Inheritance and Lookup

1: Inheritance

Stéphane Ducasse and Damien Cassou

http://stephane.ducasse.free.fr/ stephane.ducasse@inria.fr damien.cassou@inria.fr

Goal

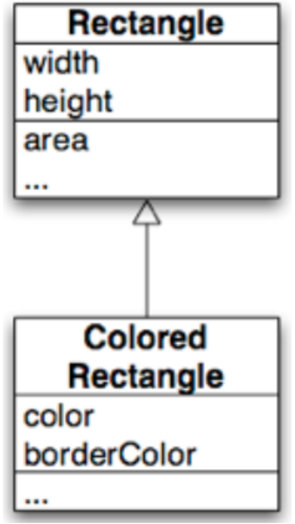
- what is inheritance?
- when to use it?

Inheritance Reminder

- do not want to rewrite everything!
- often we want small adaptations
- we would like to reuse and extend existing behavior

- Solution: class inheritance

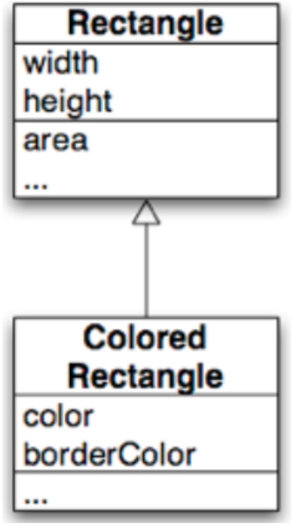
- Each class refines the definition of its superclasses



Inheritance Reminder

New classes

- can add state and behavior:
  - `color`, `borderColor`, ...
- can specialize superclass behavior
- can use superclass behavior and state
- can redefine superclass behavior



Single inheritance

- Static for the instance variables
  - At class creation time the instance variables are collected from the superclasses and the class. No repetition of instance variables.
- Dynamic for the methods
  - Late binding (all virtual) methods are looked up at runtime depending on the dynamic type of the receiver.

Root of Inheritance

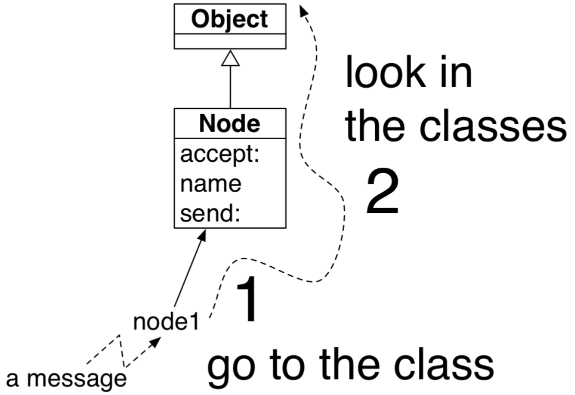
- `Object` is the root of most classes
- `ProtoObject` (the superclass of `Object`) is for special purposes...
  - ...but we will ignore it as it is not important

Inheritance of Instance Variables

- Inheritance of instance variables is made at class definition time
  - The instance variables of a new class are computed based on its own instance variables and the ones of its superclass
  - This happens at class definition time

Inheritance of Behavior and the Lookup

- Inheritance of behavior is dynamic and done at runtime
- The *method* corresponding to the *message* is *looked up*
  - starting from the class of the receiver
  - if not found there, the **lookup** follows the inheritance chain



What you should Know

- inheritance of instance variables is made at class-definition time;
- inheritance of behavior is dynamic