class Pt
   attr_accessor :x, :y
   def distToOrigin
      Math.sqrt(x * x + y * y)
   end
end

class ColorPt < Pt
   attr_accessor :color
   def darken # error if @color not already set
      self.color = "dark" + self.color
   end
end

class Pt3D < Pt
   attr_accessor :z
   def distToOrigin
      Math.sqrt(x * x + y * y + z * z)
   end
end

# This does not exist in Ruby (or Java/C#, it does in C++)
# class ColorPt3D_3 < ColorPt, Pt3D
# end
# two ways we could actually make 3D Color Points:
class ColorPt3D_1 < ColorPt
   attr_accessor :z
   def distToOrigin
      Math.sqrt(x * x + y * y + z * z)
   end
end

class ColorPt3D_2 < Pt3D
   attr_accessor :color
   def darken
      # error if @color not already set
      self.color = "dark" + self.color
   end
end

########## mixins ##########

module Doubler
   def double
      self + self # uses self’s + message, not defined in Doubler
   end
end

class Pt
   attr_accessor :x, :y
   include Doubler
   def + other
      ans = Pt.new
      ans.x = self.x + other.x
      ans.y = self.y + other.y
      ans
   end
end

class String
   include Doubler
end

# these are probably the two most common uses in the Ruby library:
# Comparable and Enumerable
def initialize(x, y, z, color="clear")
  @x = x
  @y = y
  @z = z
  @color = color

  include Color

end