

# Prototype-based Object-Orientation

Carl Friedrich Bolz, David Schneider  
Dynamische Programmiersprachen  
Heinrich-Heine-Universität Düsseldorf  
Sommersemester 2010

# Instances and Classes

typical ingredients of object-orientation:

- ▶ classes (with methods)
- ▶ instances (with attributes)
- ▶ inheritance

# Relationships between Objects

- ▶ two types of arrows:
  - ▶ *is-instance-of*
  - ▶ *inherits-from*
- ▶ types are special objects
- ▶ object creation via type

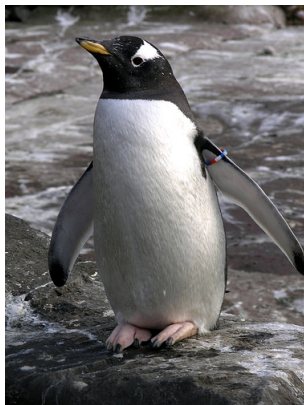
# Classes

- ▶ classes describe the shape of instances
- ▶ taxonomy: classify all objects into some hierarchy of categories
- ▶ basic idea: a group of objects belong to the same category, if they share some common properties
- ▶ problems: often no “right” way to classify things
- ▶ some categories seem to have no common properties

# Example

- ▶ common properties of a bird?
- ▶ feather, beak, ability to fly

But:



(by paulboxley from Flickr)

# Overcoming the Problems

- ▶ Prototype Theory (Eleanor Rosch) in cognitive psychology
- ▶ graded categorization
- ▶ “good examples” of a category

# Prototype-Based Object-Orientation

- ▶ Every object has one or more parents
- ▶ anything can be a parent object
- ▶ object creation via *cloning*



# Why?

- ▶ because we can

# Why?

- ▶ because it reduces the number of concepts
- ▶ simpler
- ▶ more powerful than class-based object-orientation

# Languages

- ▶ SELF
- ▶ Io
- ▶ JavaScript (but messy)
- ▶ Slate, Keto (research languages)

# SELF

- ▶ built by Dave Ungar and Randall Smith in 1986 at Xerox PARC
- ▶ Smalltalk-like language, but Prototype-based
- ▶ pioneered many implementation techniques (VM with JIT compiler, generational GC)

# SELF Object-Model

- ▶ an object contains a set of slots
- ▶ a method is an object that is activatable
- ▶ a data slot can have an additional assignment slot
- ▶ an object can have one or several parent slots

# Message Sends

- ▶ go along the chain of parents until you find the message
- ▶ receiver of the message is the original object
- ▶ multiple inheritance via priorities, not linearization
- ▶ in case of ambiguity, raise error