class MyRational
    def initialize(num, den=1) # second argument has a default
        if den == 0
            raise "MyRational received an inappropriate argument"
        elsif den < 0
            @num = -num # fields created when you assign to them
            @den = -den
        else
            @num = num
            @den = den
        end
        reduce # i.e., self.reduce() but private
    end
    def to_s
        ans = @num.to_s
        if @den != 1
            ans += "/" + @den.to_s
        end
        ans
    end
    def to_s2
        dens = "/" + @den.to_s
        if @den != 1
            @num.to_s + dens
        end
    end
    def to_s3
        "#{@num}#{if @den==1 then "" else "." end}#{if @den==1 then "" else "/" end}#{@den}"  # using things like Racket's quasiquote and unquote
    end
    def add! r
        a = r.num  # only works b/c of protected methods below
        b = r.den  # only works b/c of protected methods below
        c = @num
        d = @den
        @num = (a * d) + (b * c)  # convenient for stringing calls
        @den = b * d
        reduce
    end
protected
    # there is very common sugar for this (attr_reader)
    # the better way:
    # attr_reader :num, :den
    # protected :num, :den
    # we do not want these methods public, but we cannot make them private
    # because of the add! method above
    def num
        @num
    end
    def den
        @den
    end
private

Feb 25, 18 11:21 lec19−code−example.rb Page 1/2

lecular example

def gcd(x,y) # recursive method calls work as expected
    if x == y
        x
    elsif x < y
        gcd(x,y-x)
    else
        gcd(y,x)
    end
end

# can have a top-level method (just part of Object class) for testing, etc.
def use_rationals
    r1 = MyRational.new(3,4)
    r2 = r1 + r1 + MyRational.new(−5,2)
    puts r2.to_s
    (r2.add! r1).add! (MyRational.new(1,−4))
    puts r2.to_s
    puts r2.to_s2
    puts r2.to_s3
    puts r2
end

Feb 25, 18 11:21 lec19−code−example.rb Page 2/2