CIS 22B
Final Project
Team #1

Members:
Vien Van
Eric Kim
Syed Tihami
Ronald Martin

CIS 22B Project-
Serendipity Bookstore Cashier Project

Requirements Analysis

We were required to create a cashier application for Serendipity Bookstore with the following modules: Cashier, Inventory Database, and Report using many features of object-oriented programming. To do this, we created three subclasses that all inherit from one abstract base class. We created a pure virtual function in the base class called userInteraction which is used in every module to display a list of options to the user and accept a selection. We also defined three classes: Inventory, Book, and Date to facilitate operations. When the program runs, it calls the Control Module (which also inherits from the base class Module), which defines an Inventory object and asks the user which module to go to. The Control Module then uses a base class pointer to call the requested module, implementing polymorphism.

In the Cashier Module, we prompt the user for the name of the book and quantity needed. Once entered, we search for the book in the inventory and check if there is enough quantity. Then we ask the user for the next book he wants to purchase. Then we calculate the subtotal, tax, and total price. Once transaction is done, we decrease the quantity of the books purchased by the necessary amount.

In the Inventory Database Module, we display to the user a list of operations, specifically: Add book, delete book, edit book using the methods defined in the Inventory class.

In the Report Module, we display to the user a list of reports to print. Once the user selects an option, we call a function to print the corresponding report.

When the program closes, it calls the Inventory destructor which instructs the user to enter a file path to output the updated inventory.

Serendipity Books: Psuedocode/Documentation:

Module Class – super class with base classes ControlModule, InventoryDatabaseModule(IDM), ReportModule, and CashierModule

```
Module Constructor
        Initializes Inventory Object to be passed to all base classes
userInteraction
        abstract method to be overridden by all subclasses
```

CashierModule

```
userInteraction
        do while loop to
    •   read in book title and quantity and wantsToContinue
    •   search for title
            o   if title found, add index of title to bookIndices array
            o   check quantity of books
                    ▪   if quantity is less than quantity entered, prompt for new
                        quantity
                    ▪   if quantity is 0 or less, let user know
            o   assign quantity to an array of bookQuantity Array matching
                indexes of bookIndices array
            o   if title not found, let user know
        call printScreen
        call updateInventory
        reset bookQuantity and bookIndices arrays elements to 0
getTotal()
        in a for loop
    •   add running total of quantities in bookQuantity array
    •   calculate price by getting book retail * quantity
    •   return totalPrice
updateInventory()
        in a for loop
    •   subtract quantity from userInteraction from book object's quantity
    •   if quantity is 0 or less, delete book from inventory
printScreen()
        print quantity of book, book isbn, book title, retail price, and total
        print subtotal
        print salestax
        print total
```

ControlModule

```
Constructor initializes
    •   inventory module
    •   cashier module
    •   report module
```

- inventorydatabase module

Deconstructor deletes all modules
userInteraction
    in a do while loop

- provides a menu to enter each module
- each case calls userInteraction in each Module

Date
setDate() initializes day, month, year
Constructor

- takes in string argument,
- checks if string length is 10
    - if not ten, throw error
    - else
        - convert string m, d, y to integer

Call setDate()
Overload operators ==, !=, >, < >=, and <= to compare dates
Overload operators >> and << to output mm/dd/yyyy

<u>Book class</u>
Overloaded constructor takes 9 params and set author, title, isbn, publisher, date, quantity, retailmarkup, and wholesale price to params
Inline accessors and mutators for all variables
setupBook function takes string param

- string param is a line from input file (read from another function)
- split string param with \t and set book variables

friend function operator << to format book output, returns ostream
friend function operator>> to read from file using getline and \t as delimiter, returns istream

<u>Inventory class</u>
Constructor takes 3 param, set filepath, retail markup, and sales tax
Initializes inventory array with book objects
Call pullInventoryfromfile

pullInventoryFromFile()
    open books.txt
read from file in a while loop, check for end of file
call setupbook for each line of the file
close file

addBook()
    takes book object as params
    if the inventory is full, throw full inventory error
    else, dynamically add book to inventory array

```
        increment currentSize

deleteBook()
        check if size of current inventory is 0
        if 0, throw empty inventory error
        else, delete the Book object
        in a for loop, shift the pointers until size – 1
        once done, assign a newly allocated Book to the last pointer

totalWholesale()
        in a for loop, add running total of all wholesale prices
        return running total

searchTitle()
        takes string argument,
        sort inventory by title,
        perform binary search
        return index
sortByTitle()
        perform selection sort based on book.getTitle()
sortByQuantity()
        perform selection sort based on book.quantity()
sortByCost()
        perform selection sort based on cost of book.getWholesale()
sortByDate()
        perform selection sort based on book.getDate()
```

overloaded operator<< to cout each book object

printFile function to print a book to an output file in the same way it is read by the pullInventory function

<u>InventoryDatabaseModule</u>

```
userInteraction()
        in a do while loop
            •   print menu
            •   use switch case to enter menu
addBook()
        calls inventory's addbook function
deletebook
        calls inventory's searchTitle
        calls inventory's deleteBook, passing in index returned from searchTitle
```

<u>ReportModule</u>

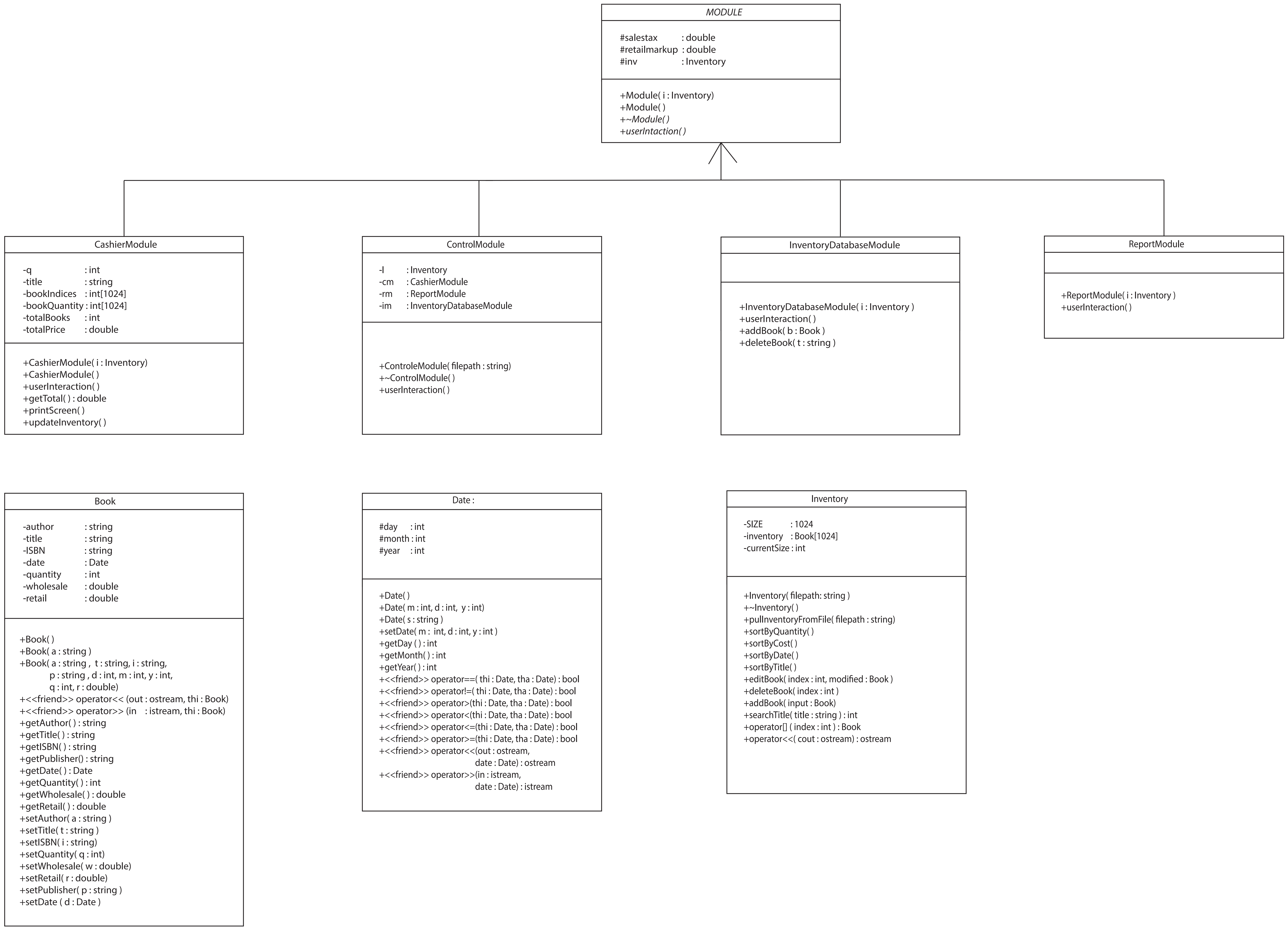
```
userInteraction()
```

in a do while loop

- print menu
- use switch case to enter menu

<u>Main Function (driver)</u>

calls ControlModule


MODULE
#salestax : double
#retailmarkup : double
#inv : Inventory
+Module(i : Inventory)
+Module( )
+~Module( )
+userIntaction( )
CashierModule
-q : int
-title : string
-bookIndices : int[1024]
-bookQuantity : int[1024]
-totalBooks : int
-totalPrice : double
+CashierModule(i : Inventory)
+CashierModule( )
+userInteraction( )
+getTotal( ) : double
+printScreen( )
+updateInventory( )
ControlModule
-I : Inventory
-cm : CashierModule
-rm : ReportModule
-im : InventoryDatabaseModule
+ControleModule(filepath : string)
+~ControlModule( )
+userInteraction( )
InventoryDatabaseModule
+InventoryDatabaseModule(i : Inventory)
+userInteraction( )
+addBook(b : Book)
+deleteBook(t : string)
ReportModule
+ReportModule(i : Inventory)
+userInteraction( )
Book
-author : string
-title : string
-ISBN : string
-date : Date
-quantity : int
-wholesale : double
-retail : double
+Book( )
+Book(a : string)
+Book(a : string , t : string, i : string,
p : string , d : int, m : int, y : int,
q : int, r : double)
+<<friend>> operator<< (out : ostream, thi : Book)
+<<friend>> operator>> (in : istream, thi : Book)
+getAuthor( ) : string
+getTitle( ) : string
+getISBN( ) : string
+getPublisher() : string
+getDate( ) : Date
+getQuantity( ) : int
+getWholesale( ) : double
+getRetail( ) : double
+setAuthor(a : string)
+setTitle(t : string)
+setISBN(i : string)
+setQuantity(q : int)
+setWholesale(w : double)
+setRetail(r : double)
+setPublisher(p : string)
+setDate (d : Date)
Date :
#day : int
#month : int
#year : int
+Date( )
+Date(m : int, d : int, y : int)
+Date(s : string)
+setDate(m : int, d : int, y : int)
+getDay ( ) : int
+getMonth( ) : int
+getYear( ) : int
+<<friend>> operator==(thi : Date, tha : Date) : bool
+<<friend>> operator!=(thi : Date, tha : Date) : bool
+<<friend>> operator>(thi : Date, tha : Date) : bool
+<<friend>> operator<(thi : Date, tha : Date) : bool
+<<friend>> operator<=(thi : Date, tha : Date) : bool
+<<friend>> operator>=(thi : Date, tha : Date) : bool
+<<friend>> operator<<(out : ostream,
date : Date) : ostream
+<<friend>> operator>>(in : istream,
date : Date) : istream
Inventory
-SIZE : 1024
-inventory : Book[1024]
-currentSize : int
+Inventory(filepath: string)
+~Inventory( )
+pullInventoryFromFile(filepath : string)
+sortByQuantity( )
+sortByCost( )
+sortByDate( )
+sortByTitle( )
+editBook(index : int, modified : Book)
+deleteBook(index : int)
+addBook(input : Book)
+searchTitle(title : string) : int
+operator[] (index : int) : Book
+operator<<(cout : ostream) : ostream