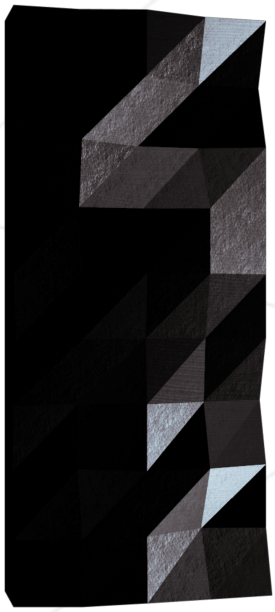

OOP - Part 1

Essentials of Object Oriented Programming

Michael Lu mlu@student.42.fr

Summary: This project will help you learn the essentials of objective oriented programming.

Contents

Chapter I

Foreword

Did you know I love cooking and cooking is a great way to learn stuff?

Cooking teaches you a lot of skills that is beneficial to you. Regardless if you prefer your mom's home cooking or maybe your dad's barbecue, but have you ever tried following a recipe and learning it yourself?

If you ever start learning how to cook you will soon realize you need a couple of things first. You need measuring tools, some kind of hot plate, a way to cut or prep ingredients. Each of these objects are their own entity but they work together to provide you a delicious meal.

Object oriented programming is very similar to this concept. Hah! Bet you thought I wouldn't bring up programming eh? Just like cooking, you will be creating objects in objected oriented programming (I wonder why it's called that), and learning how to utilize them to help you create some cool stuff.

Chapter II

Goals

The goal of this project is to introduce you to the basics object oriented programming. By the end of this project you should know how to:

- Create classes
- Initiate an instance of a class
- Assign class methods and attributes
- Use variables specific to each instance
- Design classes that inherit from each other.

You will be exploring a fundamental topic of object oriented programming so take advantage of all the resources including articles, videos, your neighbor, StackOverflow and so forth. There are many tutorials on classes and inheritance.

Chapter III

General instructions

- This project will only be corrected by actual human beings. You are therefore free to organize and name your files as you wish, although you must respect some requirements listed below.
- You must follow the exercise details and instruction clearly
- You must turn in all the requested files
- Ask your peers, mentor, slack or anywhere else if you need any help, and make sure to have fun!

Chapter IV

Exercise 00

	ex00: Your first class
Topics to study :	
Files to turn in : `main.(rb/py), first_class.(rb/py)`	
Notes : `n/a`	

Make your first class (a class named FirstClass in a file named first_class) with a constructor that says "Hello World". Instanciate the class inside of a main function located in main.(rb/py).

```
?> ruby first_class.py
Hello World
?>
```

`Google classes or check ft_arena or ft_boardgame tutorial videos`

```
if __name__ == "__main__":
```

`Ruby: require_relative first_class. Python: from . import first_class`

Chapter V

Exercise 01

	ex01 : Your second class and first inheritance
Topics to study :	
Files to turn in : `main.(rb/py), first_class.(rb/py), second_class.(rb/py)`	
Notes :	

Make your second class (a class named SecondClass in a file named second_class) that will inherit from the first class. Its constructor should directly call the first class constructor that says "Hello World". You must instantiate the second class only in your main.

```
?> ruby main.rb
Hello World
?>
```

`Google class inheritance or check ft_arena or ft_boardgame tutorial videos`

Chapter VI

Exercise 02

	ex02 : Your first parameter and passing parameter
Topics to study :	
Files to turn in : `main.(rb/py), first_class.(rb/py), second_class.(rb/py)`	
Notes :	

You now need your second class to take a parameter "name" (which will be your login name) and pass it into the first class which will display "Hello ". You can hard-code your login name into the program without calling input() or gets() to fetch it. You must instantiate the second class only in your main.

```
?> ruby main.rb
Hello mlu
?>
```

`Google how to send parameter into classes or check ft_arena or ft_boardgame tutorial videos`

Chapter VII

Exercise 03

	ex03 : Your first method
Topics to study :	
Files to turn in : `main.(rb/py), first_class.(rb/py), second_class.(rb/py)`	
Notes :	

You now need to create a method inside your first class called say_hello which will take the name from the constructor and print out "Hello ". When the method is called pring out a sentance stating that it has been called. You must instantiate the second class only in your main.

```
?> ruby main.rb
Method say_hello in FirstClass is called
Hello mlu
?>
```

`Google class methods or check ft_arena or ft_boardgame tutorial videos`

Chapter VIII

Exercise 04

	ex04 : Your second method
Topics to study :	
Files to turn in : `main.(rb/py), first_class.(rb/py), second_class.(rb/py)`	
Notes :	

You now need to create a method inside your second class called roll_dice which will randomly generate a number from 1 to 6. Roll_dice will then call the Hello method in its parent class and pass it the random number. You should see an output of "Hello <username>, your number is ". Remember, every method your write should print something to announce it has been called. You must instantiate the second class only in your main.

```
?> ruby main.rb
Method roll_dice in SecondClass called
Method hello in FirstClass is called
Hello mlu, your number is 2
?>
```

`Google class methods/methods interaction or check ft_arena or ft_boardgame tutorial videos`

Chapter IX

Exercise 05

	ex05 : Your first class variables, and more methods!
Topics to study :	
Files to turn in : `main.(rb/py), first_class.(rb/py), second_class.(rb/py)`	
Notes :	

Your second class will now take in a second parameter, a string called "hobby", when it is instantiated. Store the hobby in an instance variable. You will write a method called get_hobby in your second class that returns this variable. In your main you need to print out "Your hobby is " where must be returned from the get_hobby method. You must instantiate the second class only in your main.

```
?> ruby main.rb
Method roll_dice in SecondClass called
Method hello in FirstClass is called
Hello mlu, your number is 5
Your hobby is being lazy
?>
```

Read about the difference between class variables and instance variables. What is an instance of a class?