# CSC121 Lab 11: Classes and OO Programming

## Goals

In this lab assignment, students will demonstrate the ability to:

* Create and use classes to store data attributes
* Create and use classes and methods which work with other classes and data types

NOTE: This lab document is a copyrighted work of Wake Tech Community College and the course instructor. Any posting of this document outside of Wake Tech is considered a copyright violation. Students who post these documents outside of Wake Tech are subject to academic and possible legal actions.

## Instructions

In this lab, you will demonstrate your understanding of classes and object-oriented programming.

Follow the instructions in each problem and submit the specified files.

Problems 1 will consist of program files that you create from scratch that meets the problem specification. Problem 2 will reuse the class module from Problem 1 and will also provide a starter file that you will use to implement the problem specification.

## Problems

### Problem 1

This program demonstrates your ability to implement classes and create objects from those classes.

You will create two files, one for your class and one for your main module:

**inventory\_item.py**

* Create a file named **inventory\_item.py** to hold a new class for defining inventory items for Trish's bookstore.
* After the module comment header, define a named constant called CATEGORY\_LIST as a list containing the strings "Book", "DVD", and "Game".
* Define a class named InventoryItem. This class will contain 4 **public** attributes: name, count, cost and category.
* In this class, the \_ \_init\_ \_ method should accept 4 arguments for the item name (str), item count (int), item cost (float), and item category (string).
* The class should have a get\_item\_input method that will ask the user for an item name, item count, item cost, and category, and then set the object attributes to those values. The count, cost, and category should be validated as we've done in previous labs.
  + HINT: Code implemented in previous labs can certainly be reused here.
* The class must also have a \_ \_str\_ \_ method that will return a string that describes the object. If an object gets printed directly, it should have a display similar to this:

Science Book

Count: 10, Cost: 12.95

Category: Book

The following class diagram shows the design of this class:

| **InventoryItem** |
| --- |
| +name: String  +count: Int  +cost: Float  +category: String |
| +\_ \_init\_ \_(name:String, count:Int, cost:Float, category:String)  +get\_item\_input()  +\_ \_str\_ \_() |

**Lab11P1.py**

* Create a file named **Lab11P1.py**. This file will be the main module that holds the main program. The purpose of this file is to test the implementation of the class.
* In the main module, write 3 lines of code that creates three InventoryItem objects with the following information:

| Item Name | Item Count | Unit Cost | Category |
| --- | --- | --- | --- |
| Python for All | 10 | 12.95 | Book |
| Barbie | 15 | 6.95 | DVD |
| Uno | 32 | 4.50 | Game |

HINTS:

* + The program will be using 3 assignment statements.
  + Each assignment statement will reference the class.
  + The values from the table should be passed in when each object is created so the three objects represent the data from the table.
* Print the 3 objects that were created by printing the objects **directly**. That means to **not** explicitly call \_ \_str\_ \_() but to just print the object.
* After printing the 3 objects from the table, create one additional InventoryItem object with no fixed data.
* For this additional InventoryItem object, use the method you created to ask the user for the item's information.
* After creating this object from the user input, print that object directly.

Sample output:

Python for All

Count: 10, Cost: 12.95

Book

Barbie

Count: 15, Cost: 6.95

DVD

Uno

Count: 32, Cost: 4.50

Game

Enter the item name: Advanced Python

Enter the item count: 15

Enter the unit cost: 23.90

Enter the category: Book

Advanced Python

Count: 15, Cost: 23.90

Book

Submit the program file **Lab11P1.py** and **inventory\_item.py** to Blackboard for credit.

### Problem 2

In this problem, you will reuse the InventoryItem class you created in Problem 1 to create an inventory collecting program.

You have been given a starter file which needs to be updated. With your updates, you will have a program that can collect inventory information and save it to a binary file. That file is loaded when the program starts if the file is present.

The program has TODO comments which indicate where you need to make changes:

* In the main() function, there is a while loop in which the program asks for inventory item data from the user and stores those items in a list. Implement the part which creates the InventoryItem object, asks the user for input, and then appends the object to the inventory list.
* In the load\_inventory() function, attempt to open a binary file named "inventory.dat".
  + If the file exists, load it into the inventory list and return that list to the calling routine.
  + If the file doesn't exist, just return an empty list ot the calling routine.
* In the save\_inventory()function, open a binary file named inventory.dat and dump the inventory list that was passed as a parameter to the function.
* In the display\_category()function, display each InventoryItem object that is in that category, or display "No items." if there are no items in that category.
* In the display\_inventory()function, display each InventoryItem object that is in the list, or display "Inventory is empty." if the list is empty.

Once all the TODO sections are completed, the program should be able to maintain the list of inventory items between executions of the program.

Sample Output – First run

Current Inventory

-----------------

Inventory is empty.

-----------------

Enter the item name: Python for Beginners

Enter the item count: 12

Enter the unit cost: 3.59

Enter the category: Book

Do you want to enter more items? y

Enter the item name: Advanced Python

Enter the item count: 15

Enter the unit cost: 12.35

Enter the category: Book

Do you want to enter more items? y

Enter the item name: Pythonopoly

Enter the item count: 8

Enter the unit cost: 8.90

Enter the category: Game

Do you want to enter more items? n

Items in Book

-------------

Python for Beginners

Count: 12, Cost: 3.59

Book

Advanced Python

Count: 15, Cost: 12.35

Book

Items in DVD

------------

No items.

Items in Game

-------------

Pythonopoly

Count: 8, Cost: 8.90

Game

Current Inventory

-----------------

Python for Beginners

Count: 12, Cost: 3.59

Book

Advanced Python

Count: 15, Cost: 12.35

Book

Pythonopoly

Count: 8, Cost: 8.90

Game

-----------------

Inventory was saved in inventory.dat.

Sample Output – Second run

Current Inventory

-----------------

Python for Beginners

Count: 12, Cost: 3.59

Book

Advanced Python

Count: 15, Cost: 12.35

Book

Pythonopoly

Count: 8, Cost: 8.90

Game

-----------------

Enter the item name: Python 2: The Revenge

Enter the item count: 7

Enter the unit cost: 3.10

Enter the category: DVD

Do you want to enter more items? n

Items in Book

-------------

Python for Beginners

Count: 12, Cost: 3.59

Book

Advanced Python

Count: 15, Cost: 12.35

Book

Items in DVD

------------

Python 2: The Revenge

Count: 7, Cost: 3.10

DVD

Items in Game

-------------

Pythonopoly

Count: 8, Cost: 8.90

Game

Current Inventory

-----------------

Python for Beginners

Count: 12, Cost: 3.59

Book

Advanced Python

Count: 15, Cost: 12.35

Book

Pythonopoly

Count: 8, Cost: 8.90

Game

Python 2: The Revenge

Count: 7, Cost: 3.10

DVD

-----------------

Inventory was saved in inventory.dat.

Submit the program file **Lab11P2.py** to Blackboard for credit.

## Grading Rubric

### Grading rubric for Problem 1 (50 points)

* Program has a well-formatted and correct header [5 points]
* Program does execute correctly and produces correct results [45 points]

### Grading rubric for Problem 2 (50 points)

* Program has a well-formatted and correct header [5 points]
* Program does execute correctly and produces correct results [45 points]