Objects and Design in Smalltalk

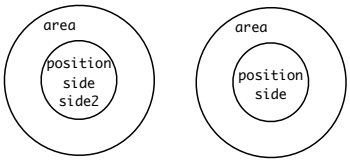
6. Two Kinds of Inheritance

Éric Tanter
etanter@dcc.uchile.cl





Beauty and the beast



aRectangle aSquare

- Design problem
  - should Rectangle be superclass of Square?
  - vice versa?

- Squares are a special kind of rectangles
  - with just one property more
- Inheriting is about adding 'qualities'
  - a rectangle has one more instance variable
- Both are valid
  - and even a third alternative: no inheritance at all

Dilemma

- Consumer point of view: values consistency
  - respecting the intuitive a-kind-of hierarchy
  - a square is a special kind of rectangle

"beauty"
subtype inheritance

- Developer point of view: values practicality
  - it is simpler to make rectangle inherit from square
  - doing the other way, every square has a useless iv

"beast"
implementation inheritance

Forms of inheritance

- First noted in 1986 [Snyder, Lalonde et al., etc.]

Subtype Inheritance (beauty)	Subclass Inheritance (beast)
subtyping	subclassing
specification inheritance	implementation inheritance
visible inheritance	invisible inheritance
essential inheritance	incidental inheritance

Polymorphism

- Type matters for polymorphism
- an algorithm that expects a Rectangle can accomodate with a Square
  - this is called substitutability
  - Square is a subtype of Rectangle
- the reverse is not true!
  - rectangle won't work if I expect only squares
  - Rectangle is not a subtype of Square
- substitutability and polymorphism amount to the same idea

Being a subtype

- What does it mean?
  - a-kind-of
  - substitutable
  - consistent
  - polymorphic
- More formally, what is consistency?
  - anarchy: you can do what you want
  - conformance: restriction on signatures (covariance, contravariance)
  - behavioral consistency: weaken preconds, strengthen postconds
  - rigidity: forbid alternate implementations of a method