Object Orientation in Ruby

4010-350
Personal Software Engineering

Declaring a Class

```
class Point
    . . .
end
```

- Technically "Point" is a constant (as is any other entity whose name begins with a capital.
- By default, the super-class of Point is "Object."
- Note – we can extend a class at any point by simply opening it up and adding behavior
- Do so *very* carefully.

Creating an Object in a Class

p = Point.new(x, y)

- new is a class method (like static in Java).
- It allocates space and calls the initialize method in of the new object.
- initialize looks like a constructor, but it is just a method called by the new class method.
- Since Ruby is dynamically typed, there is no way to create multiple initialize methods.

Initialization

```
class Point
   def initialize(x, y)
      @x = x ;  @y = y
   end
end
```

- Arguments to initialize: x and y
- @x and @y are object instance variables.
- Instance variables are private – to access you need setters and getters – see below
- Class variables (rarely used) are prefixed by @@
- Global variables (even rarer) are prefixed by $
- Instance variables & arguments begin with a lower case letters.

Default Arguments

```
class Point
    def initialize(x = 0, y = 0)
        @x = x ;  @y = y
    end
end
```

- p = Point.new – p is initialized to the origin.
- p = Point.new(5) – p is initialized to (5, 0).
- p = Point.new(3, 7) – p is initialized to (3, 7)

Setters & Getters – The Wrong Way

```
class Point
    def initialize(x = 0, y = 0)
        @x = x ;  @y = y
    end

    def x
        @x
    end

    def x=(newx)
        @x = newx
    end
end
```

Setters & Getters – The Right Way

```
class Point
   def initialize(x = 0, y = 0)
      @x = x ;  @y = y
   end

   attr_accessor :x, :y
end
```

- attr_accessor is a method that takes symbols and
  - defines instance variables from those symbols
  - defines the setter and getter methods
- For more control: attr_reader and attr_writer
- The previous form can be used for "pseudo" variables
- Example: rho & theta for polar coordinates

Other Instance Methods

```
class Point
   def move_by(deltax, deltay)
      @x += deltax ;  @y += deltay
      self                                          # hmmmm???
   end

   def move_to(other_point)
      @x = other_point.x ; @y = other_point.y
      self
   end

   def to_s       #override default converter to String
      "(#{@x}, #{@y})"
   end
end
```

Class Methods & Variables

```
class Point
   @@count = 0
   def initialize(x = 0, y = 0)
      @@count += 1

      @x =x ;  @y =x
   end

   def Point.count
      @@count
   end
end
```

ON TO THE ACTIVITY