CS 162P
Programming Lab 7

For this exercise you will create a linked list that uses classes based off of composition and inheritance.

Program Description

This problem is designed to give you practice at using function overloading and coding a linked list. If you have questions, it is suggested you review the relevant Moodle documents for examples.

For this lab you will use the Person class from last week. Then you will create a Player class with derived Character classes from it. Finally, you will create a linked list that will hold instances of these Characters.

Program Requirements

For this, as in all assignments, you are expected to follow the course programming style guidelines.

You should have two class module files, one for the Person class and its methods and another for the Character class, its methods, and all its descendant classes with their methods. Note that the Character module must import the Person module.

Person Class

The Person class is the same as last week, it has private variables for firstName, lastName, and age. It sets these in the __init__ and has setters and getters for them.

Player Class

The Player class is similar to the Employee class in Lab 6 but is not abstract. It includes an instance of Person. It should have private variables for className, classAction, and level.

- __init__ creates a Person passing in the values from the parameter list. It also sets className and action to “unknown”. Level should be set to 0.
- getPlayerName – returns a string that is firstName, space, lastName from the Person data
- getPlayerAge – returns the age from the Person data
- getClassName – return the value of the className variable
- getAction – return the value of the classAction variable.
- getLevel – return the value of the level variable.

Player does not need any setters for this lab, everything will be defined in the __init__ method for an instance of Player.

The Player class also needs to override the following built-in functions

- __eq__ – return true when the two objects have the same firstName, lastName, age, and className
  __gt__ – return true when the first object has a higher level than the second object
- __str__ – return playerName + “aged” + playerAge + “playing a” + className + action

Character Classes

There are four classes derived from Player. These are all based on Role Playing Game character options. Each has an __init__ with parameters firstName, lastName, and age that is used to initialize Player. They define the className and classAction variables as follows:

- className Ranger – “shoots arrows”
- className Wizard – “casts fireballs”
- className Rogue – “picks pockets”
- className Priest – “heals”

Link Class

The Link class has private variables for a Player and next. It has an __init__ setting the player to the parameter value and setting next to None . It also has setNext, getNext, and getValue.

LinkedList Class

The LinkedList class has a private variable head and an __init__ setting it to None, it has methods for

- addHead(value) – adds a new link containing value, returns nothing
- removeHead() – removes the first link in the list, raising index_error if the list is empty
- getHead() – returns the Player value in the first link, raising index_error if the list is empty
- findValue(value) – returns True if there is a link containing a Player equal to value, else False

Helper Functions

None required.

Main functionality

The provided driver has tests for Person, Player and Characters, Player Overloaded operators, and List. You should run them in that order to make sure that your classes work properly.