

Metaclasses in 7 Steps

Classes are objects too...
Classes are instances of other classes
...
One model applied twice



Metaclasses in 7 points

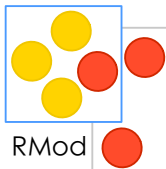
1. Every object is an instance of a class
2. Every class eventually inherits from Object
3. Every class is an instance of a metaclass
4. The metaclass hierarchy parallels the class hierarchy
5. Every metaclass inherits from Class and Behavior
6. Every metaclass is an instance of Metaclass
7. The metaclass of Metaclass is an instance of Metaclass

Adapted from Goldberg & Robson, *Smalltalk-80 — The Language*

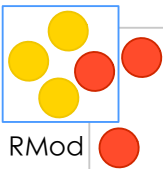
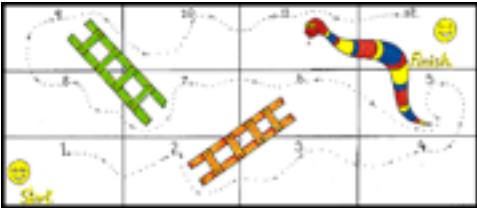


Metaclasses in 7 points

1. **Every object is an instance of a class**
2. Every class eventually inherits from Object
3. Every class is an instance of a metaclass
4. The metaclass hierarchy parallels the class hierarchy
5. Every metaclass inherits from Class and Behavior
6. Every metaclass is an instance of Metaclass
7. The metaclass of Metaclass is an instance of Metaclass

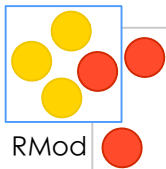


I. Every object is an instance of a class



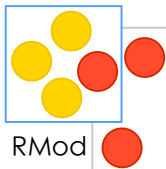
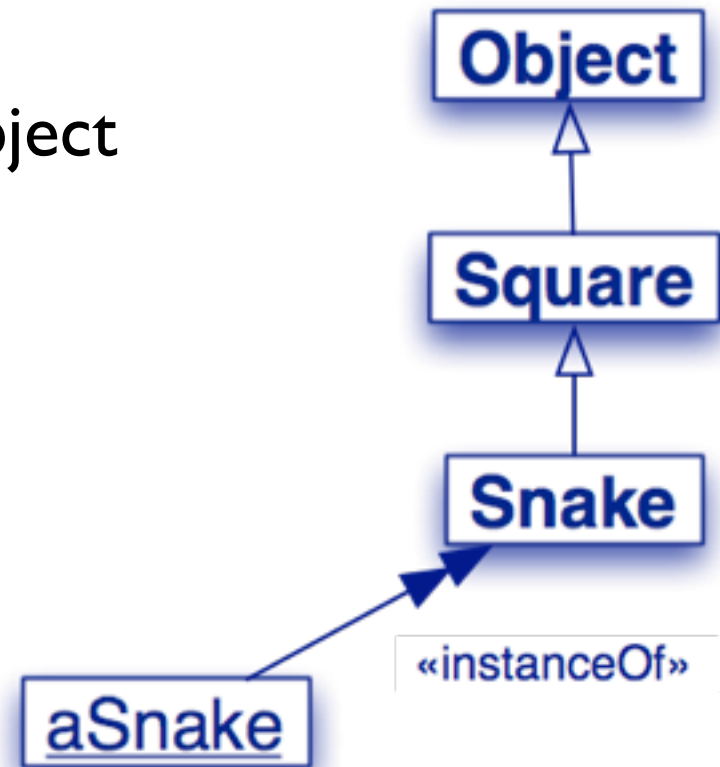
Metaclasses in 7 points

1. Every object is an instance of a class
2. **Every class eventually inherits from Object**
3. Every class is an instance of a metaclass
4. The metaclass hierarchy parallels the class hierarchy
5. Every metaclass inherits from Class and Behavior
6. Every metaclass is an instance of Metaclass



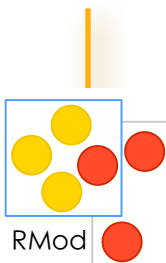
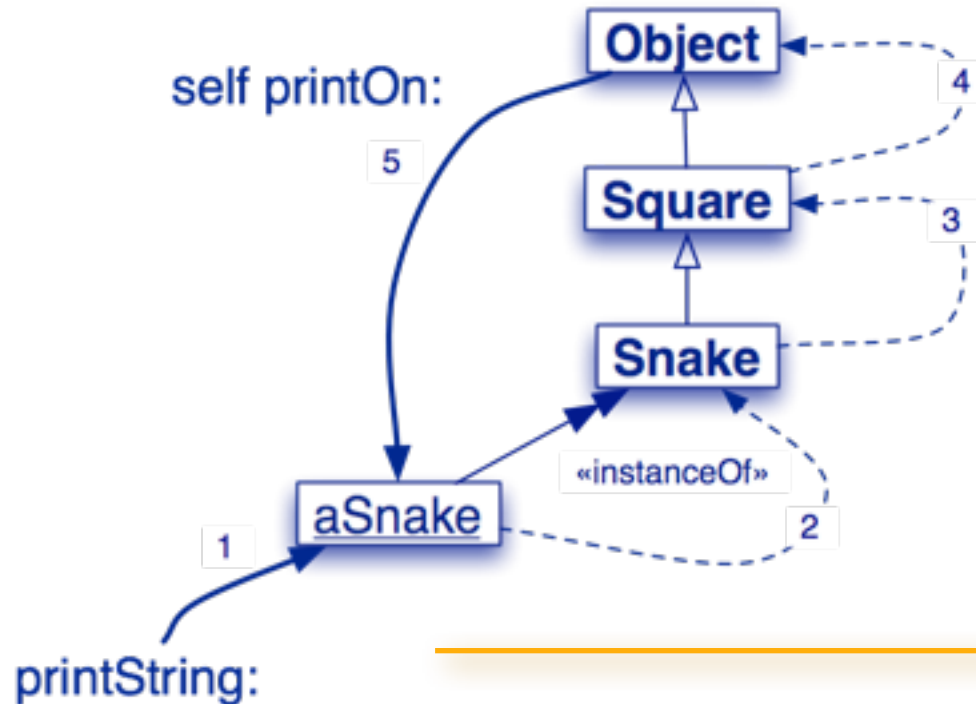
2. Every class inherits from Object

Every object is-an Object
The class of every object
ultimately inherits from Object



The Meaning of is-a

When an object receives a message, the method is looked up in the method dictionary of its class, and, if necessary, its superclasses, up to Object



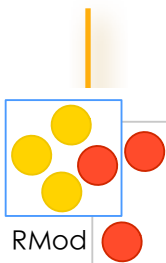
Responsibilities of Object

Object

represents the common object behavior

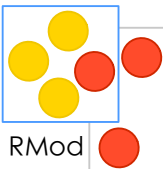
error-handling, halting ...

all classes should inherit ultimately from Object



Metaclasses in 7 points

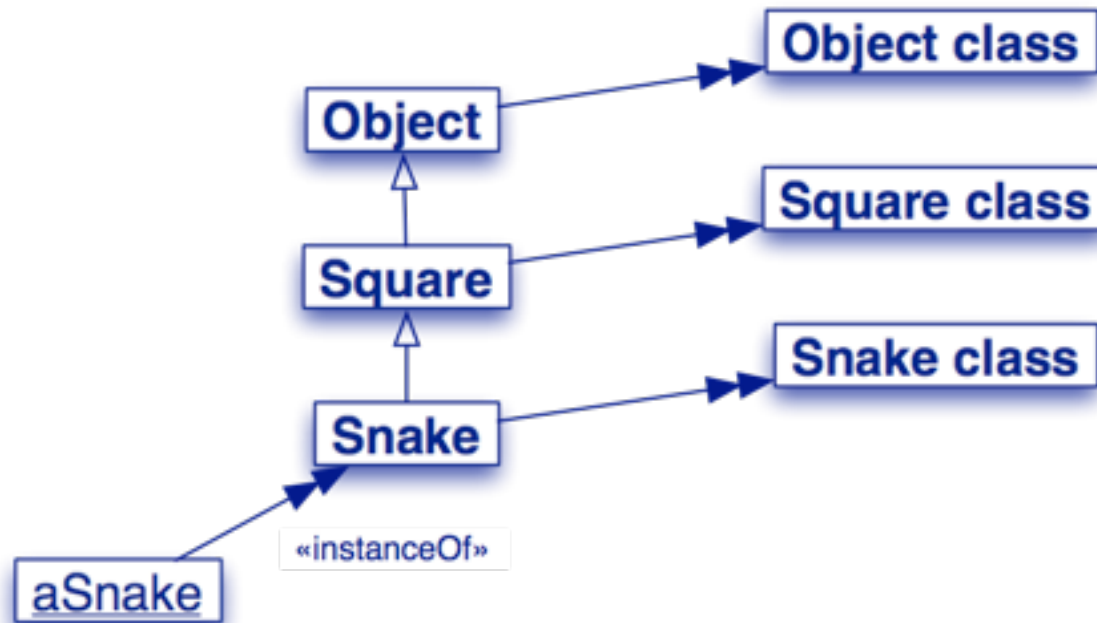
1. Every object is an instance of a class
2. Every class eventually inherits from Object
3. **Every class is an instance of a metaclass**
4. The metaclass hierarchy parallels the class hierarchy
5. Every metaclass inherits from Class and Behavior
6. Every metaclass is an instance of Metaclass
7. The metaclass of Metaclass is an instance of



3. Every class is an instance of a metaclass

Classes are objects too!

Every class X is the unique instance of its metaclass, called X class

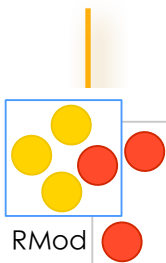


Metaclasses are implicit

There are no explicit metaclasses

Metaclasses are created implicitly when classes are created

No sharing of metaclasses (unique metaclass per class)



Metaclasses by Example

Square allSubclasses

Snake allSubclasses

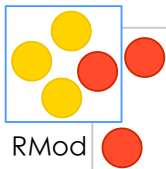
Snake allInstances

Snake instVarNames

Snake back: 5

Snake selectors

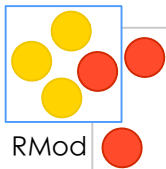
Snake canUnderstand: #new



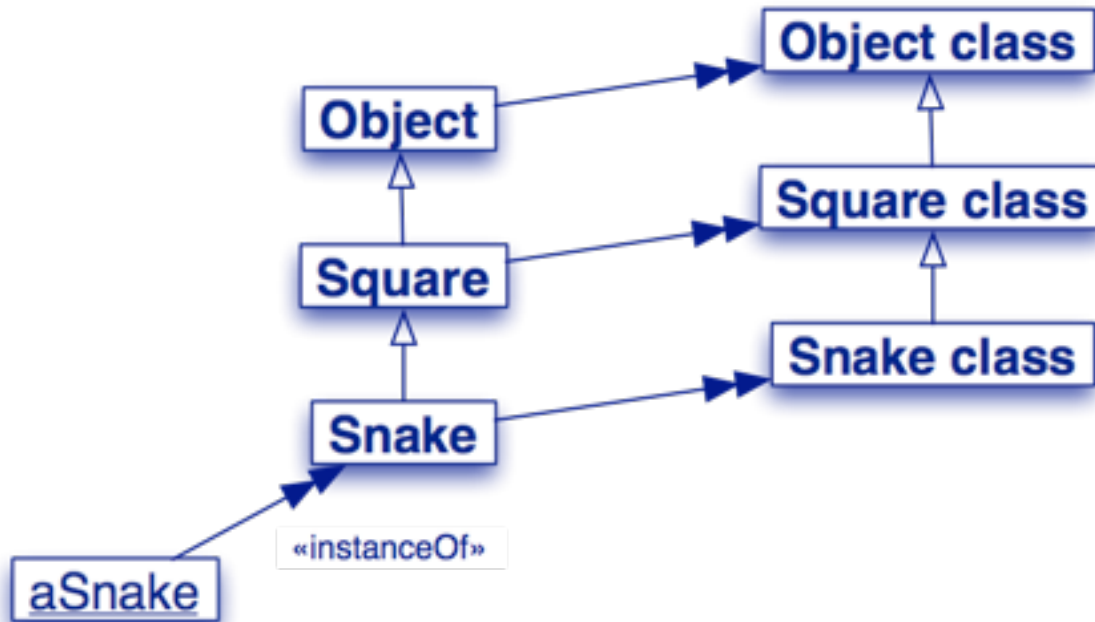
RMod

Metaclasses in 7 points

1. Every object is an instance of a class
2. Every class eventually inherits from Object
3. Every class is an instance of a metaclass
4. **The metaclass hierarchy parallels the class hierarchy**
5. Every metaclass inherits from Class and Behavior
6. Every metaclass is an instance of Metaclass
7. The metaclass of Metaclass is an instance of



4. The metaclass hierarchy parallels the



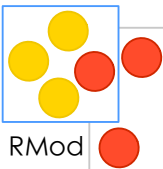
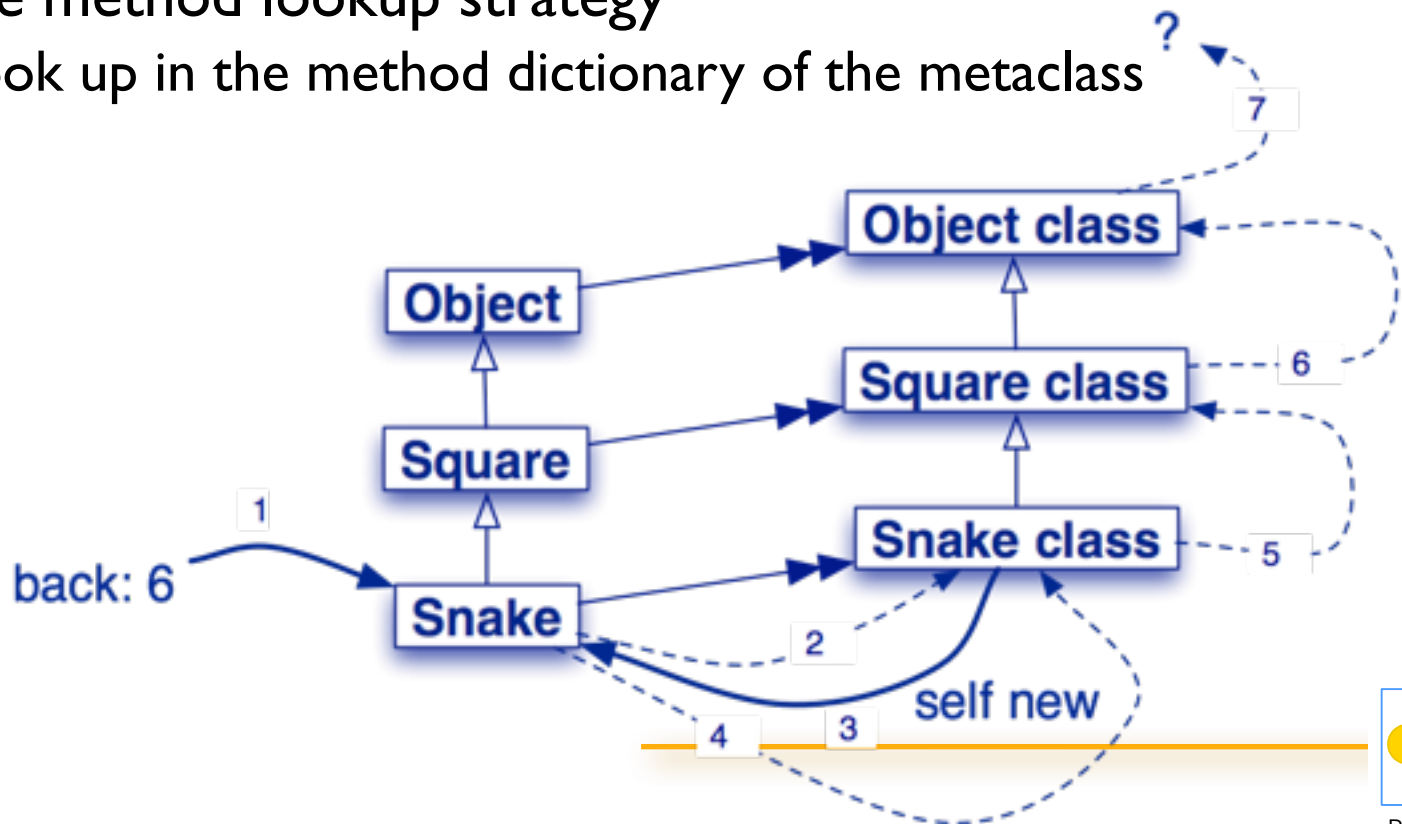
Uniformity between Classes and Objects

Classes are objects too, so ...

Everything that holds for objects holds for classes as well

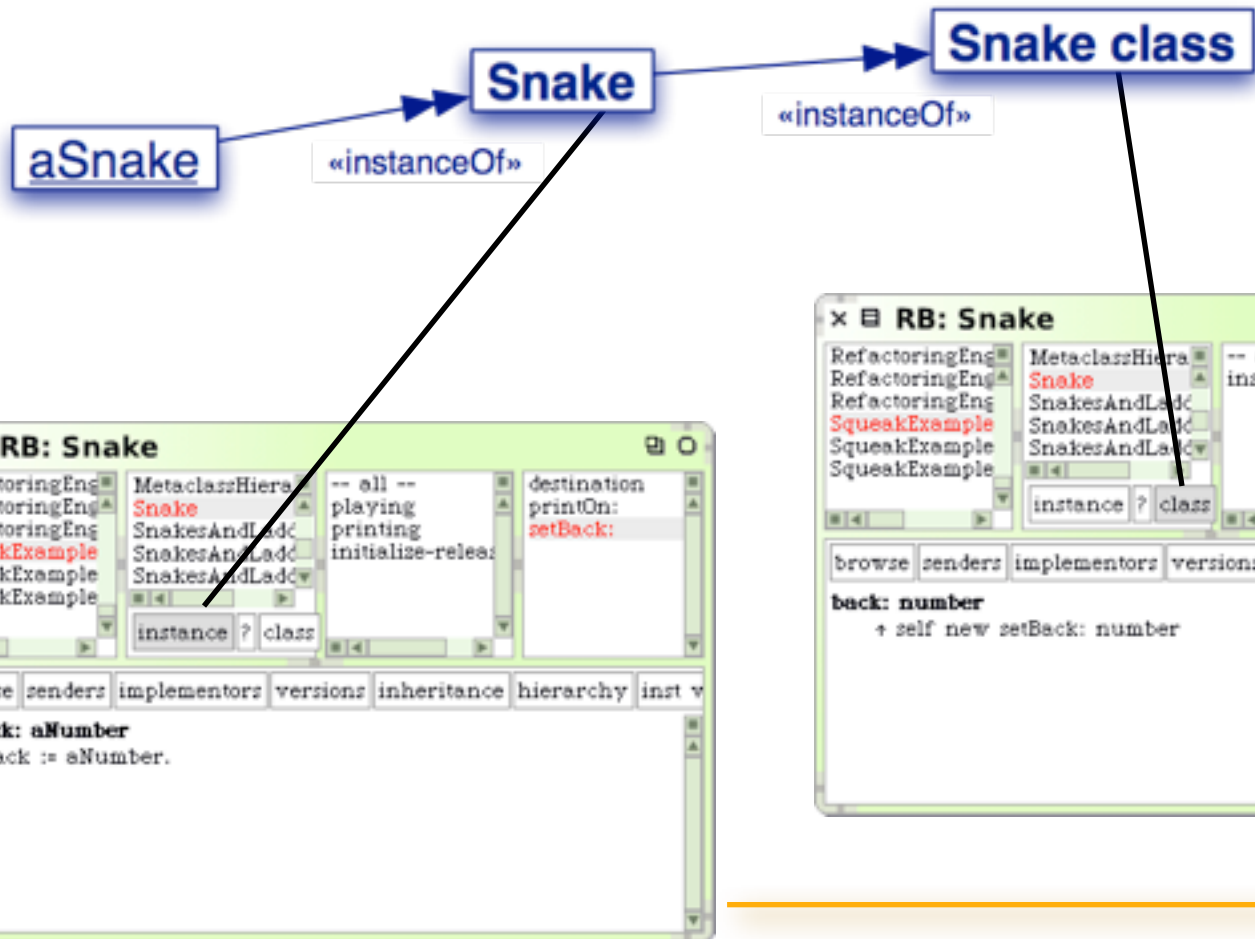
Same method lookup strategy

Look up in the method dictionary of the metaclass



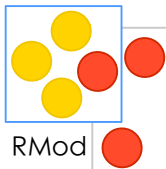
RMod

About the Buttons



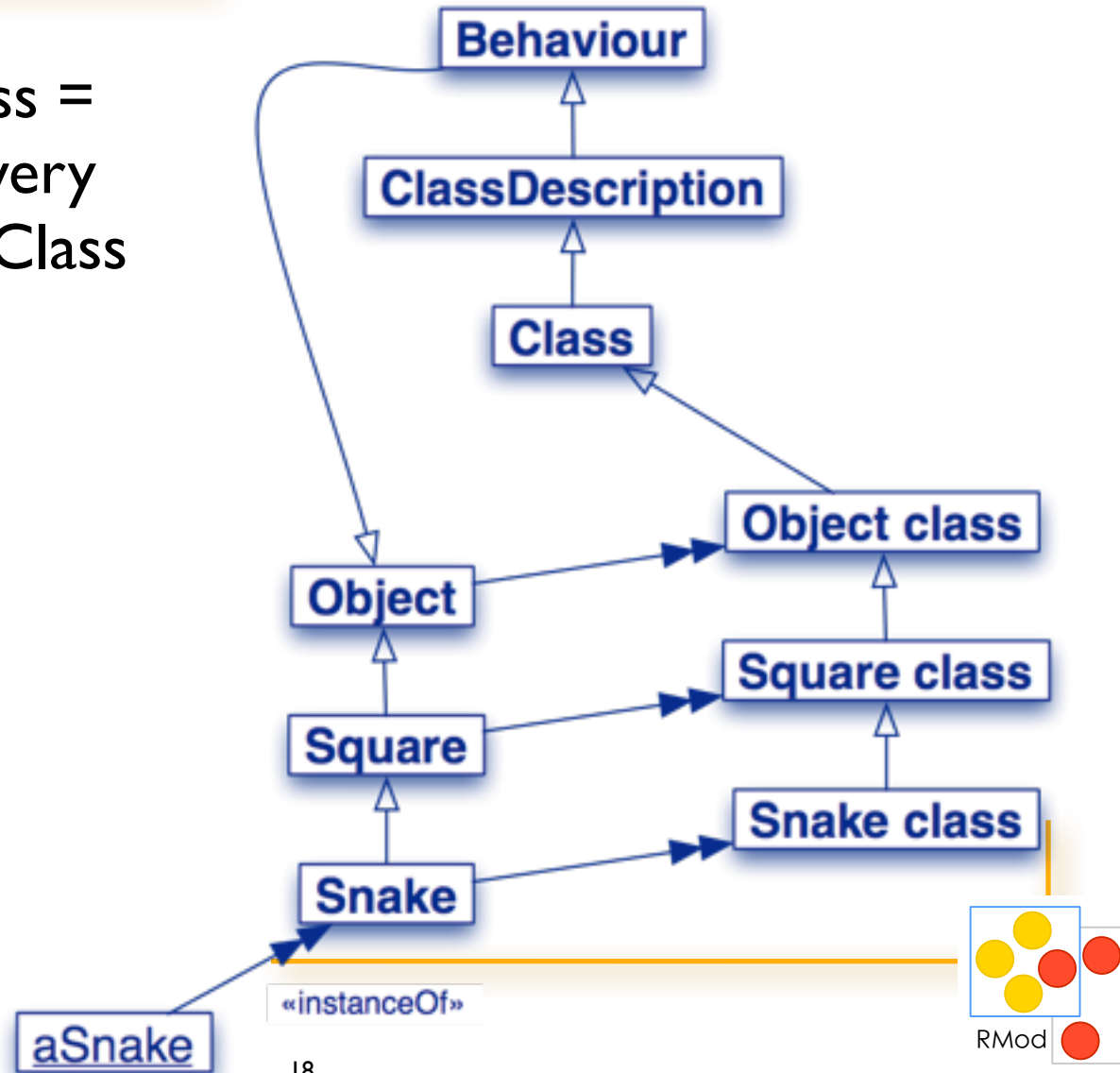
Metaclasses in 7 points

1. Every object is an instance of a class
2. Every class eventually inherits from Object
3. Every class is an instance of a metaclass
4. The metaclass hierarchy parallels the class hierarchy
5. **Every metaclass inherits from Class and Behavior**
6. Every metaclass is an instance of Metaclass
7. The metaclass of Metaclass is an instance of

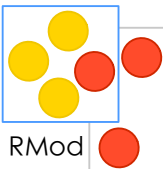
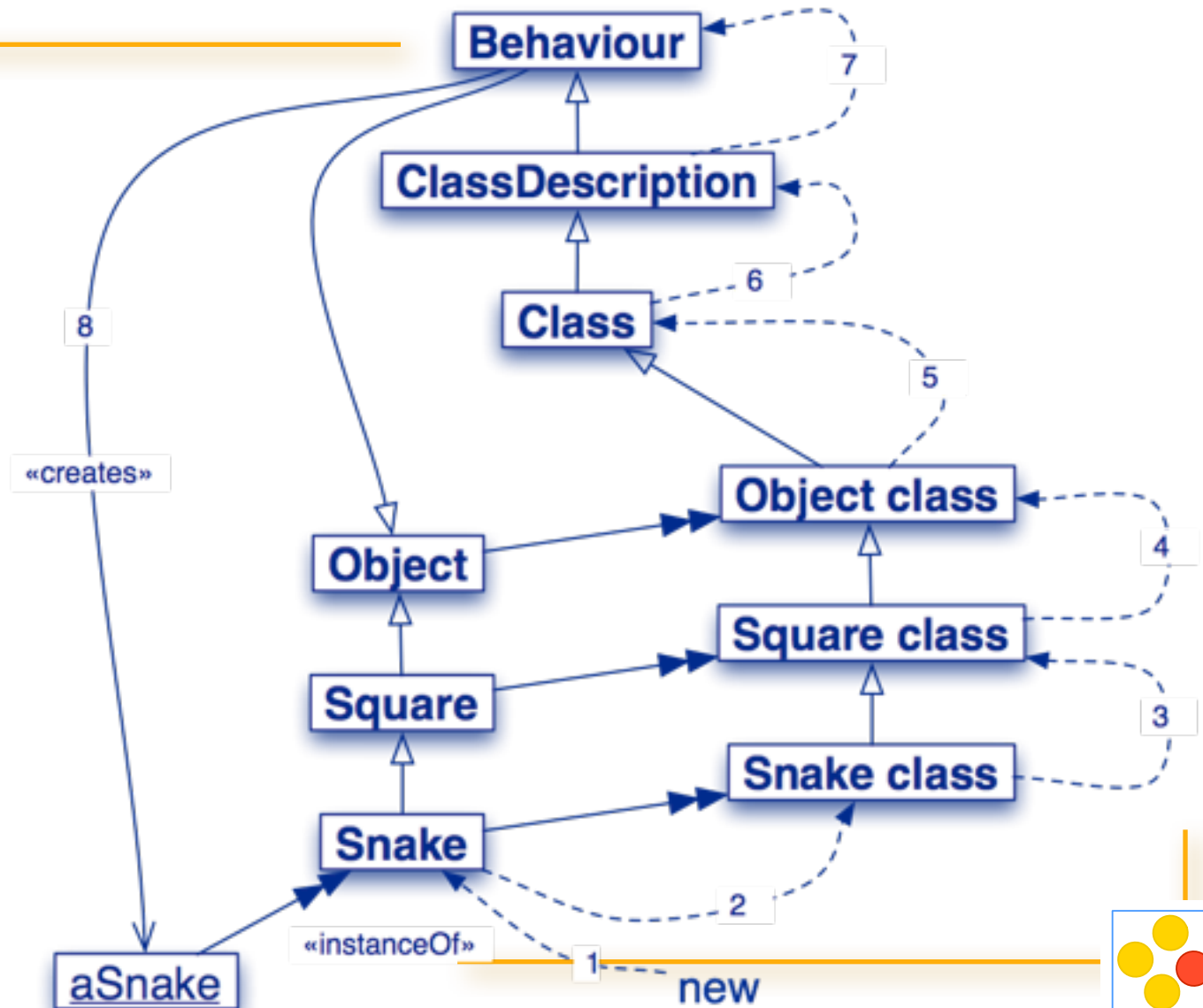


5. Every metaclass inherits from Class and

Every class is-a Class =
The metaclass of every
class inherits from Class



Where is new defined?



Responsibilities of Behavior

Behavior

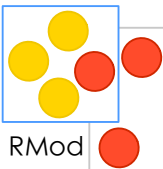
Minimum state necessary for objects that have instances.
Basic interface to the compiler.

State:

class hierarchy link, method dictionary, description of instances (representation and number)

Methods:

creating a method dictionary, compiling method
instance creation (new, basicNew, new:, basicNew:)
class hierarchy manipulation (superclass:, addSubclass:)
accessing (selectors, allSelectors, compiledMethodAt:)
accessing instances and variables (allInstances, instVarNames)



Responsibilities of ClassDescription

ClassDescription

adds a number of facilities to basic Behavior:

- named instance variables

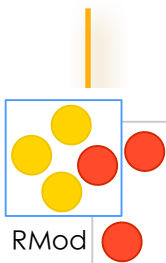
- category organization for methods

- the notion of a name (abstract)

- maintenance of Change sets and logging changes

- most of the mechanisms needed for fileOut

ClassDescription is an abstract class: its facilities are intended for inheritance by the two subclasses, Class and Metaclass.



Responsibilities of Class

Class

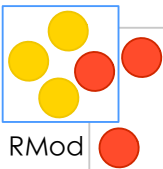
represents the common behavior of all classes

name, compilation, method storing, instance variables ...

representation for classVariable names and shared pool variables (`addClassVarName:`, `addSharedPool:`, `initialize`)

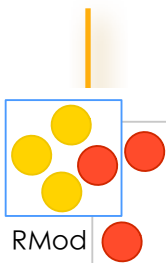
Class inherits from Object because Class is an Object

Class knows how to create instances, so all metaclasses should inherit ultimately from *Class*

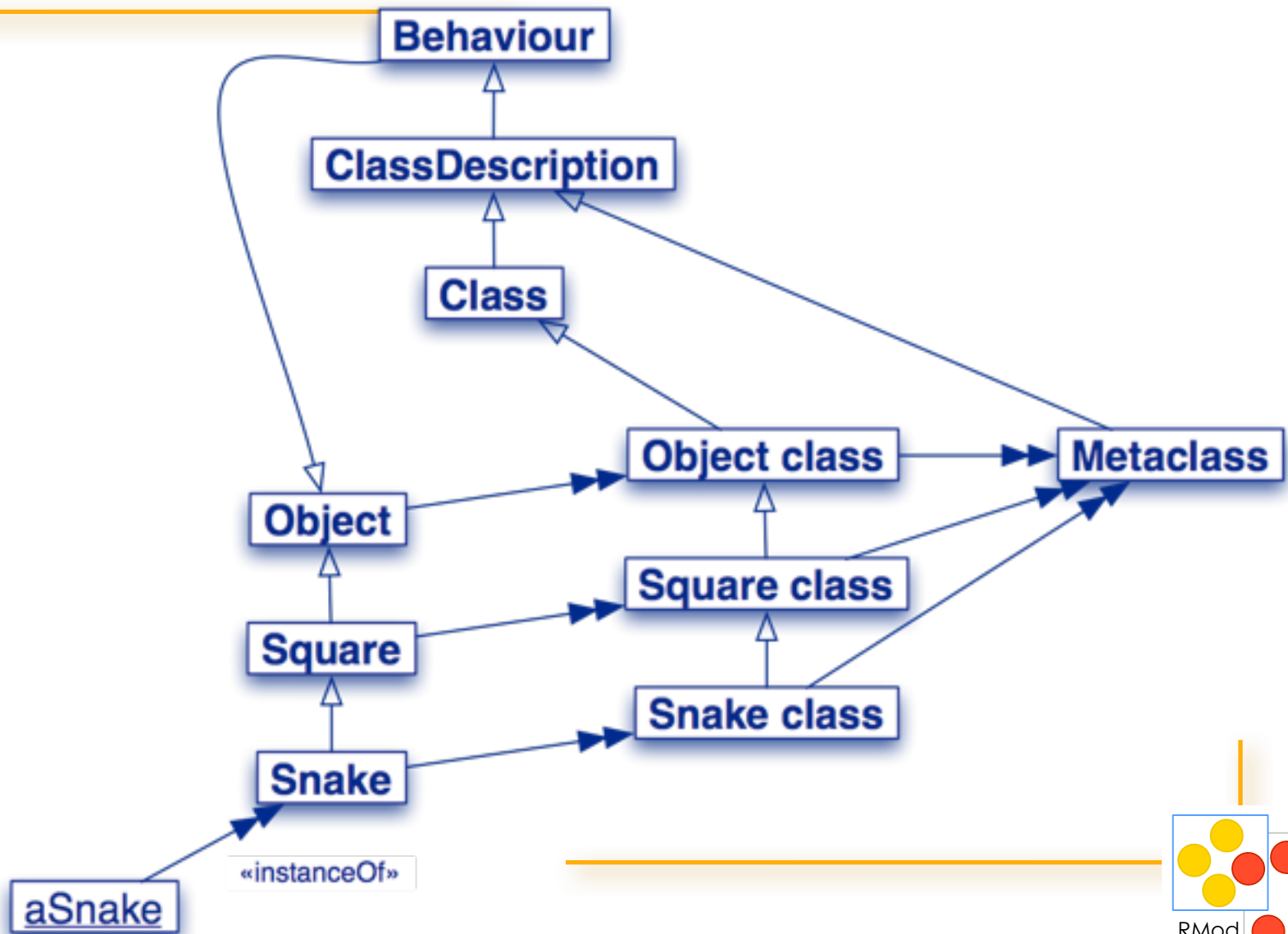


Metaclasses in 7 points

1. Every object is an instance of a class
2. Every class eventually inherits from Object
3. Every class is an instance of a metaclass
4. The metaclass hierarchy parallels the class hierarchy
5. Every metaclass inherits from Class and Behavior
6. **Every metaclass is an instance of Metaclass**
7. The metaclass of Metaclass is an instance of Metaclass



6. Every metaclass is an instance of Metaclass



Metaclass Responsibilities

Metaclass

Represents common metaclass Behavior

- instance creation (subclassOf:)

- creating initialized instances of the metaclass's sole instance

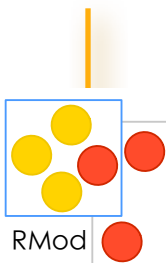
- initialization of class variables

- metaclass instance protocol

- (name:inEnvironment:subclassOf:....)

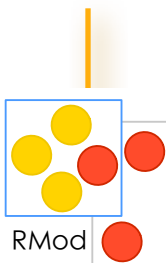
- method compilation (different semantics can be introduced)

- class information (inheritance link, instance variable, ...)

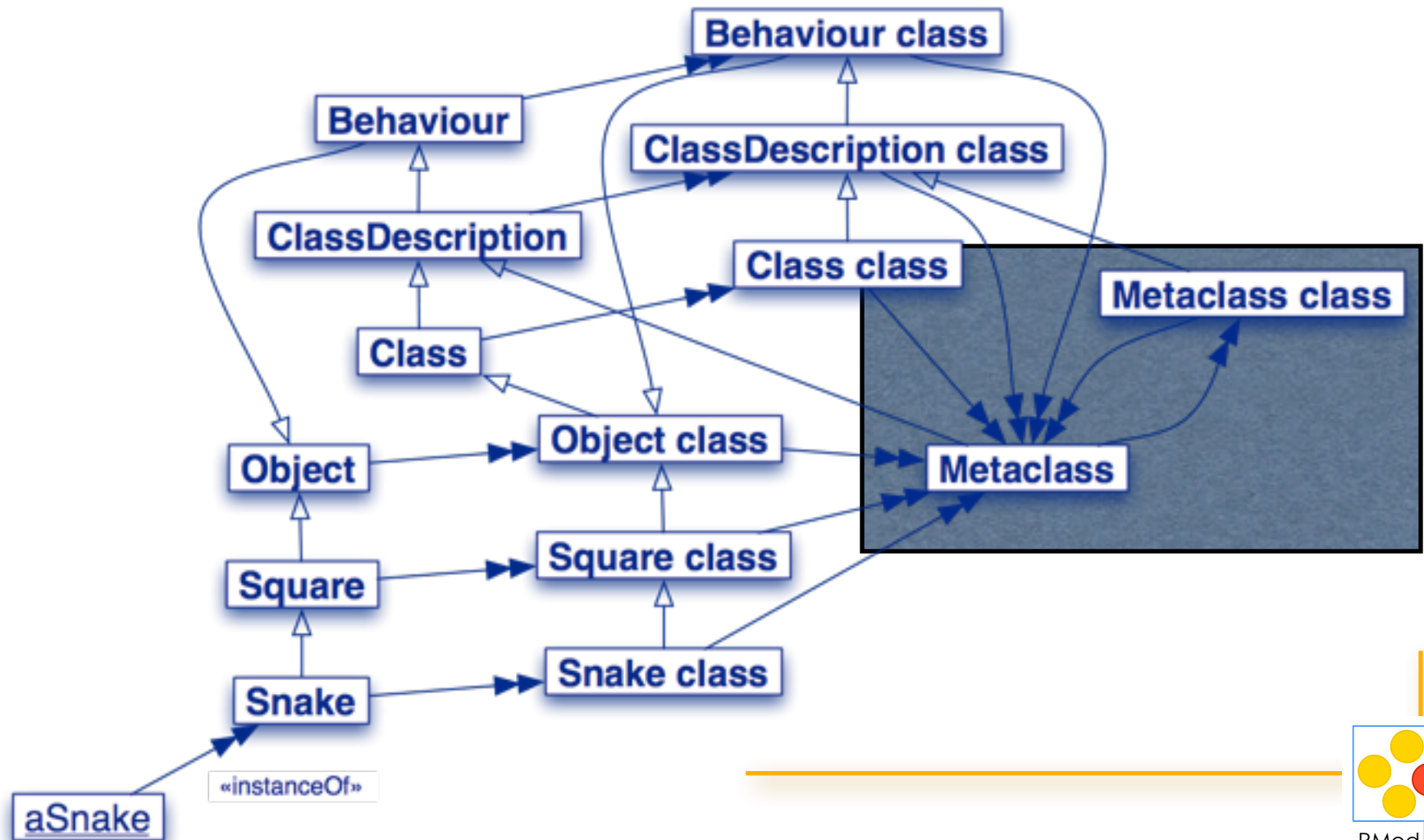


Metaclasses in 7 points

1. Every object is an instance of a class
2. Every class eventually inherits from Object
3. Every class is an instance of a metaclass
4. The metaclass hierarchy parallels the class hierarchy
5. Every metaclass inherits from Class and Behavior
6. Every metaclass is an instance of Metaclass
7. **The metaclass of Metaclass is an instance of Metaclass**

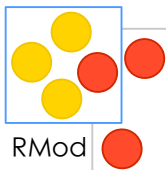


7. The Metaclass Of of Metaclass



Navigating the metaclass hierarchy

```
MetaclassHierarchyTest>>testHierarchy
"The class hierarchy"
self assert: Snake superclass = Square.
self assert: Square superclass = Object.
self assert: Object superclass superclass = nil. "skip ProtoObject"
"The parallel metaclass hierarchy"
self assert: Snake class name = 'Snake class'.
self assert: Snake class superclass = Square class.
self assert: Square class superclass = Object class.
self assert: Object class superclass superclass = Class.
self assert: Class superclass = ClassDescription.
self assert: ClassDescription superclass = Behavior.
self assert: Behavior superclass = Object.
"The Metaclass hierarchy"
self assert: Snake class class = Metaclass.
self assert: Square class class = Metaclass.
self assert: Object class class = Metaclass.
self assert: Class class class = Metaclass.
self assert: ClassDescription class class = Metaclass.
self assert: Behavior class class = Metaclass.
self assert: Metaclass superclass = ClassDescription.
"The fixpoint"
self assert: Metaclass class class = Metaclass.
```



Summary



Just one model applied systematically.

The key: messages sent to an object are looked in its class then in the superclass.