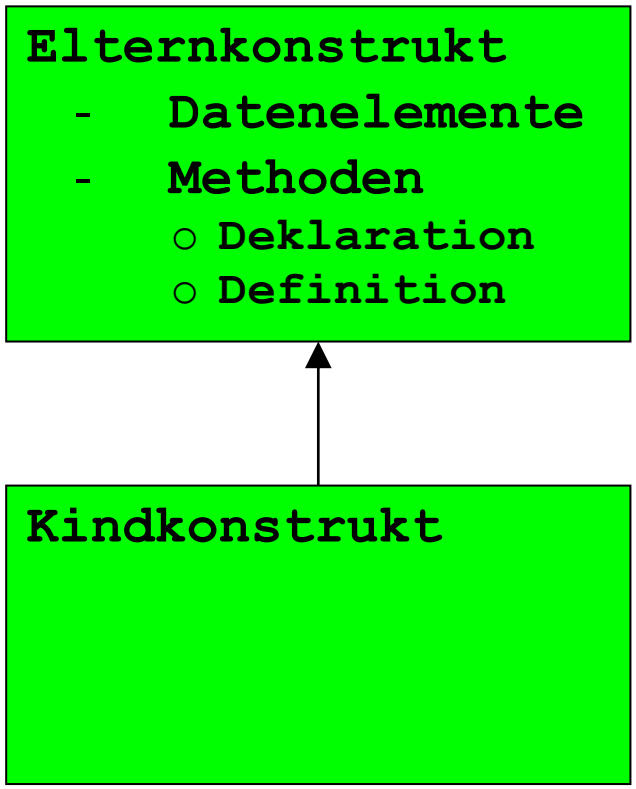
Vererbung

Was versteht man unter dem Begriff „Vererbung“ ?




```
... Main ...

Elternkonstrukt b =
    new Kindkonstrukt()

b.fn()
```

Vererbung

Vererbungsvarianten

- Schnittstellenvererbung
  - Deklaration wird vererbt, so dass Kindkonstrukt Definition vornehmen muss
- Implementierungsvererbung
  - Deklaration und Definiton wird vererbt, so dass Kindkonstrukt die Methode (Implementierung) direkt verwenden kann oder Implementierung überschreiben kann
- Substituierbarkeit
  - Verwendung eines Kindobjektes im Sinne des Basiskonstruktes (Kovarianz)
  - Bsp.: Basiskonstrukt b = new Kindkonstrukt()
  - Polymorphismus

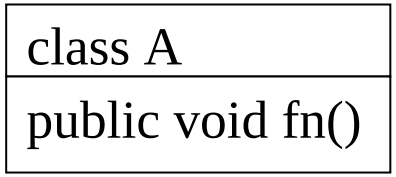
Vererbung

Vererbung und Wiederverwendung

- **Implementierungsvererbung**
  Durch Bereitstellung einer Implementierung im Basiskonstrukt kann diese im abgeleiteten Konstrukt wieder verwendet und bei Bedarf sogar angepasst werden. Der Nachteil ist hierbei die enge Kopplung der beiden Konstrukte. Insbesondere führen nachträgliche Änderungen an der Implementierung des Basiskonstruktes zu Problemen.
- **Schnittstellenvererbung und Substituierbarkeit**
  - **Wiederverwendbarkeit von Abläufen und Architekturen**
  - **Programmieren auf Schnittstellen hin**
  - **Plug-in-Architektur (Bsp. Eclipse)**

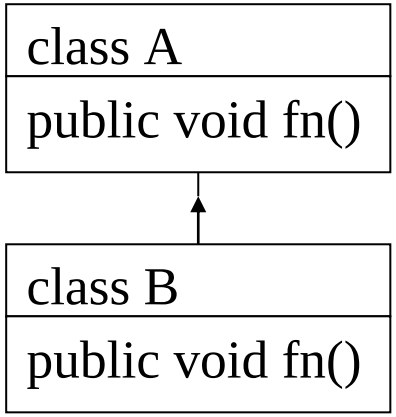
Vererbung

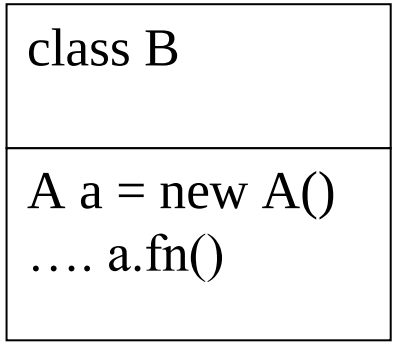
Alternativen zur Implementierungsvererbung



Wie kann die Methode fn() (wieder-) verwendet werden?

- **Aufruf nur über Instanz möglich (Verwendung)**
- **Vererbung oder Objektkomposition**





Vererbung

Problem der zerbrechlichen Basisklasse in Java

```
class B_alt{public void fn(){System.out.println("B");}}
class B_neu{public void fn(){this.delete(); }
    public void delete(){System.out.println("Delete");}
}
//class C extends B_alt{
class C extends B_neu{
  public void delete(){System.out.println("Loesche
Platte");}
}
public class A {
  public static void main(String[] args) {
    C c = new C();
    c.fn();}
}
```

Vererbung

Zerbrechliche Basisklasse @Override (Java!)

- Annotation @Override
  - kann bei Methoden in abgeleiteter Klasse verwendet werden
  - bedeutet, dass die annotierte Methode eine geerbte Methode überschreibt
  - Fehlermeldung falls Methode mit gleicher Signatur nicht in Basisklasse existiert
  - Garantiert, dass Entwickler wirklich Methoden überschreiben und nicht aus Versehen Methoden mit falschen Parametern überladen
- Workaround für Problem der zerbrechlichen Basisklasse
  - falls Basiklasse nachträglich modifiziert und @Override in abgeleiteter Klasse nicht gesetzt
    => Hinweis auf mögliches Problem per Compilerwarnung

Vererbung

Vererbungsvarianten in C#

- Methoden sind
  - virtuell
  - abstrakt
  - nicht virtuell
- Konstrukte
  - Klasse (keine Mehrfachvererbung)
  - Struktur (nur Vererbung von Interfaces)
  - Abstrakte Basisklasse (keine Mehrfachvererbung)
  - Interface (kann von Interface erben)
- Ableitung durch Benennung der Basiskonstrukte
- Lösung des Problems der zerbrechlichen Basisklasse
  - neue Schlüsselworte:
    new (verdecken) und
    override (überschreiben)

Vererbung

Vererbung

- Objekte der abgeleiteten Klasse haben (erben) alle Elemente der Basisklasse
- Abgeleitete Klasse hat keinen Zugriff auf private Elemente der Basisklasse (siehe protected)
- keine Mehrfachvererbung (keine reine Implementierungsvererbung d. h. private-, protected-Ableitung wie in C++)
- nicht vererbt werden: Konstruktoren
- Syntax: class Abgeleitet : Basis {Klassenkoerper}
- Substituierbarkeit
  Instanzen der abgeleiteten Klasse können an allen Stellen verwendet werden, an denen eine Instanz der Basisklasse erwartet wird jedoch nicht umgekehrt

Vererbung

Polymorphismus

```
class GeometrischesObjekt
    {
        public virtual void zeichnen()
        {Console.WriteLine("G");}
    }
class Dreieck : GeometrischesObjekt
    {
        public override void zeichnen()      // new ?
        {Console.WriteLine("D");}
    }

...
GeometrischesObjekt g1 = new Dreieck();
g1.zeichnen();
```

Vererbung

Virtuelle Funktionen

- Kennzeichnung einer Methode mit dem Schlüsselwort virtual macht diese zu einer polymorphen Methode (späte Bindung)
- override (= aufheben, übersteuern) kennzeichnet bewusstes Überschreiben einer virtuellen Methode
- wird eine virtuelle Methode in der abgeleiteten Klasse nicht mit override gekennzeichnet, so wird beim Zugriff über Basisklassenreferenzen die Basisklassenmethode verwendet(entspricht new)
- wird in abgeleiteter Klasse eine Methode mit gleicher Signatur definiert erzeugt der Compiler eine Warnung falls weder new noch override verwendet wurde
- virtuelle Funktion ist immer die Wurzel der virtuellen Weitergabe

Vererbung

Verdecken von Methoden

- werden in abgeleiteter Klasse Methoden mit gleicher Schnittstelle wie in Basisklasse mit new definiert, so verdecken diese die Basisklassenmethode
- Die Markierung mit new verhindert eine versehentliche Neudefinition (kein polymorpher Aufruf)

Überschreiben (übersteuern) von Methoden

- werden in abgeleiteter Klasse Methoden mit gleicher Schnittstelle wie in Basisklasse mit override definiert, so überschreiben diese die Basisklassen-methode (muss virtuell in Basisklasse sein)
  => Polymorphismus wird wirksam

Vererbung

Abstrakte Klassen

- Abstrakte Klassen ermöglichen die Deklaration abstrakter Methoden, können auch nicht abstrakte Methoden haben
- abstrakte Methoden dürfen keinen Funktionskörper haben
- Syntax-Beispiel: `abstract public void fn();`
- abstrakte Methoden müssen in allen abgeleiteten Klassen mit Schlüsselwort override (da sonst verdeckt) definiert werden
- abstrakte Methoden sind automatisch virtuell
- von abstrakter Klasse kann nicht direkt eine Instanz erstellt werden
- falls eine Methode abstrakt ist, muss auch die Klasse als abstrakt deklariert sein

Vererbung

Zugriffsmodifikatoren

- Regeln die Zugriffsrechte auf Klassen und Strukturen sowie auf deren Elemente
- Zugriffsmodifikatoren für Elemente
  - public = keine Beschränkung
  - private = Zugriff nur über Methoden der Klasse
  - protected = wie private jedoch zusätzlich Methoden der abgeleiteten Klasse
  - internal = Zugriff über alle Methoden der Assembly
  - protected internal = protected ODER internal
  - private protected = private oder protected in aktueller Assembly
- Defaultwert: private
- Das Zugriffsrecht ist Klassen- und nicht Instanzbezogen
- werden für jedes Element angegeben

Vererbung

Zugriffsmodifikatoren auf Assemblyebene

- regeln den Zugriff auf Klassen und Strukturen aus Sicht einer Assembly

- Zugriffsmodifikatoren auf Assemblyebene
  - public = keine Beschränkung für andere Assemblies
  - internal = Zugriff ist nur innerhalb derselben Assembly erlaubt
    (Anm.: Interface kann Zugriffsschutz internal haben)

  - Defaultwert: internal