Metaobjektprotokolle

Alexander Konstantinov

Anwendungen 1 – WiSe 2011/12

HAW Hamburg – Masterstudiengang Informatik

Motivation

(objektorientierte) Programmiersprachen

- Abstraktionsmittel
  - Klassen, Methoden, Funktionen, Threads, Packages...
- Semantik

  "Jede Klasse hat eine Superklasse und 0..n Interfaces"
  "Es wird die spezifischste Methode im Reciever gewählt"
  - spezifiziert (in einer Sprachspezifikation)

modifizierte Abstraktionsmittel/Semantik ?

...neue Programmiersprache?

...Framework erstellen?

ein Metaobjektprotokoll nutzen [9]

Agenda

Motivation

Agenda

Begriffsdefinition: Metaobjektprotokoll (MOP)

Beschaffenheit von MOPs

- Introspektion
- Modifikation
- Interzession

(aktuelle) Entwicklungen

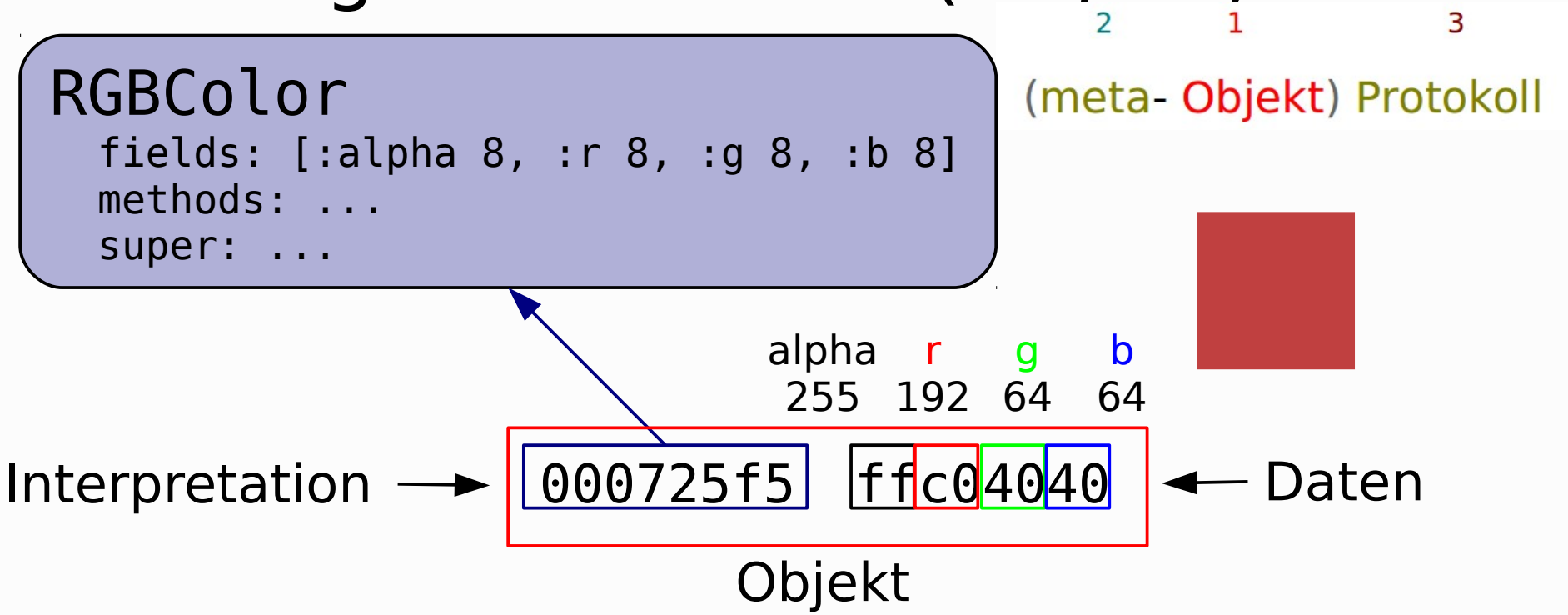
Begriffsdefinition

Metaobjektprotokoll

2 1 3

(meta- Objekt) Protokoll

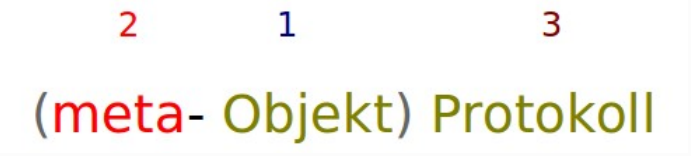
Begriffsdefinition (Objekt)



Objekt:

- Daten mit eingebetteter Interpretation

Begriffsdefinition (meta-)



Metakommunikation:

- Kommunikation, welche Kommunikation beschreibt

Metamodell:

- ein Modell, welches ein Modell beschreibt

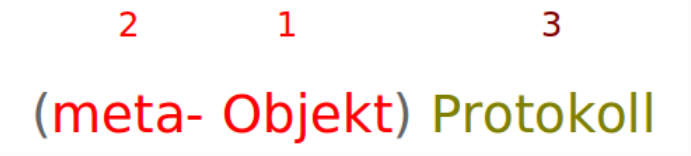
Metatheorie:

- eine Theorie, welche eine Theorie beschreibt

Meta<X>:

- ein <X>, welches ein <X> beschreibt

Begriffsdefinition (Metaobjekt)



Metakommunikation:

- Kommunikation, welche Kommunikation beschreibt

Metamodell:

- ein Modell, welches ein Modell beschreibt

Metatheorie:

- eine Theorie, welche eine Theorie beschreibt

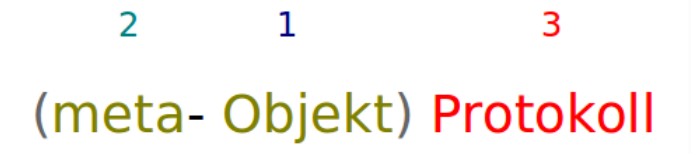
Metaobjekt:

- ein Objekt, welches ein Objekt beschreibt

  z.B. Klassenobjekte:

  ```
  Person.class Queue.class Apple.class etc.
  ```

Begriffsdefinition (Protokoll)



Operationen eines Datentyps

```
anApple // Fruit
```

Accessoren	`Fruit -> Other`	`anApple.weight()` `anApple.color()`
Mutatoren	`Fruit -> Fruit`	`anApple.wash()` `hans.wax(anApple)`
Konstruktoren	`Other -> Fruit`	`new Apple(seed,time)` `store.buyOrange()`

Begriffsdefinition (Metaobjektprotokoll)

2 1 3

(meta- Objekt) Protokoll

Operationen eines Metaobjekts

```
aplCls = anApple.class()
```

Accessoren (Introspektion)

```
aplCls.getMethods()
aplCls.getFields()
```

Mutatoren (Modifikation)

```
???
```

Konstruktoren (Interzession)

```
???
```

Agenda

Motivation

Agenda

Begriffsdefinition: Metaobjektprotokoll (MOP)

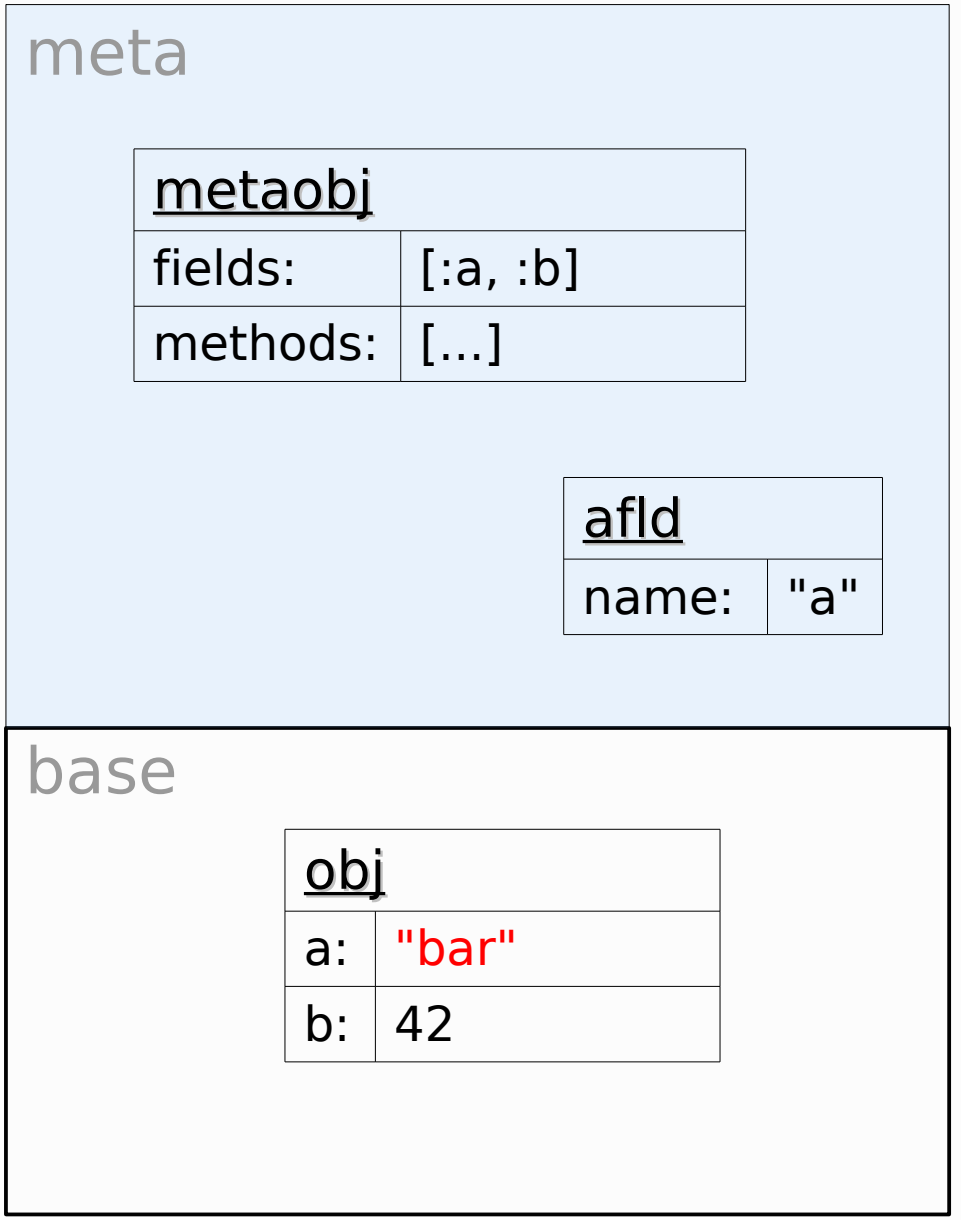
Beschaffenheit von MOPs

- Introspektion
- Modifikation
- Interzession

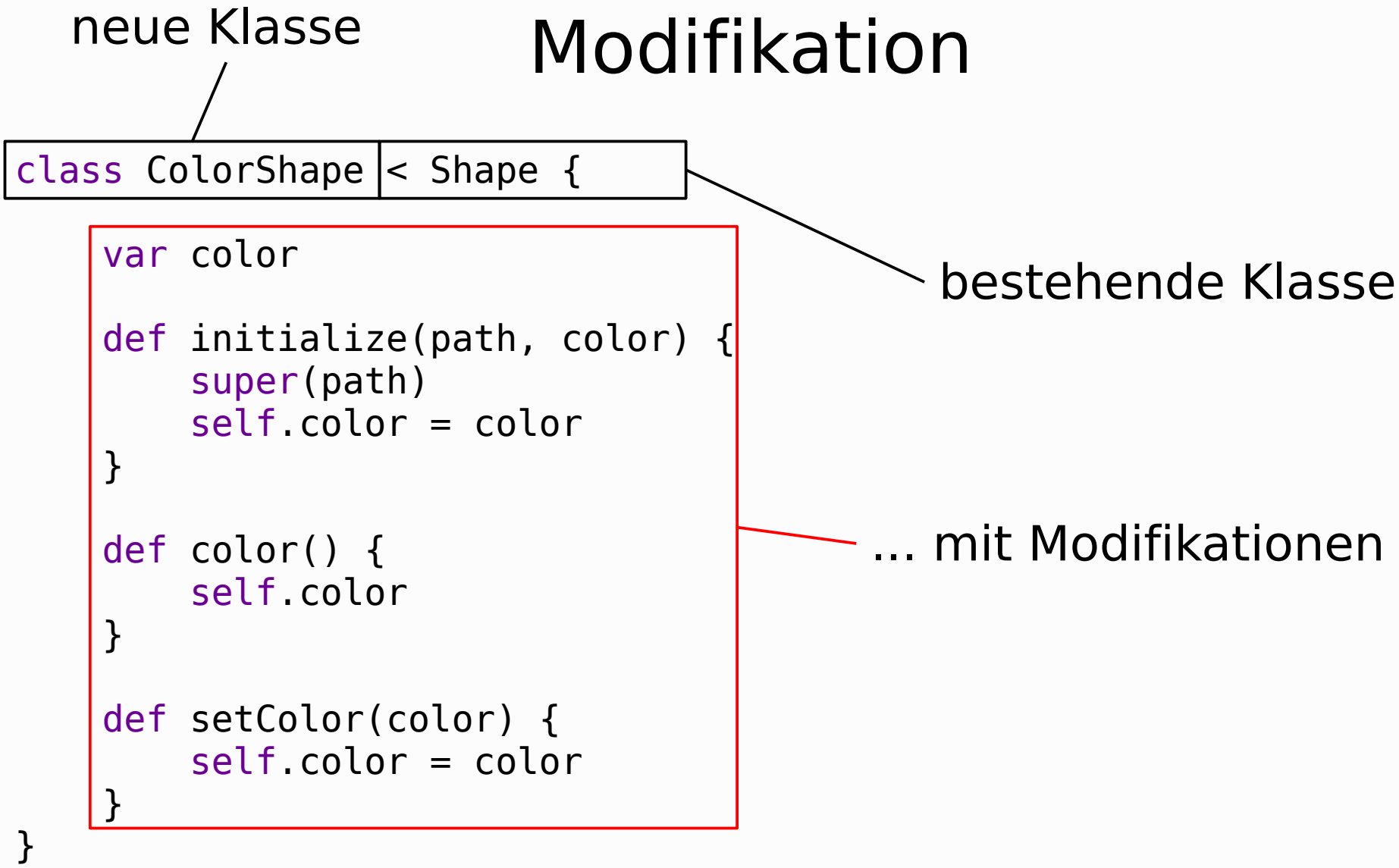
(aktuelle) Entwicklungen

Introspektion

```
obj
metaobj = obj.class()

afld = metaobj.getField(:a)
afld.name() // => "a"

afld.get(obj) // => "foo"
afld.set(obj, "bar")

obj.a // => "bar"
```

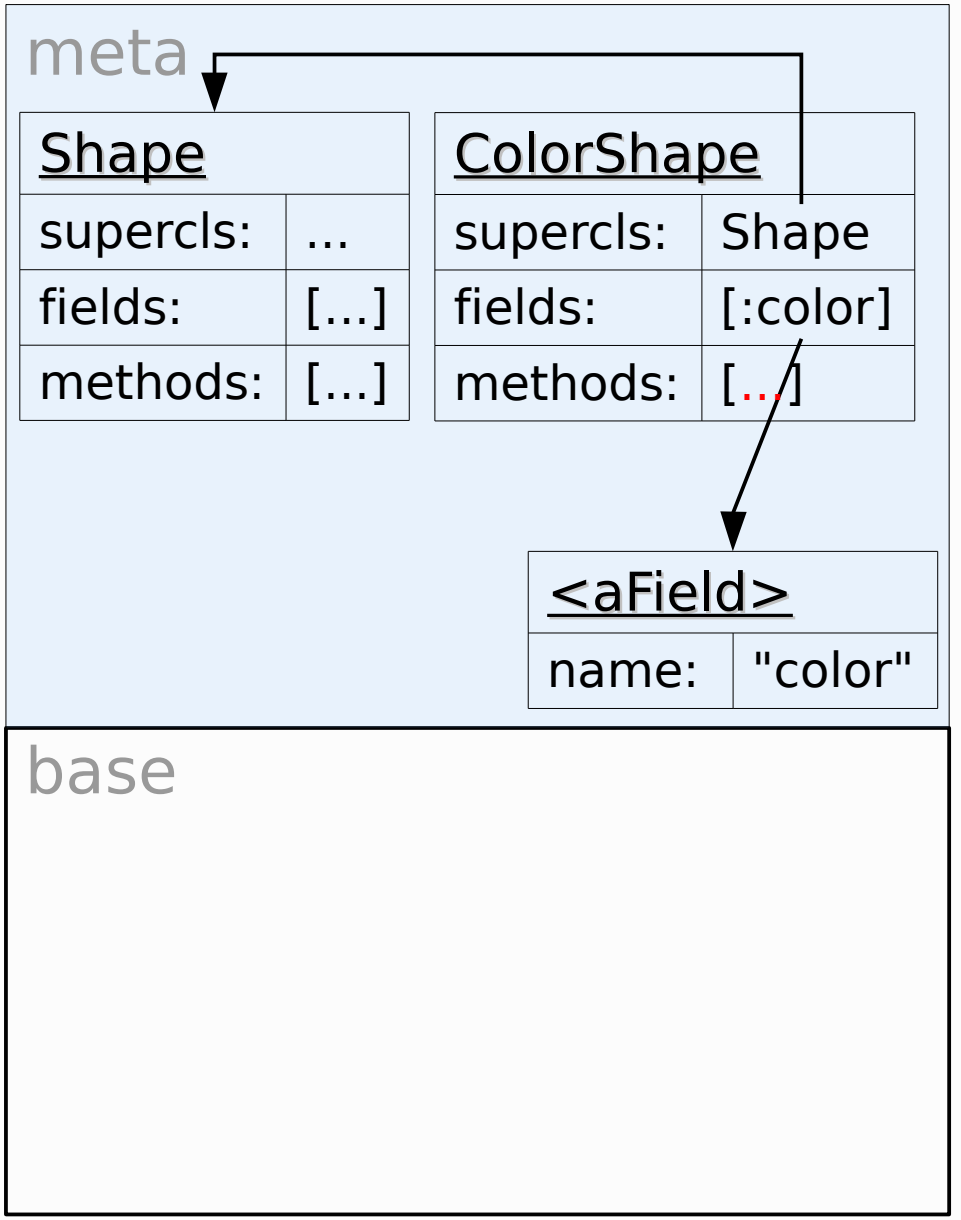


Modifikation



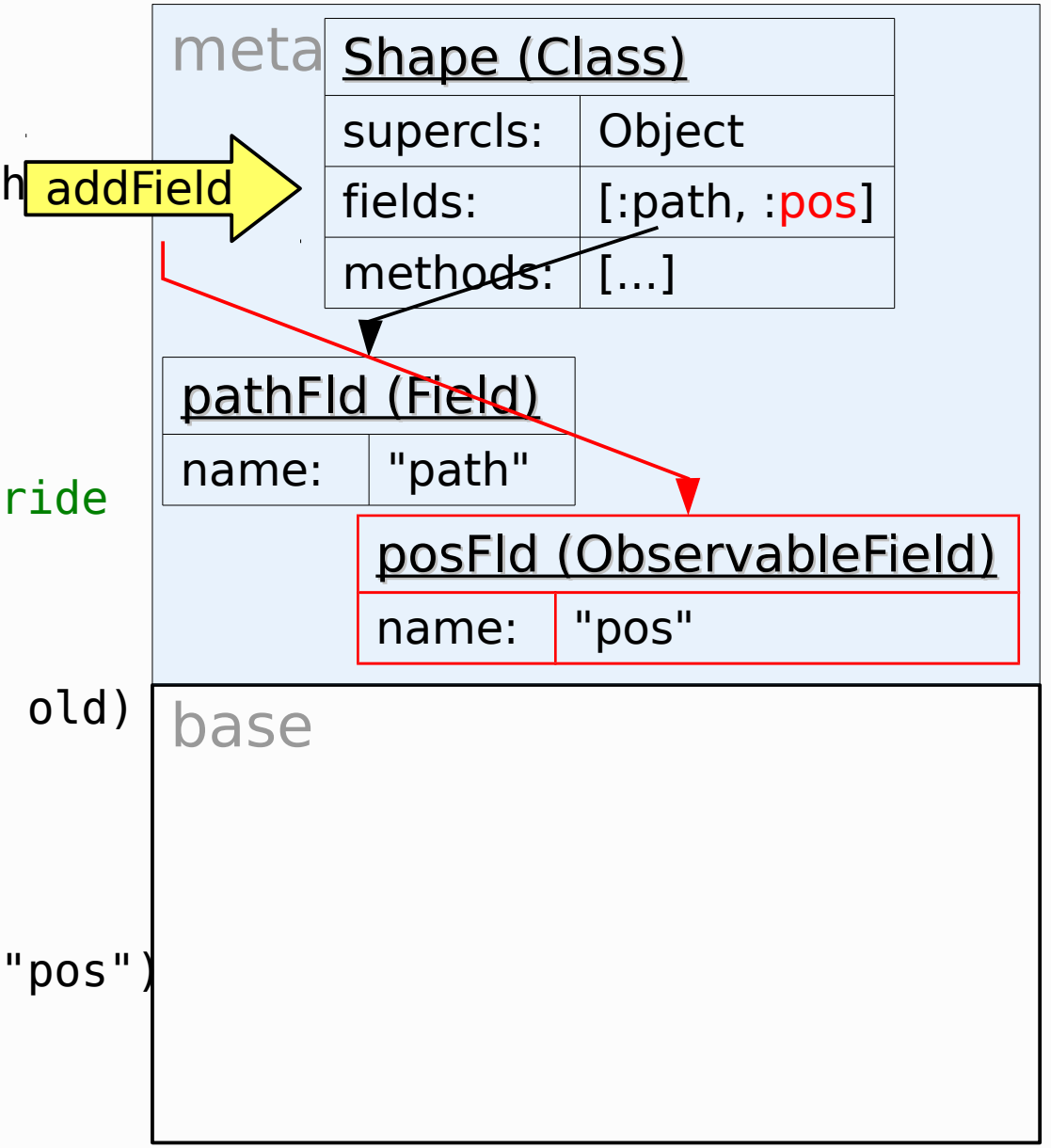
Modifikation

```
ColorShape = Shape.subclass()

ColorShape.addField(:color)
CS = ColorShape
CS.addMethod(:initialize, {|p, c|
    super(p)
    this.color = c
})

CS.addMethod(:color, {||
    this.color
})

CS.addMethod(:setColor, {|c|
    this.color = c
})
```



Interzession

```
Shape

pathFld = Shape.getField(:path)

Shape.class() // => Class
pathFld.class() // => Field

class ObservableField < Field
    def set(obj, new) // override
        old = get(obj)
        obs = obj.observers()
        super(obj, new)
        obs.notify(this, new, old)
    end
    ...
end

posFld = new ObservableField("pos")
Shape.addField(:pos, posFld)
```

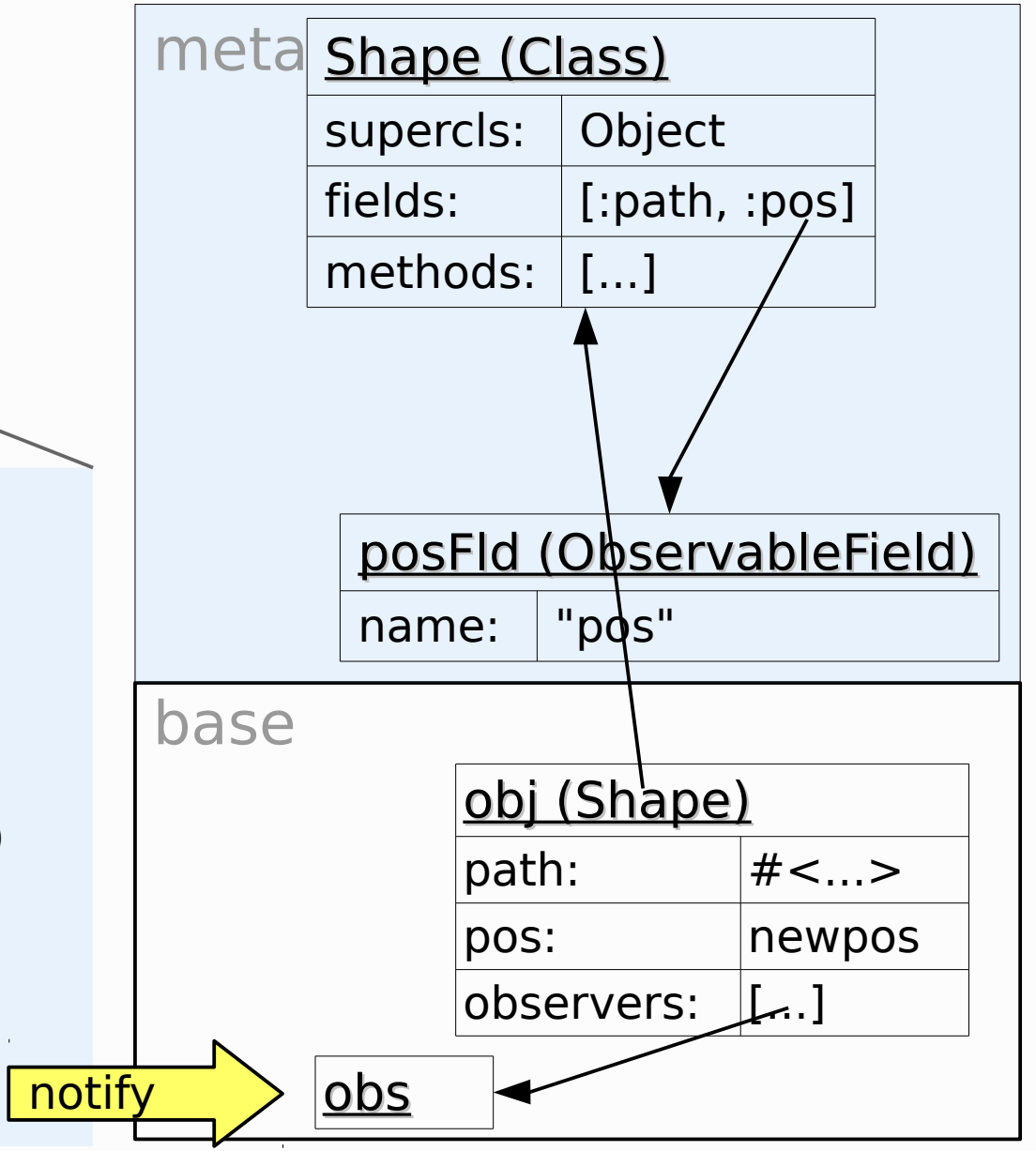


Interzession

```
obj = new Shape(...)

obj.pos = newpos
```

meta

```
// ObservableField#set
def set(obj, new)
    old = get(obj)
    obs = obj.observers()
    super(obj, new)
    obs.notify(this, new, old)
end
```



Zusammenfassung

MOP: Verhältnis zwischen Base- und Metaebene [3]

- Einsicht auf Programmstrukturen/Metaobjekte
  - Introspektion, "Reflection"
- Generierung und Modifikation von Programmen
  - (selbst-) Modifikation, Metaprogrammierung
- (re)-Definition von Programmiersprachensemantik
  - Interzession
    - Feldzugriff, Methodensuche/-aufruf, Vererbung etc.
    - "open implementation" [9]

Zusammenfassung

MOP: Verhältnis zwischen Base- und Metaebene [3]

- Einsicht auf Programmstrukturen/Metaobjekte
  - Introspektion, "Reflection"
- Generierung und Modifikation von Programmen
  - (selbst-) Modifikation, Metaprogrammierung
- (re)-Definition von Programmiersprachensemantik
  - Interzession
    - Feldzugriff, Methodensuche/-aufruf, Vererbung etc.
    - "open implementation" [9]

"We believe that providing an open implementation can be advantageous in a wide range of high-level languages and that metaobject protocol technology is a powerful tool for providing that power to the programmer."

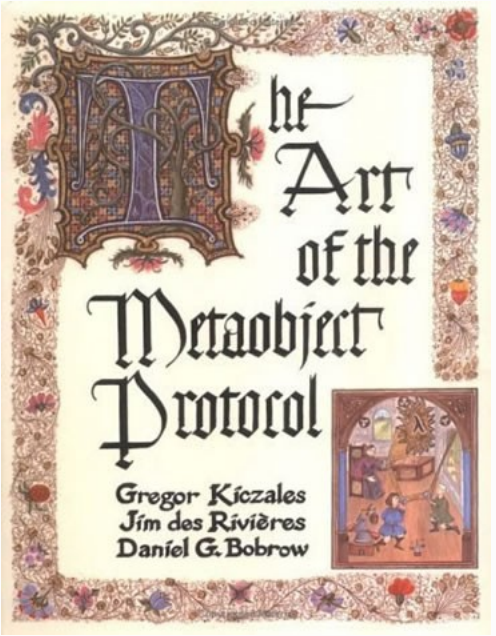
Gregor Kiczales et al. [2]

Akteure und Entwicklungen

1991:

- "The Art of the Metaobject Protocol"

  Gregor Kiczales, Jim des Rivières, Daniel G. Bobrow
- beschreibt die "open implementation" des Objektsystems von Common Lisp
- Standardwerk für Reflection, Interzession und Metaobjektprotokolle



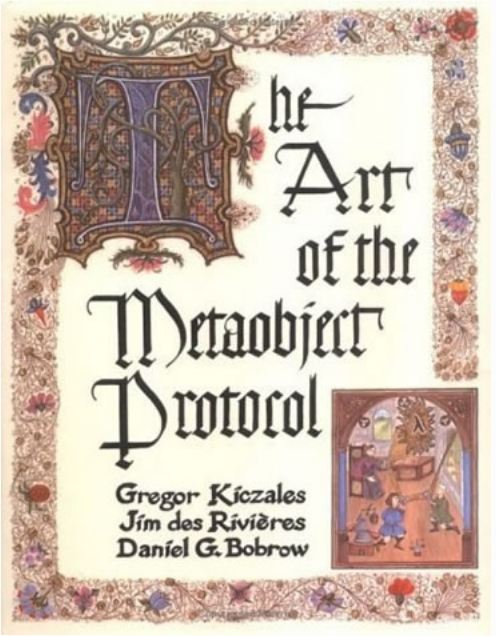
[Abb1]

Akteure und Entwicklungen

1991:

- "The Art of the Metaobject Protocol"

  Gregor Kiczales, Jim des Rivières, Daniel G. Bobrow
- beschreibt die "open implementation" des Objektsystems von Common Lisp
- Standardwerk für Reflection, Interzession und Metaobjektprotokolle



[Abb1]

"When they wrote this book, I called them up and I said, 'This is the best book anybody has written in ten years'[...] this book has some of the most profound [...] and the most pratical insights about OOP than anybody has done in the last many years."

Alan Kay. "The Computer Revolution Hasn't Happend Yet!" Keynote OOPSLA 1997 [6]

Akteure und Entwicklungen

2004:

- "Mirrors: Design Principles for Meta-level Facilities of Object-Oriented Programming Languages"

  Gilad Bracha, David Ungar

- Metaobjekte in einer prototypbasierten Sprache (Self)

2007:

- "Mirages: Behavioral Intercession in a Mirror-based Architecture."

  T. Van Cutsem, S. Mostinckx, S. Timbermont, E. Tanter.

- Interzession in AmbientTalk, einer prototypbasierten Sprache für ambient intelligence

Akteure und Entwicklungen

2010:

- "Proxies: Design principles for robust object-oriented intercession APIs"

  Tom Van Cutsem, Mark S. Miller

- Interzession für JavaScript
  - Teil des "ECMAScript Harmony"-Projekts
    - Entwicklung für den ECMAScript 6 Standard
  - Implementiert in den JS-Engines SpiderMonkey (Mozilla Firefox) und V8 (Google Chrome)

Konferenzen und SIGs

ACM SIGPLAN

(**S**pecial **I**nterest **G**roup on **P**rogramming **LAN**guages)

SPLASH Conference

(**S**ystems, **P**rogramming, **L**anguages and **A**pplications: **S**oftware for **H**umanity)

- früher: OOPSLA

(**O**bject-**O**riented **P**rogramming, **S**ystems, **L**anguages & **A**pplications)

ECOOP

(**E**uropean **C**onference on **O**bject-**O**riented **P**rogramming)

Dynamic Languages Symposium

- meistens Teil von OOPSLA/SPLASH

Danke für die Aufmerksamkeit!

Literatur

[1]Gregor Kiczales and Jim Des Rivieres. 1991. **The Art of the Metaobject Protocol**. MIT Press, Cambridge, MA, USA.

[2]Gregor Kiczales, J. Michael Ashley, Luis H. Rodriguez, Jr., Amin Vahdat, and Daniel G. Bobrow. 1993. **Metaobject protocols: why we want them and what else they can do**. In *Object-oriented programming: The CLOS Perspective*, Andreas Paepcke (Ed.). MIT Press, Cambridge, MA, USA p.101-118.

[3]Gilad Bracha and David Ungar. 2004. **Mirrors: design principles for meta-level facilities of object-oriented programming languages**. In *Proceedings of the 19th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOPSLA '04)*. ACM, New York, NY, USA, p.331-344.

[4]Tom Van Cutsem, Stijn Mostinckx, Stijn Timbermont, and Éric Tanter. 2007. **Mirages: behavioral intercession in a mirror-based architecture**. In *Proceedings of the 2007 symposium on Dynamic languages (DLS '07)*. ACM, New York, NY, USA, p.89-100.

Literatur

[5]Tom Van Cutsem and Mark S. Miller. 2010. **Proxies: design principles for robust object-oriented intercession APIs**. In *Proceedings of the 6th symposium on Dynamic languages (DLS 2010)*. ACM, New York, NY, USA, p.59-72.

[6]Alan Kay. 1997. **The Computer Revolution Hasn't Happened Yet**. Keynote in *ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages & Applications (OOPSLA '97)*, Atlanta, Georgia, October 5-9, 1997.

[7]Tom Van Cutsem, Stijn Mostinckx, Stijn Timbermont, Elisa Gonzalez Boix, Éric Tanter, and Wolfgang De Meuter. 2009. **Mirror-based reflection in AmbientTalk**. *Softw. Pract. Exper. 39, 7 (May 2009)*, p.661-699.

[8]Andreas Paepcke. 1993. **User-level language crafting: introducing the CLOS metaobject protocol**. In *Object-oriented programming: The CLOS Perspective*, Andreas Paepcke (Ed.). MIT Press, Cambridge, MA, USA p.65-99.

Literatur

[9]Gregor Kiczales. 1996. **Beyond the Black Box: Open Implementation**. IEEE Softw. 13, 1 (January 1996), 8-11.

Bilder

[Abb1]The Art of the Metaobject Protocol - The MIT Press
http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=3925