# CSC121 Lab 12: Inheritance and Polymorphism

## Goals

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

* Create and use superclasses and subclasses
* Exploit polymorphism of methods in derived classes

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 further demonstrate your understanding of classes and object-oriented programming by exploring inheritance and polymorphism.

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

Problems 1 is a program with class files that you create from scratch that meets the problem specification. Problem 2 reuses the class files from Problem 1 in which you complete an implementation begun in a starter file.

## Problems

### Problem 1

In this problem, we're going to extend the InventoryItem class so that it supports the three specific types of items in Trish's Swap Shop: Books, DVDs, and Games.

You will be provided the base class, InventoryItem, in a file you can download from Blackboard called **inventory\_item.py**. You will create 3 subclass files for each of the item categories.

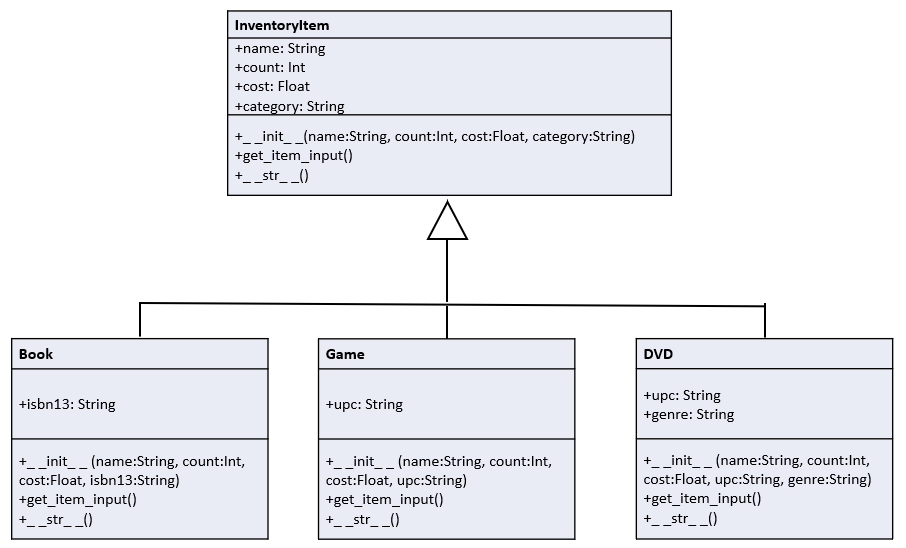
In Trish's swap shop, every item she sells has an item name, item count, unit price, and a category.

For books, she wants to be able to track the ISBN13 code. That is a 13 digit code used by barcode readers.

For games, she wants to track the UPC Code. The UPC code is a 12 digit code.

For DVDs, she wants to track the UPC Code and specify a genre for the DVD. The genre will be a string.

As the software developer working on her inventory system, you've decided on the following object-oriented design for these items:



You will create six files, one for your superclass (provided to you on Blackboard), one for each of your subclasses, and two for two different main modules:

**inventory\_item.py**

* **This file is provided to you**. This class is nearly identical to the one you created for Lab 11 Problem 1.
* Notice that in the get\_item\_input() method, the user is NOT asked to enter a category. That will get filled in by the subclasses when the specific object is created.

**book.py**

* **A starter version of this file is provided** called book-starter.py. Rename that to **book.py.**
* The **book.py** file defines a class named Book that inherits from InventoryItem. The Book class has one additional public instance variable, a string called isbn13.
* **The \_\_init\_\_ method is defined for you.** It takes four default arguments, three from InventoryItem and one additional one named isbn13.
  + The superclass \_ \_init\_ \_ method is called to initialize the superclass instance variables. When the superclass dunder init is called, it sets the category to "Book".
  + After that, the isbn13 instance variable is initialized.
* Implement the provided get\_item\_input method. This method will override the method in the superclass. This method will get user input to fill in all the Book objects attributes.
  + You should call the superclass's get\_item\_input method to fill in the name, count, and cost instance variables first before writing additional code to fill in the isbn13 instance variable.
  + Make sure to include data validation to ensure for isbn13 the user enters a string with 13 characters that is all digits.
* The \_\_str\_\_ method is provided for you. It will return a string that includes this object's data. When a Book object is printed, it should appear like this:

Science Book

Count: 100, Cost: 22.95

Category: Book

ISBN: 2345234523451

* Wherever possible, your program should call the super() function to access methods from the base class to avoid copying code from the base class module.

**game.py**

* Create a file named **game.py** and define a class named Game that inherits from InventoryItem. The Game class has one additional public instance variables, a string called upc.
* Just as you did with the Book class, override the superclass \_ \_init\_ \_, get\_item\_input, and \_ \_str\_ \_ methods to perform appropriate actions for this class.
  + Make sure the \_ \_init\_ \_ method sets the category for this object to be "Game".
  + Make sure your get\_item\_input method does data validation to ensure the UPC entered by the user is a string with 12 digits.
* If this object is created directly, here's what it should look like:

Monopoly

Count: 50, Cost: 12.95

Category: Game

UPC: 098765432121

* Wherever possible, your program should call the super() function to access methods from the base class to avoid copying code from the base class module.

**dvd.py**

* Create a file named **dvd.py** and define a class named DVD that inherits from InventoryItem. The DVD class has two additional public instance variables, a string called upc and a string called genre.
* Perform similar actions here as you did with the Game class. The category should be set to "DVD".
* If this object is created directly, here's what it should look like:

Fifth Element

Count: 30, Cost: 6.95

Category: DVD

UPC: 098765432121

Genre: Sci Fi

To test the work you did for creating subclasses, you are going to create **TWO** different main programs.

**Lab12P1a.py**

* Create a file named **Lab12P1a.py**. This file will be the main module that holds the main program.
* In the main module, create 5 objects with the following information:

| **Object Type** | **Item Name** | **Item Count** | **Unit Cost** | **ISBN13** | **UPC** | **Genre** |
| --- | --- | --- | --- | --- | --- | --- |
| Book | Python Now | 100 | 22.95 | 2345234523451 |  |  |
| Book | Even More Python | 150 | 8.95 | 3456345634561 |  |  |
| Game | Uno | 75 | 6.95 |  | 123456789012 |  |
| DVD | Barbie | 50 | 12.00 |  | 098765432121 | Comedy |
| DVD | The Piano | 10 | 2.90 |  | 321321321321 | Drama |

* After creating these 5 objects, print the objects directly.
  + In our class, "print directly" means do NOT call \_ \_str\_ \_() in the code, but just print the object, and Python will automatically call \_ \_str\_ \_() when converting the object to a string.

Sample output for Lab12P1a.py:

Python Now

Count: 100, Cost: 22.95

Category: Book

ISBN: 2345234523451

Even More Python

Count: 150, Cost: 8.95

Category: Book

ISBN: 3456345634561

Uno

Count: 75, Cost: 6.95

Category: Game

UPC: 123456789012

Barbie

Count: 50, Cost: 12.00

Category: DVD

UPC: 098765432121

Genre: Comedy

The Piano

Count: 10, Cost: 2.90

Category: DVD

UPC: 321321321321

Genre: Drama

**Lab12P1b.py**

* Create a file named **Lab12P1b.py**. This file will be the main module that holds the main program.
* Create three objects by asking the user for the item's information. You will need to ask whether the user wants to create a book, game, or DVD item.
  + Use the Sample Output as a guide for how you should be asking the user what they want to add to the inventory. You must use 1, 2, 3 as your input for this part of you will lose points.
  + The user may ask for these objects in any order and may not always ask for one of each.
  + Do not ask the user if they want to add more items. Consider using a list to hold the objects and using a for loop for asking for EXACTLY 3 times.
* After creating these 3 objects, print the objects directly.

Sample output for Lab12P1b.py:

What item type (1-Book, 2-Game, 3-DVD)? 1

Enter the item name: Of Mice and Men

Enter the item count: 12

Enter the unit cost: 3.40

What is the ISBN? 998877665544

ISBN must be a 13 digit number.

What is the ISBN? 9988776655443

What item type (1-Book, 2-Game, 3-DVD)? 2

Enter the item name: Monopoly

Enter the item count: 17

Enter the unit cost: 10.00

What is the UPC? 1234567890123

UPC must be a 12 digit number.

What is the UPC? 123456789012

What item type (1-Book, 2-Game, 3-DVD)? 3

Enter the item name: Bambi

Enter the item count: 8

Enter the unit cost: 5.95

What is the UPC? 098765432112

What is the genre of the DVD? Animation

Of Mice and Men

Count: 12, Cost: 3.40

Category: Book

ISBN: 9988776655443

Monopoly

Count: 17, Cost: 10.00

Category: Game

UPC: 123456789012

Bambi

Count: 8, Cost: 5.95

Category: DVD

UPC: 098765432112

Genre: Animation

Submit the program file **Lab12P1a.py**, **Lab12P1b.py**, **inventory\_item.py**, **book.py**, **game.py**, and **dvd.py** to Blackboard for credit.

### Problem 2

For this program, you will be using the 4 class files from Problem 1 in an inventory collection program.

You will use the starter file for this problem, and update the TODO sections:

* 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 that part which creates the appropriate 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. If it doesn't exist, just allow the empty list to be returned.
* In the save\_inventory() function, open a binary file named inventory.dat and dump the inventory list that was passed as a parameter to that file.
* In the display\_inventory() function, display each 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 of different types between executions of the program.

Sample Output – First run

Current Inventory

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

Inventory is empty.

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

What item type (1-Book, 2-Game, 3-DVD)? 1

Enter the item name: Django For Beginners

Enter the item count: 26

Enter the unit cost: 39.00

What is the ISBN? 9781735467207

Do you want to enter more items? y

What item type (1-Book, 2-Game, 3-DVD)? 2

Enter the item name: Pictionary

Enter the item count: 15

Enter the unit cost: 24.95

What is the UPC? 123451234512

Do you want to enter more items? y

What item type (1-Book, 2-Game, 3-DVD)? 3

Enter the item name: Shin Godzilla

Enter the item count: 3

Enter the unit cost: 14.90

What is the UPC? 999999999999

What is the genre of the DVD? Rargh!

Do you want to enter more items? n

Current Inventory

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

Django For Beginners

Count: 26, Cost: 39.00

Category: Book

ISBN: 9781735467207

Pictionary

Count: 15, Cost: 24.95

Category: Game

UPC: 123451234512

Shin Godzilla

Count: 3, Cost: 14.90

Category: DVD

UPC: 999999999999

Genre: Rargh!

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

Inventory.dat file was created.

Sample Output – Second run

Current Inventory

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

Django For Beginners

Count: 26, Cost: 39.00

Category: Book

ISBN: 9781735467207

Pictionary

Count: 15, Cost: 24.95

Category: Game

UPC: 123451234512

Shin Godzilla

Count: 3, Cost: 14.90

Category: DVD

UPC: 999999999999

Genre: Rargh!

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

What item type (1-Book, 2-Game, 3-DVD)? 2

Enter the item name: Go Fish

Enter the item count: 10

Enter the unit cost: 2.00

What is the UPC? 111111111111

Do you want to enter more items? n

Current Inventory

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

Django For Beginners

Count: 26, Cost: 39.00

Category: Book

ISBN: 9781735467207

Pictionary

Count: 15, Cost: 24.95

Category: Game

UPC: 123451234512

Shin Godzilla

Count: 3, Cost: 14.90

Category: DVD

UPC: 999999999999

Genre: Rargh!

Go Fish

Count: 10, Cost: 2.00

Category: Game

UPC: 111111111111

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

Inventory.dat file was created.

Submit the program file **Lab12P2.py** to Blackboard for credit.

## Grading Rubric

### Grading rubric for Problem 1 (60 points)

* Programs have a well-formatted and correct header [5 points]
* Programs execute correctly and produces correct results [55 points]

### Grading rubric for Problem 2 (40 points)

* Program has a well-formatted and correct header [5 points]
* Program does execute correctly and produces correct results [35 points]