



Delegation

Von Bernd Stärz und
Patrick Röder (Mai 2002)



Überblick

- Einführung
 - Idee und Modell
- Fähigkeiten
- Umsetzungen
 - Darwin Modell > LAVA
 - weitere
- Diskussion



Überblick

- Einführung
 - **Idee** und Modell
- Fähigkeiten
- Umsetzungen
 - Darwin Modell > LAVA
 - weitere
- Diskussion



Wie speichern Menschen Information über Ihre Umwelt?

- Prototypen im Kopf
- Beispiel: Hund
 - Von einem schließt man zunächst auf alle
 - Sieht man weitere, lernt man weitere Eigenschaften kennen
- So gewinnt man langsam eine Vorstellung über gewisse Objekte der Welt.



Wie werden Mengen von Objekten in der Mathe dargestellt?

- Abstrakte Definition der Eigenschaften oder explizite Aufzählung aller Objekte.
- Letzteres nur bei endlichen Mengen möglich.
- Beispiel Merkmalsliste: Ist Tier, hat 4 Beine, Schnauze, Größe, hat Fell, kann Bellen usw.



Welches der beiden Konzepte ist intuitiver?

- Prototypen sind eher typisch für menschliches Denken
- Prototypen sind flexibler, da noch nicht alle Eigenschaften a priori bekannt sein müssen
- Daher wird ein inkrementelles Vorgehen besser unterstützt



Math. Mengen entsprechen Vererbung

- Definieren von abstrakten Klassen
 - Attribute
 - Methoden
- Zur Verfeinerung Bildung von Subklassen
- Dadurch noch keine Objekte
 - Deswegen: Instanzieren dieser Klassen



Prototypen entsprechen Delegation

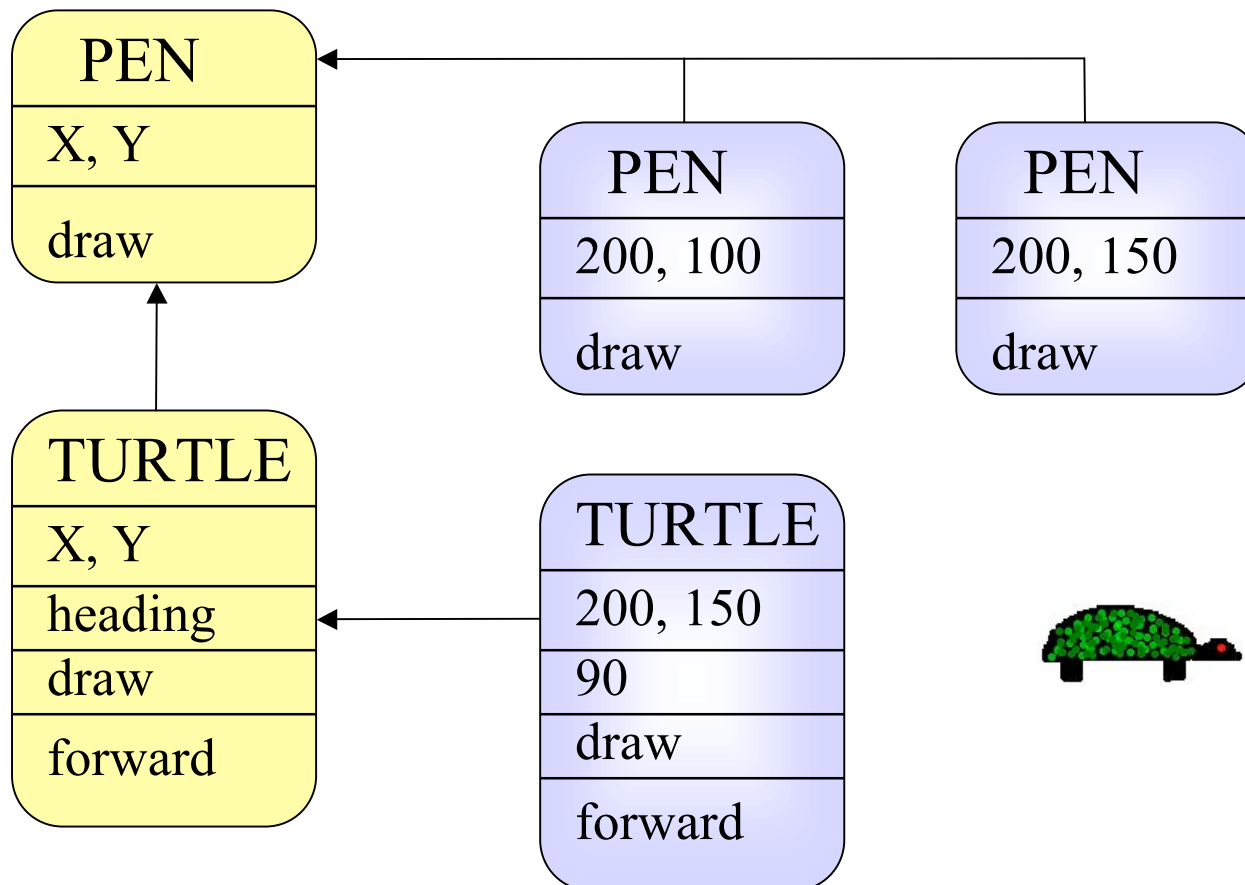
- Anlegen eines Objekts mit den gewünschten Eigenschaften
- Weiteres Objekt mit leicht abweichenden Eigenschaften
 - Neues Objekt anlegen
 - Unterschiede implementieren
 - Alles weitere: An erstes Objekt (Parent) delegieren
- Erstes Objekt fungiert als Prototyp
- Unterscheidung Klasse / Instanz entfällt
- Jedes Objekt kann als Prototyp fungieren
- „Wenn ich eine Anfrage nicht beantworten kann delegiere ich diese an meinen Prototypen



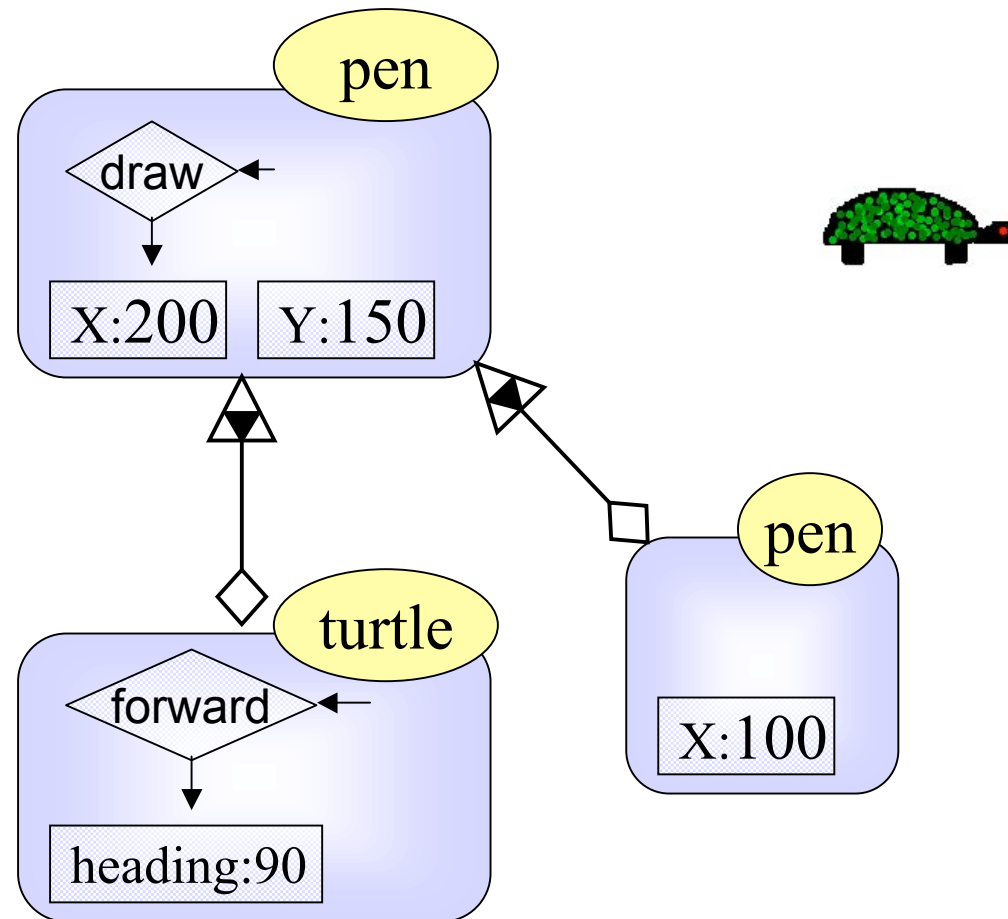
Überblick

- Einführung
 - Idee und **Modell**
- Fähigkeiten
- Umsetzungen
 - Darwin Modell > LAVA
 - weitere
- Diskussion

Turtle Grafik mit Vererbung

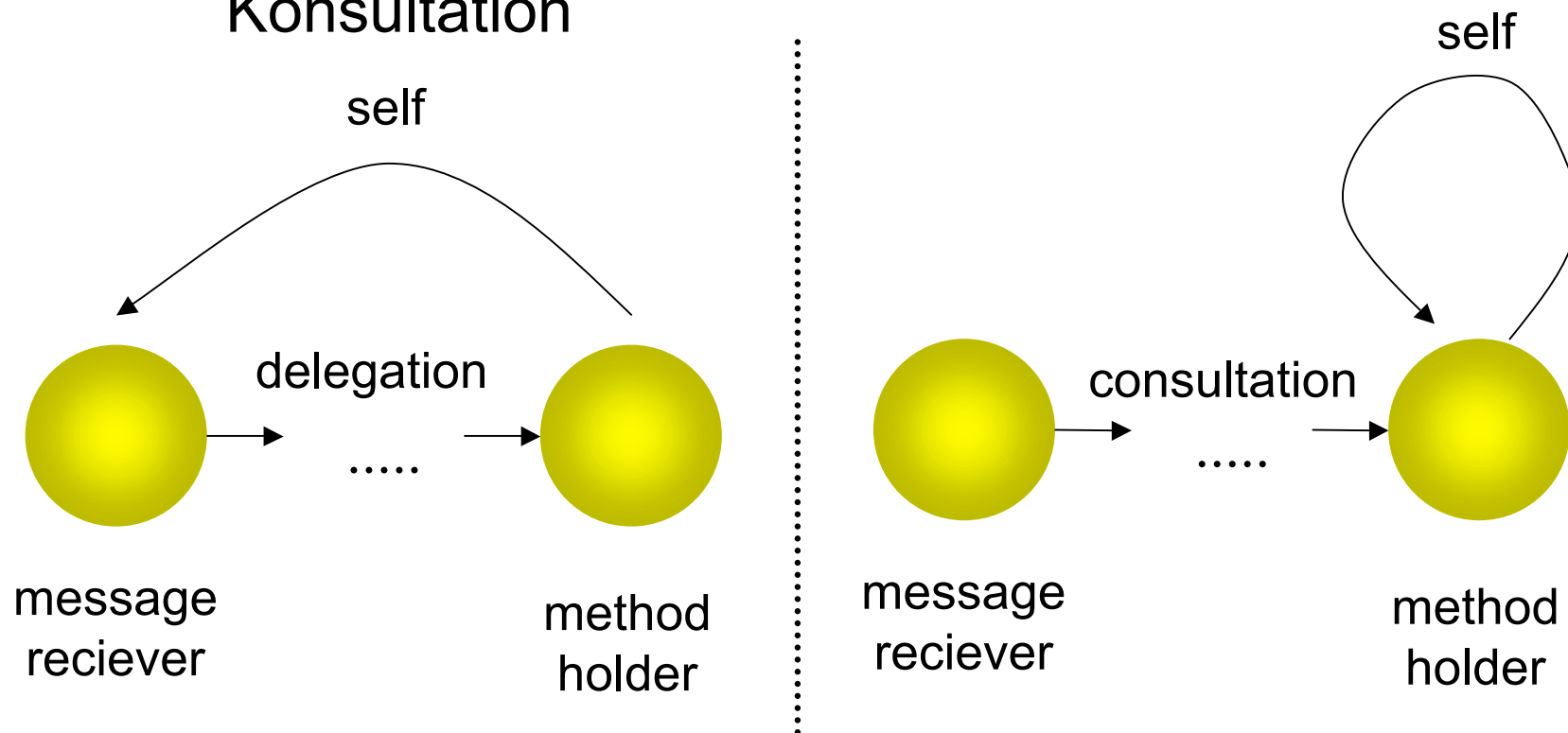


Turtle Grafik mit Delegation



Konzepte: Delegation und Konsultation

- Veranschaulichung von Delegation und Konsultation





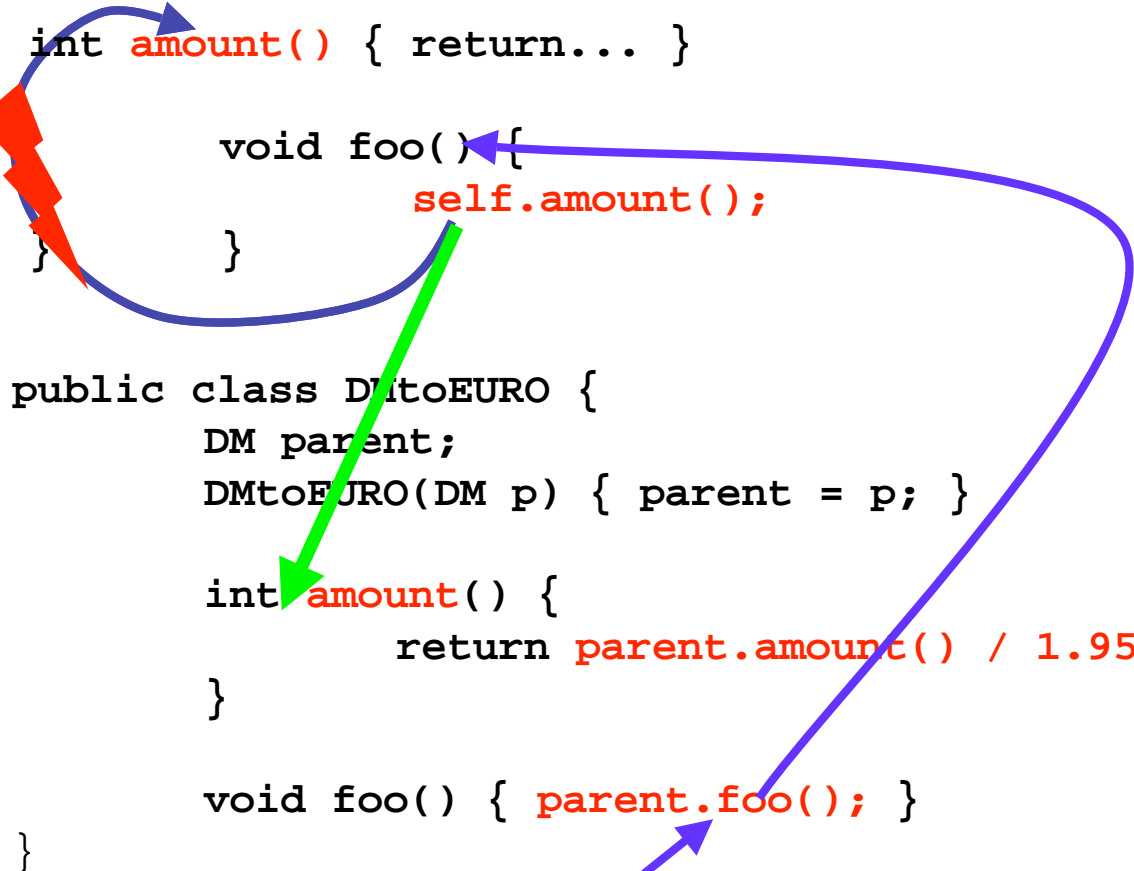
Überblick

- Einführung
 - Idee und Modell
- **Fähigkeiten**
- Umsetzungen
 - Darwin Modell > LAVA
- Diskussion

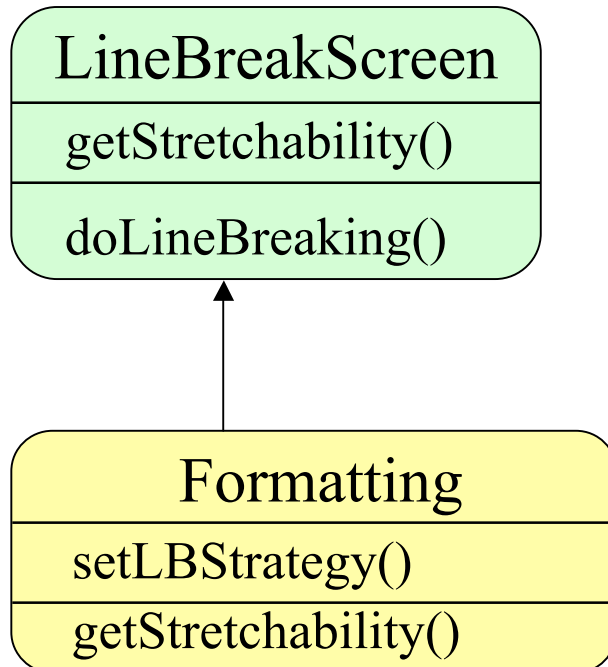


Self-Problem

```
public class DM {  
    int amount() { return... }  
    void foo() {  
        self.amount();  
    }  
}  
  
public class DMtoEURO {  
    DM parent;  
    DMtoEURO(DM p) { parent = p; }  
    int amount() {  
        return parent.amount() / 1.95;  
    }  
    void foo() { parent.foo(); }  
}
```



Dynamische Vererbung



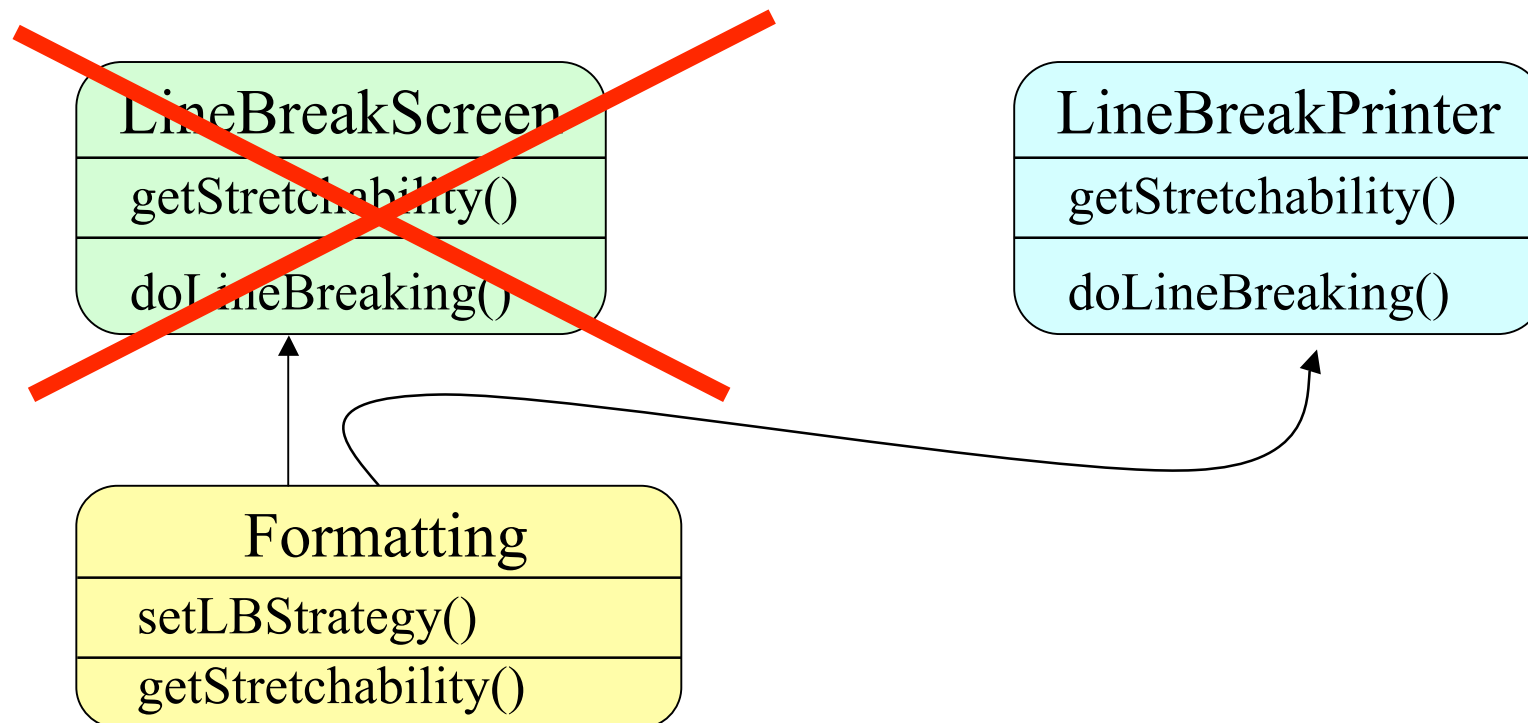
```
public class Formatting {
    mandatory delegatee LineBreaking lb;

    // Konstruktor für Sandarmethode
    Public Formatting () {
        lb = new SimpleLineBreaking();
    }

    // Strategie austauschen
    Public setLBStrategy (LineBreaking _lb) {
        lb = _lb
    }

    // Standardverhalten, wird überschrieben
    Public int[] getStretchability() { ... }
}
```

Dynamische Vererbung





Wann nutze ich was (Vererbung / Delegation)

- Vererbung bei ist-ein Beziehung
- Vererbung bei großer Anzahl gleicher Objekte
- Delegation bei agiert/fungiert als Beziehung
- Delegation bei einmaligen individuellen Objekten
- Das „ein-Objekt-Klassen Problem“



Vergleich in Sachen Effizienz (Speicher/Speed)?

- Vererbung mehr Speicher
- Prototypen mehr Nachrichten
- Nachteil von Prototypen (Delegation) durch Caching ausgleichbar.
 - Speicher wird effizienter genutzt.
 - Besseres Lokalitätsverhalten: Caches greifen besser



Überblick

- Einführung
 - Idee und Modell
- Fähigkeiten
- **Umsetzungen**
 - **Darwin Modell** > LAVA
 - weitere
- Diskussion



Darwin: Einführung

- Projekt der Uni Bonn
- Name vom Charles Darwin, dem Begründer der Evolutionstheorie abgeleitet („survival of the fittest“)
- Verbindet Vererbung auf Klassenebene mit der auf Objektebene
- Umsetzung führte zur einer Erweiterung der Sprache Java



Darwin

- Beschreibung einer typisierten, klassenbasierten OO-Sprache und deren Erweiterung um dynamische Vererbung
- Sichere Integration der statischen sowie dynamischen Delegation zusammen mit einem statischen Typkonzept
- Das Model ist implementationsunabhängig



Überblick

- Einführung
 - Idee und Modell
- Fähigkeiten
- **Umsetzungen**
 - Darwin Modell > **LAVA**
 - weitere
- Diskussion

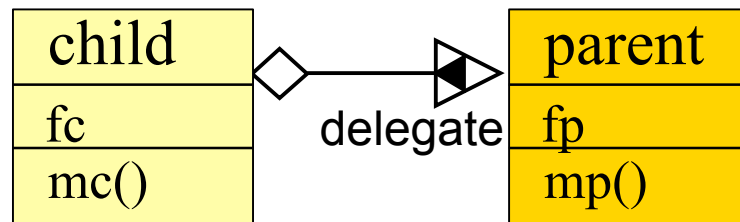


Darwin Projekt: Lava

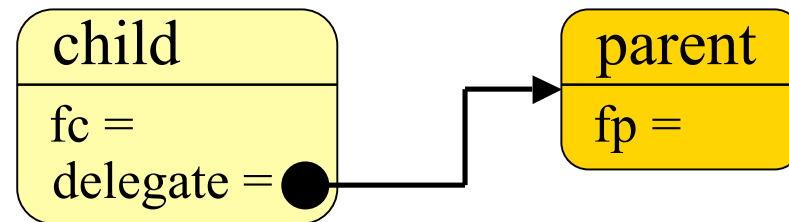
- Lava ist eine Erweiterung der Sprache Java um die Eigenschaft der dynamischen Vererbung
- Konzept der Delegation
 - Automatische Übergabe mit der Bindung von *this* an den Nachrichtenempfänger
 - wird mit dem Schlüsselwort *delegatee* eingeleitet
- Konzept der Konsultation
 - Automatische Übergabe mit der Bindung von *this* an den Methodenbesitzer
 - das Schlüsselwort *consultee* wird benutzt
- Statisches Typkonzept
 - Zur Übersetzungszeit wird festgestellt, welche Objekte welche Nachrichten verstehen

Beispiel: Erweiterung der Schnittstelle der Klasse *ChildClass* um die sichtbaren Elemente des Klassentyps *ParentClass*.

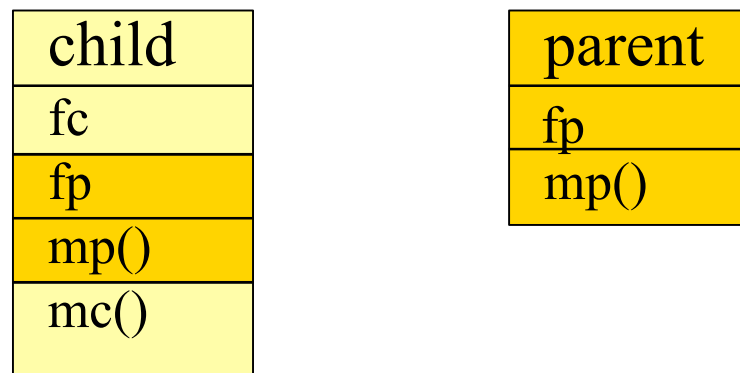
Erweiterte UML-Notation



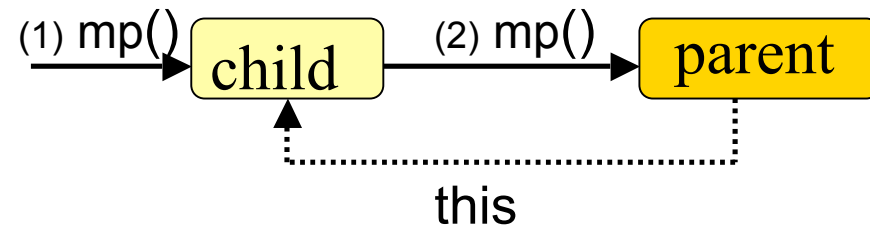
Objekt-Struktur



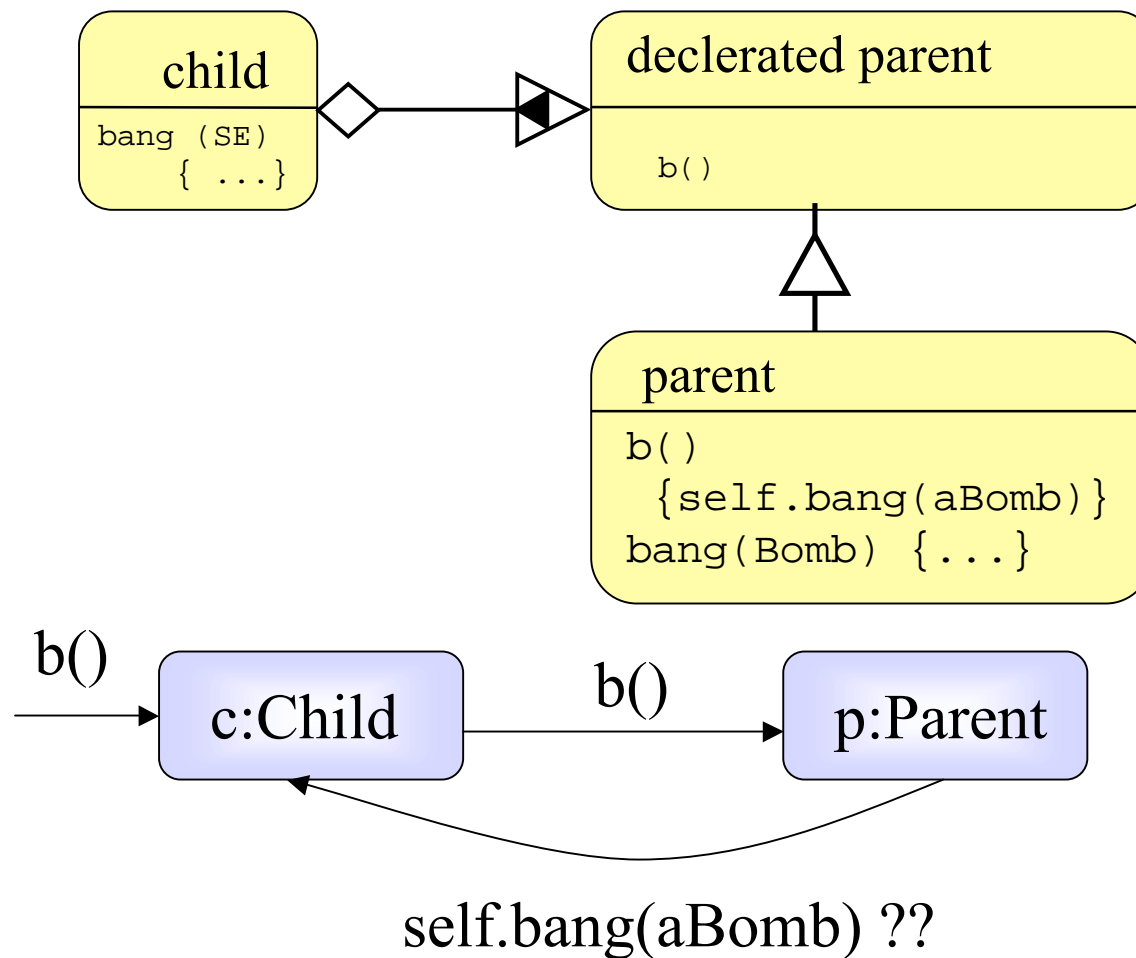
Schnittstellen Struktur



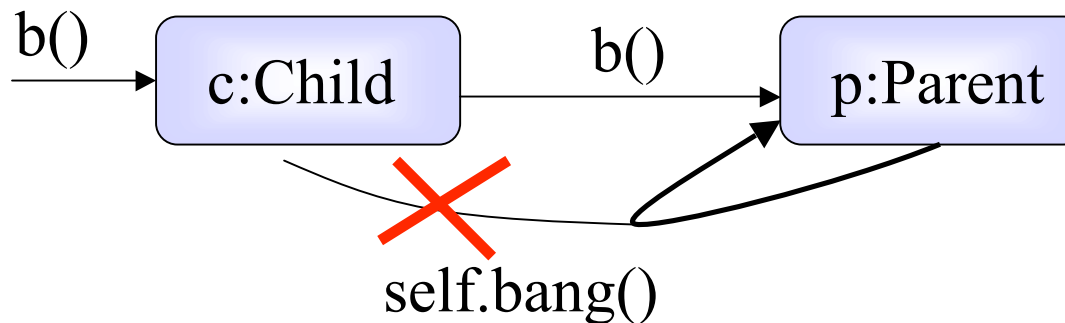
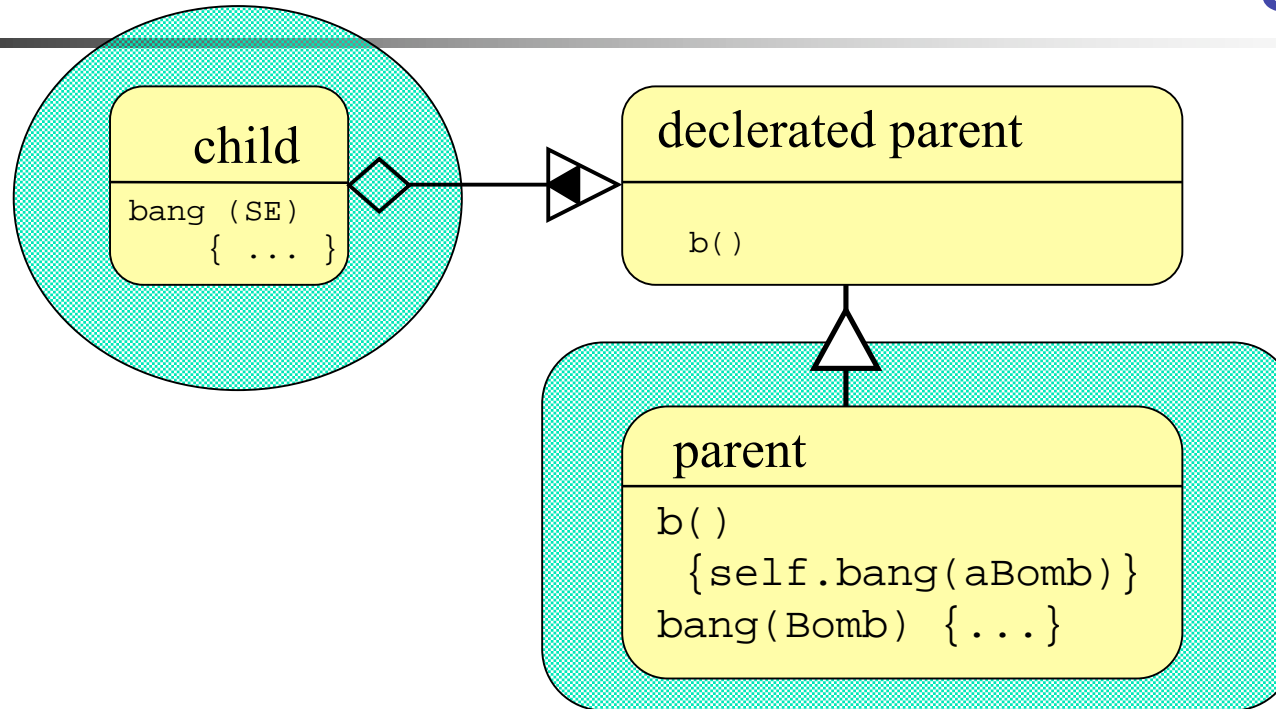
Bindung von this



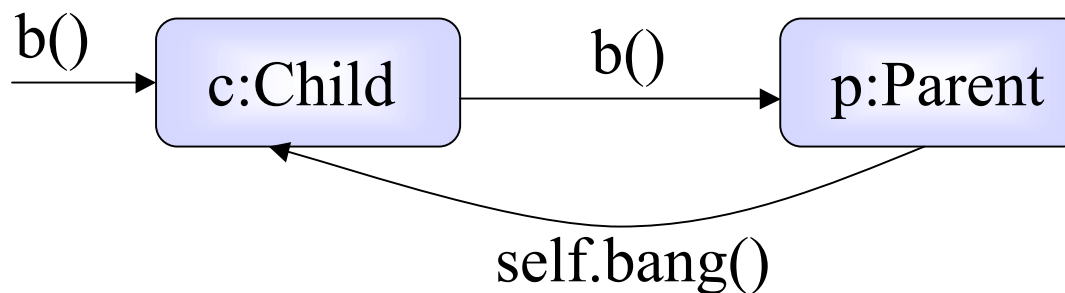
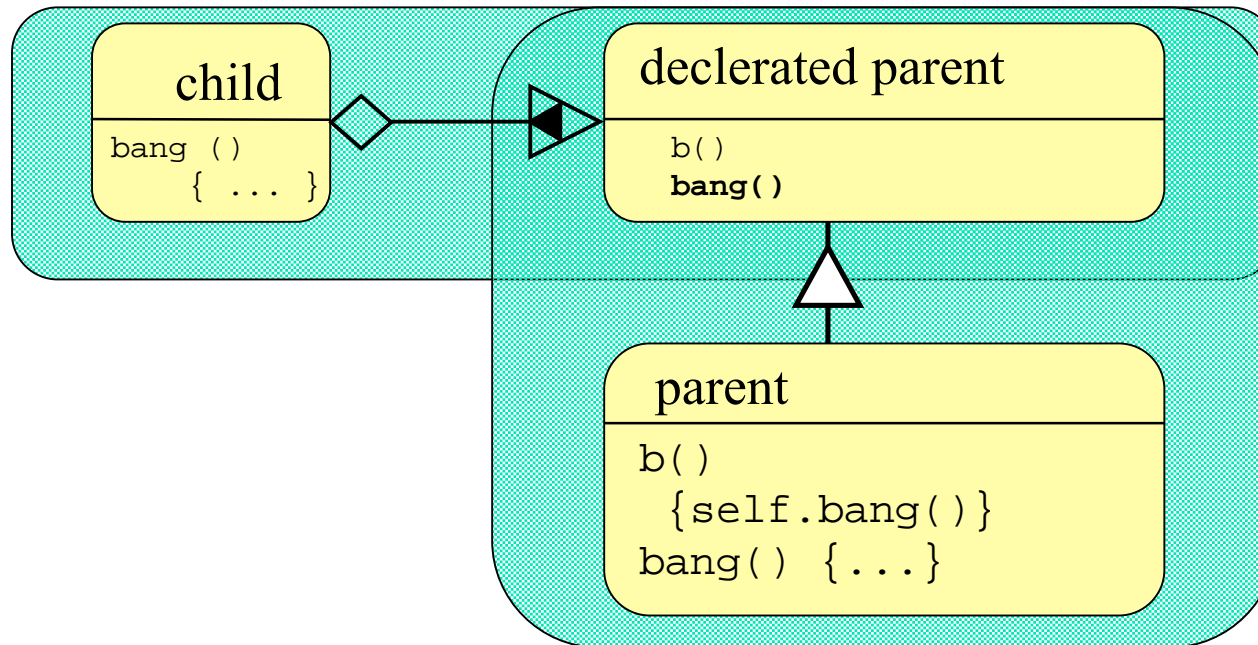
Unabhängige Erweiterbarkeit



Unabhängig eingeführte Methoden: kein overriding



Methode gemeinsamen Ursprungs: overriding aktiv





Andere Implementierungen

- Self
 - Basiert auf Smalltalk Syntax
 - Unterstützt Multiple Objektbasierte Vererbung
 - Mechanismen um Namenskonflikte zu lösen
 - Hat wesentlich zur Optimierung von Java beigetragen
- Smalltalk
 - Delegation lässt sich Nachrüsten
 - Es gibt einen Handler für unbekannte Nachrichten
 - Diesen modifizieren, so dass er Nachrichten weiterleitet



Überblick

- Einführung
 - Idee und Modell
- Eigenschaften / Probleme
- Umsetzungen
 - Darwin Modell > LAVA
 - weitere
- **Diskussion**



Schlussfragen / Diskussion

- Wie gut ist delegation?
- Würdet ihr es verwenden?

Vielen Dank für die
Aufmerksamkeit

