DOUBLE DISPATCH

- **Dispatch** is the runtime procedure for looking up which function to call based on the parameters given.
  - (In Ruby, **Single Dispatch** on the implicit self parameter)

- **Double Dispatch** is dispatching based not only on the runtime class of self, but on the first method parameter as well.
  - (Ruby/Java doesn’t have this, but we can emulate it.)

- We can go even further with **Multiple Dispatch** or **Multimethods**
DOUBLE DISPATCH

• The idea to emulating double dispatch is to use the built-in single dispatch twice.
  • Have the principal method immediately call another method on its first parameter, passing in self.
  • Then the second call will know the class of self implicitly, and the class of the first parameter through single dispatch.

• This is a common idiom in SML, using case statements
DOUBLE DISPATCH

Code examples!
MIXINS

All kinds of objects and methods!
But there are a lot of recurring methods.

Code reuse or redundant?
Inherited? What about String and FixNum?

Have <, >, <=>, but their nearest common ancestor is Object.

Mixins are perfect for code reuse between otherwise unrelated classes.
Our lookup rules get changed slightly...

- When looking for receiver obj's method m, look in obj's class, then mixins that class includes (later includes shadow), then obj's superclass, then the superclass' mixins, etc.
MIXINS

Oh boy! More examples!
MIXINS

Some standard mixins (These are your friends)

Comparable: Implement $<=$ and you get all comparison operators

Enumerable: Implement each and you get 47 methods!
VISITOR PATTERN

To the code!