
Reflection

A Framework for Language Implementations

Reflection

The ability of a program to examine and control its own implementation.

Computational reflection is good for

- programming environments
- extending a language
- concurrent programming

Disadvantages of Reflection

Reflection is usually hard to understand.

Meta is good. “Meta” is bad.

Reflection can make language hard to implement.

Reflection can make language hard to analyze.

Reflection in Smalltalk

Object

Classes

Method lookup

Blocks, Contexts

Object

- `basicSize` gives number of instance variables of any object
- `instVarAt:` and `instVarAt:put:` lets the inspector access any instance variable
- used by inspector, `storeOn:`, `copy`

Copying

`shallowCopy`

`| newObject |`

`newObject := self class basicNew.`

`1 to: self basicSize do:`

`[:i | newObject`

`instVarAt: i`

`put: (self instVarAt: i)].`

`^newObject`

Classes as objects

Point of reflection

Classes are implementation

Makes system more dynamic

What is a Class?

1) A factory for instances.

- new

2) The category to which an object belongs.

- name
- superclass

Using classes as categories

ByteCharacterEncoder

= anEncoder

^self class == anEncoder class

and: [self decoder = anEncoder
decoder]

What is a Class?

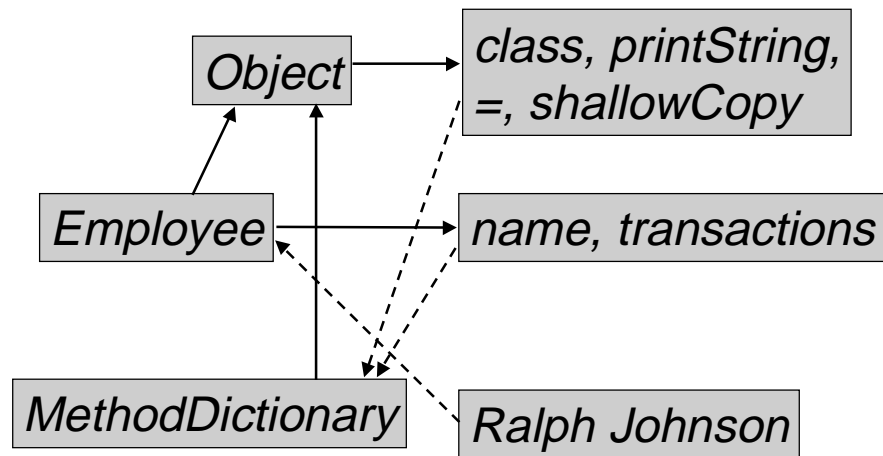
3) The definition (implementation) of an object.

- compiledMethodAt:
- instVarNames
- compile:classified:notifying:

4) A set of instances.

- allInstances

Classes as Implementation



Object-oriented Programming and Design - Copyright 1998 by Ralph E. Johnson

735

"Class" Hierarchy

Behavior ('superclass' 'methodDict'
'format' 'subclasses')

ClassDescription ('instanceVariables'
'organization')

Class ('name' 'classPool'
'sharedPools')

Metaclass ('thisClass')

Object-oriented Programming and Design - Copyright 1998 by Ralph E. Johnson

736

Implementing Variables

Instance variables

slots of an object

Global and class variables

Associations - key/value pairs

Class variables in class pool dictionary

Global variables in "Smalltalk"

Class Variables

Single variable accessible by all instances
of class and subclasses.

Implemented like a global variable.

Subclasses share the same location.

Usually used for constants. Famous
counterexample - the cut buffer.

Class Instance Variables

Instance variable of object representing class.

Not accessible by instances.

Each subclass has its own copy of the variable.

Class Variables

```
ApplicationModel subclass: #Browser
  instanceVariableNames: 'organization
  category className ...'
  classVariableNames: 'CategoryMenu
  ClassMenu ...'
  poolDictionaries: ''
  category: 'Tools-Programming'
```

Class Instance Variables

Browser class

instanceVariableNames: ”

The Metaclass Mystery

Assumption

Every object has a class.

Everything is an object.

Conclusion

A class is an object.

A class has a class.

The Metaclass Mystery

Mystery

What is the class of a class of a class
... ?

Other facts

Object has no superclass

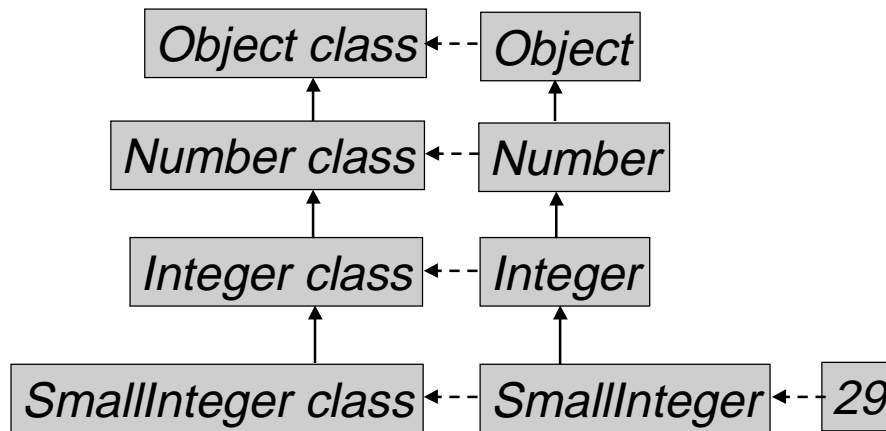
Metaclasses

The class of a class is a metaclass.

Name of "SmallInteger class" is
"SmallInteger class".

Metaclasses form a hierarchy that mirrors
the regular class hierarchy. This allows
subclasses to inherit class methods
from superclasses.

Metaclasses



Object-oriented Programming and Design - Copyright 1998 by Ralph E. Johnson

745

More Metaclass Mystery

If *X* is a *Y*, then *X* is an instance of *Y* or of a subclass of *Y*.

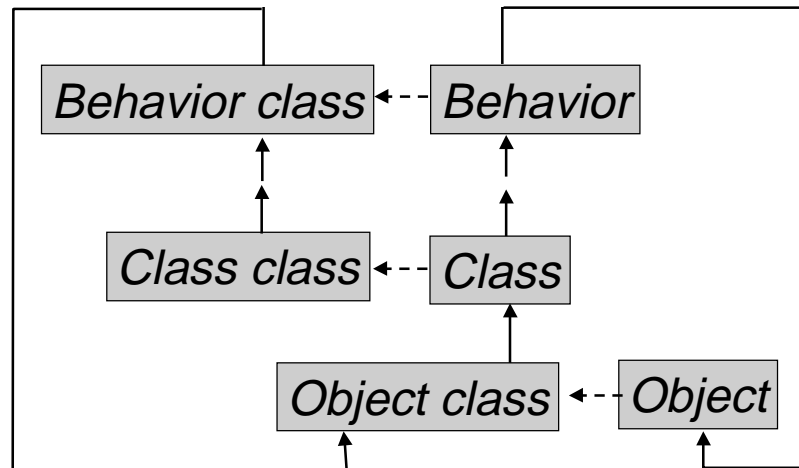
Thus, since *Object* is a class, *Object* is an instance of *Class* or of a subclass of *Class*.

Object is an instance of *Object class*, thus *Object class* is a subclass of *Class*.

Object-oriented Programming and Design - Copyright 1998 by Ralph E. Johnson

746

Metaclasses



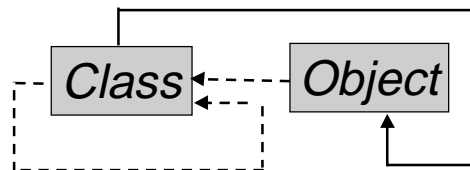
Object-oriented Programming and Design - Copyright 1998 by Ralph E. Johnson

747

Minus Metaclasses?

Metaclasses are not necessary.

Every class could be an instance of Class. Each class would then have the same behavior, so no special instance creation methods.

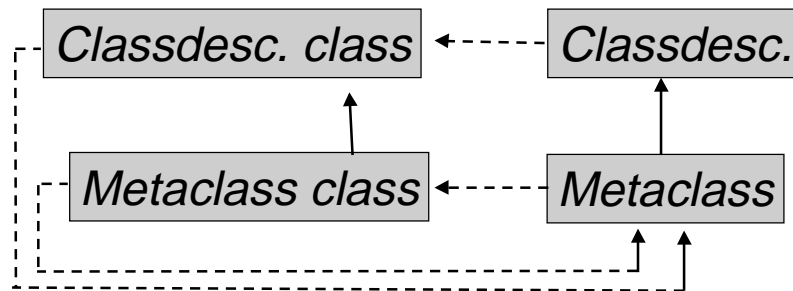


Object-oriented Programming and Design - Copyright 1998 by Ralph E. Johnson

748

Metaclass Madness

Every metaclass is an instance of Metaclass.



Object-oriented Programming and Design - Copyright 1998 by Ralph E. Johnson

749

Creating New Classes

Class creation template in the browser is just a regular Smalltalk message that is sent to a class.

Classes either have no indexible part, an indexible part that contains uninterpreted bytes, or an indexible part that contains pointers to objects.

Object-oriented Programming and Design - Copyright 1998 by Ralph E. Johnson

750

Creating New Classes

Methods defined in Class for creating subclasses:

subclass: t instanceVariableNames: f
classVariableNames: d
poolDictionaries: s category: cat
variableByteSubclass: t ...
variableSubclass: t ...

Builder Pattern

Object for creating complex object.

Builder supports set of operations for building pieces of product.

Builder hides how pieces are put together.

Class Builder

```
subclass: t instanceVariableNames: f  
classVariableNames: d poolDictionaries: s  
category: cat
```

"This is the standard initialization message for creating a new class as a subclass of an existing class (the receiver)."

Class Builder

```
^self classBuilder  
  superclass: self;  
  environment: self environment;  
  className: t; instVarString: f;  
  classVarString: d;  
  poolString: s; category: cat;  
  beFixed; reviseSystem
```

ClassDescription compiles methods

**compile: code classified: heading notifying:
requestor**

| selector |

selector := self compile: code

notifying: requestor

ifFail: [^nil].

(continued)

(methodDict at: selector)

putSource: code asString

class: self

category: heading

inFile: 2.

(continued)

self organization classify: selector
under: heading.

^selector

The Compiler

**compile: code notifying: requestor ifFail:
failBlock**

...

```
self compilerClass new
  compile: code in: self
  notifying: requestor
  ifFail: failBlock.
```

...

Patterns in Compiler

Strategy - (why isn't compiler just a method on String?)

Facade - hides parser, parse trees

Interpreter - parse trees

Lightweight Classes

Problem: changing method of an object changes method for all objects in its class

Solution: give each object its own class

Light-weight Classes

Behavior is light-weight: has only superclass, subclasses, method dictionary, and format.

Light-weight class will not be available in browser.

Have to provide compiling yourself.

Light-weight Classes

