Library System Management

GROUP 7

Whey Jonest, Ogania

Sumbillo, Jessi

Miclat, Joana Marie

Dumangas, Hannizel

Course/Year: BSCS - 1st Year Section C104

Submission Date: 3rd Week of May

Program Description:

The Library System Management is a simple Java program. This code implements a simple console-based Library System Management. It allows users to manage books by performing operations like adding, viewing, borrowing, and returning books, and it keeps track of borrowing history and current borrowing status using files.

Program Requirements:

Your program must include:

- At least one function
- At least one array
- At least one iteration/loop statement
- Proper input and output
- At least one conditional statement

Code Implementation:

The code for the Restaurant Management System is shown below. Ensure the code is properly formatted and commented for clarity.

```
/**************************************************
 * Project Title: Library System Management
 * Author: Hannizel, Dumangas & Joana M. Miclat & Whey Jonest, Ogania & Jessi, Sumbillo
 * Date: 3rd Week of May
 * Description: It is a simple console-based library management system that allows users to
add, view, borrow, and return books while maintaining borrowing history and status using file
storage.
 **************************************************/

import java.util.Scanner;

public class RestaurantManagementSystem {
```

```
import java.io.*;
import java.util.Scanner;

public class LibrarySystem {
    static String[] books = new String[100];
    static boolean[] isBorrowed = new boolean[100];
    static int bookCount = 0;
    static Scanner scanner = new Scanner(System.in);
    static final String FILE_NAME = "library.txt";
    static final String BORROW_HISTORY_FILE = "borrow_history.txt";
    static final String STATUS_FILE = "borrow_status.txt";

    public static void main(String[] args) {
        initializeFileIfNeeded();
        loadBooksFromFile();
        loadBorrowStatus();

        while (true) {
            displayMenu();
            int choice = scanner.nextInt();
            scanner.nextLine();  // consume newline

            switch (choice) {
                case 1 -> addBook();
                case 2 -> viewBooks();
                case 3 -> borrowBook();
```

```
            case 4 -> returnBook();
            case 5 -> viewBorrowHistory();
            case 6 -> {
                System.out.println("Exiting... Goodbye!");
                saveBorrowStatus();
                return;
            }
            default -> System.out.println("Invalid option. Please try again.");
        }
    }
}

public static void displayMenu() {
    System.out.println("\n--- Library Management System ---");
    System.out.println("1. Add Book");
    System.out.println("2. View Available Books");
    System.out.println("3. Borrow Book");
    System.out.println("4. Return Book");
    System.out.println("5. View Borrow History");
    System.out.println("6. Exit");
    System.out.print("Choose an option: ");
}

public static void addBook() {
    System.out.print("Enter book title: ");
    String title = scanner.nextLine();
```

```
    books[bookCount] = title;
    isBorrowed[bookCount] = false;
    bookCount++;

    try (FileWriter fw = new FileWriter(FILE_NAME, true)) {
        fw.write(title + "\n");
    } catch (IOException e) {
        System.out.println("Error writing to file.");
    }

    saveBorrowStatus();
    System.out.println("Book added successfully!");
}

public static void viewBooks() {
    System.out.println("\n--- Available Books ---");
    boolean hasAvailable = false;
    for (int i = 0; i < bookCount; i++) {
        if (!isBorrowed[i]) {
            System.out.println((i + 1) + ". " + books[i]);
            hasAvailable = true;
        }
    }
    if (!hasAvailable) {
        System.out.println("No available books.");
    }
```

```
}

public static void borrowBook() {
    viewBooks();
    System.out.print("\nEnter the number of the book you want to borrow: ");
    int index = scanner.nextInt() - 1;
    scanner.nextLine(); // consume newline

    if (index < 0 || index >= bookCount || isBorrowed[index]) {
        System.out.println("Invalid or already borrowed.");
        return;
    }

    System.out.print("Enter your name: ");
    String name = scanner.nextLine();

    isBorrowed[index] = true;
    String record = name + " borrowed \"" + books[index] + "\"";

    try (FileWriter fw = new FileWriter(BORROW_HISTORY_FILE, true)) {
        fw.write(record + "\n");
    } catch (IOException e) {
        System.out.println("Error writing to history file.");
    }

    saveBorrowStatus();
```

```
        System.out.println("You borrowed: " + books[index]);
    }

    public static void returnBook() {
        System.out.println("\n--- Borrowed Books ---");
        boolean hasBorrowed = false;
        for (int i = 0; i < bookCount; i++) {
            if (isBorrowed[i]) {
                System.out.println((i + 1) + ". " + books[i]);
                hasBorrowed = true;
            }
        }

        if (!hasBorrowed) {
            System.out.println("No books are currently borrowed.");
            return;
        }

        System.out.print("Enter the number of the book to return: ");
        int index = scanner.nextInt() - 1;
        scanner.nextLine(); // consume newline

        if (index < 0 || index >= bookCount || !isBorrowed[index]) {
            System.out.println("Invalid or already returned.");
            return;
        }
```

```
        System.out.print("Enter your name: ");
        String name = scanner.nextLine();

        isBorrowed[index] = false;

        // Log return in history
        String record = name + " returned \"" + books[index] + "\"";
        try (FileWriter fw = new FileWriter(BORROW_HISTORY_FILE, true)) {
            fw.write(record + "\n");
        } catch (IOException e) {
            System.out.println("Error writing return history.");
        }

        saveBorrowStatus();
        System.out.println("Book returned: " + books[index]);
    }

    public static void viewBorrowHistory() {
        System.out.println("\n--- Borrow History ---");
        try (BufferedReader br = new BufferedReader(new FileReader(BORROW_HISTORY_FILE)))
{
            String line;
            boolean hasHistory = false;
            while ((line = br.readLine()) != null) {
                System.out.println(line);
```

```
            hasHistory = true;
        }
        if (!hasHistory) {
            System.out.println("No borrow history found.");
        }
    } catch (IOException e) {
        System.out.println("No borrow history found.");
    }
}

public static void loadBooksFromFile() {
    try (BufferedReader br = new BufferedReader(new FileReader(FILE_NAME))) {
        String line;
        while ((line = br.readLine()) != null && bookCount < books.length) {
            books[bookCount] = line;
            bookCount++;
        }
    } catch (IOException e) {
        // File should exist from initializeFileIfNeeded
    }
}

public static void saveBorrowStatus() {
    try (FileWriter fw = new FileWriter(STATUS_FILE)) {
        for (int i = 0; i < bookCount; i++) {
            fw.write((isBorrowed[i] ? "1" : "0") + "\n");
```

```
            }
        } catch (IOException e) {
            System.out.println("Error saving borrow status.");
        }
    }

    public static void loadBorrowStatus() {
        File file = new File(STATUS_FILE);
        if (!file.exists()) return;

        try (BufferedReader br = new BufferedReader(new FileReader(file))) {
            String line;
            int index = 0;
            while ((line = br.readLine()) != null && index < books.length) {
                isBorrowed[index] = line.equals("1");
                index++;
            }
        } catch (IOException e) {
            System.out.println("Error loading borrow status.");
        }
    }

    public static void initializeFileIfNeeded() {
        File file = new File(FILE_NAME);
        if (!file.exists()) {
            try (FileWriter fw = new FileWriter(FILE_NAME)) {
```

```
            fw.write("Dictionary\n");
            fw.write("To Kill a Mockingbird\n");
            fw.write("The Grapes of Wrath\n");
            fw.write("The Great Gatsby\n");
            fw.write("Jane Eyre\n");
            fw.write("Pride and Prejudice\n");
            fw.write("Wuthering Heights\n");
            fw.write("Anna Karenina\n");
            fw.write("Brave New World\n");
            fw.write("Don Quixote\n");
            fw.write("Alice's Adventures in Wonderland\n");
            System.out.println("Initial book list created.");
        } catch (IOException e) {
            System.out.println("Error initializing book list.");
        }
    }
  }
}
```

Explanation of Components:

Functions: The code uses functions (methods) like addBook(), viewBooks(), and borrowBook() to organize reusable blocks of logic for different library operations.

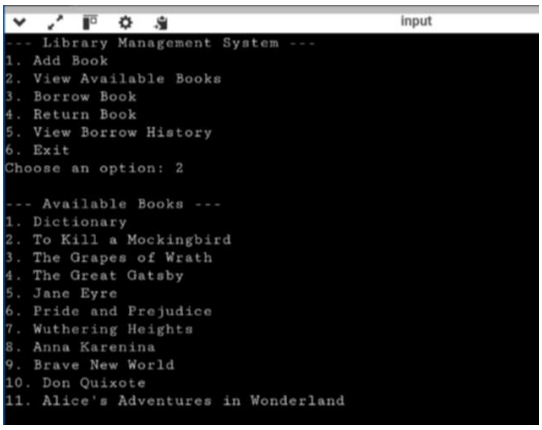
Array: Arrays books[] and isBorrowed[] store the list of book titles and their corresponding availability status.

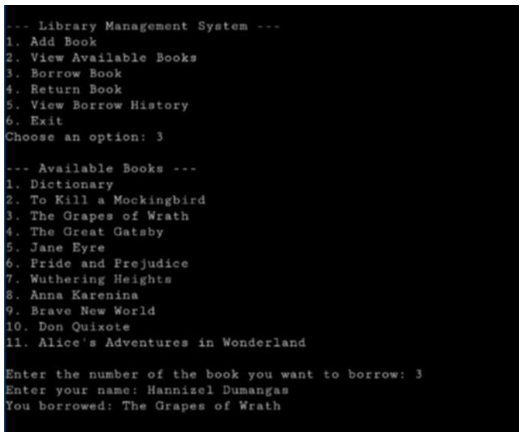
Iteration/Loop Statements: for and while loops are used to repeatedly process book data, display menus, and traverse file contents.

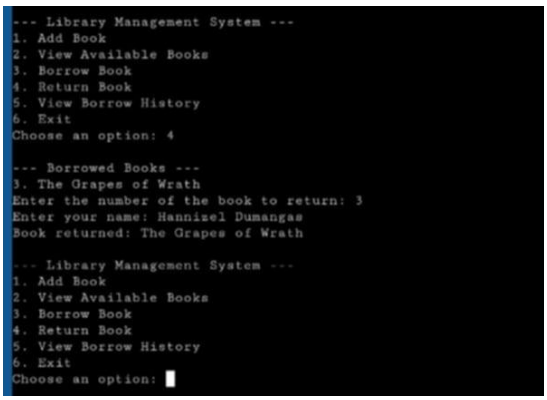
Input/Output: The program uses Scanner for user input and Java I/O classes like FileWriter and BufferedReader to read from and write to files.

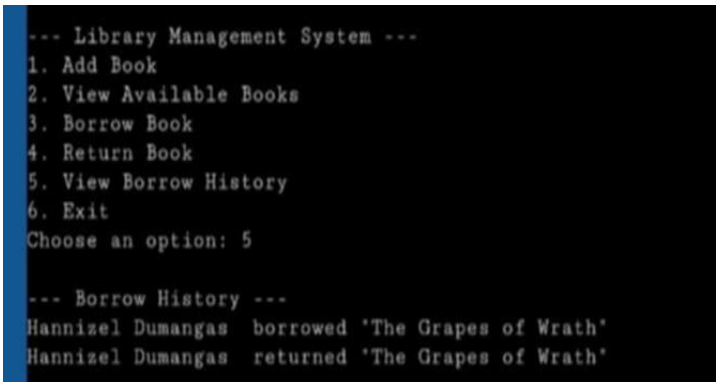
Conditional Statement: if, else if, and switch statements control the flow based on user choices and book availability conditions.

Sample Output:









Reflection:

While we creating this Library Management System, we learned how to effectively use arrays, file handling, user input, and control structures in Java. It improved our understanding of data persistence using files and how to design a simple yet functional menu-driven program. We also realized the importance of validating user input and handling exceptions to make the program more robust and user-friendly.