CSI 33 LECTURE NOTES (Ojakian)

Topic 2: Classes in Python and C++

OUTLINE

(References: Ch. 9)

1. Python Classes

   PROBLEM 1. *In Python start writing (will complete in C++ not in Python) a class for a general role-playing character (like "D and D"), called* `RPGchar`*. It should have two kinds of internal data: hit points (a non-negative integer) and alignment (a string that can take on the value 'good', 'bad', or 'neutral'). It should have at least the following methods: 1) returns hit points, 2) return alignment, 3) damage self (lose hit points), 4) improve self (gain hit points)*

   PROBLEM 2. *Try the following experiments:*

   (a) From an object, try accessing methods and data directly.

   (b) Define an external function with the same name as a class method. What happens when the method is called? What happens when the function is called outside the class?

   (c) Try the last experiment again, but now comment out the method definition.

   READ: Look at the Markov Gibberish Generator (read about it - chapter 3, pages 95-99: can skip details). To look at soon.

2. C++ Classes

   PROBLEM 3. *Now start writing the RPG Character class in C++, conducting the following experiments:*

   (a) Make everything public and see how it is like Python.

   (b) Remove the access indicator and see what happens.

   (c) Consider the above experiments done in Python - do them here.

   (d) Now try putting in a public and private part.

3. Python versus C++ Classes

   (a) Usual Differences still apply:

      i. Bodies: Python Indentation versus C++ Braces.

      ii. Static versus dynamic typing

   (b) Python: Everything public

   (c) C++: Options of public, protected, and private

   (d) Python: "self" parameter; C++: not

   (e) Member methods:

      i. Python defines syntactically within class

      ii. C++: defined like Python in class, or outside class declaration, using the class prefix

   (f) Contructor/initializer:

      i. Python: "init".

      ii. C++: method with same name as class

   (g) Passing an object to a function:

      PROBLEM 4. *In both Python and C++ try passing a RPG Character object, and changing the hit points to see if it persists after the function call. Try this in two ways in C++ (usual and by reference).*

   (h) Operator Overloading: Basically just different syntax

      PROBLEM 5. *Overload equivalence checking so that two characters are equal if they have the same alignment and their hit points are within one of eachother.*

      In Python: __eq__ (with self and other argument)

      In C++: operator== (with one argument)

4. Inheritance

   (a) In Python

      PROBLEM 6. *Begin in Python (but do not complete - will complete in C++) a child class for* `RPG_Char` *called* `Madman`*. It has the following additional internal data: A stash of weapons (just represented as single word strings). Write methods to add a weapon and see the inventory.*

   (b) In C++

      PROBLEM 7. *In C++ write a child class for* `RPG_Char` *called* `Madman`*. It has the following additional internal data: A stash of weapons (just represented as single word strings). Write methods to add a weapon and see the inventory. There is also a method:* `fling`*, where a random weapon is drawn - either the the most recent weapon acquired or the least recent weapon acquired. If he tries to throw a weapon when he has none he losses some number of hit points (we'll decide).*

5. Finish the problems

   (a) Look at and try out the Marvoe Gibberish Generator.

   (b) Complete the C++ RPG Character and Madman classes.