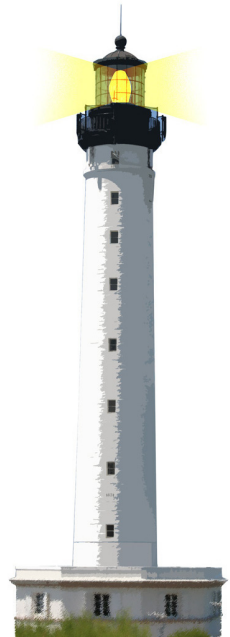
Understanding Metaclasses

Damien Cassou, Stéphane Ducasse and Luc Fabresse

W7S03

Pharo

http://www.pharo.org

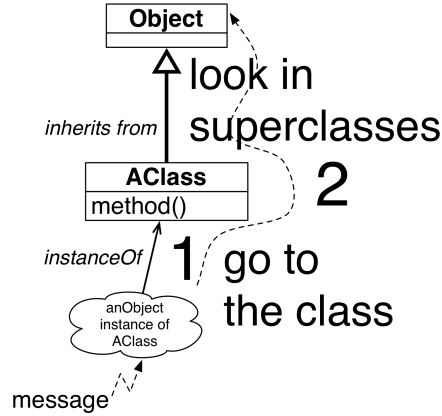


What You Will Learn

- Not needed to program in Pharo :)
- But intellectually exciting
  - Where `new` is defined?
  - What is the class of a metaclass?
  - Uniformity of the instance relationship

The Key: Only One Method Lookup

When a message is sent to an object, a method is searched starting from the class of the object and following the inheritance chain



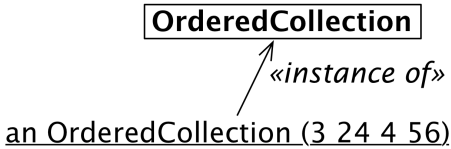
Metaclasses in 7 Points

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 up to Behavior
6. Every metaclass is an instance of Metaclass
7. The metaclass of Metaclass is an instance of Metaclass

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 up to Behavior
6. Every metaclass is an instance of Metaclass
7. The metaclass of Metaclass is an instance of Metaclass

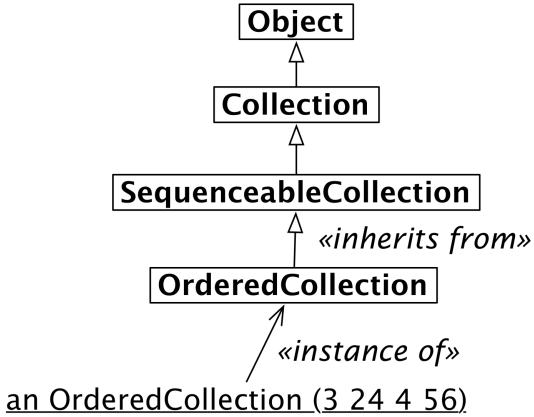
Every Object is an Instance of a Class



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 up to Behavior
6. Every metaclass is an instance of Metaclass
7. The metaclass of Metaclass is an instance of Metaclass

Every Class Eventually Inherits From Object



Responsibility of Object

Class Object represents the common object behavior

- error handling, halting, announcements
- all classes eventually inherit from Object

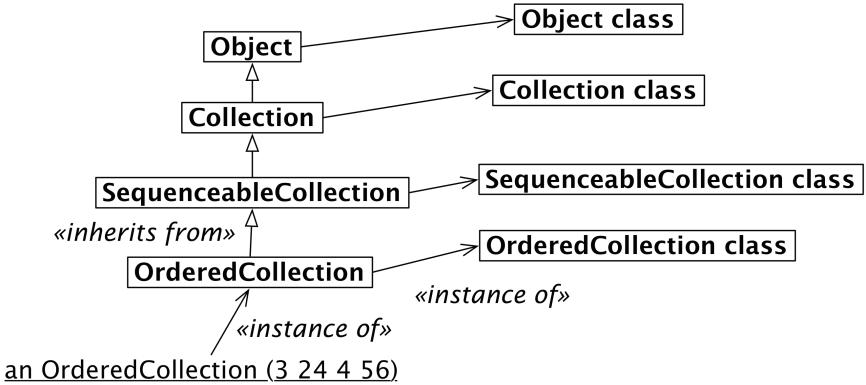
Metaclasses in 7 Points

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 up to Behavior
6. Every metaclass is an instance of Metaclass
7. The metaclass of Metaclass is an instance of Metaclass

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



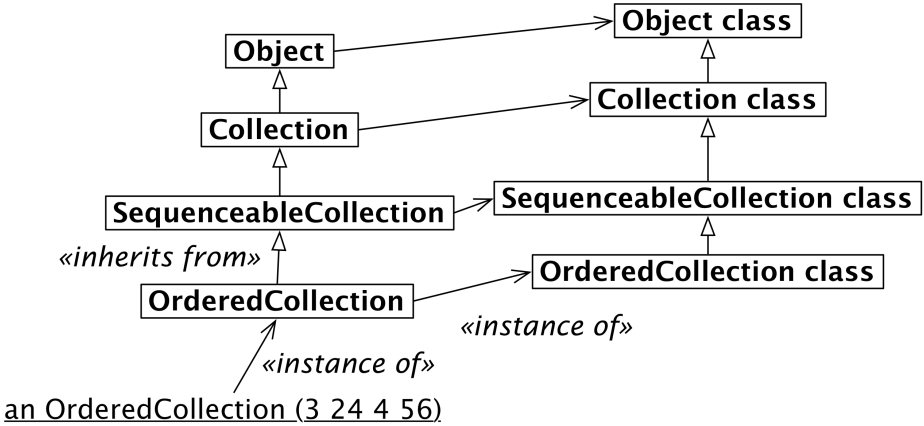
Metaclass are Implicit

Metaclasses are automatically created when creating a class

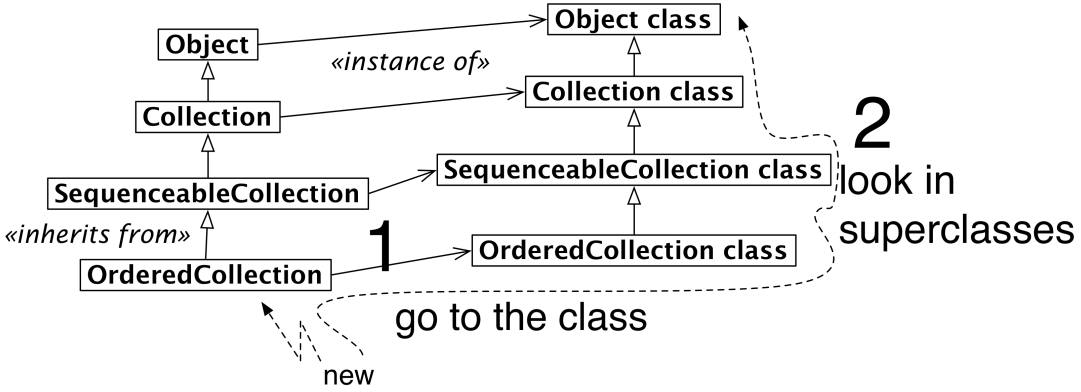
Metaclasses in 7 Points

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 up to Behavior
6. Every metaclass is an instance of Metaclass
7. The metaclass of Metaclass is an instance of Metaclass

Metaclass Hierarchy Parallels Class Hierarchy

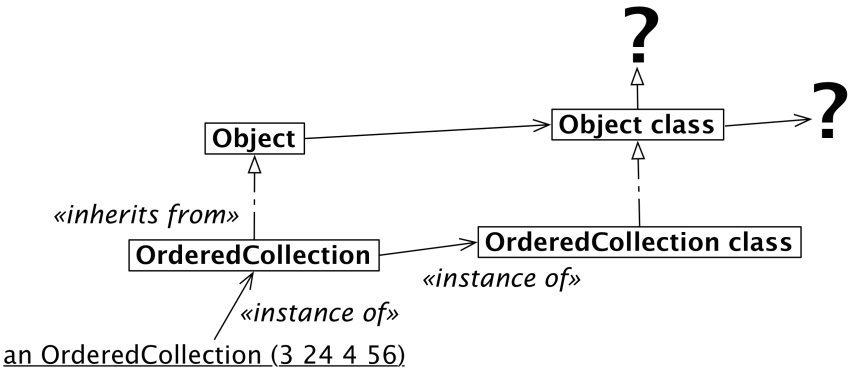


Sending a Message to a Class



Questions

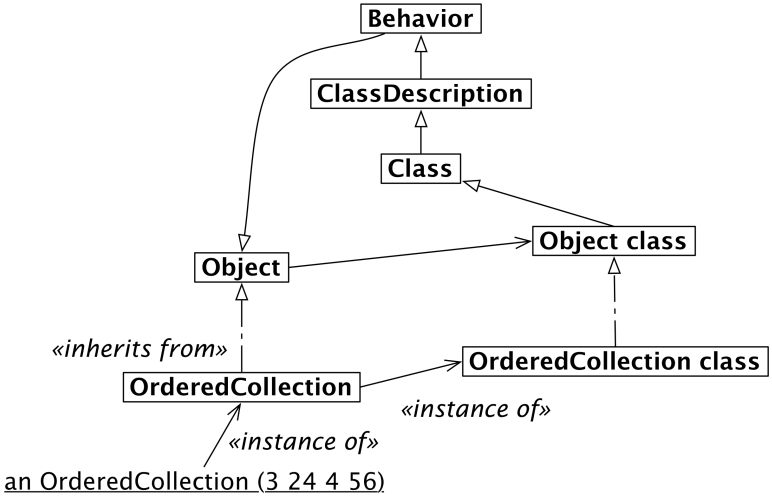
- What is the class of a metaclass?
- What is the superclass of Object class?
- What about sending a message to a metaclass?



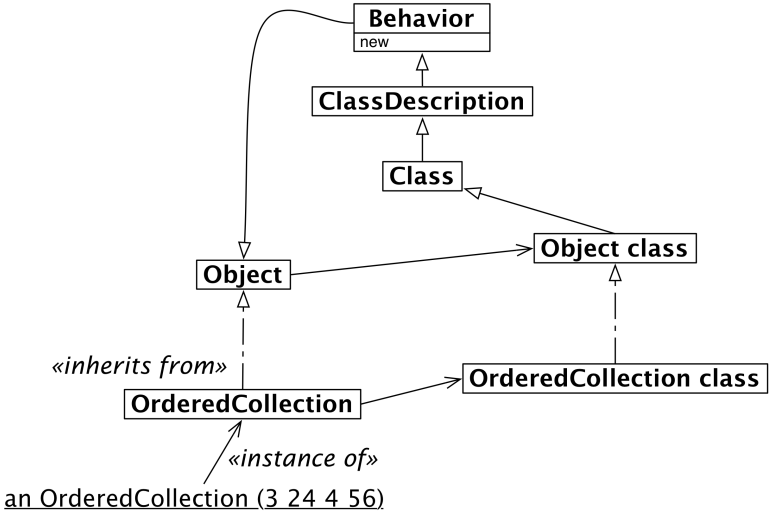
Metaclasses in 7 Points

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 **up to** Behavior
6. Every metaclass is an instance of Metaclass
7. The metaclass of Metaclass is an instance of Metaclass

Every Metaclass Inherits from Class up to Behavior



Where new is Defined?



Responsibilities of Behavior

Behavior

- Minimum state for objects that **have instances**
- State:
  - superclass link, method dictionary, description of instances (representation and number)
- Methods:
  - method dictionary, compiling method
  - instance creation (new, basicNew, new:, basicNew:)
  - class hierarchy manipulation (superclass:, addSubclass:)
  - accessing (selectors, allSelectors, compiledMethodAt:, allInstances, instVarNames)

Responsibilities of ClassDescription

ClassDescription

- Abstract class superclass of Class and Metaclass
- Adds a number of facilities to 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

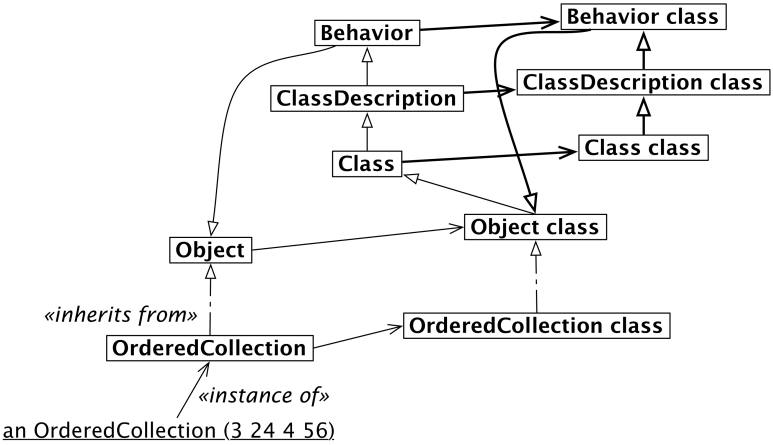
Responsibilities of Class

Class

- represents the common behavior of all classes
  - name, compilation, method storing, instance variables ...
- representation for classVariable names
  (addClassVarName:, initialize)

Classes are Instances of Metaclasses

Metaclass and class inheritances are parallel

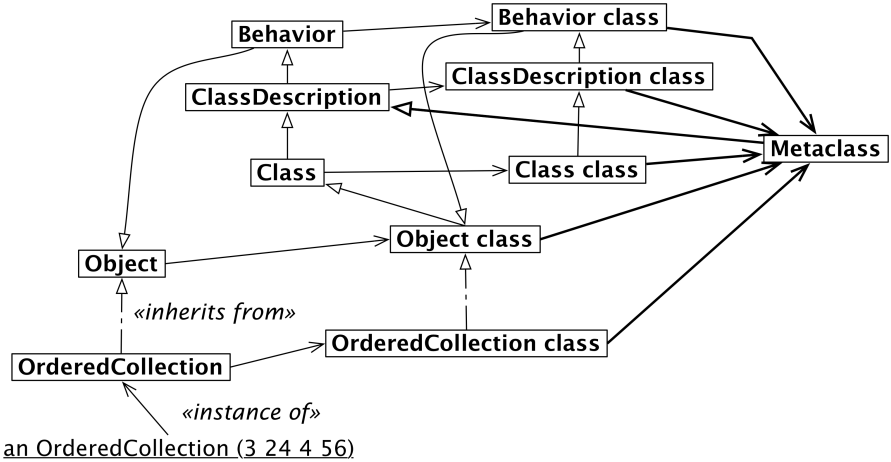


Metaclasses in 7 Points

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 up to Behavior
6. **Every metaclass is an instance of** Metaclass
7. The metaclass of Metaclass is an instance of Metaclass

Every Metaclass is an Instance of Metaclass

Metaclass inherits from ClassDescription



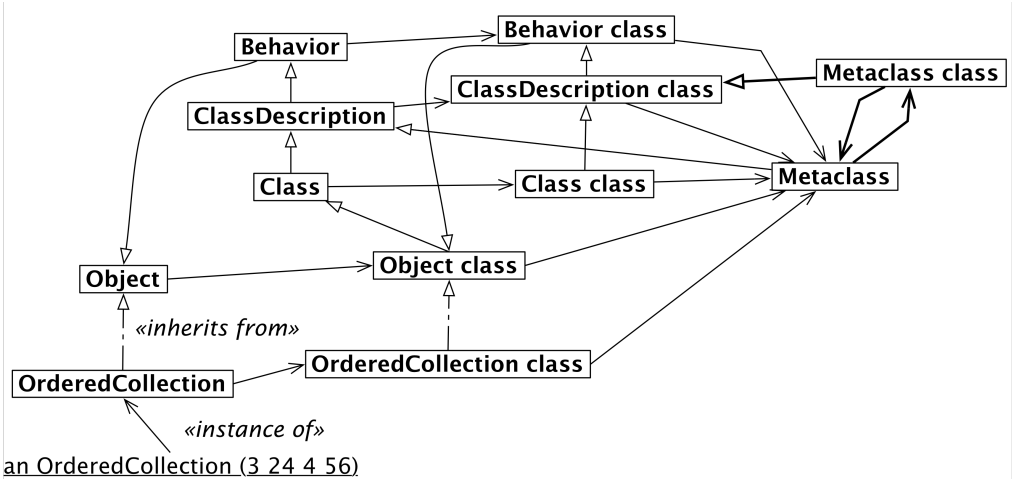
Metaclass Responsibilities

Metaclass is responsible for creating and initializing a metaclass's sole instance (a Class)

Metaclasses in 7 Points

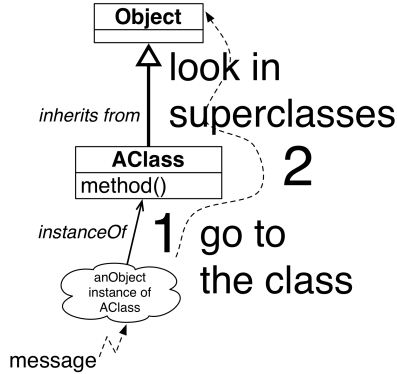
1. Every object is an instance of a class
2. Every class eventually inherits from Object
3. Every class is an instance of a metaclass
4. The metaclass hierarchy parallels the class hierarchy
5. Every metaclass inherits from Class up to Behavior
6. Every metaclass is an instance of Metaclass
7. **The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

The metaclass of Metaclass is an Instance of Metaclass

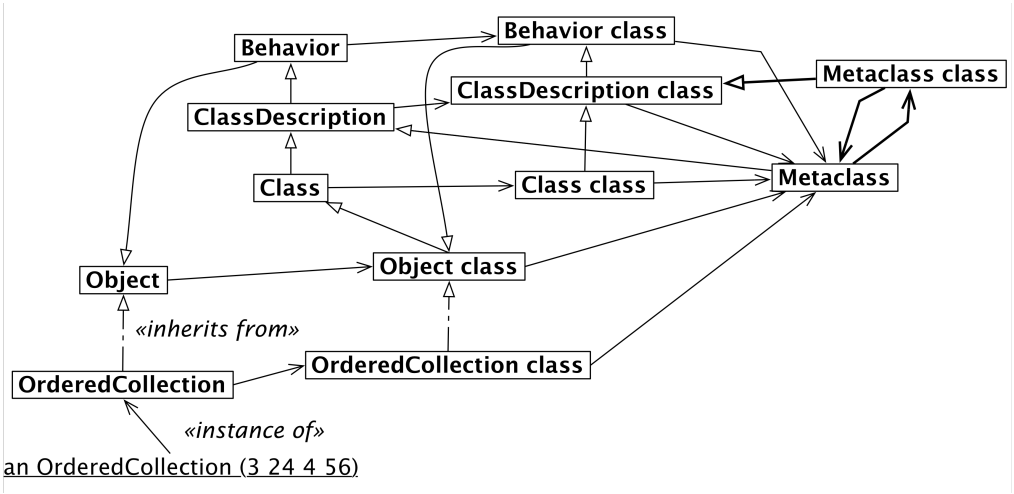


Observations

- When programming we do not really care about that!
- Now the full graph is consistent
  - Any class can receive a message
  - Only one message lookup



A Consistent World



What You Should Know

- Classes are objects and can receive messages
- The process is **exactly** the same as for any other objects

A course by

and

in collaboration with