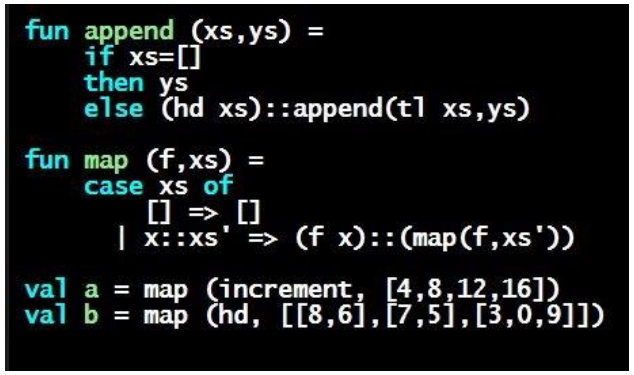

Programming Languages

Dan Grossman

Class Definitions are Dynamic

Changing classes

- Ruby programs (or the REPL) can add/change/replace methods while a program is running

- Breaks abstractions and makes programs very difficult to analyze, but it does have plausible uses
  - Simple example: Add a useful helper method to a class you did not define
    - Controversial in large programs, but may be useful

- For us: Helps re-enforce “the rules of OOP”
  - Every object has a class
  - A class determines its instances’ behavior

Examples

- Add a **`double`** method to our **`MyRational`** class
- Add a **`double`** method to the built-in **`FixNum`** class
- Defining top-level methods adds to the built-in **`Object`** class
  - Or replaces methods
- Replace the **`+`** method in the built-in **`FixNum`** class
  - Oops: watch **`irb`** crash

The moral

- Dynamic features cause interesting semantic questions

- Example:
  - First create an instance of class `C`, e.g., `x = C.new`
  - Now replace method method `m` in `C`
  - Now call `x.m`

  Old method or new method? In Ruby, new method

  The point is Java/C#/C++ do not have to ask the question
  - May allow more optimized method-call implementations as a result