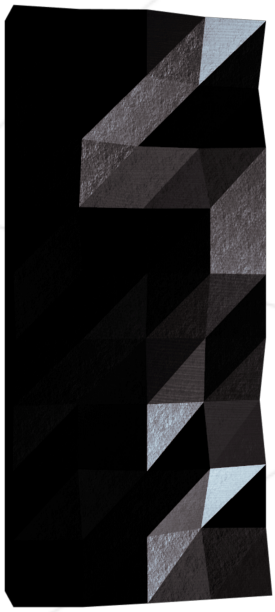

# oop_introduction

## Basic Object Oriented Programming

Michael Lu mlu@student.42.fr

*Summary: This project will help you learn the very basic of objective oriented programming*

# Contents


| | | |
|---|---|---|
| I | Foreword | 2 |
| II | Introduction | 3 |
| III | Goals | 4 |
| IV | General instructions | 5 |
| V | Exercise 00 | 6 |
| VI | Exercise 01 | 7 |
| VII | Exercise 02 | 8 |
| VIII | Exercise 03 | 9 |
| IX | Exercise 04 | 10 |
| X | Exercise 05 | 11 |
| XI | Turn-in and peer-evaluation | 12 |

# 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 stuff 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

# Introduction

The goal of this project is to complete a sequence of exercises which will teach you the basics of object oriented programming.

If you are using python or another language approved by hack high school make sure to research into python equivalent concepts on your own. This project can be completed in any approve project language, however the tutorial and video guides will be in Ruby.

So what are you waiting for? Get going.

# Chapter III

# Goals

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

- Create classes
- Create methods
- Create inheritances
- Be awesome

You will be exploring a fundamental topic of object oriented programming so take advantage of all the resources including the videos that are present in ft_arena and ft_boardgame (the other object oriented projects available) , your neighbor and google. There are many tutorials on classes and inheritance.

# Chapter IV

# 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
- Your project must be written in a language approved by the hack high school program
- Ask your peers, mentor, slack or anywhere else if you need any help, and make sure to have fun

# Chapter V

# Exercise 00

|  | Exercise 00 |
|---|---|
| ex00 : Your first class | |
| Turn-in directory : $ex00/$ | |
| Files to turn in : `main.(rb/py), first.class.(rb/py)` | |
| Allowed functions : `All` | |
| Notes : `n/a` | |

Make your first class (called first.class) with a constructor that says "Hello World". You must instanciate your first class only in a main.

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

`Google classes or check ft_arena or ft_boardgame tutorial videos`

# Chapter VI

# Exercise 01

|  | Exercise 01 |
|---|---|
| ex01 : Your second class and first inheritance | |
| Turn-in directory : $ex01/$ | |
| Files to turn in : `main.(rb/py), first.class.(rb/py), second.class.(rb/py)` | |
| Allowed functions : `All` | |
| Notes : `n/a` | |

Make your second class (called second.class) that will inherit from the first class and call the first class constructor that says "Hello World". You must instanciate 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 VII

# Exercise 02

|  | Exercise 02 |
|---|---|
| ex02 : Your first parameter and passing parameter | |
| Turn-in directory : *ex*02/ | |
| Files to turn in : `main.(rb/py), first.class.(rb/py), second.class.(rb/py)` | |
| Allowed functions : `All` | |
| Notes : `n/a` | |

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 must instanciate 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 VIII

# Exercise 03

|  | Exercise 03 |
|---|---|
| ex03 : Your first method | |
| Turn-in directory : $ex03/$ | |
| Files to turn in : `main.(rb/py), first.class.(rb/py), second.class.(rb/py)` | |
| Allowed functions : `All` | |
| Notes : `n/a` | |

You now need to create a method inside your first class called Hello which will take the name from the constructor and print out "Hello ". When the method is call put some sort of output for identification. You must instanciate the second class only in your main.

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

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

# Chapter IX

# Exercise 04

|  | Exercise 04 |
|---|---|
| ex04 : Your second method | |
| Turn-in directory : *ex*04/ | |
| Files to turn in : `main.(rb/py), first.class.(rb/py), second.class.(rb/py)` | |
| Allowed functions : `All` | |
| Notes : `n/a` | |

You now need to create a method inside your second class called number which will randomly generate a number from 1 to 6. It than will pass it to Hello method in first class which will print out "Hello , your number is ". When any method is called it must put some sort of output for itself. You must instanciate the second class only in your main.

```
?> ruby main.rb
Number method 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 X

# Exercise 05

|  | Exercise 04 |
|---|---|
| ex05 : Your first class variables, and more methods! | |
| Turn-in directory : $ex04/$ | |
| Files to turn in : `main.(rb/py), first.class.(rb/py), second.class.(rb/py)` | |
| Allowed functions : `All` | |
| Notes : `n/a` | |

Your second class will now take a second parameter called hobby and it must be stored as a variable. You will write a method called getHobby 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 getHobby method. You must instanciate the second class only in your main.

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

Google what ever you want (maybe creating variables inside classes?)
or check ft_arena or ft_boardgame tutorial videos

# Chapter XI

# Turn-in and peer-evaluation

Turn your work in using your `GiT` repository, as usual. Only work present on your repository will be graded in defense.

Good luck and remember to have fun!