CSE 215: Programming Language II

Faculty - Dr. Mohammad Rashedur Rahman (RRn)

Assignment

Fall - 2022

Lab Instructor's Information:

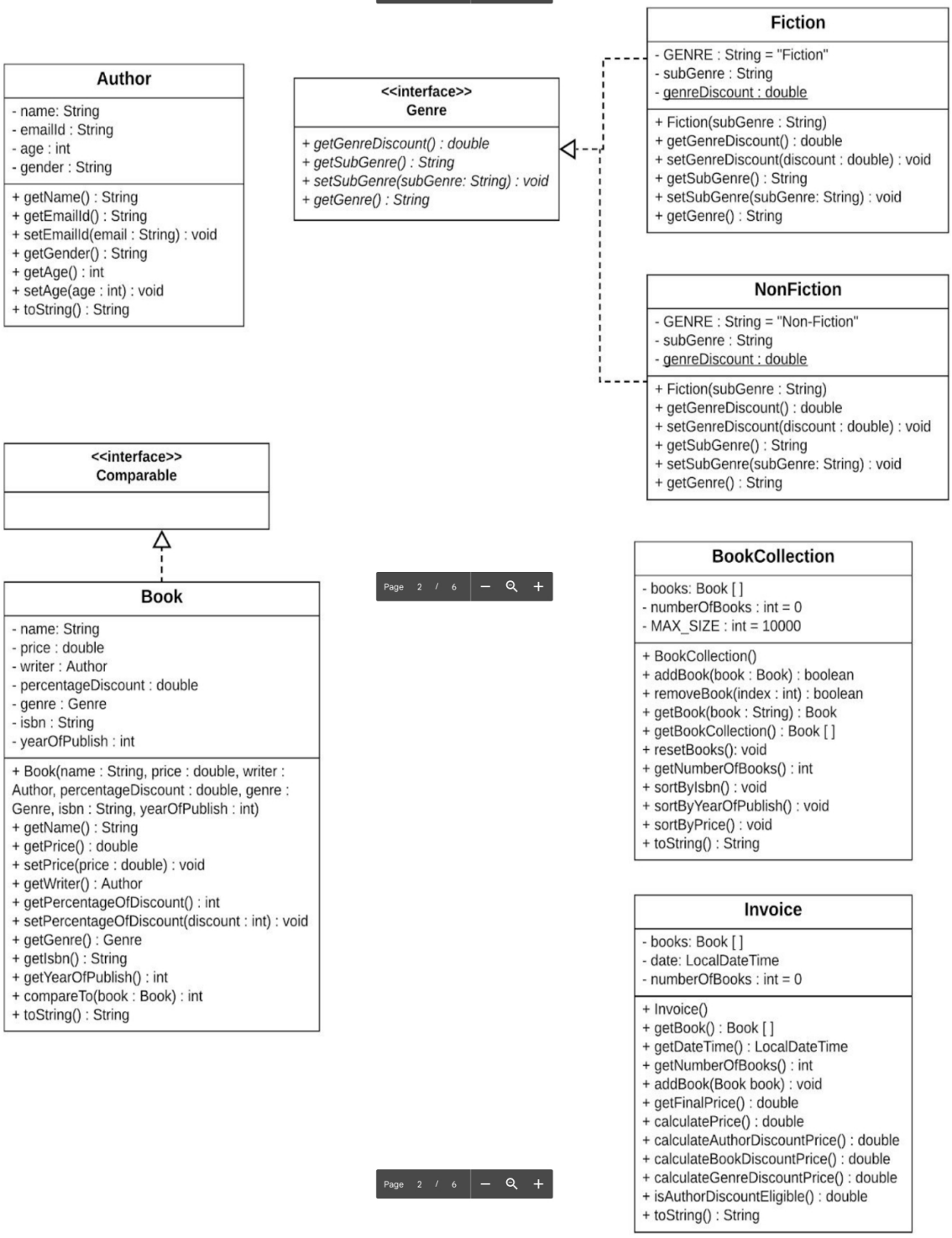
- **Name: Md. Mustafizur Rahman**
- **Email:** rahman.mustafizur@northsouth.edu
- **Office Hours:**
- S - 8:00 AM - 12:00 PM (LIB600-C5)
- T - 8:00 AM - 12:00 PM (LIB600-C5)

General Instructions:

1. Go through the problem, UML diagrams and class descriptions carefully.
2. The solution that you submit must be your own work. You are expected to write the required methods by yourself, that is, without copying anyone else's coded solution. **Plagiarism will lead you to a straight zero.**
3. Maintain the provided architecture. You may add helper methods and classes, but you must be able to explain the need of them. If you can not give a proper explanation, then **marks will be deducted.**
4. If you add any helper class, add it in a different package named helpers.
5. Use proper naming conventions for naming fields. Most of the IDE's and code editors do proper indentation automatically. So, please use it.

Problem:

Suppose you are hired by a company to develop an **inventory management and billing system** for them. The company has categorized their **books** into two main genres - **fiction** and **non-fiction**. As a startup, the company is giving attractive **discounts** on books to gain more customers. They are giving two kinds of discounts - **book-discount** and **genre-discount**. **Book-discount** is given on a book and may vary from book to book. On the other hand, **genre-discount** is given on **genres** of books set by the company. Currently the company gives 9% discount on Fiction and 10% for non-fiction books. There is also another kind of discount named **author-discount** which is, if someone buys 3 or more books by the same author he/she will get 45% discount on the price of those books. When **billing** an **invoice** the company compares between the eligible discounts and give the one which is more than the other. The company wants to keep the books **sorted** according to their **ISBN**. However, you should also keep the option to sort the books by their price and year published if required. **UML Class Diagram** is given below.


Author
- name: String
- emailId : String
- age : int
- gender : String
+ getName() : String
+ getEmailId() : String
+ setEmailId(email : String) : void
+ getGender() : String
+ getAge() : int
+ setAge(age : int) : void
+ toString() : String
<<interface>>
Genre
+ getGenreDiscount() : double
+ getSubGenre() : String
+ setSubGenre(subGenre: String) : void
+ getGenre() : String
Fiction
- GENRE : String = "Fiction"
- subGenre : String
- genreDiscount : double
+ Fiction(subGenre : String)
+ getGenreDiscount() : double
+ setGenreDiscount(discount : double) : void
+ getSubGenre() : String
+ setSubGenre(subGenre: String) : void
+ getGenre() : String
NonFiction
- GENRE : String = "Non-Fiction"
- subGenre : String
- genreDiscount : double
+ Fiction(subGenre : String)
+ getGenreDiscount() : double
+ setGenreDiscount(discount : double) : void
+ getSubGenre() : String
+ setSubGenre(subGenre: String) : void
+ getGenre() : String
<<interface>>
Comparable
Book
- name: String
- price : double
- writer : Author
- percentageDiscount : double
- genre : Genre
- isbn : String
- yearOfPublish : int
+ Book(name : String, price : double, writer : Author, percentageDiscount : double, genre : Genre, isbn : String, yearOfPublish : int)
+ getName() : String
+ getPrice() : double
+ setPrice(price : double) : void
+ getWriter() : Author
+ getPercentageOfDiscount() : int
+ setPercentageOfDiscount(discount : int) : void
+ getGenre() : Genre
+ getIsbn() : String
+ getYearOfPublish() : int
+ compareTo(book : Book) : int
+ toString() : String
BookCollection
- books: Book [ ]
- numberOfBooks : int = 0
- MAX_SIZE : int = 10000
+ BookCollection()
+ addBook(book : Book) : boolean
+ removeBook(index : int) : boolean
+ getBook(book : String) : Book
+ getBookCollection() : Book [ ]
+ resetBooks(): void
+ getNumberOfBooks() : int
+ sortByIsbn() : void
+ sortByYearOfPublish() : void
+ sortByPrice() : void
+ toString() : String
Invoice
- books: Book [ ]
- date: LocalDateTime
- numberOfBooks : int = 0
+ Invoice()
+ getBook() : Book [ ]
+ getDateTime() : LocalDateTime
+ getNumberOfBooks() : int
+ addBook(Book book) : void
+ getFinalPrice() : double
+ calculatePrice() : double
+ calculateAuthorDiscountPrice() : double
+ calculateBookDiscountPrice() : double
+ calculateGenreDiscountPrice() : double
+ isAuthorDiscountEligible() : double
+ toString() : String

Page 2 / 6

Class, Method & Variable Description:

Class: Fiction
This class will implement the Genre Interface

Class: NonFiction
This class will implement the Genre interface

Class: Book	
BookDiscount:double	Variable will store the percentage of discount for a book. **Possible values:** 0 - 100.
CompareTo(Book):int	This is a method of the comparable interface. you have to implement it. It helps you when to want to compare classes with each other. In this case you have to compare the prices. Please develop a meaningful logic. For more information about comparable interface please look here Here the key of comparing will be the isbn number. You have to find a way to compare two Strings lexically. You can write a helper class / method for this purpose.

Class: BookCollection
To find the quantity of a certain book you will have to count the number of book objects.

addBook(Book):boolean	This method will take a Book object as a parameter, add the book into the books array and return a boolean value, depending on whether the operation was successfully done

	or not.
removeBook(int):boolean	This method will take the index of the book you want to remove and remove the book from the bookCollection. It will replace the empty space by shifting the book objects left by one index, starting from index + 1.
getBook(Book):Book	Will search for a book, return the book and remove it from the books array.
printBookInfo(Book) : String	Will search for a book and return a String containing the information of the book.
sortByIsbnBooks():void	This method will sort the books in the books array according to their ISBN. **Hint:** As you implemented the **compareTo():int** method in the **Book** class you can now compare two classes with each other.
sortByYearOfPublish():void	You can have only one implementation of compareTo():int method in a class. So you have to write an algorithm to sort. There are some
resetBooks():void	This method will reset the nextIndex variable to 0.

Class: Invoice

date:LocalDateTime	This variable will store the date & time of the invoice. LocalDateTime is an immutable date-time object that represents a date-time. For more information please look here
getFinalPrice():double	This method will compare the prices after giving the eligible discounts and then return the lowest price. **Hint:** a -> price after authorDiscount (if possible) b -> price after bookDiscount c -> price after genreDiscount

	min(a,b,c)
getDateTime():String	This method will return the date object formatted in "dd-MM-yyyy HH:mm:ss" pattern _____
calculateBookDiscountPrice():double	This method calculates and returns the price after giving the **book discount**, if any.
calculateAuthorDiscountPrice():double	This method calculates and returns the price after giving the **author discount**, if any.
calculateGenreDiscountPrice():double	This method calculates and returns the price after giving the **genre discount**, if any.
ifAuthorDiscountEligable():boolean	This method returns a boolean value, which depends on whether the **author discount** is eligible or not.
toString():String	This method will return a string, which should have a format like this - Date & time in "dd-MM-yyyy HH:mm:ss" in this pattern. The name of books in the invoice - and their price. The total price of the books. The price after discount.

Tasks:

Objects
Objects are given in the following file. **Link:** See the other file **NOTE: You must use the provided objects for inputs, else your assignment will not be graded.**

Task	Expected Output
1. Sort the collection by isbn number and	`The Burning Maze,9123567891201` `The Philosopher's Stone,9123567891202`
print the book names, and isbn	`The Chamber of Secrets,9123567891203`

numbers only.	The Prisoner of Azkaban,9123567891204 The Goblet of Fire,9123567891205 Looking for Alaska,9123567891206
2. Sort the collection by yearOfPublish and print the author name, book name, and yearOfPublish only.	The Philosopher's Stone,JK Rowling,1997 The Chamber of Secrets,JK Rowling,1998 The Prisoner of Azkaban,JK Rowling,1999 The Goblet of Fire,JK Rowling,2000 Looking for Alaska,John Green,2005 The Burning Maze,Rick Riordan,2018
3. Find the writers who are aged between 50 - 55 **(50 and 55 exclusive)**, and print their name and age.	JK Rowling,54 Nicholas Sparks,53
4. Print the name and the price of the cheapest book.	The House of Hades, 150.0
5. Print the name, price and the discount of the book with highest available bookDiscount.	The Green Mile,270.0,11.5 Origin,370.0,11.5 Da Vinci Code,470.0,11.5
6. Print name of Books and their writers, which are of crime subgenre and published after 2010(2010 exclusive).	Origin, Dan Brown
7. Add book object b1, b2, b3, b8 to an invoice and print it using the toString method.	purchase date: 21-11-2019 19:33:22 1. The Philosopher's Stone:550.0 2. The Chamber of Secrets:450.0 3. The Prisoner of Azkaban:330.0 4. The Lightning Thief:450.0 price: 1780.0 price after discount: 1181.5
8. Take Books with **ISBN** number "9123567891201" and "9123567891202" from the collection and add them to a new Invoice object. Print the invoice object and the collection.	purchase date: 21-11-2019 19:33:22 1. The Burning Maze:250.0 2. The Philosopher's Stone:550.0 price: 800.0 price after discount: 716.85 1. A Brief History of Time 2. The Prisoner of Azkaban 3. The Goblet of Fire 4. The Chamber of Secrets