**1.PROGRAM CODE**

**BOOK CLASS**

public class Book {  
 *// Private member variables for book details* private int bookID;  
 private String title;  
 private String author;  
 private boolean isAvailable; *// Indicates whether the book is available for borrowing*

*// Constructor to initialize a new Book object* public Book(int bookID, String title, String author) {  
 this.bookID = bookID;  
 this.title = title;  
 this.author = author;  
 this.isAvailable = true; *// By default, a new book is available* }  
  
 *// Getter method for book ID* public int getBookID() {  
 return bookID;  
 }  
  
 *// Getter method for title* public String getTitle() {  
 return title;  
 }  
  
 *// Getter method for author* public String getAuthor() {  
 return author;  
 }  
  
 *// Method to check if the book is available* public boolean isAvailable() {  
 return isAvailable;  
 }  
  
 *// Setter method to update the availability status* public void setAvailable(boolean available) {  
 isAvailable = available;  
 }  
  
 *// Returns a string representation of the book details* @Override  
 public String toString() {  
 return "Book ID: " + bookID + ", Title: " + title + ", Author: " + author + ", Available: " + isAvailable;  
 }  
}

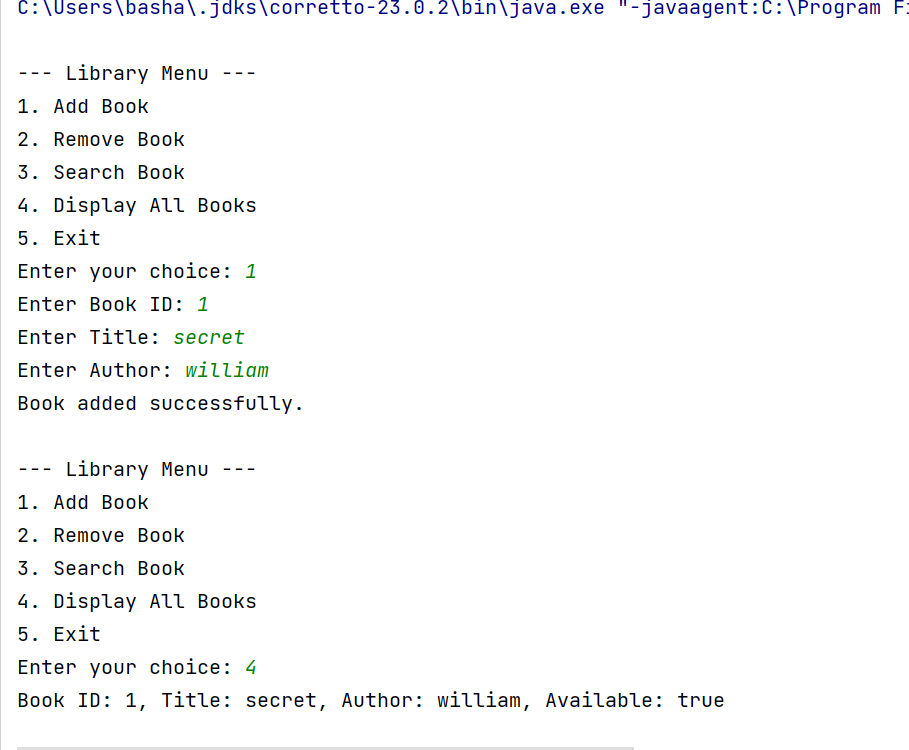
**LIBRARY CLASS**

public class Library {  
 private Book[] books; *// Array to store book objects* private int count; *// Number of books currently in the library  
  
 // Constructor to initialize the library with a given size* public Library(int size) {  
 books = new Book[size]; *// Create a book array with the specified capacity* count = 0; *// Initially, there are no books* }  
  
 *// Method to add a book to the library* public void addBook(Book book) {  
 if (count < books.length) {  
 books[count] = book; *// Add book at current count position* count++; *// Increment count* System.*out*.println("Book added successfully.");  
 } else {  
 System.*out*.println("Library is full. Cannot add more books.");  
 }  
 }  
  
 *// Method to remove a book by its ID* public void removeBook(int bookID) {  
 int index = -1;  
 *// Search for the book with the given ID* for (int i = 0; i < count; i++) {  
 if (books[i].getBookID() == bookID) {  
 index = i; *// Store index of the book to remove* break;  
 }  
 }  
  
 if (index != -1) {  
 *// Shift all books after the found book to the left* for (int i = index; i < count - 1; i++) {  
 books[i] = books[i + 1];  
 }  
 books[count - 1] = null; *// Remove last duplicate book* count--; *// Decrease the count* System.*out*.println("Book removed successfully.");  
 } else {  
 System.*out*.println("Book not found.");  
 }  
 }  
 *// Method to search for a book by its ID* public void searchBook(int bookID) {  
  
 for (int i = 0; i < count; i++) {  
 if (books[i].getBookID() == bookID) {  
 System.*out*.println(books[i]); *// Display the book* }else {  
 System.*out*.println("Book Not Found");  
 }  
 }  
  
  
 }  
  
 *// Method to display all books in the library* public void displayBooks() {  
 if (count == 0) {  
 System.*out*.println("No books in the library.");  
 } else {  
 for (int i = 0; i < count; i++) {  
 System.*out*.println(books[i]);  
 }  
 }  
 }  
}

**BOOKMANAGEMENTSYSTEM CLASS**

*// Main class to run the Book Management System*public class BookManagementSystem {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
  
 *// Create a Library object with capacity for 100 books* Library library = new Library(100);  
 int choice;  
  
 *// Display menu repeatedly until user chooses to exit (choice == 5)* do {  
 *// Display menu options* System.*out*.println("\n--- Library Menu ---");  
 System.*out*.println("1. Add Book");  
 System.*out*.println("2. Remove Book");  
 System.*out*.println("3. Search Book");  
 System.*out*.println("4. Display All Books");  
 System.*out*.println("5. Exit");  
 System.*out*.print("Enter your choice: ");  
  
 *// Read user choice* choice = sc.nextInt();  
  
 *// Perform action based on user choice* switch (choice) {  
 case 1:  
 *// Add a new book* System.*out*.print("Enter Book ID: ");  
 int id = sc.nextInt();  
 sc.nextLine(); *// consume newline left by nextInt()* System.*out*.print("Enter Title: ");  
 String title = sc.nextLine();  
 System.*out*.print("Enter Author: ");  
 String author = sc.nextLine();  
 Book newBook = new Book(id, title, author);  
 library.addBook(newBook); *// Add book to the library* break;  
  
 case 2:  
 *// Remove a book by ID* System.*out*.print("Enter Book ID to remove: ");  
 int removeId = sc.nextInt();  
 library.removeBook(removeId);  
 break;  
  
 case 3:  
 *// Search for a book by ID* System.*out*.print("Enter Book ID to search: ");  
 int searchId = sc.nextInt();  
 library.searchBook(searchId);  
 break;  
  
 case 4:  
 *// Display all books in the library* library.displayBooks();  
 break;  
  
 case 5:  
 *// Exit the system* System.*out*.println("Exiting the system.");  
 break;  
  
 default:  
 *// Handle invalid input* System.*out*.println("Invalid choice. Try again.");  
 }  
  
 } while (choice != 5); *// Repeat until user chooses to exit* sc.close(); *// Close scanner to prevent resource leak* }  
}

**OUTPUT**

****

**2.PROGRAM CODE**

**TAXABLE INTERFACE**

public interface Taxable {  
 double *salesTax* = 0.07;  
 double *incomeTax* = 0.105;

*// Abstract method to calculate tax* void calcTax();  
}

**EMPLOYEE CLASS**

public class Employee implements Taxable {  
  
 *// Instance variables to store employee details* int empid;  
 double salary;  
 String name;  
  
 *// Constructor to initialize the Employee object* Employee(int empid, String name, double salary) {  
 this.empid = empid;  
 this.name = name;  
 this.salary = salary;  
 }  
  
 *// Implementation of the calcTax() method from the Taxable interface* @Override  
 public void calcTax() {  
 *// Calculate income tax based on annual salary* double tax = (salary \* 12) \* *incomeTax*;  
 System.*out*.println("Income Tax for " + name + " is: ₹" + tax);  
 }  
}

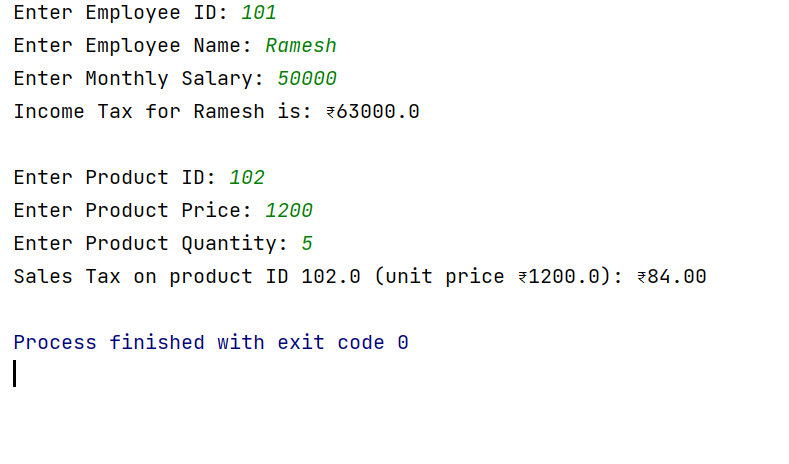
**PRODUCT CLASS**

public class Product implements Taxable {  
  
 *// Instance variables for product details* double price; *// Unit price of the product* double pid; *// Product ID* double quantity; *// Quantity of the product  
  
 // Constructor to initialize product details* public Product(int pid, double price, int quantity) {  
 this.pid = pid;  
 this.price = price;  
 this.quantity = quantity;  
 }  
  
 *// Implementation of calcTax() method from the Taxable interface* @Override  
 public void calcTax() {  
 *// Calculate sales tax on the unit price* double tax = price \* *salesTax*;  
  
 *// Display formatted tax result* System.*out*.printf("Sales Tax on product ID " + pid + " (unit price ₹" + price + "): ₹");  
 System.*out*.printf("%.2f\n", tax);  
 }  
}

**DRIVERMAIN CLASS**

import java.util.Scanner;  
public class DriverMain {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
  
 *// Input for Employee* System.*out*.print("Enter Employee ID: ");  
 int empId = sc.nextInt();  
 sc.nextLine(); *// Consume newline* System.*out*.print("Enter Employee Name: ");  
 String name = sc.nextLine();  
  
 System.*out*.print("Enter Monthly Salary: ");  
 double salary = sc.nextDouble();  
  
 Employee emp = new Employee(empId, name, salary);  
 emp.calcTax();  
  
 *// Input for Product* System.*out*.print("\nEnter Product ID: ");  
 int pid = sc.nextInt();  
  
 System.*out*.print("Enter Product Price: ");  
 double price = sc.nextDouble();  
  
 System.*out*.print("Enter Product Quantity: ");  
 int quantity = sc.nextInt();  
  
 Product prod = new Product(pid, price, quantity);  
 prod.calcTax();  
  
 sc.close();  
 }  
}

**OUTPUT**

****