(int... x)

### public static void varargs(int[] x)

### Regeln

* Der Typ einer Vararg-Deklaration ist beliebig
* Die Parameterliste einer Methode kann weitere Parameter beliebigen Typs enthalten,  
  jedoch muß die Vararg-Deklaration immer am Ende der Parameterliste stehen.
* Es kann nur eine Varargdeklaration geben

# JavaFX

Seit Java 8 ist JavaFX 8 direkt eingebettet.

## Einrichtung

* Scene Builder von Gluon installieren: <http://gluonhq.com/open-source/scene-builder/#download> (Oracle entwickelt nicht mehr weiter)
* Eclipse Plugin e(fx)clipse installieren
* Preferences > JavaFX > Scenebuilder executable setzen

## First Steps

* <https://docs.oracle.com/javafx/scenebuilder/1/use_java_ides/sb-with-eclipse.htm>
* <http://code.makery.ch/blog/javafx-8-event-handling-examples/>