CIS4350_Lab03: Arrays of Objects

Points: **100** points.

Objective:

- Learning how to write a simple class.
- Learning how to use a prewritten class. (I will provide a Java class that you need to convert to Swift).
- Understanding how object oriented design can aid program design by making it easier to incorporate independently designed pieces of code together into a single application.

Instructions:

- Create a project called **`Lab03`** that contains three files called **`Book.swift`** , **`Bookstore.swift`** and **`TempleBookstore.swift`**
- I've provided a screenshot of how your project should look like.
- Be sure to document your code (add comments on top of your swift files).
- In the comments add your name, date, course, homework number, and statement of problem.
- Make sure your output matches mine (I attached a demo file for your convenience).
- Once you are done, Zip **`Lab03`** project and upload through Blackboard.

Description:

In this assignment, you will essentially write a console application that runs a simple bookstore.
Three classes will be used in this homework: **`Book`**, **`Bookstore`** and **`TempleBookstore`**.

1. I've provided half-way written class called **`Bookstore`** that you need to complete (it is posted in Blackboard next to the assignment description). You can add its Swift version it to your project. You will just use the methods in **`Bookstore`** class only. The methods of **`Bookstore`** class are given below.

2. You have to implement a class called **`Book`** as described below. The name of this class MUST be **`Book`** and it MUST contain the methods below. Do not change the names of the methods, because they are accessed from the **`Bookstore`** class.

3. You also have to implement another class called **`TempleBookstore`**. This class will be main class to do testing. In this main method, you will create a **`Bookstore`** object first, and then you will perform certain menu-driven operations using this **`Bookstore`** object and the methods in the **`Bookstore`** class.

Book

The **Book** class needs to written by you. A **Book** object keeps track of information for one particular book that could potentially be stored in a bookstore. Here are the properties for this class:

```
private String title;
private int numOfPages;
private double price;
private int quantity;
```

You have to implement the following methods in your **Book** class:

```
// an Initializer: Takes in the title of the book, the number of pages in the book, the cost of the book and
// the number of copies (quantity) of books and initializes each of the appropriate properties in
//the object.
public Book(theTitle: String, pages:Int, cost:Double, num: Int)

// Returns the title of the Book object the method is called on.
public func getTitle()-> String

// Returns the price of the Book object the method is called on.
public func getPrice()-> Double

// Returns the quantity of the Book object the method is called on.
public func getQuantity()-> Int

// Returns all the information about a Book object as a String. (Add spaces or tabs to make it readable!)
public func toString()-> String

// Decrements the number of copies, by the given amount, for the Book object the method is called on.
public subtractQuantity(amount:Int)

//Increments the number of copies, with the given amount, for the Book object the method is called on.
public addQuantity(amount:Int)
```

Bookstore

You need to use the methods in **`Bookstore`** class in order to create your console application that will "run" a bookstore.
Here are the methods in that class:

```
// an Initializer that creates a new, empty Bookstore object.
public Bookstore()

// Adds a new Book b to the stock of the Bookstore object.
public addNewBook(b:Book)

// Adds quantity number of books to the book already in stock in the Bookstore object with
// the title title. If the book is not in the Bookstore object, nothing is done.
public addBookQuantity(title:String, quantity:Int)

// Returns true if quantity or more copies of a book with the title are contained in the Bookstore object.
public inStock(title:String, quantity:Int)->Bool

// Executes selling quantity number of books from the Bookstore object with the title to the
// buyer. (Note: there is no I/O done in this method, the Bookstore object is changed to reflect
// the sale. The method returns true is the sale was executed successfully, false otherwise.
public sellBook(title:String, quantity:Int)->Bool

// Lists all of the titles of the books in the Bookstore object.
public listTitles()

// Lists all of the information about the books in the Bookstore object.
public listBooks()

// Returns the total gross income of the Bookstore object.
public getIncome()->Double
```

TempleBookstore

The **TempleBookstore** class should be act as a main class. In this class, you will create a **Bookstore** object first, and then you will perform certain menu-driven operations using this **Bookstore** object and the methods in this class. The main method of your **TempleBookstore** class should do the following:

- Create a new **Bookstore** object.
- Prompt the user with the following menu choices:

```
1) Add a book to the stock
2) Sell a book in stock
3) List the titles of all the books in stock (in the Bookstore object)
4) List all the information about the books in stock (in the Bookstore object)
5) Print out the gross income of the bookstore
6) Quit
```

- Execute the choice the user chooses and then continue until they quit.
- What each menu choice should do:

1. *Adding a book*: Ask the user for the title of the book. If it is already in stock, simply ask the user to enter how many extra books to stock, and then do so. If not, ask the user for the rest of the information (number of pages, price, and quantity) and add that book into the stock.
2. *Selling a book*: Ask the user for the title of the book they want to buy. If it is not in stock, print a message to that effect. Otherwise, ask the user how many copies of that book they want to buy. If there are enough copies of the book, make the sale. If not, print out a message explaining why the transaction could not be completed.
3. *List the titles of all the books in stock* (in the **Bookstore** object)
4. *List all the information about the books in stock* (in the **Bookstore** object)
5. *Print out the gross income of the bookstore*
6. *Quit*.

- The End -