Funktionale OOP

- Mögliche Sichtweise: OO = Organisationsparadigma, Berechnungsmodell kann imperativ, funktional, etc. sein
- Widersprüche: Objekt hat Zustand, Verhalten, Identität; Referentielle Transparenz = kein Zustand, keine Identität
- Praktisch: Ein- und Ausgabe nicht referentiell transparent
  $\rightarrow$ auch in FP gibt es Zustände und Identität
  $\rightarrow$ aber in FP gibt es auch referentiell transparente Teile
- Tatsächlich: FP+OOP = FP-Teil getrennt von OOP-Teil
- Pattern Matching (als FP-Konzept) mit OOP verträglich
- Generizität geht in Richtung referentieller Transparenz

Delegate in C#

```
using System;
delegate void Sample(string message);
class MainClass {
    static void Method(string message)
        { Console.WriteLine(message); }
    static void Main() {
        // Instantiate delegate with named method:
        Sample d1 = Method;
        // Instantiate delegate with anonymous method:
        Sample d2 = delegate(string message)
                    { Console.WriteLine(message); };
        d1("Hello");
        d2(" World");
    }
}
```

Lambda-Ausdrücke in OOP

- *Delegate* bzw. *Closure* kapselt eine Funktion in ein Objekt; im Gegensatz zu Funktionszeigern ist Umgebung enthalten (anonymisiert, nur über gekapselte Funktion zugreifbar)
- *Lambda-Ausdruck* = direkt erzeugtes Delegate/Closure
- Einfache Schreibweise (syntactic suggar + Intelligenz)
- Umgebung wird implizit aufgebaut (ähnlich innerer Klasse)
- Typinferenz für Parameter (da kein dynamisches Binden)
- Eher bei applikativem Programmierstil große Bedeutung

Geschichte der OOP

- Sprachen: Simula, Smalltalk, Objective-C, C++, Eiffel, CLOS, Oberon, Java
- Konzepte: Strukturierte P., Abstraktion, Vererbung, Ersetzbarkeit, Schnittstellenspezifikation, Parametrisierung
- Methoden: Faktorisierung, Use-Cases, graphische Darstellung (UML), Patterns, Pair-Programming
- Konflikte: Funktionale Prog., relationale DB, Collections + kovariante Probleme, formale Komplexität, Nebenläufigkeit
- Entwicklungslinie: objektbasiert – objektorientiert – (teilautomatisiert) – typisiert – team+architekturintegriert

Zukunft der OOP

- OOP allgegenwärtig $\Rightarrow$ Innovations-Charakter verloren
- erleben heute Aufsplittung in viele Rand- u. Detailthemen
- wichtige Bereiche in absehbarer Zukunft: Parallelität, Verteiltheit, Datenintegration, vielschichtige Architekturen, Cloude-Computing, komplexe Schnittstellenprotokolle
- in vielen dieser Bereiche derzeit mehr Fragen als Antworten
- Sprachunterstützung zu erwarten, sobald die wichtigsten Fragen beantwortet sind