CSC 134 Final project

Create an object oriented program that will provide a menu of six books for a customer to order.

- The customer can only order three books of the six possible.
- The customer can order as many of each book as they would like.
- The customer must be in the system (customer.txt) in order to order books, the customer has to enter the customer id to start.
- Once the customer is finished with their order, the program will print an invoice.

See the class structure and sample input and output below. ***You must follow and implement this class structure.***

See sample test program. ***Note that the test program does not supply interaction from the user as your program should.***

Your client/test program should:

1) Ask for customer id
2) Show menu of 6 possible books to order
3) Once user chooses a book to order, ask for the quantity of that book (this is for each order)
4) Allow the user to order maximum of three books (line items)
5) If user has ordered 3 books and/or chooses “quit” automatically print the invoice
6) Exit gracefully

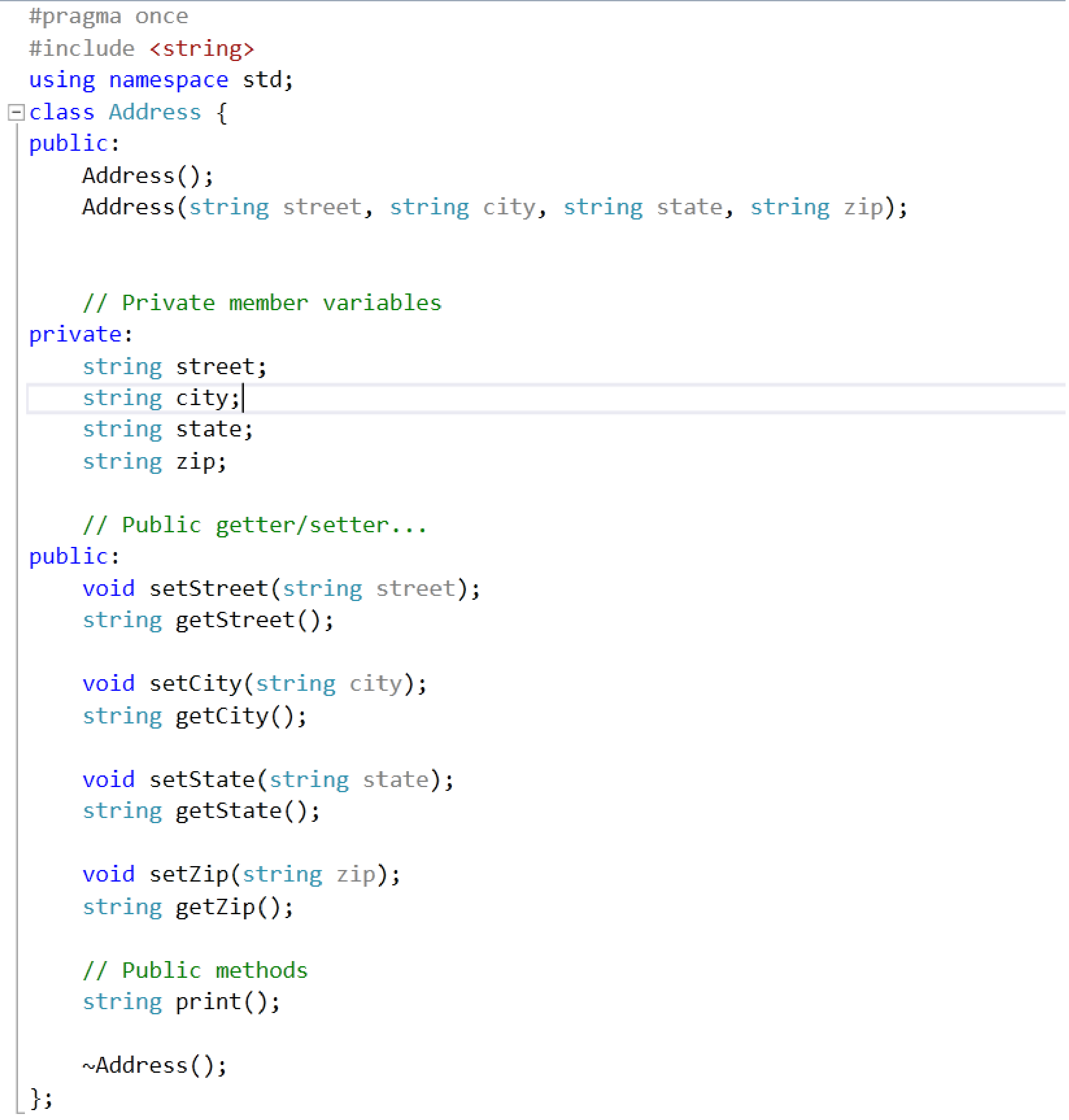
Required Classes:

- Address
- Customer
- Book
- LineItem
- DataStore (Singleton)

Extra Credit Class: (10 points)

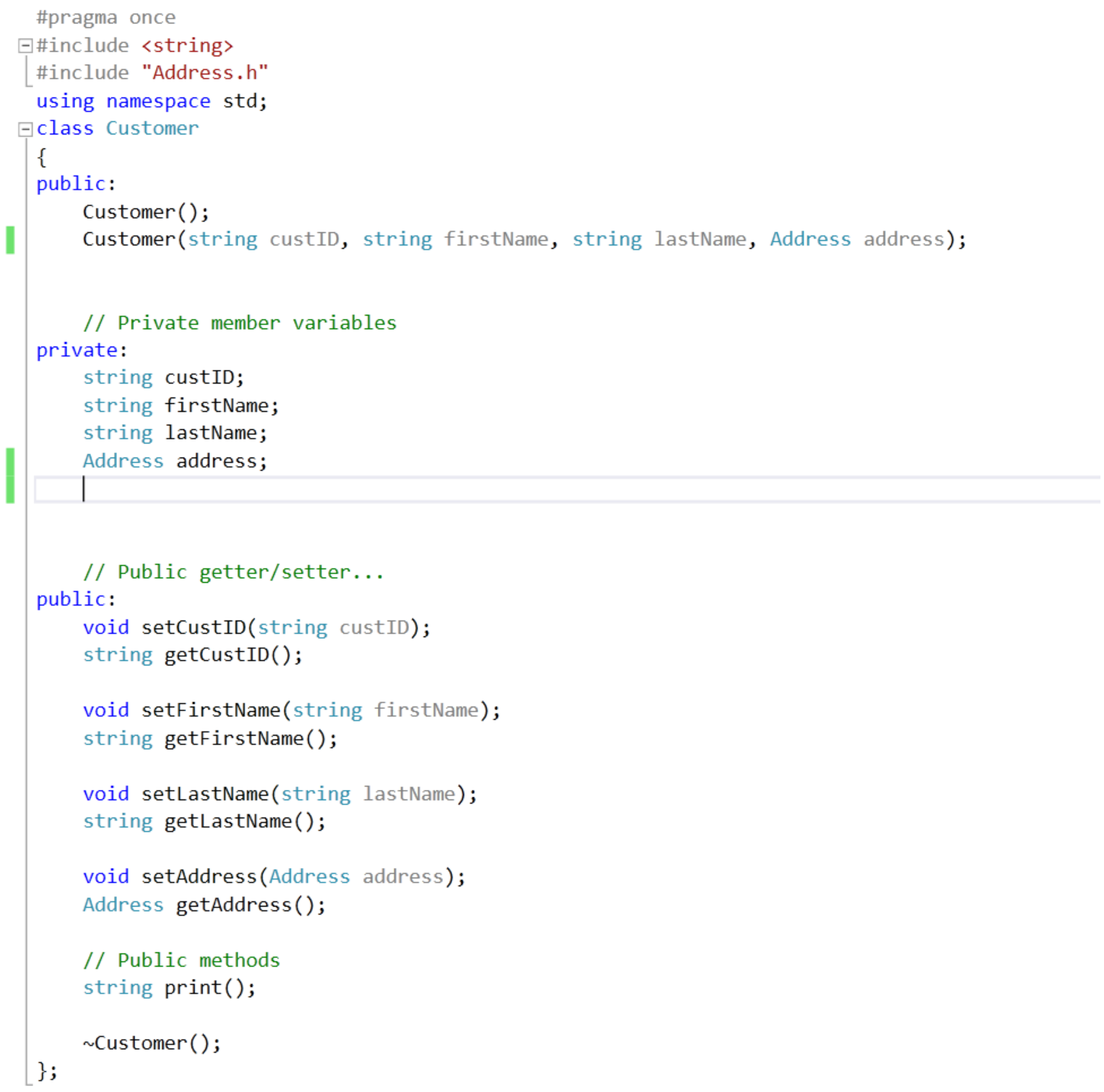
- Invoice

Address Class header (you implement the implementation file – Address.cpp)



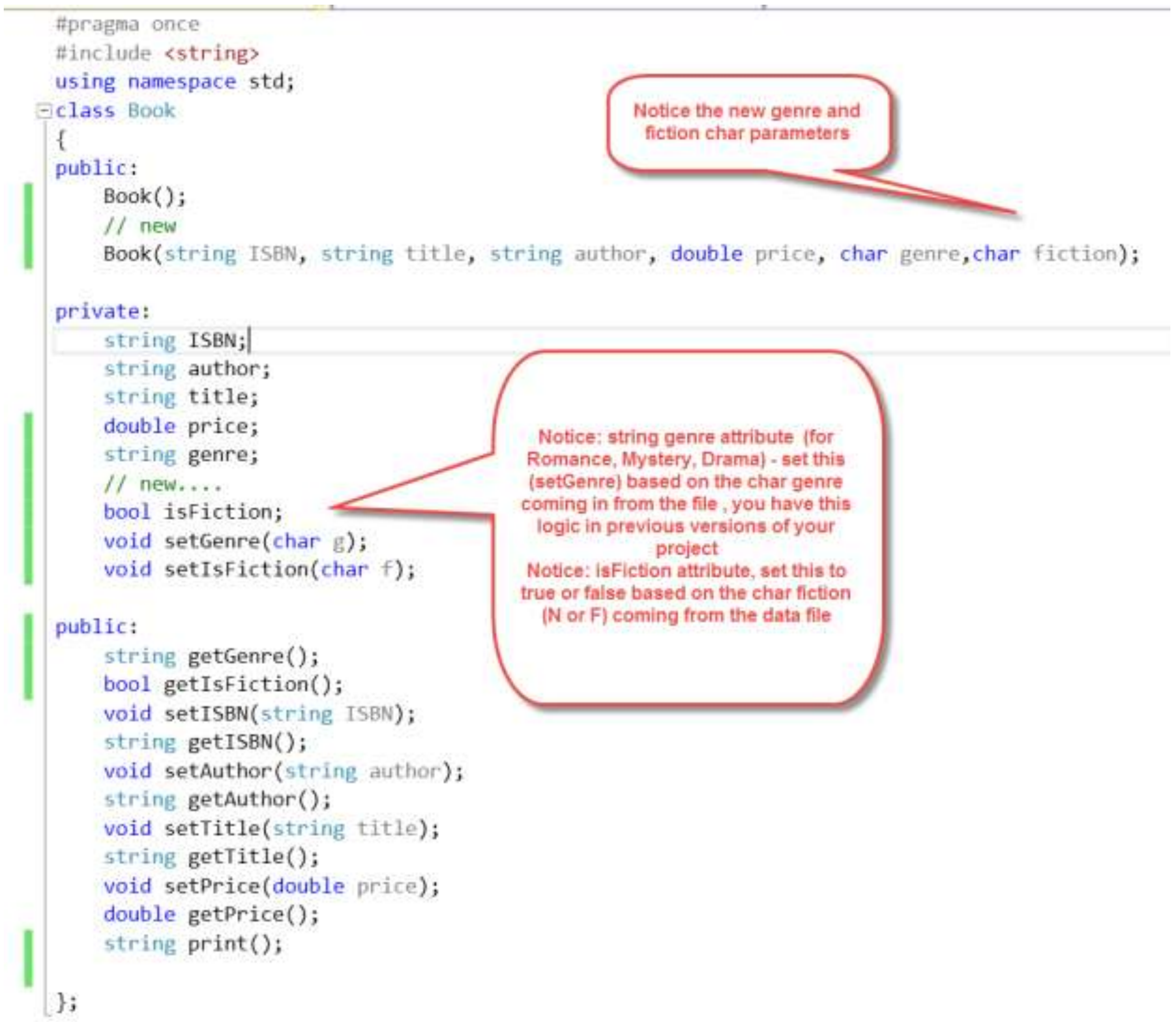
```cpp
#pragma once
#include <string>
using namespace std;
class Address {
public:
    Address();
    Address(string street, string city, string state, string zip);


    // Private member variables
private:
    string street;
    string city;
    string state;
    string zip;

    // Public getter/setter...
public:
    void setStreet(string street);
    string getStreet();

    void setCity(string city);
    string getCity();

    void setState(string state);
    string getState();

    void setZip(string zip);
    string getZip();

    // Public methods
    string print();

    ~Address();
};
```

Customer Class header (you implement the implementation file – Customer.cpp)



```cpp
#pragma once
#include <string>
#include "Address.h"
using namespace std;
class Customer
{
public:
    Customer();
    Customer(string custID, string firstName, string lastName, Address address);


    // Private member variables
private:
    string custID;
    string firstName;
    string lastName;
    Address address;
    


    // Public getter/setter...
public:
    void setCustID(string custID);
    string getCustID();

    void setFirstName(string firstName);
    string getFirstName();

    void setLastName(string lastName);
    string getLastName();

    void setAddress(Address address);
    Address getAddress();

    // Public methods
    string print();

    ~Customer();
};
```

Book Class header (you implement the implementation file – Book.cpp)

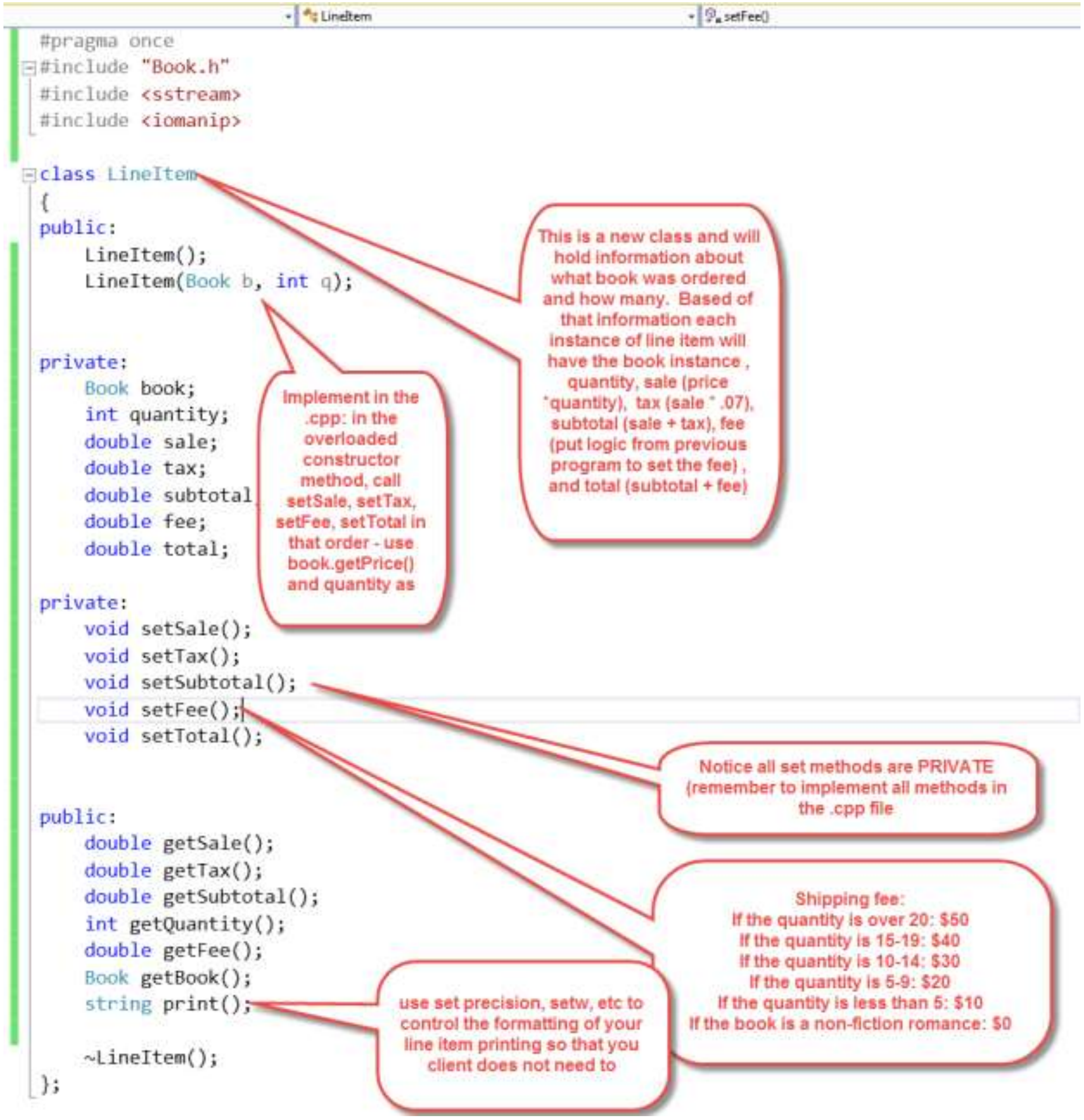


```
#pragma once
#include <string>
using namespace std;
class Book
{
public:
    Book();
    // new
    Book(string ISBN, string title, string author, double price, char genre,char fiction);

private:
    string ISBN;
    string author;
    string title;
    double price;
    string genre;
    // new....
    bool isFiction;
    void setGenre(char g);
    void setIsFiction(char f);

public:
    string getGenre();
    bool getIsFiction();
    void setISBN(string ISBN);
    string getISBN();
    void setAuthor(string author);
    string getAuthor();
    void setTitle(string title);
    string getTitle();
    void setPrice(double price);
    double getPrice();
    string print();

};
```

Notice the new genre and fiction char parameters

Notice: string genre attribute (for Romance, Mystery, Drama) - set this (setGenre) based on the char genre coming in from the file , you have this logic in previous versions of your project
Notice: isFiction attribute, set this to true or false based on the char fiction (N or F) coming from the data file

LineItem Class header (you implement the implementation file – LineItem.cpp) – much of your previous logic for totals, fees, tax, etc will be in the implementation file of this class



```
#pragma once
#include "Book.h"
#include <sstream>
#include <iomanip>

class LineItem
{
public:
    LineItem();
    LineItem(Book b, int q);


private:
    Book book;
    int quantity;
    double sale;
    double tax;
    double subtotal
    double fee;
    double total;

private:
    void setSale();
    void setTax();
    void setSubtotal();
    void setFee();
    void setTotal();


public:
    double getSale();
    double getTax();
    double getSubtotal();
    int getQuantity();
    double getFee();
    Book getBook();
    string print();

    ~LineItem();
};
```

This is a new class and will hold information about what book was ordered and how many. Based of that information each instance of line item will have the book instance, quantity, sale (price *quantity), tax (sale * .07), subtotal (sale + tax), fee (put logic from previous program to set the fee), and total (subtotal + fee)

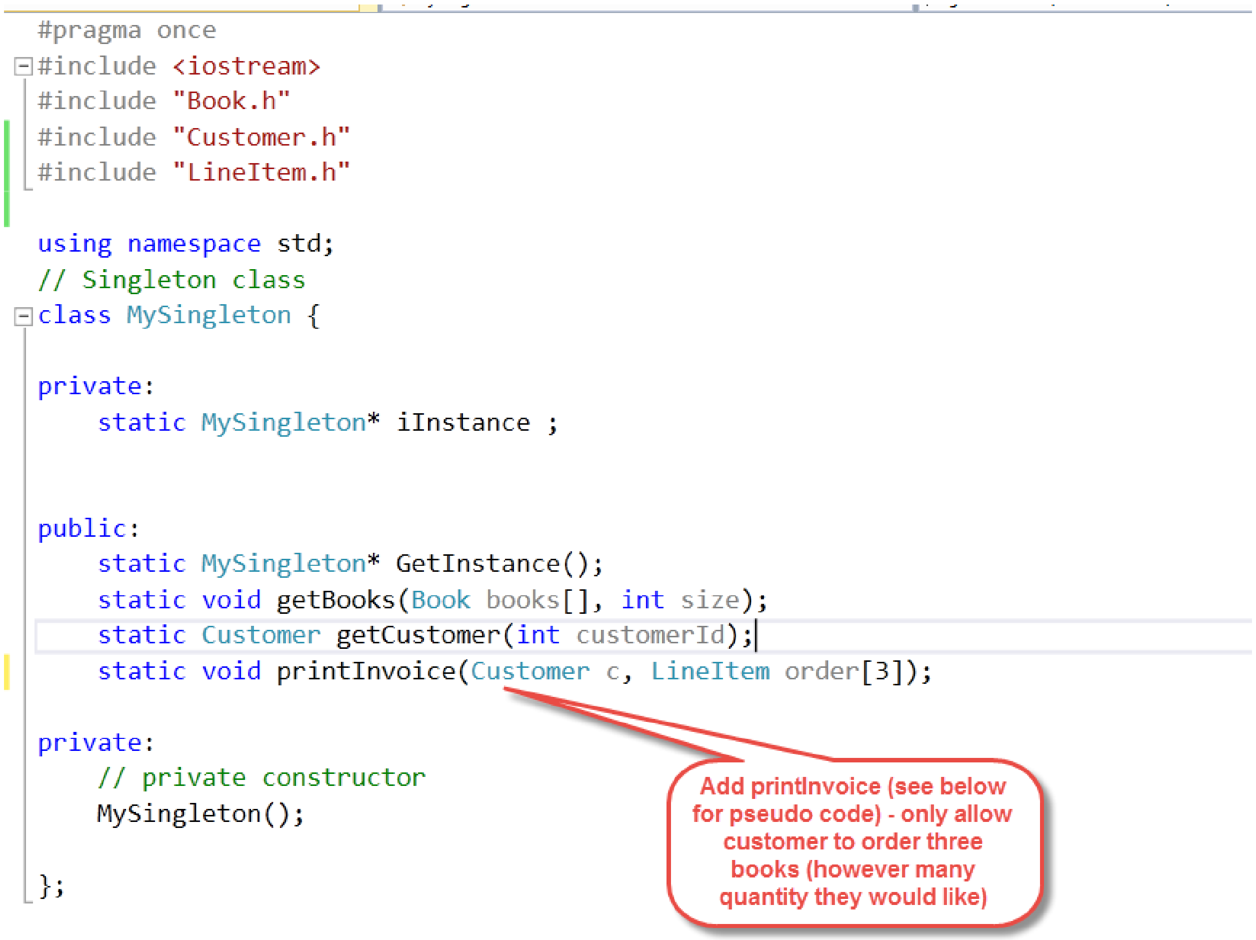
Implement in the .cpp: in the overloaded constructor method, call setSale, setTax, setFee, setTotal in that order - use book.getPrice() and quantity as

Notice all set methods are PRIVATE (remember to implement all methods in the .cpp file

Shipping fee:
If the quantity is over 20: $50
If the quantity is 15-19: $40
If the quantity is 10-14: $30
If the quantity is 5-9: $20
If the quantity is less than 5: $10
If the book is a non-fiction romance: $0

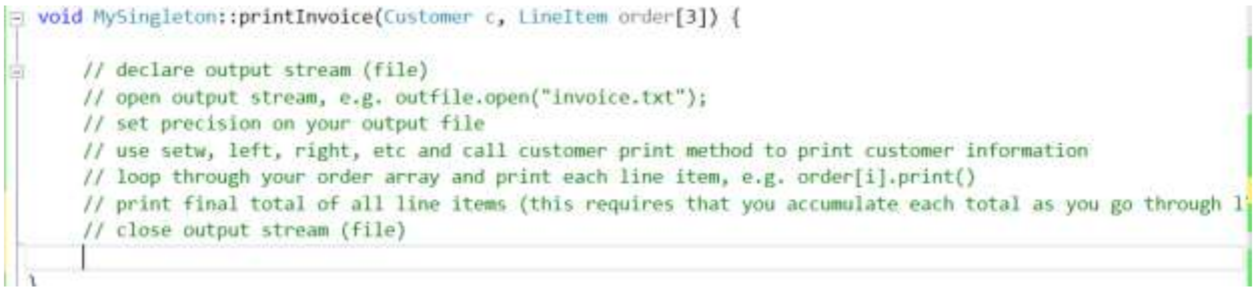
use set precision, setw, etc to control the formatting of your line item printing so that you client does not need to

Singleton (yours should be named ***DataStore***) Class header (you implement the implementation file – DataStore.cpp)



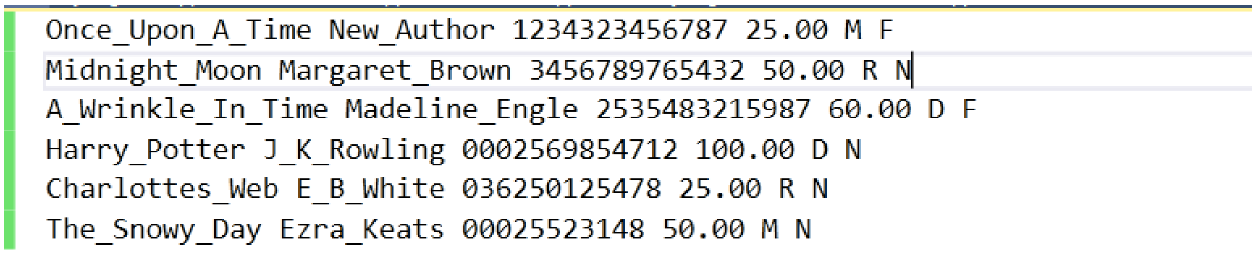
```cpp
#pragma once
#include <iostream>
#include "Book.h"
#include "Customer.h"
#include "LineItem.h"

using namespace std;
// Singleton class
class MySingleton {

private:
    static MySingleton* iInstance ;


public:
    static MySingleton* GetInstance();
    static void getBooks(Book books[], int size);
    static Customer getCustomer(int customerId);
    static void printInvoice(Customer c, LineItem order[3]);

private:
    // private constructor
    MySingleton();

};
```

Add printInvoice (see below for pseudo code) - only allow customer to order three books (however many quantity they would like)



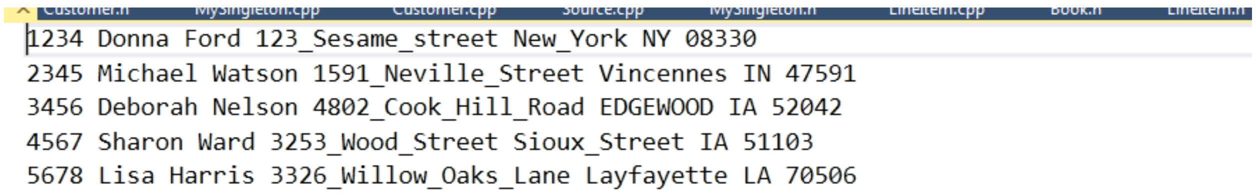
```cpp
void MySingleton::printInvoice(Customer c, LineItem order[3]) {

    // declare output stream (file)
    // open output stream, e.g. outfile.open("invoice.txt");
    // set precision on your output file
    // use setw, left, right, etc and call customer print method to print customer information
    // loop through your order array and print each line item, e.g. order[i].print()
    // print final total of all line items (this requires that you accumulate each total as you go through l
    // close output stream (file)

}
```

Sample Book Input Data (notice the genre and fiction/non-fiction chars to read)



```
Once_Upon_A_Time New_Author 1234323456787 25.00 M F
Midnight_Moon Margaret_Brown 3456789765432 50.00 R N
A_Wrinkle_In_Time Madeline_Engle 2535483215987 60.00 D F
Harry_Potter J_K_Rowling 0002569854712 100.00 D N
Charlottes_Web E_B_White 036250125478 25.00 R N
The_Snowy_Day Ezra_Keats 00025523148 50.00 M N
```

Sample Customer Data (your test program should only ask for customer id, then search via singleton for the customer data)



```
1234 Donna Ford 123_Sesame_street New_York NY 08330
2345 Michael Watson 1591_Neville_Street Vincennes IN 47591
3456 Deborah Nelson 4802_Cook_Hill_Road EDGEWOOD IA 52042
4567 Sharon Ward 3253_Wood_Street Sioux_Street IA 51103
5678 Lisa Harris 3326_Willow_Oaks_Lane Layfayette LA 70506
```

Here is some sample test program code, allow the user to choose 3 books (I've hard coded three books below) – the user will also have to enter quantity for each book order- instantiate the line items and then print the invoice once their order is complete.

```
// declare a book array...
Book books[6];
// send array to get books...
MySingleton::getBooks(books, 6);
// loop through the filled in book array and just show authors
for (Book b : books) {
    cout << b.getAuthor() << endl;
}


Customer c = MySingleton::getCustomer(2345);
cout << c.print();

LineItem line = LineItem(books[2], 3);
LineItem order[3];
order[0] = line;
LineItem line2 = LineItem(books[3], 2);
order[1] = line2;
LineItem line3 = LineItem(books[1], 2);
order[2] = line3;

MySingleton::printInvoice(c, order);
```

Sample Output (Yours should be formatted a little better)

```
Invoice
Customer:___________________________________

Customer ID:                   2345
First Name:                 Michael
Last Name:                   Watson
Address:1591_Neville_Street, Vincennes, IN, 47591

___________________________________

 Madeline_Engle A_Wrinkle_In_Time 60 Drama Fiction

     3@60.00     Total: 180.00

Total Book Sales:       180.00
Tax:                     12.60
Subtotal:               192.60
Extra Fee:               10.00
Total:                  202.60
________________________________________________________________________________
 J_K_Rowling Harry_Potter 100 Drama Non-Fiction


     2@100.00     Total: 200.00

Total Book Sales:       200.00
Tax:                     14.00
Subtotal:               214.00
Extra Fee:               10.00
Total:                  224.00
________________________________________________________________________________
 Margaret_Brown Midnight_Moon 50 Romance Non-Fiction


     2@50.00     Total: 100.00

Total Book Sales:       100.00
Tax:                      7.00
Subtotal:               107.00
Extra Fee:                0.00
Total:                  107.00
________________________________________________________________________________
***** GRAND TOTAL: 533.60
```