# OOP-lab2

**Michele Pratusevich**

January 27, 2014

## 1 Lab 4: OOP, classes, and objects

1. Make a new folder called Lab4 (in your MEET-YL1 folder) to store all of your work. If you do not have a MEET-YL1 folder on your desktop (or are logging in as Guest), `cd` onto the Desktop and `git clone http://github.com/YOURUSERNAME13-meet/MEET-YL1` to get your MEET-YL1 folder.
2. Open an empty text file and save it as `number.py` in the Lab4 folder.
3. In the `number.py` file, create a class called `Integer`. This class should have one variable that stores the number, and one method called `display` that just prints out the number.
4. Below your class definition, make a `__main__` method. It should look like this:

   ```
   if __name__=="__main__":
       print "Michele"
   ```

   Inside this main method, create a new `Integer` called `test` of the class that you created, and prove that the `display` method works by calling `test.display()`. This piece of code will serve as your tests for the classes you create.
5. Add a new parameter (variable) to your `Integer` class that will describe whether the integer is a positive or a negative number. (What kind of information will this variable store?). You will need to change your contructor to include this second variable. Change the `display` method to display a number correctly, based on whether it is positive or negative. For example, instead of displaying "9", display "-9" if necessary. Make sure your main method tests still work correctly.
6. Create a subclass of your `Integer` class called `NegativeInteger`. The constructor for this class should only have one parameter - asking for the number. Because the class is called `NegativeInteger`, one of the variables from `Integer` will already be set. Make sure to take care of this in the constructor.
7. In your main method, create a new object of type `NegativeInteger` and call it's `display` method, making sure it works correctly.
8. Override the `display` method of the `NegativeInteger` class. It should be the SAME as the `display` method of the `Integer` class, but afterwards, it should also print to the screen "This is an object of the NegativeInteger class". Remember, to call a method from a superclass inside a subclass, do:

   ```
   SuperClassName.methodname(self, arguments)
   ```

   Test to make sure these display methods work correctly.
9. Now instead of calling the `display` method separately for all the objects you created, create an `Integer` object and a `NegativeInteger` object. Put these objects in a list. Make a loop that iterates over all the objects in that list and calls their `display` methods.

10. (extra) Instead of hard-coding the numbers that you want to create in your main method, ask the user to give you a positive number and a negative number to create. (Which class types will you create from these objects? Does it matter?)
11. (extra) Ask the user how many numbers they want to create, and for each number, ask them whether it is positive or negative, then create these numbers.
12. (extra) Figure out a way to do arithmetic (adding, subtracting, multiplying, dividing) with the `Integers` and `NegativeIntegers`. (*Hint: One way of doing this is to make four new methods to the `Integer` class, called something like `add()`, `subtract()`, `multiply()`, and `divide()`. Start with addition, and figure out how to test it and work up from there.*)

For reference, here is the example `Student` and `UniversityStudent` class I made for you in class.

In [1]:
```
class Student(object):
    def __init__(self, name_to_set="", year_to_set=0, score=100):
        self.name = name_to_set
        self.year = year_to_set
        self.score = score

    def letter_grade(self):
        if self.score >= 90:
            return "A"
        elif self.score >= 80:
            return "B"
        elif self.score >= 70:
            return "C"
        elif self.score >= 65:
            return "D"
        else:
            return "F"
```

In [2]:
```
class UniversityStudent(Student):
    def __init__(self, name_to_set="", year_to_set="", uni=""):
        super(UniversityStudent, self).__init__(name_to_set, year_to_set)
        self.university = uni

    def signature(self):
        return self.name + ", " + self.university + " " + str(self.year) +
```

In [5]:
```
if __name__=="__main__":
    m = Student("Michele Pratusevich", 2013)
    m.score = 85
    print m.score
    print m.letter_grade()

    n = UniversityStudent("Kyle Hannon", 2013, "MIT")
    print n.name
    print n.score
    print n.year
    print n.signature()
```

```
85
B
Kyle Hannon
100
2013
Kyle Hannon, MIT 2013 Grade: A
```