Advanced Topics in Comp Sci — Project—Wardrobe

ASSIGNMENT OVERVIEW

In this culminating assignment, we'll pull together a little bit of everything we know about Python at this point: conditionals, loops, lists, and object-oriented programming.

BACKGROUND

A common application for a computer program is to manage a collection of items:

- An `AddressBook` manages a collection of `Contacts`.
- A `CardDeck` manages a collection of `Cards`.
- A `Bank` manages a collection of `BankAccounts`.
- A `CheckersGame` manages a `Board` and a collection of `Checkers`.

In this project you'll be writing a `Wardrobe` program that manages a collection of `Clothing` objects, some of which will be one of two different types of subclasses of the `Clothing` class.

PROGRAM SPECIFICATION

Create a Python program `wardrobe_runner.py` that will keep track of and manipulate a list of `Clothing` objects. Your final project will consist of six files(!) contained in a folder called `wardrobe`. Those six files will include:

1. The main program `wardrobe_runner.py` that will interact with the `Wardrobe` class.
2. A file called `wardrobe.py` that contains a description of the `Wardrobe` class which will manage your collection of `Clothing` objects.
3. A file called `clothing.py` that contains a description of the `Clothing` class, which will be imported by classes or programs that need it.
4. A file called `shirt.py` that contains a description of the `Shirt` class, a subclass that inherits from the `Clothing` class.
5. A file that contains a second class description that inherits from `Clothing`. The specifics of this subclass will be determined by you. (A `Socks` class? A `Pants` class? A `Jacket` class? Something else?)
6. A file `README.txt` that documents your Wardrobe project in detail.

These files, contained in the directory `wardrobe`, will be zipped together and uploaded to the server as `wardrobe.zip`.

DELIVERABLES

`wardrobe.zip`

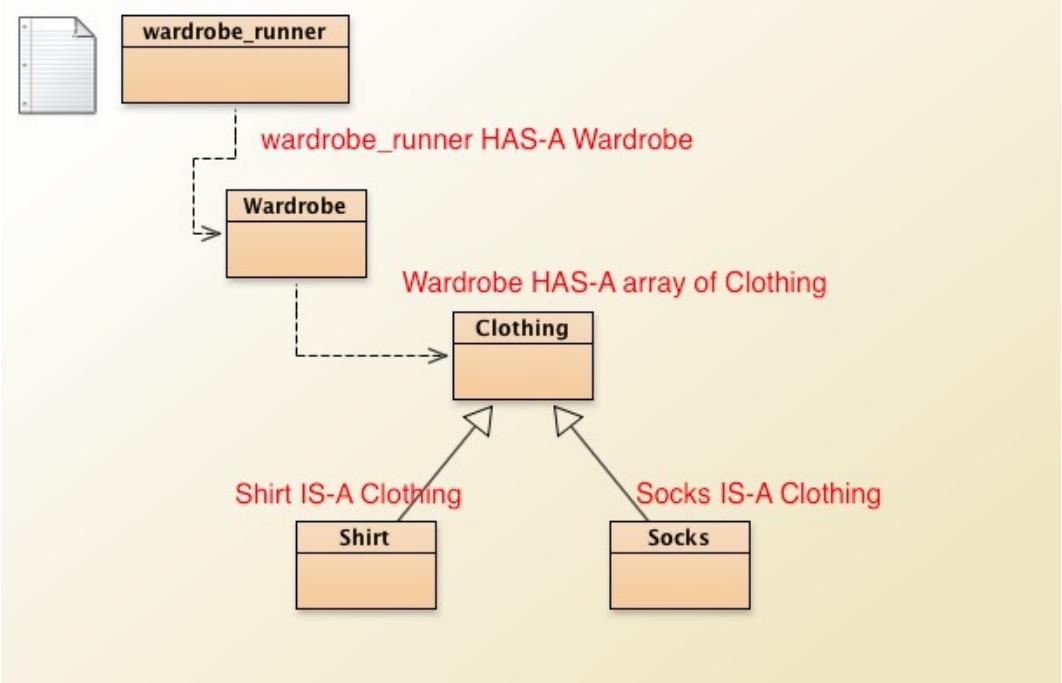
This zipped directory will contain six files as outlined in the specification above. You may use variables as you see fit in your classes, but the methods for each required class are given here:

1. `Clothing` class
   a. Instance variables `name`, `color`, `max_wears`, `times_worn`
   b. Methods `get_name()`, `get_color()`, `wear()`, `is_dirty()`, `wash()`, `__repr__()`
2. `Shirt` class (inherits from `Clothing`)
   a. Boolean instance variables `shortsleeves` plus those inherited from `Clothing`
   b. Methods `has_short_sleeves()`, plus those inherited from `Clothing`
3. `Wardrobe` class (manages a list of `Clothing` items)
   a. Instance variables `clothes` (the list)
   b. Methods `add(item), find(name), remove(item), get_all(), get_by_color(color), get_clean(), get_dirty, wear(item), do_laundry()`

To submit your assignment for grading, copy `wardrobe.zip` to your directory in `/home/studentID/forInstructor/` at *crashwhite.polytechnic.org* before the deadline.

ASSIGNMENT NOTES

- Part of the challenge in this assignment is figuring out where to begin. Do you want to start with the `wardrobe_runner.py` program and get some initial output on the screen before you start writing classes? Would you prefer to begin with the superclass `Clothing` and get that figured out before writing the main? Do you want to draw a diagram showing how all the pieces fit together?



- When writing classes, it's a good idea to write down what instance variables and methods you think the class should have, and then write the class, all by itself in its own file. Then, in an adjacent window, start up Python in interactive mode, import the module, and start testing it out interactively. Switch back and forth between the two windows, with both of them open so that you can quickly scan both source code and run results. (See next page for example.)

  You can make adjustments to the Class file as you write it, but you'll have to restart interactive mode every time you want to load a new version of your class.

- Now is the time to get into the habit of using two windows on screen, side-by-side. Figure out how to use hot-keys to jump back and forth between the two windows, and try to reduce your use of the trackpad/mouse. This project will be a good one for helping you to get more efficient in your programming/debugging.

```python
#!/usr/bin/env python3
"""
clothing.py
Describes the Clothing class, which represents an article of clothing.
"""
__author__  = "Richard White"
__version__ = "2023-03-21"

class Clothing(object):
    """Describes an article of clothing by it name, color, how many
    times it can be worn before being considered dirty, and how many
    times it has been worn since washing.
    """
    def __init__(self, name, color, max_wears):
        self.name = name
        self.color = color
        self.max_wears = max_wears
        self.times_worn = 0
    def get_name(self):
        return self.name
    def get_color(self):
        return self.color
    def wear(self):
        self.times_worn += 1
    def is_dirty(self):
        return self.times_worn >= self.max_wears
    def wash(self):
        self.times_worn = 0
    def __repr__(self):
        return super().__repr__() + "[name=" + self.name \
                                  + ",color=" + self.color \
                                  + ",max_wears=" + str(self.max_wears) \
                                  + ",times_worn=" + str(self.times_worn) + "]"
```

```
(base) rwhite@VingtMille Desktop % python
Python 3.9.7 (default, Sep 16 2021, 08:50:36)
[Clang 10.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license
" for more information.
>>> from clothing import *
>>> blue_jeans = Clothing("jeans","blue",3)
>>> blue_jeans.get_name()
'jeans'
>>> blue_jeans.get_color()
'blue'
>>> blue_jeans.__repr__()
'<clothing.Clothing object at 0x7faef80f2340>[n
ame=jeans,color=blue,max_wears=3,times_worn=0]'
>>> blue_jeans.wear()
>>> blue_jeans.wear()
>>> blue_jeans.wear()
>>> blue_jeans.is_dirty()
True
>>> blue_jeans.wash()
>>> blue_jeans.is_dirty()
False
>>> _
```

- In writing the wardrobe project, at some point you'll need to check to see what type of object you're working with. If you have a list of `Clothing` in your wardrobe and you want to print out all your short-sleeved shirts, for example, you can't call the `has_short_sleeves` method on a non-`Shirt`. To check whether or not you can call `has_short_sleeves` on a piece of clothing, you can use the `isinstance` operation:

```
if isinstance(wardrobe[i], Shirt):
    if wardrobe[i].has_short_sleeves():
        print(wardrobe[i])
```

  The `isinstance` operation checks the first parameter (`wardrobe[i]` in this case) against the second parameter, an `Object` type. If the first parameter's type is the same as the second parameter, `True` is returned. Otherwise, `False` is returned.

- A final version of the `wardrobe_runner.py` program might very well be interactive, allowing the user options to enter `Clothing` items, and giving the user the option to choose which capability of the `Wardrobe` class to use. Testing user interactions, and having to enter test data every time you run your program, takes a lot of time. Avoid having your program take any user input for this activity. The `wardrobe_runner.py` program will be responsible for creating `Clothing` items, `Shirt` items, `Socks` items (or whatever you've chosen for you project), etc.

GETTING STARTED

1. Decide which strategy listed above you think you'd like to take in working on this project. Will you start with a drawing? Outlining classes on the whiteboard? Writing the start of your tester program? Writing a superclass?

2. Find someone else in the class that you think you might like to work alongside and share ideas with.

Oftentimes, in discussing your work with others, you'll identify some problem that needs solving, a problem that you wouldn't otherwise have thought of.

3. It makes sense to write a superclass before you write subclasses that inherit from it. Write the Clothing class first and test it thoroughly, interactively or in a program, before you move on to the subclasses. Testing interactively is fine for a few first tests, but ultimately, you want to be able to conduct a bunch of tests quickly from a main program, so make sure you get that written before too much time has passed.

4. Make sure you check with the instructor if you start to run into difficulties. Although some aspects of the project have been specified in this document, there may be additional design decisions that we'll have to take a look at.

5. Reference documents containing working code may be available upon request.

QUESTIONS FOR YOU TO CONSIDER (NOT HAND IN)

1. At this stage in your programming development, how often do you use a mouse to navigate your computer, your windows, or your text editor? Do you feel the mouse speeds up your work, or slows it down?

2. There are differences between the statements

```
import Clothing;
```

...and...

```
from Clothing import *;
```

Which still of import do you prefer, and why? What are the advantages and disadvantages of your preference?

SAMPLE INTERACTIONS

The following printout from a wardrobe_runner.py is just a short example. Your own Runner will test a somewhat larger collection of clothing items.

```
========= Testing the Wardrobe class =============
Creating pants, socks, shirt items...done
--------------
Creating wardrobe object...done
--------------
Adding items to wardrobe...done
--------------
Getting list of all items in wardrobe
0. Clothing[name=jeans,color=blue,max_wears=3,times_worn=0]
1. Socks[Clothing[name=lucky,color=white,max_wears=1,times_worn=0][paired=True]]
2. Shirt[Clothing[name=t-shirt,color=white,max_wears=2,times_worn=0][shortsleeves=True]]
3. Clothing[name=sweater,color=black,max_wears=10,times_worn=0]
--------------
Getting list of white items in wardrobe
0. Socks[Clothing[name=lucky,color=white,max_wears=1,times_worn=0][paired=True]]
1. Shirt[Clothing[name=t-shirt,color=white,max_wears=2,times_worn=0][shortsleeves=True]]
--------------
Wearing all items twice...done
--------------
Getting list of dirty items
0. Socks[Clothing[name=lucky,color=white,max_wears=1,times_worn=2][paired=True]]
1. Shirt[Clothing[name=t-shirt,color=white,max_wears=2,times_worn=2][shortsleeves=True]]
--------------
Washing dirty clothes...Oh, no! We lost a sock!
done
--------------
Showing all items in wardrobe
0. Clothing[name=jeans,color=blue,max_wears=3,times_worn=2]
1. Socks[Clothing[name=lucky,color=white,max_wears=1,times_worn=0][paired=False]]
2. Shirt[Clothing[name=t-shirt,color=white,max_wears=2,times_worn=0][shortsleeves=True]]
3. Clothing[name=sweater,color=black,max_wears=10,times_worn=2]
--------------
Removing the socks if unpaired...done
--------------
Showing all items in wardrobe
0. Clothing[name=jeans,color=blue,max_wears=3,times_worn=2]
1. Shirt[Clothing[name=t-shirt,color=white,max_wears=2,times_worn=0][shortsleeves=True]]
2. Clothing[name=sweater,color=black,max_wears=10,times_worn=2]
```