MetaclassTalk

a Testbed for Exploring Programming Paradigms

Noury Bouraqadi
Computer Science Laboratory (CSL)
Ecole des Mines de Douai
France

Starting Point

- ***Goal: Experiment Various Paradigms***
  - *Multiple-Inheritance*
  - *Aspect-Oriented Programming*
  - *Mixin-based Inheritance*
  - *Types*
  - *...*
- ***Requirements: a "Ball of Mud" (© R. Fateman ;-)***
  - *OO Programming Language*
  - *API + "framework" to ease changes*

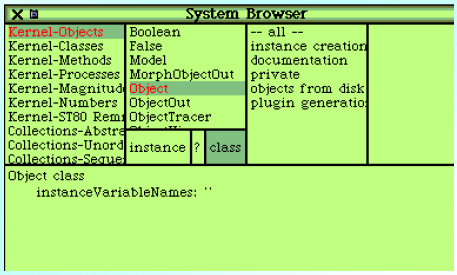
Why Extend Smalltalk?

- ✓ ***Powerfull (Reflective Facilities)***
- ✓ ***Simple & Uniform***

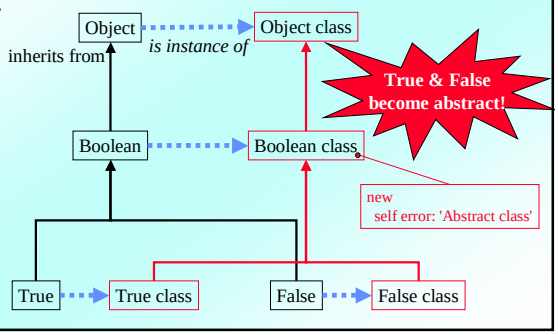
- × ***Difficult to define new kinds of classes***
  - × *Metaclasses are Implicit*
  - × *Parallel Inheritance Hierarchies*
- × ***Complex to change the evaluation process***
- × ***No API or framework easing extensions***

Smalltalk Metaclasses are Implicit

Metaclasses: Classes which instances are also classes



Smalltalk Parallel Inheritance Hierarchies

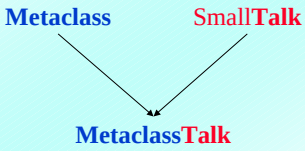


Outline

- ***What is MetaclassTalk?***
- ***Class Properties Reuse Using Mixins***
- ***Aspect-Oriented Programming Using MetaclassTalk***
- ***Conclusion***

What is MetaclasTalk?

Meta… what?



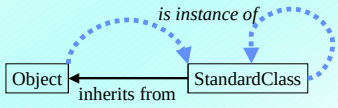
Programming with Explicit Metaclasses

What to do with Metaclasses?

- **Class Properties**
  - Definition of new kinds of classes
- **Control of Program Execution**
  - instance creation
  - IV reads & writes
  - message sends & reception
  - method lookup & evaluation

MetaclassTalk "Conceptual" Kernel

The ObjVLisp Model [Cointe 87]
= The Smalltalk-76 Kernel + ability to subclass the root metaclass



Metaclass Core Protocol

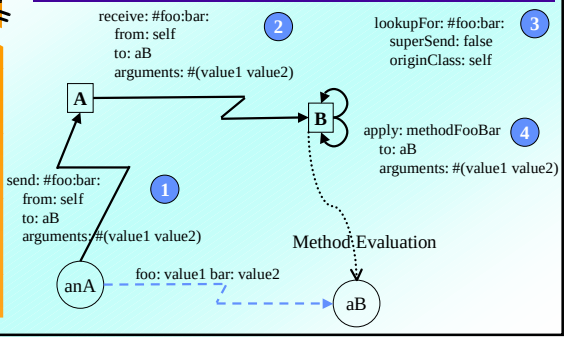
Each class controls
the evaluation process for its instances

- *instance memory allocation* (allocate)
- *object creation* (new = allocate ∘ initialize)

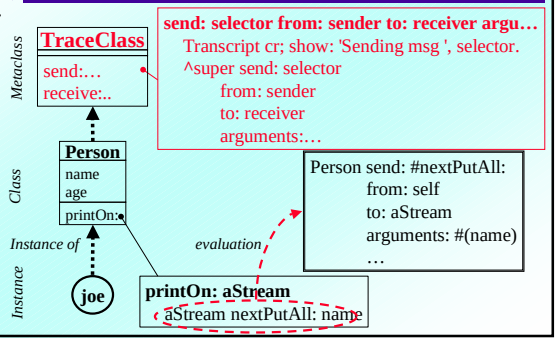
- *reading instance variables* (atIV:of:)
- *writing variables d'instance* (atIV:of:put:)

- *sending messages* (send:…)
- *receiving messages* (receive:…= lookupFor:… ⊕ apply:…)
- *method lookup* (lookupFor:…)
- *method evaluation* (apply:…)

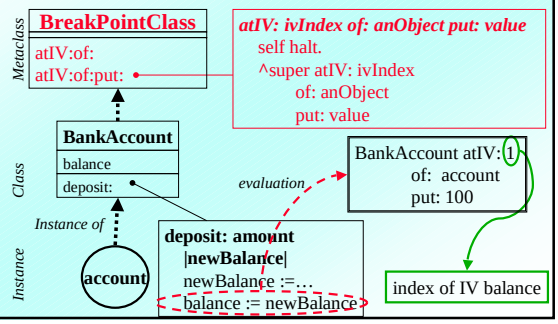
Decomposition of a Message Sending



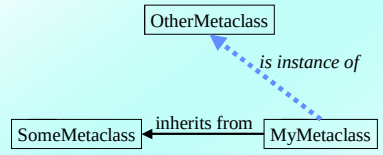
Example of Message Sending Control



Example of IV Read-Write Control

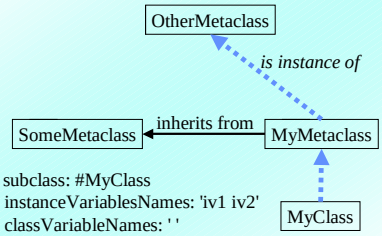


How to Use MetaclassTalk?




```
SomeMetaclass subclass: #MyMetaclass
        instanceVariablesNames: 'x y'
        classVariableNames: ' '
        poolDictionaries: ' '
        category: 'MetaclassTalk-Example'
        metaclass: OtherMetaclass
```


How to Use MetaclassTalk?




```
Object subclass: #MyClass
        instanceVariablesNames: 'iv1 iv2'
        classVariableNames: ' '
        poolDictionaries: ' '
        category: 'MetaclassTalk-Example'
        metaclass: MyMetaclass
```


Some Possible Uses of the MetaclassTalk's MOP

	Allocate	New	Read	Write	Send	Receive	Lookup	Apply
Abstract Class		✗					✗	
Asynchronous Communication					✗	✗		
Break Point			✗	✗		✗		
Lazy Memory Allocation	✗		✗	✗				
Message Caching						✗		
Multiple Inheritance							✗	
Persistent Objects (Data Base)	✗		✗	✗				
Sole Instance (Pattern)		✗						
Subject Class (Observer Pattern)				✗				✗
Synchronisation			✗	✗				✗
Tracing			✗	✗	✗	✗		
Type Checking							✗	

Class Properties Reuse Using Mixins

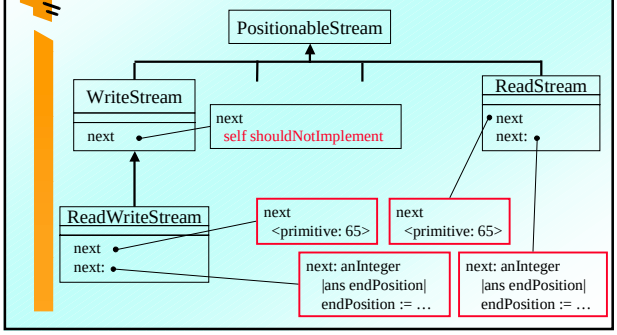
19

What are Mixins?

- **Context:**
  - Single inheritance
  - Unreleated class hierarchies
- **Problem:**
  - Reuse shared properties
  - Avoid code duplication
- **Solution:**
  - Mixin = Subclass builder [Bracha 90]

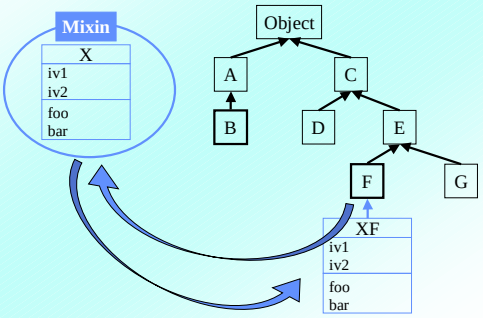
20

Example of Code Duplication



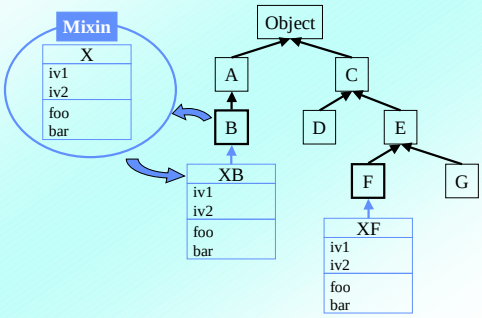
21

How Do Mixins Work?



22

How Do Mixins Work?



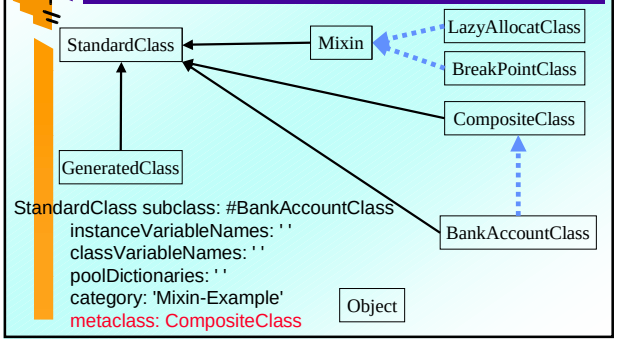
23

Mixins Implementation in MetaclassTalk

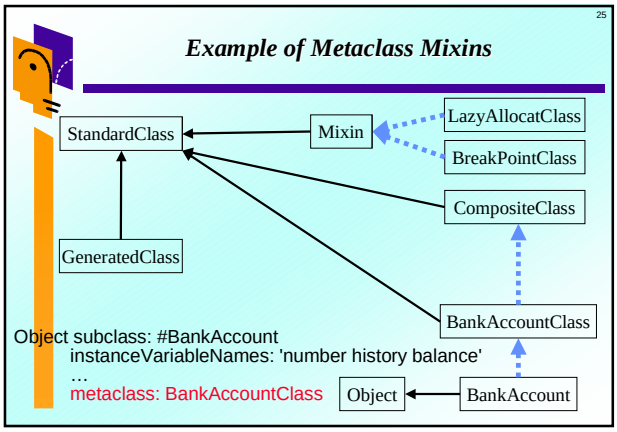
- **3 metaclasses (i.e. 3 class properties):**
  - Mixins
  - Classes generated by mixins
  - Classes that make use of mixins
- **Generated classes are updated on mixin change**
- **Use of Smalltallk reflective facilities:**
  - Method Compilation
  - Class Building
  - Adds & Removals of Methods and Instance Variables

24

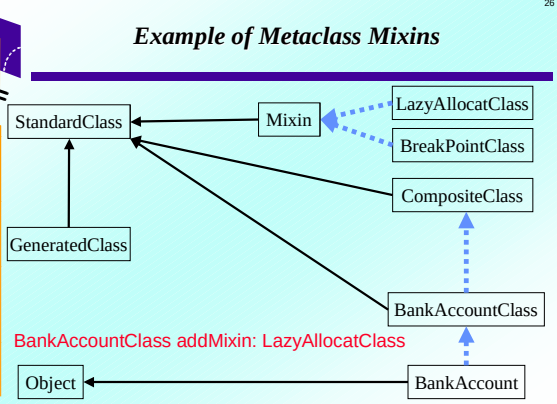
Example of Metaclass Mixins



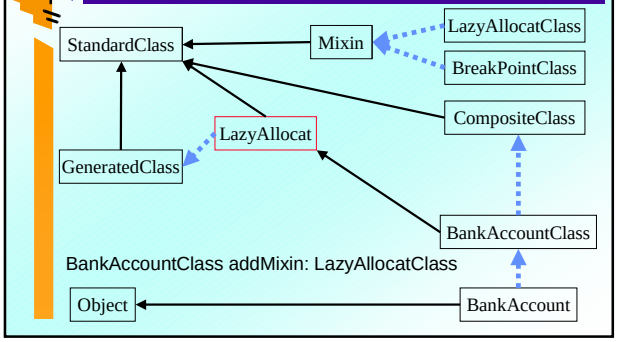
Example of Metaclass Mixins



Example of Metaclass Mixins



Example of Metaclass Mixins



Example of Metaclass Mixins

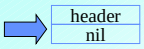
```
StandardClass subclass: #BankAccountClass
    instanceVariableNames: ' '
    classVariableNames: ' '
    poolDictionaries: ' '
    category: 'MetaclassTalk-Example'
    metaclass: CompositeClass.

BankAccountClass mixins: {LazyAllocatClass}
```

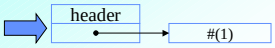

Example of Metaclass Mixins

```
Object subclass: #BankAccount
    instanceVariableNames: 'number history balance'
    classVariableNames: ' '
    poolDictionaries: ' '
    category: 'MetaclassTalk-Example'
    metaclass: BankAccountClass.
```

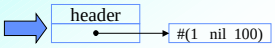
```
|account|
account := BankAccount new.
```



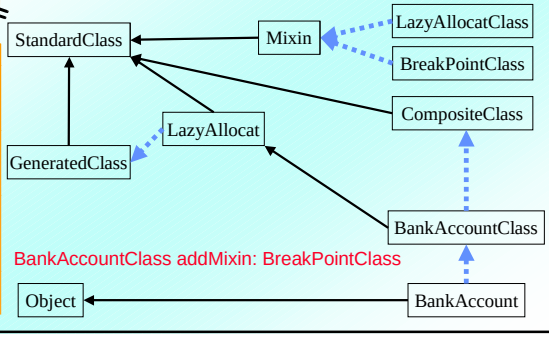

```
account number: 1.
```



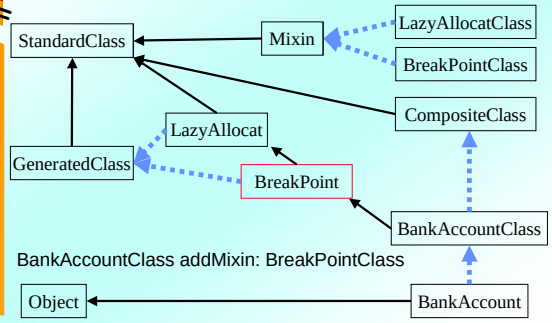

```
account balance: 100.
```



Example of Metaclass Mixins



Example of Metaclass Mixins



Example of Metaclass Mixins

```
StandardClass subclass: #BankAccountClass
   …
   metaclass: CompositeClass.

BankAccountClass mixins:
                        {LazyAllocatClass, BreakPointClass}
```

```
Object subclass: #BankAccount
   instanceVariableNames: 'number history balance'
   …
   metaclass: BankAccountClass.

BankAccount breakOnSelectors: #( ) andIVs: #( )
```

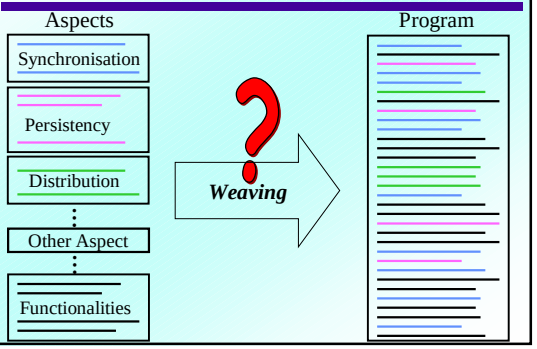

Mixins: a Usefull Extension

- ***Experiment of a New Paradigm***
- ***A Tool for Composing class Properties***

Aspect-Oriented Programming Using MetaclassTalk

Experimenting a New Programming Paradigm

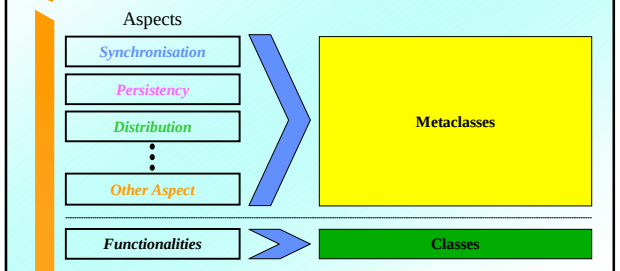
AOP



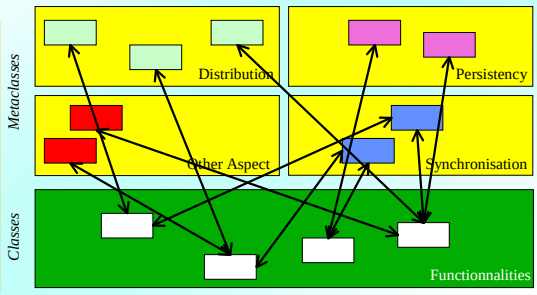
Issues & Solutions

- ***How to define isolated aspects?***
  - *Metaclasses*
- ***How to weave aspects to build an application?***
  - *The "instance-of" link*
  - *Mixin-based inheritence*
- ***What about reuse?***
  - *Generic Metaclasses*

A First Natural Separation



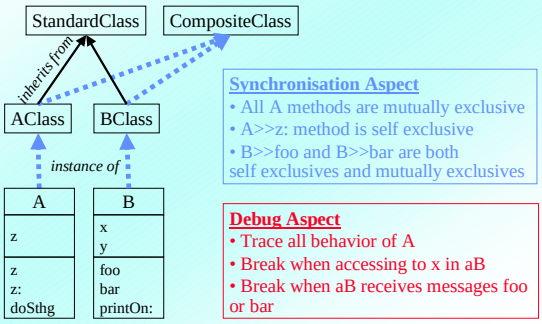
How to Isolate & Weave Aspects?



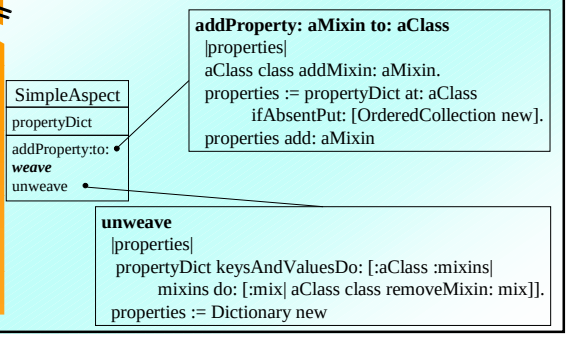
Aspects in MetaclassTalk

1. **Set of Class Properties: Metaclasses**
2. **Set of Classes**
3. **Class-Metaclass relationships**
4. **Class initialization (Properties setup)**

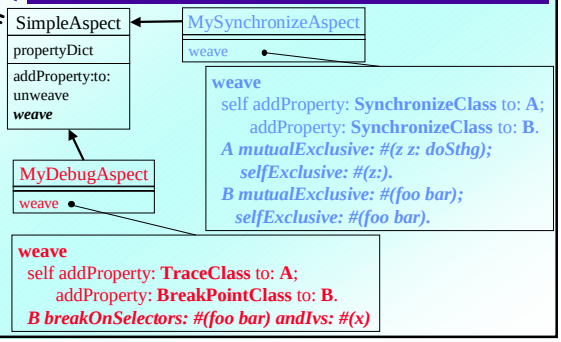
Example - Starting Point



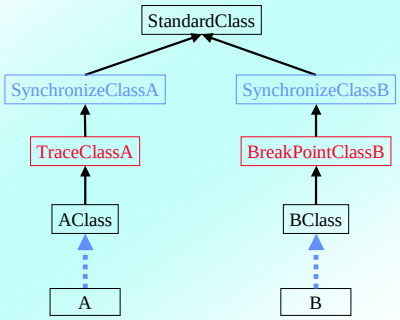
Easy Implementation



Example - Aspects Implementation



Example - After Weaving



MetaclassTalk for AOP

- ***Easy implementation of AOP***
- ***Reuse***
  - *Metaclasses are generic*
- ***Dynamicity***
  - *Run-time weaving/unweaving*

Conclusion

Summary

- ***MetaclassTalk eases experiments***
  - *Mixins*
  - *AOP*
- ***MetaclassTalk is usefull***
  - *Definition of new kinds of classes*
  - *Change the evaluation process*
    - *creation*
    - *IV access*
    - *Message handling*
- ***An implementation is available for Squeak 3.0***

Near Future Plans

- ***Extending the library of metaclasses***
- ***Migration to the latest Squeak release***
- ***Improving Performance:***
  - *Code inlining?*
  - *Extending the VM?*
  - *Other?*

Long Term Plans

- ***Rebuild a complete image with explicit metaclasses***
  - *New Kernel for Smalltalk*
- ***How to compose conflicting class properties?***
  - *e.g. two redefinitions of new*
  - *CLOS like before/after methods*
  - *other?*

Happy Birthday Smalltalk!

Smalltalk was born
on September 1972

Questions? Comments?

Download
http://csl.ensm-douai.fr/MetaclassTalk