Gerry Lopez

Professor Walauskis

CEN-3024C

18 May 2024

Learning Outcomes | SDLC Assignment Part 1:

Library Management System

Library Management System:

You have been asked to create a simple Library Management System (LMS) software, following the Software Development Life Cycle (SDLC) approach. Your assignment is to write a software development plan for the LMS, based on the description provided below. Your software development plan must include the following information:

Define Requirements:

The Library Management System (LMS) software is intended to be utilized by library staff to manage the library's book collections. The software is intended to add correctly formatted book information from a text file into the library's collection of books. The library staff will use this program to add books to their current collection, remove books from their collection and to display a list of all books currently within the collection.

Library Management System:

System Users:

- Library Saff will use the system to perform tasks to the catalogue of books.

Features and Functionality:

- Add books from a text file to the collection.
- Remove books from collection by book ID.
- Display all books in the current collection.

Constraints:

- Text file records must be properly formatted.

Gather Requirements:

According to library staff, the program should be easy to use. It should have a menu or menu options to choose whether to add, remove, or display books until the user quits out of the program. The library staff would like to be able to input the path for the text file they want to use in case they have text files in multiple locations.

Gathering software requirements:

After speaking with local library staff members, the following requirements were gathered from potential system users. The software requirement desired from for the LMS program are as follows:

- The program must be able to add new books from a text file to the library collection.

  (text file should include a unique ID number, Title, and author)

- The program must be able to remove a book from the collection utilizing the book's ID number.

- The program must be able to list the books in the current collection.

Implementation Plan:

The Library Management System will be developed in stages. The tasks required have been ordered and broken down for implementation as follows:

Breakdown of Tasks:

1. Add books to collection: Develop the addBooksToCollection method for adding books from a text file to the collection.
2. Display books from the collection: Develop the displayBooks method for displaying the books within the collection.
3. Remove books from collection: Develop the removeBooksFromCollection method for removing a book from the text file using the unique book_ID.

Methods required:

- addBookToCollection(Path file) : Reads a file and adds books to the collection
- displayBooks() : Lists all books currently in the collection.
- removeBooksFromCollection(String bookID) : removes the book from the file that corresponds the provided bookID.

Testing Plan:

Testing will be conducted after each method has been successfully written to ensure that the method is fully functioning before moving onto the next method. Once the components have been completed a full system test will be performed by selected users to allow users a chance to give feedback, report bugs or perform unexpected user errors that might break the system.

Tests:

Individual component testing:

- addBookToCollection method will be tested after completion to ensure that the books in the text file are added to the Array List of books in the collection.
- The displayBooks will be tested after the addBookToCollection method has been completed. This should display the list of books that we just added to the collection.
- removeBooksFromCollection method will be tested to ensure that books can be removed using the bookID once the two prior methods have been completed.

Full integrated system testing:

- The entire system will be tested together once all completed components are integrated into the Library Management System to test component functionality.
- Users will be selected and allowed to test the system to ensure the system meets all requirements and ease of use requested, and to allow user errors to occur for further program debugging.

Deploy Software:

Write the code to implement your software and run your tests to ensure it works correctly. Once working, copy/paste your code into your development plan.

Software code will be placed here once completed

```
* Gerry Lopez

* CEN-3024C

* CLASS FUNCTION: This class is intended to store a book object consisting of a the book ID, the book title and book author.

  */


public class Book {


	private String bookID;

	private String bookTitle;

	private String bookAuthor;



	public Book(String bookID, String bookTitle, String bookAuthor) {

		this.bookID = bookID;

		this.bookTitle = bookTitle;

		this.bookAuthor = bookAuthor;
```

```
}
public void setBookID(String bookID) {
	this.bookID = bookID;
}
public void setBookTitle(String bookTitle) {
	this.bookTitle = bookTitle;
}
public void setBookAuthor(String bookAuthor) {
	this.bookAuthor = bookAuthor;
}
public String getBookID() {
	return bookID;
}
public String getBookTitle() {
	return bookTitle;
}
public String getBookAuthor() {
	return bookAuthor;
}

//OVERRIDDEN TOSTRING METHOD FOR DISPLAYING THE BOOK DATA
@Override
public String toString(){
	return "ID#" + bookID + ", Book Title: " + bookTitle + ", Book Author: " + bookAuthor;
```

```
}

}
```

```
/* GERRY LOPEZ

  * CEN-3024C

  * CLASS FUNCTION: THIS CLASS IS INTENDED TO ADD BOOKS TO A COLLECTION OF BOOK OBJECTS IN AN ARRAY LIST,

  * ALLOW THE USER TO REMOVE BOOKS FROM THE COLLECTION AND DISPLAY THE BOOKS IN THE COLLECTION.

  */

import java.io.BufferedInputStream;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.nio.file.Files;

import java.nio.file.Path;

import java.util.ArrayList;

import static java.nio.file.StandardOpenOption.READ;


public class BookCollection {

        private ArrayList<Book> bookCollection;


        public BookCollection() {
```

```
        this.bookCollection = new ArrayList<>();
    }

    /*METHOD FOR ADDING BOOK DATA FROM A TEXT FILE TO THE COLLECTION. THE METHOD USES A PATH PARAMATER TO GET THE TEXT FILE LOCATION OF THE DATA
    * THE METHOD WILL COLLECT THE DATA FROM THE TEXT, SPLIT THE COLLECTED DATA INTO THREE STRINGS AND THEN STORE THE STRING DATA IN A BOOK OBJECT.
    * THE BOOK OBJECT WILL THEN BE STORED TO AN ARRAY LIST OF BOOK OBJECTS TO CREATE THE BOOK COLLECTION.
    */
    public void addBookToCollection(Path file){
        try{
            InputStream input = new BufferedInputStream(Files.newInputStream(file, READ));
            BufferedReader reader = new BufferedReader(new InputStreamReader(input));

            String emptyString = reader.readLine();

            while(emptyString != null){

                String delimit = ",";
                String[] fileStrings = new String[3];
                fileStrings = emptyString.split(delimit);

                if(fileStrings.length == 3){
                    Book book = new Book(fileStrings[0], fileStrings[1], fileStrings[2]);
```

```
                    bookCollection.add(book);



                    emptyString = reader.readLine();
                }
            }
        }catch(IOException exception){
            System.out.println("Error>>>" + exception.getMessage());
        }
    }

    /* METHOD FOR REMOVING A BOOK FROM THE COLLECTION OF BOOKS.
     * THE METHOD TAKES A STRING ARGUMENT AND CHECKS TO SEE IF THE PROVIDED ID STRING MATCHES ONE IN THE BOOK COLLECTION.
     * IF THE ID MATCHES, THE BOOK IS REMOVED FROM TEH COLLECTION AND THE METHOD ENDS. */

    public void removeBooksFromCollection(String id){
        Book removedBook = null;

        for(Book book : bookCollection){
            if(book.getBookID().equals(id)){
                removedBook = book;
                break;
            }
```

```
            }


                if(removedBook != null){
                    bookCollection.remove(removedBook);
                }
        }

        /* METHOD FOR DISPLAYING THE BOOKS CONTAINED WITHIN THE COLLECTION.
          * USES THE OVERRIDDEN TOSTRING CREATED IN THE BOOK OBJECT TO PROVIDE
          * THE DATA FOR THE BOOKS IN THE COLLECTION. */
        public void displayBooks(){
            for(Book book : bookCollection){
                System.out.println(book);
            }
        }
}
```

```
/* GERRY LOPEZ
  * CEN-3024C
  * CLASS FUNCTION: THIS IS THE MAIN CLASS OF THE LMS SYSTEM AND IS INTENDED TO ALLOW THE USER TO SELECT FROM A MENU
  * WHETHER THEY WANT TO ADD, REMOVE, OR DISPLAY BOOKS IN THE COLLECTION. THE MAIN METHOD WILL CONTINUE TO RUN UNTIL
  * THE USER SELECTS THE OPTION TO QUIT.
  */
```

```
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.*;

public class LMS {

	public static void main(String[] args) {

		Scanner keyboard = new Scanner(System.in);
		char userChoice;
		Path file;
		boolean isrunning = true;
		BookCollection collection = new BookCollection();

		while (isrunning) {

			System.out.println(
					"Please select a menu choice: A to add books, R to remove books, D to display books and Q to quit");
			userChoice = keyboard.nextLine().charAt(0);
			userChoice = Character.toUpperCase(userChoice);

			switch (userChoice) {
```

```
                case 'A':
                    System.out.println(
                            "Please provide a file path for the text file. WARNING **File must be in .txt format and lines formatted correctly**");
                    String path = keyboard.nextLine();
                    file = Paths.get(path);
                    collection.addBookToCollection(file);
                    break;

                case 'R':
                    System.out.println("Please provide a book id to remove from the collection. ");
                    String bookID = keyboard.nextLine();

                    collection.removeBooksFromCollection(bookID);
                    break;

                case 'D':
                    collection.displayBooks();
                    break;

                case 'Q':
                    System.out.println("Exiting the program. Have a great day.");
                    isrunning = false;
```

```
                default:
                    System.out.println("Please make a valid choice");
                    break;

            }

        }

        keyboard.close();

    }
}
```