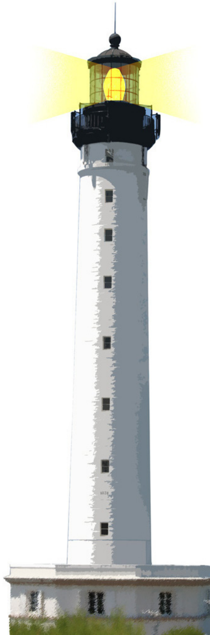
Advanced Object-Oriented Design

Inheritance Basics

S. Ducasse and G. Polito

http://www.pharo.org



Goal

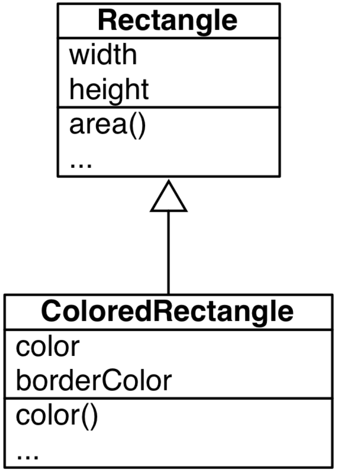
- What is inheritance?
- When to use it?
- BTW, Pharo has the same inheritance as Java

Inheritance

- It is a reuse mechanism
  - We do not reimplement the code of the superclasses
  - We extend it or customize it
- It is based on the expression of a delta
  - Only specify the differences to the superclasses

The basics

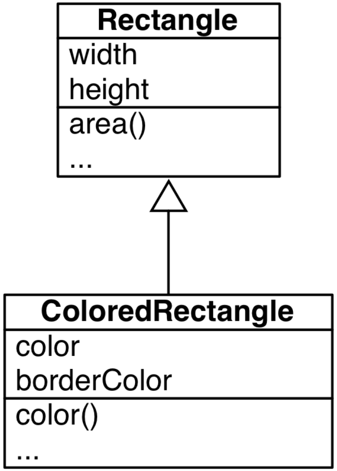
- Often we want small adaptations
- We want to extend existing behavior and state
- We do not want to reimplement everything: We want to reuse
- Solution: **class inheritance**
- A class extends the definition of its superclass



Basic subclass behavior

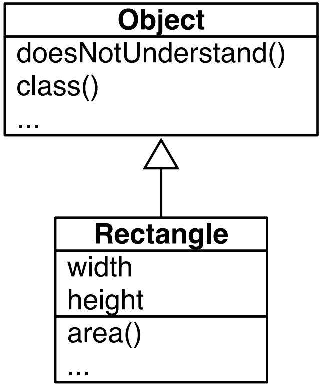
A subclass

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



Root of inheritance hierarchy

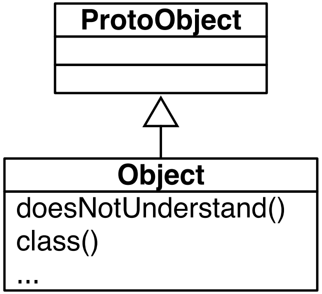
- Object is the root of most classes
  - defines the common behavior of all objects
  - raising an error, class access, ...



In Pharo: ProtoObject

ProtoObject (Object's superclass) has a special purpose:

- e.g. raising as much as errors as possible
- so that the system can catch such errors and do something with them
- useful for building advanced techniques such as proxy objects



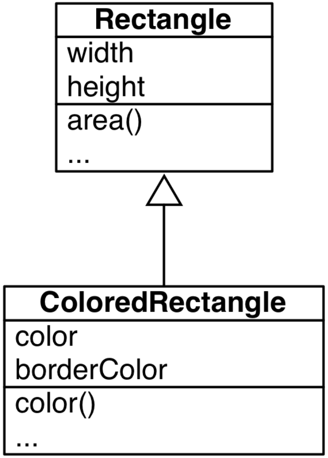
Two aspects of inheritance

Inheritance is

- **static** for state/instance variables (i.e., during class creation)
- **dynamic** for behavior (i.e., during execution)

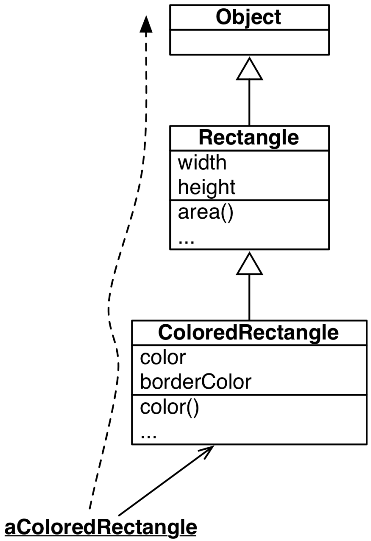
Inheritance of instance variables

- Happens during **class definition**
- Computed from
  - the class own instance variables
  - the ones of its superclasses
  - usually no duplicate in the chain
- ColoredRectangle **has a** width, height, color, **and** borderColor



Inheritance of behavior

- Happens at **run time**
- The method is looked up
  - starting from the receiver's class
  - then going to the superclass



What you should know

- Inheritance allows a class to "refine" and "add" state and behavior
- A class has 1 and only 1 superclass
- A class eventually inherits from `Object`
- Inheritance of state is static
- Inheritance of behavior is dynamic

A course by

S. Ducasse, G. Polito, and Pablo Tesone