# Dan Grossman, CSE341, Programming Languages
# Lecture 19 Introduction to Ruby
# This has mostly silly code explaining features simply.
# See also lec19_example.rb for a full example of a class implementing
# rational numbers like we did when studying the ML module system

class Hello  def my_first_method    puts "Hello, World!"  end  end

x = Hello.new
x.my_first_method

class A  def m1    34  end

def m2 (x,y)    z = 7
    if x > y
        false
    else
        x + y * z
    end
  end
end

class B  def m1    4  end

def m3 x    x.abs * 2 + self.m1  end
end

# returning self is convenient for "stringing method calls"  class C  def m1    print "hi "
        self
    end

def m2
        print "bye "
        self
    end

def m3
        print "w"  self
    end
end

# example uses for classes A, B, and C (can type into irb)
# here in a multiline comment, which is not well-known
=begin
a = A.new
thirty_four = a.m1
b = B.new
four = b.m1
forty_seven = B.new.m3 -17
thirty_one = a.m2(3,four)
c = C.new
=end

# example uses for classes D, E, and F
=begin
x = D.new
y = D.new # different object than x
z = x # alias to x
x.foo # get back nil because instance variable not initialized
x.m2 3 # error because try to add with nil object
x.ml # creates @foo in object x refers to
=end
z.foo # remember, x and z are aliases
z.m2 3
x.foo
y.m1
y.m2 4
y.foo
x.foo

w = E.new 3
y = E.new
w.foo + v.foo

d = F.new 17
d.m2 5
e = F.new
e.m2 6
d.bar
forty = F::Dans_Age + d.bar
=end

# some examples of adding or changing methods in existing classes

class MyRational # this example extends a class in lec19_example.rb
def double
  self + self
end
end
class Fixnum
def double
  self + self
end
end

def m
  42
end
class Object
def m
  43
end
end