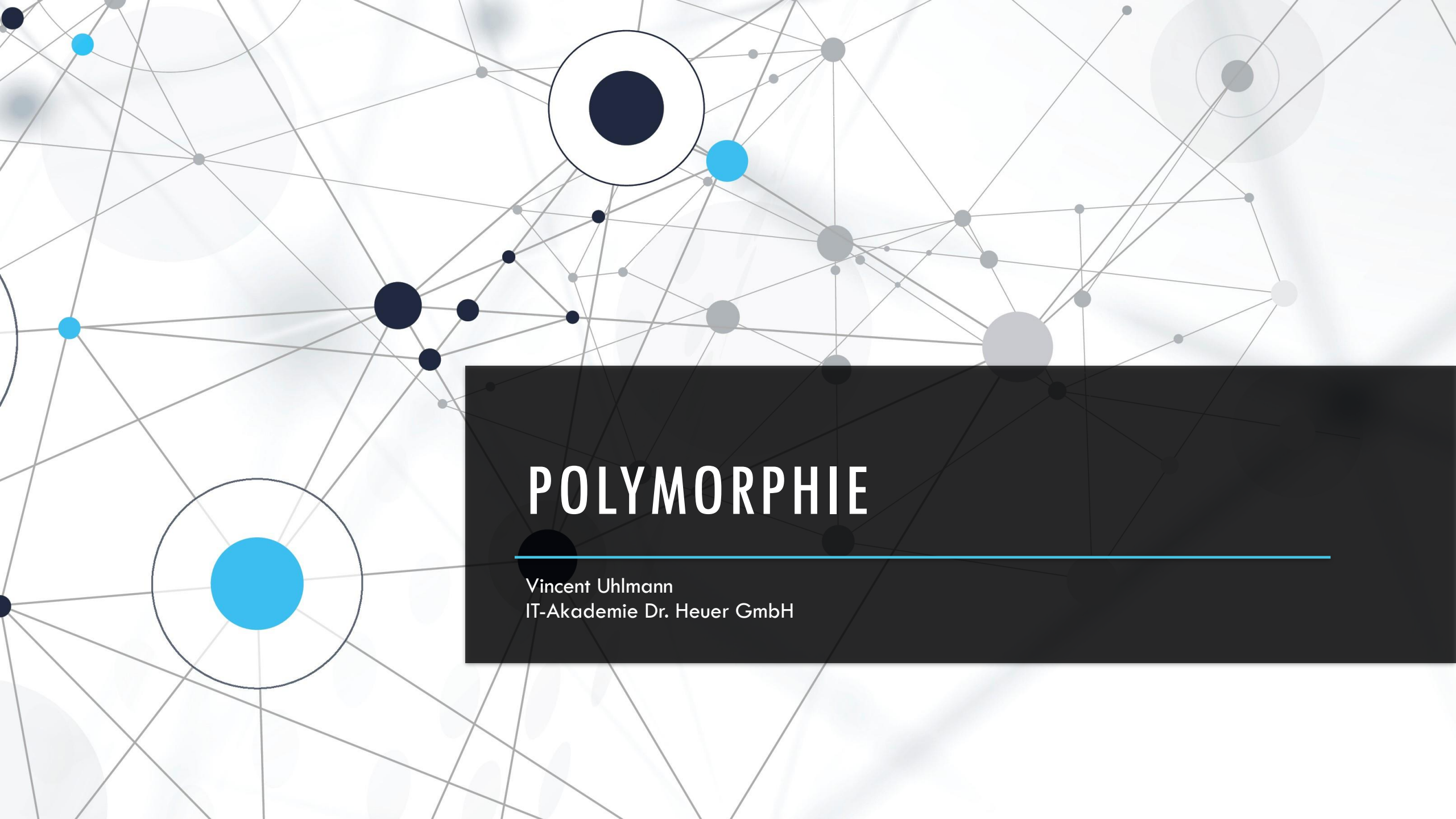
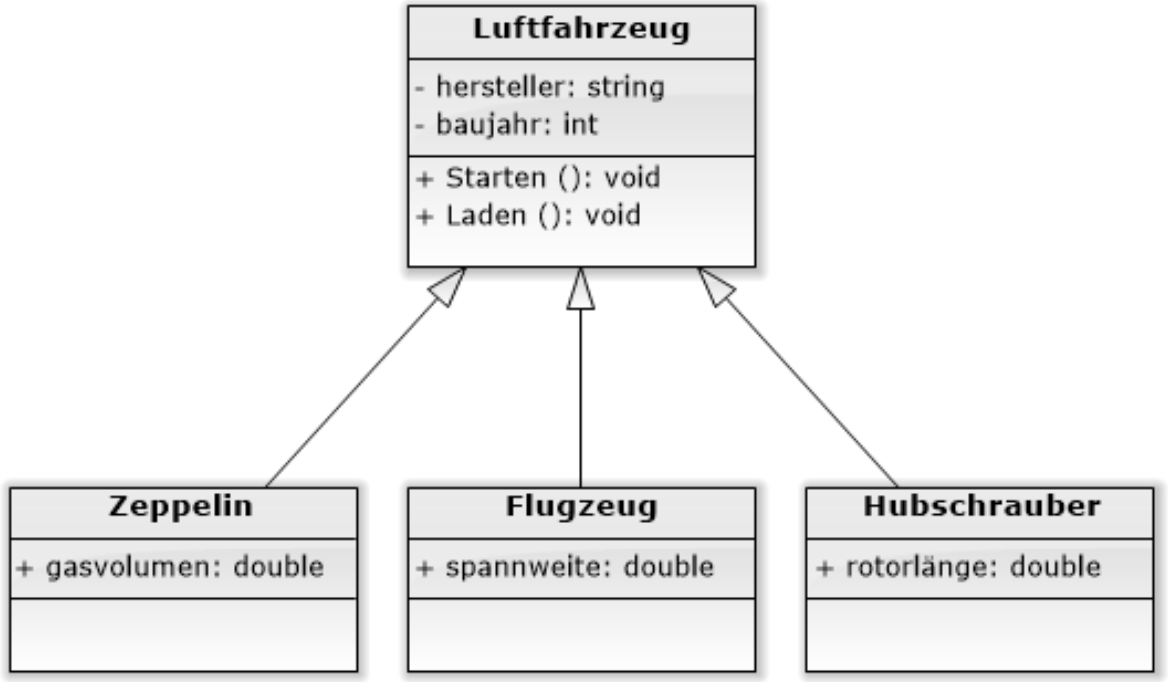


POLYMORPHIE

Vincent Uhlmann
IT-Akademie Dr. Heuer GmbH

POLYMORPHIE

- Eine Unterklasse kann gleichzeitig als ein Objekt der Oberklasse betrachtet werden
- Diese Beziehung wird als "ist-eine-Beziehung" bezeichnet
- Beispiel: Ein Hubschrauber wird als Luftfahrzeug betrachtet



POLYMORPHIE

- Es ist möglich, dass die Referenz eines Objektes der Unterklasse einer Referenz der Oberklasse zugewiesen wird
- Allgemein gilt Oberklassenobjekt = Unterklassenobjekt

```
Hubschrauber hubschrauber = new Hubschrauber();
Luftfahrzeug luftfahrzeug = hubschrauber;
```

TYPUMWANDLUNG

- Typumwandlung ermöglicht eine flexible Handhabung von Objekten
- Die Variablen zeigen auf denselben Speicherbereich
- Die Variable luftfahrzeug wird als Objekt vom Typ Luftfahrzeug betrachtet
- Dadurch ist kein Zugriff mehr auf Attribute oder Methoden möglich, die in der Klasse Hubschrauber definiert sind

```
Hubschrauber hubschrauber = new Hubschrauber();
Luftfahrzeug luftfahrzeug = hubschrauber;
```

TYPUMWANDLUNG

- Nützlich für z.B. Parameter in Methoden
- Nutzt implizite Konvertierung

- Ruft von jedem Objekt das von Luftfahrzeug abgeleitet wurde die geerbte Methode Starten auf

```
public void StarteLuftfahrzeug(Luftfahrzeug luftfahrzeug)
{
    luftfahrzeug.Starten();
}
```

EXPLIZITE TYPUMWANDLUNG

- Oberklassen-Objekt kann in Unterklassen-Objekt konvertiert werden
- Allgemein gilt Unterklassenobjekt = (Zielklasse) Oberklassenobjekt

- Eine horizontale Umwandlung ist nicht möglich
- Keine Konvertierung vom Flugzeug zum Hubschrauber möglich

- Nur möglich nach vorheriger impliziter Konvertierung

```
Hubschrauber hubschrauber = (Hubschrauber) luftfahrzeug;
```

IS-OPERATOR

- Bestimmung des Types mittels des is-Operators

```
if(luftfahrzeug is Hubschrauber)
{
    Console.WriteLine("Objekt ist ein Hubschrauber");
}
```

AS-OPERATOR

- Verwendung des as-Operators zur Typumwandlung
- Beide Ausdrücke erzielen dasselbe Ergebnis, jedoch gibt es einen Unterschied im Fehlerfall
- Die Umwandlung mit (Flugzeug) löst eine Ausnahme (Exception) aus
- Der as-Operator liefert null zurück

```
if(luftfahrzeug as Hubschrauber != null)
{
    // ..
}
```

VERERBUNGSVERHALTEN

- Methode in der Oberklasse implementieren und verdecken mit new
- Vererbung ist statisch
- Mittels new Keyword keine Polymorphie

- Abstrakte oder Virtuelle Methoden in der Oberklasse überschreiben mit override
- Vererbung ist polymorph (dynamisch)
- Mittels override keine statische Bindung

STATISCHE BINDUNG

- Bei der statischen Bindung werden Typen und Methoden während der Kompilierzeit bestimmt
- Die Entscheidung darüber, welche Methode aufgerufen wird, basiert auf dem statischen Typ der Variable
- Beispiel: Gibt “Das Luftfahrzeug ist gestartet” aus

```
internal class Program
{
    static void Main(string[] args)
    {
        Luftfahrzeug luftfahrzeug = new Zeppelin();
        luftfahrzeug.Starten();
    }
}

public class Luftfahrzeug
{
    public void Starten()
    {
        Console.WriteLine("Das Luftfahrzeug ist gestartet");
    }
}

public class Zeppelin : Luftfahrzeug
{
    public new void Starten()
    {
        Console.WriteLine("Der Zeppeling ist gestartet");
    }
}
```

DYNAMISCHE BINDUNG

- Bei der dynamischen Bindung werden Typen und Methoden zur Laufzeit bestimmt
- Die Entscheidung darüber, welche Methode aufgerufen wird, basiert auf dem tatsächlichen Typ des Objektes zur Laufzeit
- Polymorphie heißt: Aus einer Oberklassen-Referenz die typspezifische Methode aufrufen
- Beispiel: Gibt “Der Zeppelin ist gestartet” aus

```
internal class Program
{
    static void Main(string[] args)
    {
        Luftfahrzeug luftfahrzeug = new Zeppelin();
        luftfahrzeug.Starten();
    }
}

public class Luftfahrzeug
{
    public virtual void Starten()
    {
        Console.WriteLine("Das Luftfahrzeug ist gestartet");
    }
}

public class Zeppelin : Luftfahrzeug
{
    public override void Starten()
    {
        Console.WriteLine("Der Zeppelin ist gestartet");
    }
}
```

VIRTUAL

- Das Keyword virtual wird verwendet, um z.B. eine Methode in einer Basisklasse zu kennzeichnen, die von abgeleiteten Klassen überschrieben werden kann
- Klassenmethoden (static) können nicht virtuell sein
- Eine Kombination von virtual, abstract oder override ist nicht möglich
- Kann nicht bei privaten Methoden genutzt werden

```
public virtual void Starten()
{
    Console.WriteLine("Das Luftfahrzeug ist gestartet");
}
```