CMSC 201 Fall 2015
Lab 11 – Classes and Objects

Assignment: Lab 11 – Classes and Objects
Due Date: During discussion, November 16th through November 19th
Value: 1% of final grade

Part 1: Introduction to Classes

As we have discussed in our lectures, ***classes*** are structures that allow us to group together related data and functions which act on the data. They act as a blueprint for a specific thing including the related attributes and functions of that thing. When we create an ***instance*** of the class, we call it an ***object***. The two main parts of a class are the ***data members*** (class variables and instance variables) and the ***methods*** (functions).

The data members (class variables and instance variables) are the attributes that describe whatever the class is. Class variables are attributes that describe all instances of that class. Instance variables are attributes of a specific instance of a class.

Methods are functions in a class. The methods of a class look and act like special functions that are designed to be used specifically on an object. There are some special methods that we use to help to create an instance of a class called a ***constructor***. The constructor method that we use is called **`__init__`** and it is used to define the initial state of the object.

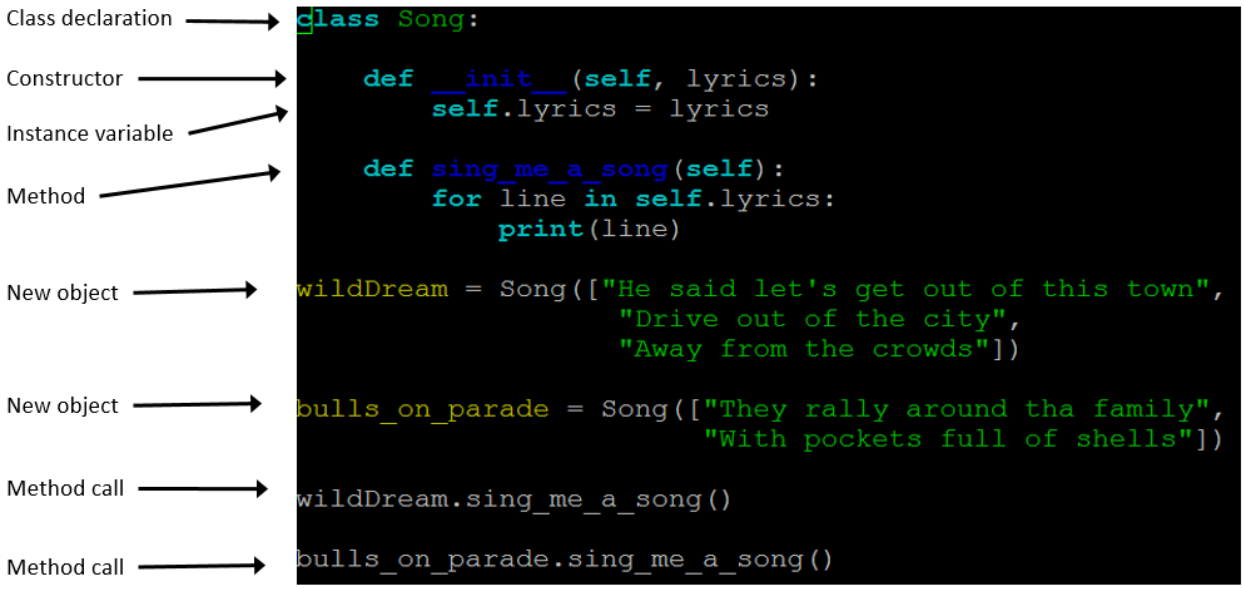
Let's look at an example:

```
class song:

    def __init__(self, lyrics):
        self.lyrics = lyrics

    def sing_me_a_song(self):
        for line in self.lyrics:
            print(line)
        print() # empty line at end

wildDream = song(["He said let's get out of this town",
                  "Drive out of the city",
                  "Away from the crowds"])

bulls_on_parade = song(["They rally around tha family",
                        "With pockets full of shells"])

wildDream.sing_me_a_song()

bulls_on_parade.sing_me_a_song()
```

And the output:

```
bash-4.1$ python ex1.py
He said let's get out of this town
Drive out of the city
Away from the crowds
They rally around tha family
With pockets full of shells
```

And a diagram defining each of the parts:



Part 2: Introduction to UML

Throughout the semester, we have been talking about the characteristics of writing good code. One thing that we have tried to emphasize throughout is the notion of design. We don't want to sit down and just try to write the code from scratch but rather we should try to plan our approach as to how to try and solve the problem at hand. Until now, we have discussed some simple techniques that we can use to help design our solution including flowcharts and pseudocode. Both of these offer us the ability to try and describe a solution to a problem either by using a diagram (flowchart) or using a written description (pseudocode). The ***Unified Modeling Language (UML)*** is another way that we can describe the solution to a problem using a more standardized approach.

UML is a modeling language that is used to describe the architecture of an application. UML uses a variety of diagrams to describe the structure and/or the behavior of the software. Structure diagrams describe the things that must be present in the system being modeled. Behavior diagrams describes what must happen in the system being modeled. There are 14 different types of UML diagrams commonly used (7 structural and 7 behavior). Figure 1 shows a variety of UML diagrams (Wikipedia, 2015).

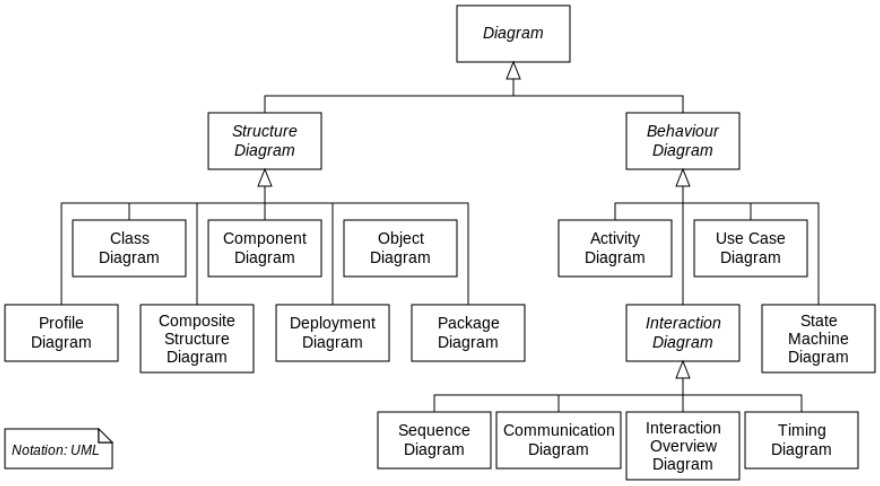
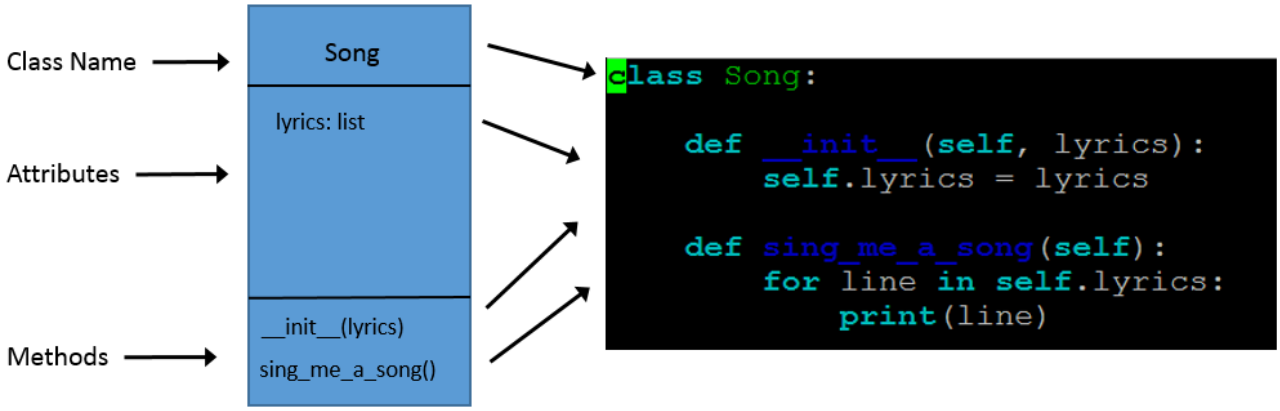


Figure 1. UML Diagrams

For this lab, we are focused on classes and so we will focus on class diagrams. Class diagrams show the classes of the software including the attributes, methods, and relationships between classes. Class diagrams can be used to show many characteristics of a class. They can even be used to show the relationship between classes.

Here is an example of a *simple* class diagram for a single class and the corresponding code:



Part 4: The "Ong" Language

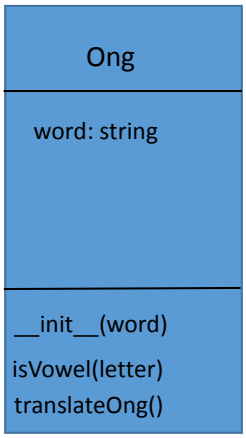
The "Ong" language is a childish encoding of English (similar to Pig Latin) that adheres to the following rules:

1. The vowels are 'a', 'e', 'i', 'o', 'u'. All other letters are consonants. For example, "yearly" has two vowels ('e' and 'a') and four consonants (the first 'y', 'r', 'l', and the second 'y').
2. For each vowel, make no change to the letter. For each consonant in the word, add "ong" after each consonant. For example, "Candy" would be "Conganongdongyong".

There is more information about the "ong" language here:
http://www.wikihow.com/Learn-to-Speak-ONG

Part 5: Creating the "Ong" Translator

For the hands-on part of the lab, we are going to use our techniques to try and generate the code required to create the "ong" language translator. For this effort, you will be building a class with one attribute, one constructor, and two methods (**`isVowel()`** and **`ongTranslate()`**). Here is the class diagram for the program:



We are going to give you the pseudocode to build each of the two functions (**`isVowel(letter)`** and **`translateOng()`**)

`isVowel(letter)` – This is a simple function that just checks to see if the letter is a vowel (a, e, i, o, or u). Returns either True or False.

`translateOng()` – This function loops through each letter in a string to decide if it is a vowel or not and either does nothing (vowel) or adds "ong" (consonants). Prints the output string when completed.

In this case, **`main()`** just asks the user for the word that they would like to translate.

Example output:

```
-bash-4.1$ python ong.py
Enter a string to translate: Baltimore
Bongalongtongimongoronge
-bash-4.1$
```

Part 8: Completing Your Lab

To test your program, first enable Python 3, then run **`ong.py`**. Test your code to make sure that it runs both encryption and decryption functions. Take the output of the encryption function to make sure that the decryption function returns what you originally input.

Since this is an in-person lab, you do not need to use the **`submit`** command to complete your lab. Instead, raise your hand to let your TA know that you are finished.

They will come over and check your work – they may ask you to run your program for them, and they may also want to see your code. Once they've checked your work, they'll give you a score for the lab, and you are free to leave.

IMPORTANT: If you leave the lab without the TA checking your work, you will receive a **zero** for this week's lab. Make sure you have been given a grade before you leave.

References:

Wiki-How (2015). "How to Learn to Speak ONG". Retrieved from http://www.wikihow.com/Learn-to-Speak-ONG

White, W. (2015). "Lab 5". Retrieved from http://www.cs.cornell.edu/courses/cs1110/2015fa/labs/