www.IT-Visions.de

Dr. Holger Schwichtenberg

Objektorientierte Programmierung
OOP

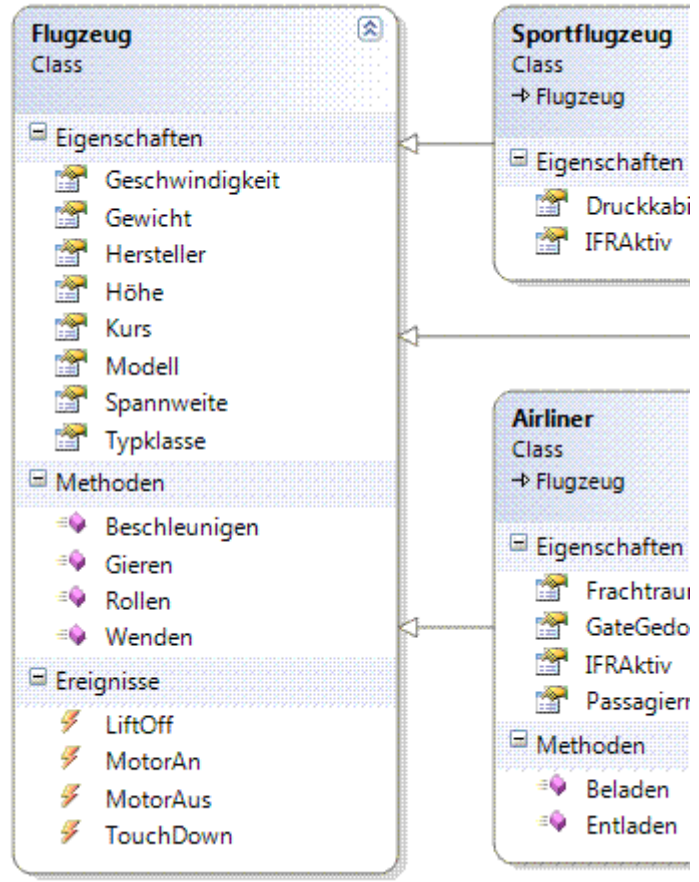
Olaf Lischke – Berater, Trainer, Entwickler

Objektorientierte Paradigmen

- Abstraktion
  - „Die Welt mit konkreten Begriffen greifbar und darstellbar machen"
- Kapselung
  - „Single Point of Responsibility"
  - interne Vorgänge bleiben intern
- Persistenz
  - Lebensdauer und Erreichbarkeit
- Vererbung
  - Möglichst wenig Code, Redundanzen vermeiden
- Polymorphie
  - Vereinheitliche, nachvollziehbare Vorgehensweisen

Ziele des Objektorientierten Programmierens

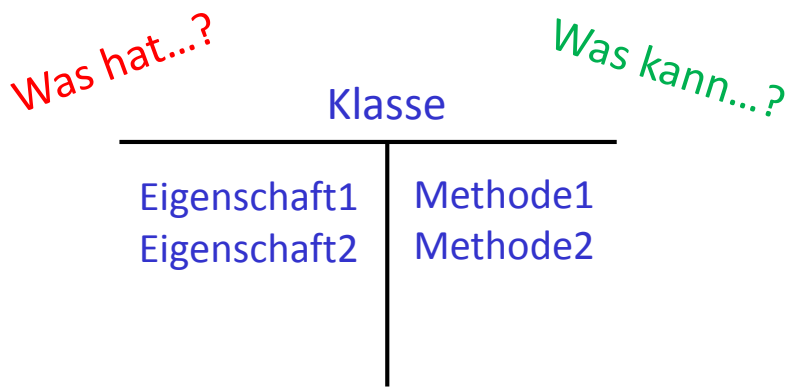
- Abstraktion
  - bildet „die Welt" so ab, wie sie ist (oder wie der Entwickler sie sich vorstellt…)
- Vereinfachung
  - Verständlichkeit und Wartbarkeit des Codes
- Wiederverwendbarkeit
  - spart Code und Entwurfszeit
- Weiterverwendbarkeit
  - durch Vererbung und Kapselung



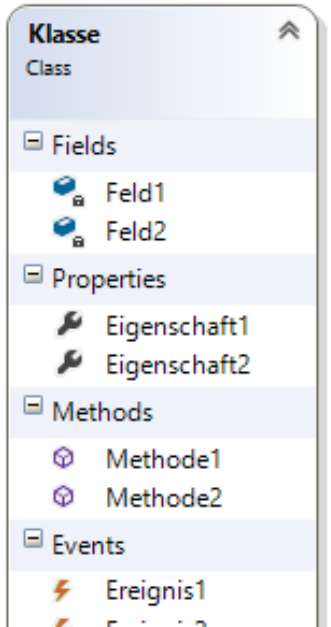
Objektorientierter Entwurf

⇨ **Vorher planen!**

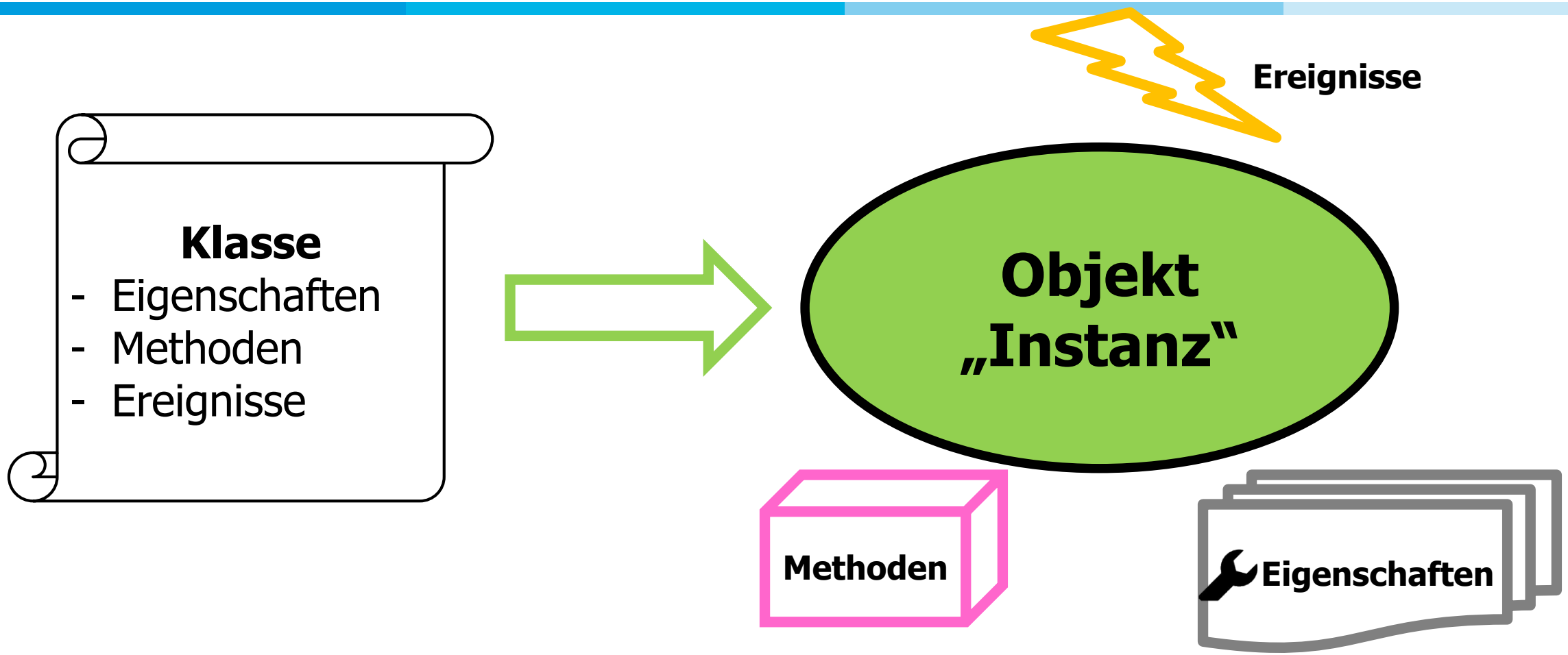
- UML v1:
  - gut mit Zettel und Stift



- UML v2:
  - gut mit Softwareunterstützung
  - eventuell mit Code-Erzeugung (Studio)



Grundbegriffe



Syntax – C#

- Klasse

```
private int _intFeld1;
public int Eigenschaft1
{
     get { return _intFeld1; }
     set { _intFeld1 = value; }
}

public string Eigenschaft2 { get; set; }

public void Methode1()
{ ... }

public bool Methode2(int Parameter)
{
        ...
        return bolErgebnis;
}
```

- Anwendung

```
Klasse meinObjekt = new Klasse();


int Wert = meinObjekt.Eigenschaft1;

meinObjekt.Eigenschaft2 = „Hallo“;


meinObjekt.Methode1();

bool bolWert = meinObjekt.Methode2(45);
```

Syntax – VB.NET

- Klasse

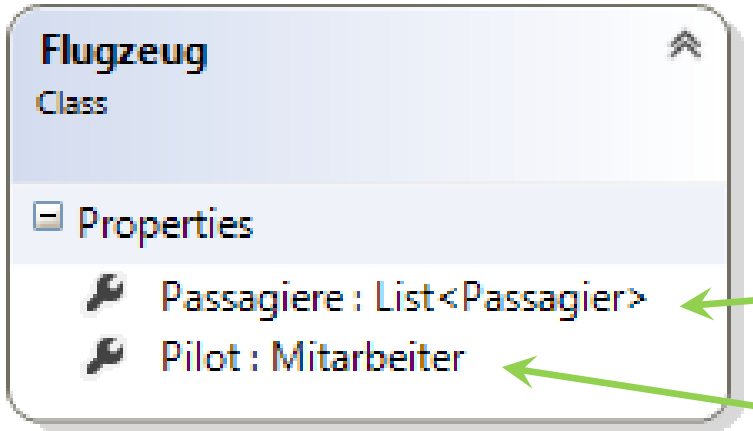
```
Private _intFeld1 As Integer
Public Property Eigenschaft1() As Integer
   Get
      Return _intFeld1
   End Get
   Set(ByVal value As Integer)
       _intFeld1 = value
   End Set
End Property

Public Property Eigenschaft2 As Integer

Public Sub Methode1()
   ...
End Sub

Public Function Methode2(Parameter As Integer) As Boolean
   ...
   Return bolErgebnis
End Function
```

- Anwendung

```
Dim meinObjekt As New Klasse()

Dim Wert As Integer = meinObjekt.Eigenschaft1

meinObjekt.Eigenschaft1 = „Hallo“

meinObjekt.Methode1();

Dim bolWert As Boolean = meinObjekt.Methode2(45)
```

Objekt-Beziehungen

- Eigenschaften können Objekte sein
- Eigenschaften können Mengen von Objekten sein

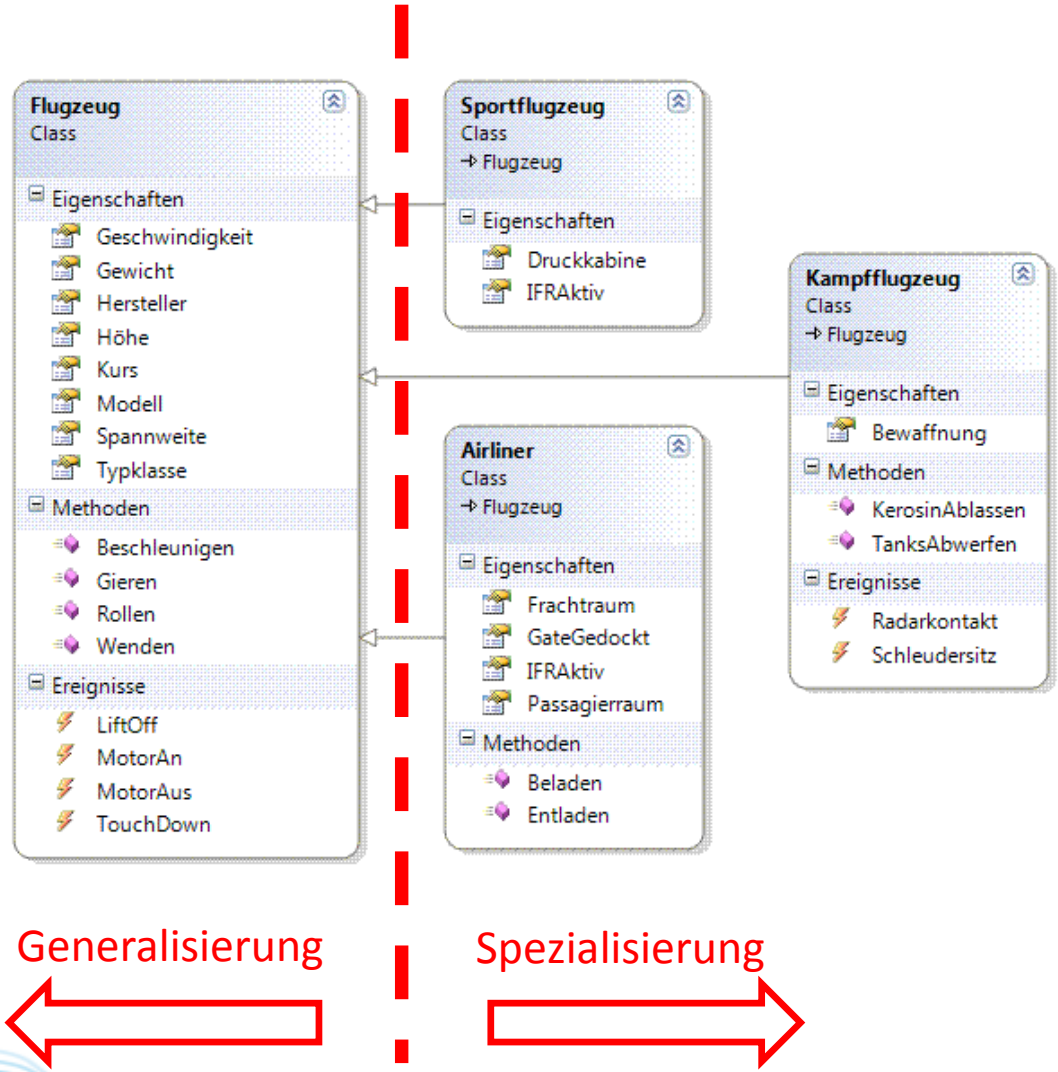


1:n Beziehung:
1 Flugzeug kann viele (n) Passagiere haben

1:1 Beziehung:
1 Flugzeug kann 1 Pilot haben

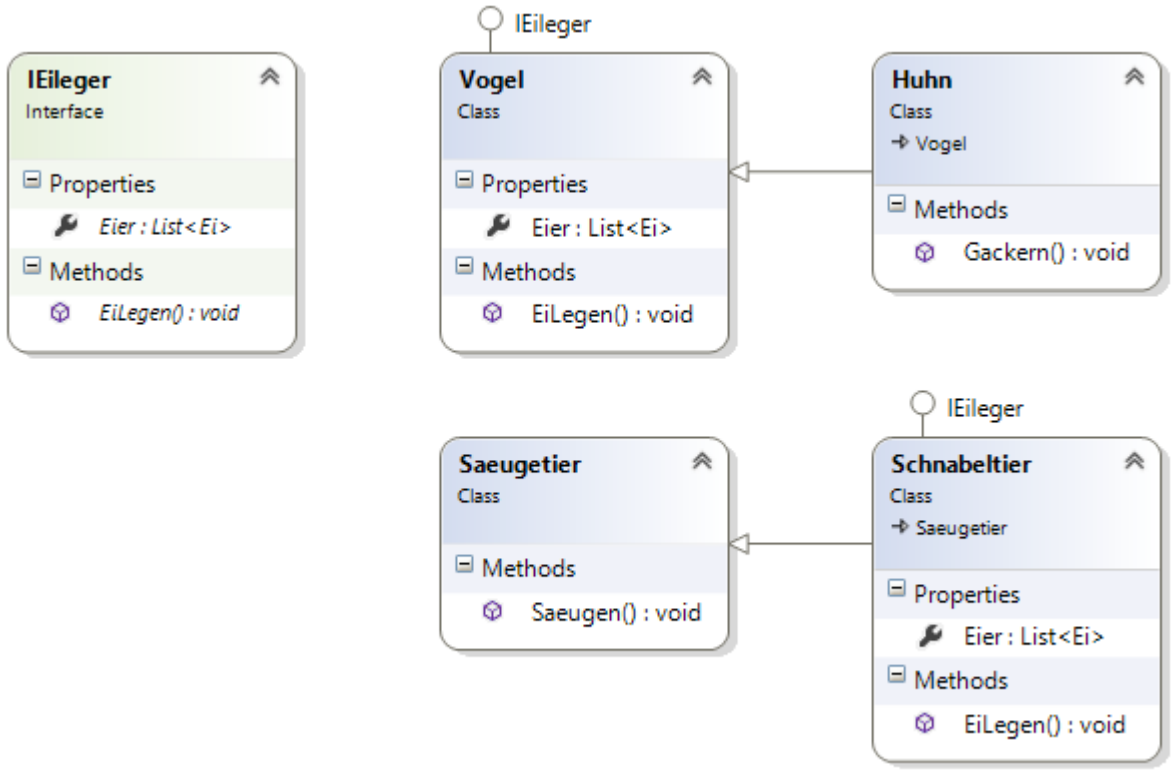
Vererbung

- Member werden von Basisklasse geerbt
  - „Erbe kann/hat alles, was der Vorfahre auch kann/hat, plus eigene Dinge."
  - Geerbt wird grundsätzlich **ALLES** (außer Konstruktoren)
- Basisklasse kann von anderer Basisklasse erben, die von weiterer Basisklasse erbt (Mehrfach**ver**erbung)
- **ABER:** Geerbt werden kann nur von **einer** Basisklasse (**keine** Mehrfach**be**erbung)



Interfaces

- Satz von Funktionalitäten ohne Implementierung
  - Implementierung geschieht in der jew. Klasse
- Eine Klasse kann beliebig viele Interfaces implementieren
  - Löst das Problem der nicht vorhandenen Mehrfachbeerbung
- „Diese Eigenschaften und Methoden brauchst Du, um hier mitspielen zu können, egal, was Du eigentlich bist."
  - Lose Koppelung, keine strikte Bindung an eine Basisklasse



Polymorphie

- Konzept:
  - Ziel: Einheitliche Vorgehensweisen
  - Gleiche/Ähnliche Dinge heißen gleich!

- .NET Framework:
  - Überladung von Methoden
  - Overridable/Abstract-Methoden in der Vererbung
  - Interfaces