Frameworks: an OO library approach

Key idea: library includes a set of classes (often abstract) from which users make subclasses, not just instances

Morphic is a good example
- Morph abstract class defines many methods, which call each other via sends to self
- subclasses of Morph provide a small amount of specific behavior (e.g. the shape, or the interaction)

Users of Morphic proceed by defining subclasses of Morph (or of one of its more specialized subclasses)
- override a few methods (e.g. initialize, step, mouseDown:) to do their client-specific work

First-class classes

In Smalltalk, classes are themselves objects
+ uniformity of language design
+ can pass around classes just like any other object
+ can send messages to classes just like any object
  - e.g. new et al., G

But
if every object has a class, and every class is an object, then what is the class of a class?
- the class of the class holds the methods for the class

The class of a class is called a metaclass

Metaclass designs

In Smalltalk-76:
- all classes were instances of the single class Metaclass, which was an instance of itself
  + “simple”
    - every class had to have the same operations
      ⇒ couldn’t have class-specific initialization methods

In Smalltalk-80 & Squeak:
- each class was an instance of its own unique metaclass (e.g. the class Point had a unique class Point class)
- each metaclass was an instance of the class Metaclass, which was an instance of Metaclass class, which was an instance of Metaclass
- browser hides metaclasses from user
+ allows each class to have its own class methods
  - massively complicated

An alternative: drop classes

Prototype-based, or classless languages (e.g. Self, Cecil, ...)

Idea:
- let objects store their own methods directly, without recourse to a class
- let objects inherit directly from other objects
- new objects created by copying existing ones, or by making fresh objects that inherit from existing ones
- can build separate factory objects to hold things that used to be in a class
- browser and inspector are merged
+ no metaclasses
+ simpler language
  - less structure
Other alternatives

Make classes second-class (C++)
- classes aren’t real objects
- can’t send them messages
  ⇒ don’t have to worry about what their class is
- special second-class constructor “methods”
  - no dispatching or inheritance for second-class “methods”

Treat class “methods” in a second-class way (Java)
- classes are objects, but have a common set of methods
  (as in Smalltalk-76)
- introduce second-class static methods, static fields, and
  constructors to do some of what Smalltalk-80 classes can do