Object-Oriented Programming (OOP) Basics

León Jaramillo

softserve

Object-Oriented Programming (OOP) Basics

- Ruby is a strongly **object-oriented language**, where everything are objects.
- It is also **class-based**. It means that typically every object is an instance of an existing class.
- Here we might review some key OOP **concepts**, such as: classes, objects, properties, attributes, operations, methods, and constructors.
- Also, we could talk about OOP **pillars** like abstraction, inheritance, polymorphism, and encapsulation.

Exploring OOP Basics with Ruby

Let's open `oop-example.rb` with your favorite code editor and explore the implementation of the following concepts:

- Declaring a class
- Declaring class attributes
- Defining an accessor (`reader`, `writer` or `accessor`)
- Defining the `initialize` method
- Defining attributes and calling them
- Defining instance methods
- Using the `to_s` method
- Defining a class method
- Using another way to access attributes
- Creating an object from an existing class
- Calling an object's attributes and methods

Code Example (only use if necessary)

```
class Television
  #Class attributes
  @@brand = "LG"
  @@min_channel = 0
  @@max_channel = 10000

  #Defined an accessor
  attr_accessor :volume

  #Initialize method (like a constructor)
  def initialize(channel, volume, power, bluetooth)
    @channel = channel
    @volume = volume
    @power = power
    @bluetooth = bluetooth
  end

  #Instance methods
  def turn_on()
    @power=true
  end

  def turn_off()
    @power=false
  end

  def shift_channel_up()
      if @power == true and @channel < @@max_channel
        @channel = @channel+1
      end
    end

    def shift_channel_down()
      if @power == true and @channel > @@min_channel
        @channel = @channel-1
      end
    end

    def change_channel(number)
      if @power == true and number >= @@min_channel and
number <= @@max_channel
        @channel = number
      end
    end

    #to_s method (like any "to string" method)
    def to_s
      return "The TV is turned " + (@power ? "ON" : "OFF") +
", in the channel " + @channel.to_s + ", with a volume of "
+ @volume.to_s + " and with bluetooth in " +
@bluetooth.to_s
    end
```

Code Example (just use if necessary)

```
#Class method
  def self.get_brand()
    @@brand
  end

  #Classical getter and setter
  def set_bluetooth(value)
    @bluetooth = value
  end

  def get_bluetooth()
    @bluetooth
  end
end
```

```
my_tv = Television.new(3, 10, false, false)
puts my_tv
my_tv.turn_on
puts my_tv
my_tv.shift_channel_up
my_tv.shift_channel_up
my_tv.shift_channel_up
puts my_tv
my_tv.shift_channel_down
puts my_tv
my_tv.change_channel(20000)
puts my_tv
my_tv.change_channel(74)
puts my_tv
my_tv.volume = 25
puts "The new volume is " + my_tv.volume.to_s
puts my_tv
puts "The brand is " + Television.get_brand
puts my_tv
my_tv.set_bluetooth(true)
puts my_tv
```

Useful Resources

- Learning Ruby: From Zero to Hero: https://www.freecodecamp.org/news/learning-ruby-from-zero-to-hero-90ad4eecc82d/

- Ruby - Object Oriented: https://www.tutorialspoint.com/ruby/ruby_object_oriented.htm

- Implementing OOP concepts with Ruby: https://www.geeksforgeeks.org/object-oriented-programming-in-ruby-set-1/

Homework

1. How do private, public and protected modifiers work in Ruby?

2. How can I do to set the initialize method, so it works receiving optional arguments?

3. Create a Student class with the following components:
   - Instance attributes: full name, address, phone, age
   - A class attribute with the name of the university
   - The initialize method
   - Only reader accessors for all the attributes
   - A method that returns true if the student is underage
   - A method that returns true if the student's age is more than 27


COLOMBIA
DC
Thanks! Any question?
softserve