**Programming Assignment Unit 2**

University of the People

CS 1102-01: Programming 1 AY2025-T2

Naeem Ahmed

29 November 2024

# Programming Assignment Unit 2

## Problem Statement

Write a Java program that accomplishes the following tasks:

**1. Implement the following options:**

* Add Books
* Borrow Books
* Return Books
* Exit

**2.     For "Add Books":**

* Prompt the user to enter the book title, author, and quantity.
* If the book already exists in the library, update the quantity.
* If the book is new, add it to the library.

**3.     For "Borrow Books":**

* Prompt the user to enter the book title and the number of books to borrow.
* Check if the requested number of books is available in the library.
* If the books are available, update the quantity and display a success message.
* If not, display an error message.

**4.     For "Return Books":**

* Prompt the user to enter the book title and the number of books to return.
* Check if the books being returned belong to the library system.
* If they do, update the quantity and display a success message.
* If not, display an error message.

**5.     Handle invalid input and display appropriate error messages.**

**6.     Implement an exit option to allow the user to exit the program.**

**Abstract**

This project presents a Java-based implementation of a basic library system, providing functionalities for adding, borrowing, and returning books. The system features a menu-driven interface where users can interact with the library by selecting options from the menu. The program maintains a record of available book quantities and handles user input validation to ensure smooth operation. Through modular design and error handling mechanisms, the project offers a user-friendly experience for managing library resources efficiently. Overall, this project serves as a foundational framework for developing more advanced library management systems and demonstrates fundamental concepts of Java programming and software engineering principles.

**Solution Code (VSCode Snapshot)**

****

**Solution Code (VSCode Text)**

import java.util.\*;

class Book {

String title;

String author;

int quantity;

public Book(String title, String author, int quantity) {

this.title = title;

this.author = author;

this.quantity = quantity;

}

}

public class LibrarySystem {

private static Map<String, Book> library = new HashMap<>();

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("\nLibrary System Menu:");

System.out.println("1. Add Books");

System.out.println("2. Borrow Books");

System.out.println("3. Return Books");

System.out.println("4. Exit");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

switch (choice) {

case 1:

addBooks(scanner);

break;

case 2:

borrowBooks(scanner);

break;

case 3:

returnBooks(scanner);

break;

case 4:

System.out.println("Exiting program...");

System.exit(0);

break;

default:

System.out.println("Invalid choice. Please enter a number from 1 to 4.");

}

}

}

private static void addBooks(Scanner scanner) {

System.out.print("Enter book title: ");

String title = scanner.next();

System.out.print("Enter author name: ");

String author = scanner.next();

System.out.print("Enter quantity: ");

int quantity = scanner.nextInt();

if (library.containsKey(title)) {

Book existingBook = library.get(title);

existingBook.quantity += quantity;

System.out.println("Quantity updated for book: " + title);

} else {

library.put(title, new Book(title, author, quantity));

System.out.println("Book added to library: " + title);

}

}

private static void borrowBooks(Scanner scanner) {

System.out.print("Enter book title: ");

String title = scanner.next();

System.out.print("Enter quantity to borrow: ");

int quantity = scanner.nextInt();

if (library.containsKey(title)) {

Book book = library.get(title);

if (book.quantity >= quantity) {

book.quantity -= quantity;

System.out.println("Successfully borrowed " + quantity + " copies of " + title);

} else {

System.out.println("Requested quantity not available for book: " + title);

}

} else {

System.out.println("Book not found in library: " + title);

}

}

private static void returnBooks(Scanner scanner) {

System.out.print("Enter book title: ");

String title = scanner.next();

System.out.print("Enter quantity to return: ");

int quantity = scanner.nextInt();

if (library.containsKey(title)) {

Book book = library.get(title);

book.quantity += quantity;

System.out.println("Successfully returned " + quantity + " copies of " + title);

} else {

System.out.println("Book not found in library: " + title);

}

}

}

**Code Explanation**

**1. Book Class:**

import java.util.\*;

class Book {

String title;

String author;

int quantity;

public Book(String title, String author, int quantity) {

this.title = title;

this.author = author;

this.quantity = quantity;

}

}

* First we define a Java class named ‘Book’ and import the java.util package, which contains utility classes such as ‘Scanner’ and ‘HashMap’.
* Then ‘Book’ class represents a book entity with three attributes: ‘title’, ‘author’, and ‘quantity’.
* It has a constructor ‘public Book(String title, String author, int quantity)’ that initializes the attributes of a book object when it's created.
* Inside the constructor, the ‘this’ keyword refers to the current instance of the ‘Book’ class, and it's used to differentiate between the instance variables and the constructor parameters with the same names.
* The ‘import java.util.\*;’ statement imports all classes from the ‘java.util’ package, allowing the program to use utility classes such as ‘Scanner’ and ‘HashMap’ without specifying their full package names.

This code sets up the foundation for representing books in a library system. The Book class encapsulates the properties of a book, making it easy to create and manage book objects within the system.

**2. LibrarySystem Class:**

Here we define a Java class named ‘LibrarySystem’, which serves as the main class for managing the library system.

**Class Declaration:**

public class LibrarySystem {

private static Map<String, Book> library = new HashMap<>();

* This line declares a public class named ‘LibrarySystem’.

**Static Map:**

private static Map<String, Book> library = new HashMap<>();

* This line declares a private static variable named ‘library’, which is a map that associates book titles (strings) with ‘Book’ objects.
* The type of the map is ‘Map<String, Book>’, indicating that keys are strings (book titles) and values are ‘Book’ objects.
* It's initialized as a new instance of ‘HashMap<>’, a type of map that stores key-value pairs in a hash table for efficient retrieval.

This class serves as the central hub for managing the library system. The ‘library’ map stores information about the available books, allowing other parts of the program to interact with it, such as adding new books, borrowing books, and returning books. The use of a static map ensures that the library data is shared across all instances of the ‘LibrarySystem’ class.

**3. Main Method:**

This `main` method serves as the entry point of the program and controls the flow of execution.

**Method Signature:**

public static void main(String[] args) {

* This line declares the `main` method, which is the starting point of execution for a Java program.
* It has a `public` access modifier, indicating that it can be accessed from outside the class.
* It's declared as `static`, meaning it belongs to the class itself rather than to instances of the class.
* The method takes an array of strings (`args`) as a parameter, which allows passing command-line arguments to the program.

**Initialization:**

Scanner scanner = new Scanner(System.in);

* This line creates a new `Scanner` object named `scanner`, which is used to read input from the console (`System.in`).

**Menu Loop:**

while (true) {

* This initiates an infinite loop, ensuring that the menu is continuously displayed until the user chooses to exit.

**Display Menu:**

System.out.println("\nLibrary System Menu:");

System.out.println("1. Add Books");

System.out.println("2. Borrow Books");

System.out.println("3. Return Books");

System.out.println("4. Exit");

System.out.print("Enter your choice: ");

* These lines print the menu options to the console, prompting the user to enter their choice.

**Read User Input:**

int choice = scanner.nextInt();

* This line reads an integer input from the user using the `nextInt()` method of the `Scanner`

class.

**Switch Statement:**

switch (choice) {

case 1:

addBooks(scanner);

break;

case 2:

borrowBooks(scanner);

break;

case 3:

returnBooks(scanner);

break;

case 4:

System.out.println("Exiting program...");

System.exit(0);

break;

default:

System.out.println("Invalid choice. Please enter a number from 1 to 4.");

}

* Based on the user's choice, this switch statement directs the program flow to the corresponding action:
  + - If the user chooses 1, it calls the `addBooks` method.
    - If the user chooses 2, it calls the `borrowBooks` method.
    - If the user chooses 3, it calls the `returnBooks` method.
    - If the user chooses 4, it prints a message indicating the program is exiting and terminates the program using `System.exit(0)`.
    - If the user enters any other choice, it prints an error message indicating invalid input.

The `main` method provides a menu-driven interface for users to interact with the library system, allowing them to perform various actions such as adding, borrowing, and returning books, as well as exiting the program.

**4. Option Methods (`addBooks`, `borrowBooks`, `returnBooks`):**

These are three private static methods: `addBooks`, `borrowBooks`, and `returnBooks`, each responsible for handling a specific action in the library system.

**addBooks Method:**

* This method adds new books to the library or updates the quantity if the book already exists.
* It takes a `Scanner` object as a parameter to read user input.
* It prompts the user to enter the book title, author name, and quantity.
* It checks if the book title already exists in the `library` map using the `containsKey` method.
  + - If the book exists, it retrieves the existing book object from the map, updates its quantity, and prints a message confirming the quantity update.
    - If the book doesn't exist, it creates a new `Book` object with the provided details and adds it to the `library` map, printing a message confirming the book addition.

**borrowBooks Method:**

* This method allows users to borrow books from the library.
* It takes a `Scanner` object as a parameter to read user input.
* It prompts the user to enter the book title and the quantity to borrow.
* It checks if the book title exists in the `library` map.
  + - If the book exists, it retrieves the corresponding `Book` object from the map.
      * If the requested quantity is available (i.e., the `quantity` attribute of the book is greater than or equal to the requested quantity), it decreases the quantity of the book, prints a success message confirming the borrowing, and updates the book's quantity.
      * If the requested quantity is not available, it prints an error message indicating insufficient quantity.
  + If the book doesn't exist in the library, it prints a message indicating that the book is not found.

**returnBooks Method:**

* This method allows users to return books to the library.
* It takes a `Scanner` object as a parameter to read user input.
* It prompts the user to enter the book title and the quantity to return.
* It checks if the book title exists in the `library` map.
  + - If the book exists, it retrieves the corresponding `Book` object from the map, increases the quantity of the book by the returned quantity, and prints a success message confirming the return.
    - If the book doesn't exist in the library, it prints a message indicating that the book is not found.

These methods encapsulate the logic for adding, borrowing, and returning books, providing modular and maintainable code for managing the library system. They interact with the `library` map to perform the necessary operations based on user input.

**5. Error Handling:**

* The program checks for invalid input and displays appropriate error messages, such as when the user enters an invalid menu choice or when a book is not found in the library.

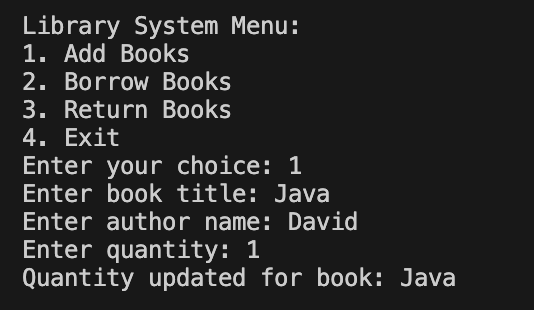
**Summary**

The project implements a basic library system in Java, allowing users to add books to the library, borrow books, and return books. It features a menu-driven interface where users can select options to perform different actions. The program maintains a record of available book quantities and handles user input validation.

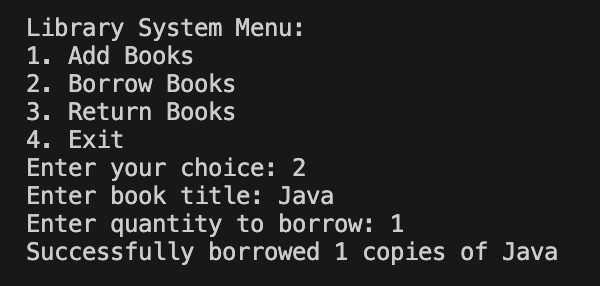
**Key Features:**

1. Users can select options from a menu to add books, borrow books, return books, or exit the program.
2. Books can be added to the library, with the system updating quantities if the book already exists.
3. Users can borrow books from the library if they are available and return books they have borrowed.
4. The program handles invalid input and provides appropriate error messages to the user.
5. The code is organized into separate methods for adding, borrowing, and returning books, enhancing readability and maintainability.

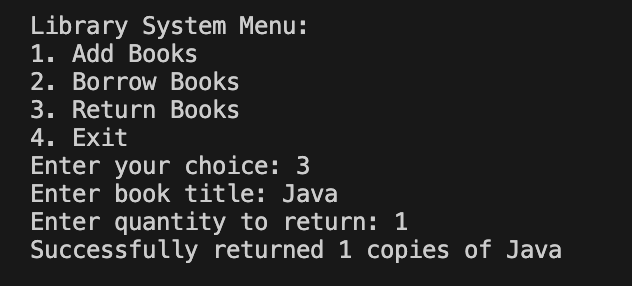
**Output**

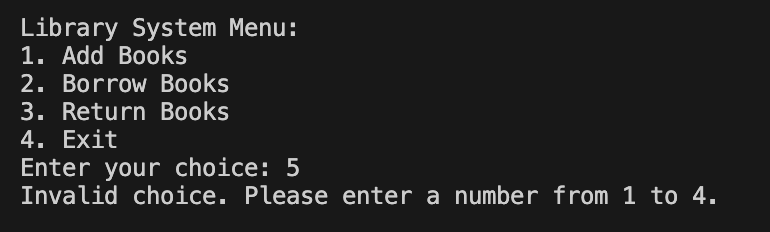
**1. Adding Books**

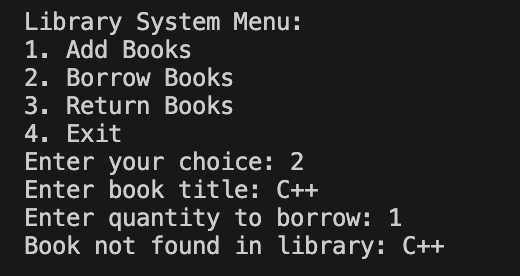
**2. Borrowing Books**

****

**3. Returning Books**

****

**4. Invalid Choice**

**5. Book Not Found**

These examples showcase different scenarios of interacting with the library system, including adding books, borrowing books, returning books, handling invalid choices, and handling cases where the requested book is not found in the library.

**References**

Oracle. *Java Documentation.* Oracle.

*https://docs.oracle.com/en/java/*

Oracle. *Java Collections Framework Tutorial.* Oracle*.*

*https://docs.oracle.com/javase/tutorial/collections/index.html*

Oracle. *Scanner (Java Platform SE 17 & JDK 17).* Oracle.

*https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Scanner.html*

GeeksforGeeks. *Java Programming Tutorials.* GeeksforGeeks.

*https://www.geeksforgeeks.org/java/*

Oracle. *The Java™ Tutorials.* Oracle.

*https://docs.oracle.com/javase/tutorial/*

Udemy*. Java Programming Masterclass for Software Developers.* Udemy.

*https://www.udemy.com/course/java-the-complete-java-developer-course/*

Sierra, K., & Bates, B. (2005)*. Head First Java.* O'Reilly Media.