## **Program Code**

import java.util.\*;

/\*\*

\* Represents a Book in the library catalog.

\*/

class Book {

private String title;

private String author;

private int itemID;

/\*\*

\* Constructs a Book object with the specified title, author, and item ID.

\*

\* @param title The title of the book.

\* @param author The author of the book.

\* @param itemID The ID of the book.

\*/

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

this.title = title;

this.author = author;

this.itemID = itemID;

}

// Getter methods for the title, author, and item ID

public String getTitle() {

return title;

}

public String getAuthor() {

return author;

}

public int getItemID() {

return itemID;

}

}

/\*\*

\* Represents a DVD in the library catalog.

\*/

class DVD {

private String title;

private String director;

private int itemID;

/\*\*

\* Constructs a DVD object with the specified title, director, and item ID.

\*

\* @param title The title of the DVD.

\* @param director The director of the DVD.

\* @param itemID The ID of the DVD.

\*/

public DVD(String title, String director, int itemID) {

this.title = title;

this.director = director;

this.itemID = itemID;

}

// Getter methods for the title, director, and item ID

public String getTitle() {

return title;

}

public String getDirector() {

return director;

}

public int getItemID() {

return itemID;

}

}

/\*\*

\* Represents a generic item in the library catalog.

\*

\* @param <T> The type of item (e.g., Book, DVD).

\*/

class LibraryItem<T> {

private String title;

private T item;

private int itemID;

/\*\*

\* Constructs a LibraryItem object with the specified title, item, and item ID.

\*

\* @param title The title of the item.

\* @param item The item (e.g., Book, DVD).

\* @param itemID The ID of the item.

\*/

public LibraryItem(String title, T item, int itemID) {

this.title = title;

this.item = item;

this.itemID = itemID;

}

// Getter methods for the title, item, and item ID

public String getTitle() {

return title;

}

public T getItem() {

return item;

}

public int getItemID() {

return itemID;

}

/\*\*

\* Gets the author or director of the item based on its type.

\*

\* @return The author or director of the item.

\*/

public String getAuthorOrDirector() {

if (item instanceof Book) {

return ((Book) item).getAuthor();

} else if (item instanceof DVD) {

return ((DVD) item).getDirector();

} else {

return "Unknown";

}

}

}

/\*\*

\* Represents a generic catalog that can store different types of library items.

\*

\* @param <T> The type of library item.

\*/

class Catalog<T extends LibraryItem<?>> {

private List<T> items;

/\*\*

\* Constructs an empty catalog.

\*/

public Catalog() {

this.items = new ArrayList<>();

}

/\*\*

\* Adds a library item to the catalog.

\*

\* @param item The library item to add.

\*/

public void addItem(T item) {

items.add(item);

}

/\*\*

\* Removes a library item from the catalog.

\*

\* @param item The library item to remove.

\*/

public void removeItem(T item) {

if (items.contains(item)) {

items.remove(item);

} else {

System.out.println("Item not found in catalog.");

}

}

/\*\*

\* Displays the items in the catalog.

\* If the catalog is empty, a message is displayed.

\*/

public void displayCatalog() {

if (items.isEmpty()) {

System.out.println("The catalog is empty.");

} else {

System.out.println("Catalog:");

for (T item : items) {

System.out.println("Title: " + item.getTitle() + ", Author/Director: " + item.getAuthorOrDirector()

+ ", ID: " + item.getItemID());

}

}

}

/\*\*

\* Gets the list of items in the catalog.

\*

\* @return The list of items in the catalog.

\*/

public List<T> getItems() {

return items;

}

}

/\*\*

\* Represents a simple command-line interface for users to interact with the

\* library catalog.

\* Users can add a new library item, remove an item, and view the current

\* catalog.

\*/

public class LibrarySystem {

// Main method to start the library system

public static void main(String[] args) {

Catalog<LibraryItem<?>> catalog = new Catalog<>();

Scanner scanner = new Scanner(System.in);

int choice;

do {

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

System.out.println("1. Add a new library item");

System.out.println("2. Remove an item");

System.out.println("3. View the current catalog");

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

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

try {

choice = scanner.nextInt();

switch (choice) {

case 1:

addItem(scanner, catalog);

break;

case 2:

removeItem(scanner, catalog);

break;

case 3:

catalog.displayCatalog();

break;

case 4:

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

break;

default:

System.out.println("Invalid choice. Please try again.");

}

} catch (java.util.InputMismatchException e) {

System.out.println("Invalid input. Please enter a number.");

scanner.next(); // Consume invalid input

choice = 0; // Reset choice to prompt the user again

}

} while (choice != 4);

}

// Methods for adding a new library item, removing an item, and viewing the

// catalog

private static void addItem(Scanner scanner, Catalog<LibraryItem<?>> catalog) {

scanner.nextLine(); // Consume newline

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

String title = scanner.nextLine();

System.out.print("Enter item author/director: ");

String authorOrDirector = scanner.nextLine();

System.out.print("Enter item ID: ");

int itemID = scanner.nextInt();

System.out.println("1. Book");

System.out.println("2. DVD");

System.out.print("Enter item type: ");

int itemType = scanner.nextInt();

if (itemType == 1) {

catalog.addItem(new LibraryItem<>(title, new Book(title, authorOrDirector, itemID), itemID));

System.out.print("Suucessfully added Book: ");

} else if (itemType == 2) {

catalog.addItem(new LibraryItem<>(title, new DVD(title, authorOrDirector, itemID), itemID));

System.out.print("Suucessfully added DVD: ");

} else {

System.out.println("Invalid item type.");

}

}

private static void removeItem(Scanner scanner, Catalog<LibraryItem<?>> catalog) {

System.out.print("Enter item ID to remove: ");

int itemID = scanner.nextInt();

LibraryItem<?> itemToRemove = null;

for (LibraryItem<?> item : catalog.getItems()) {

if (item.getItemID() == itemID) {

itemToRemove = item;

break;

}

}

if (itemToRemove != null) {

catalog.removeItem(itemToRemove);

System.out.println("Item removed successfully.");

} else {

System.out.println("Item not found in catalog.");

}

}

}

/\*\*

\* Provides testing scenarios for the library catalog and library items.

\*/

class LibraryTesting {

// Main method to execute the testing scenarios

public static void main(String[] args) {

testCatalog();

}

private static void testCatalog() {

Catalog<LibraryItem<?>> catalog = new Catalog<>();

// Adding library items

LibraryItem<Book> book1 = new LibraryItem<>("Book1", new Book("Book1", "Author1", 101), 101);

LibraryItem<DVD> dvd1 = new LibraryItem<>("DVD1", new DVD("DVD1", "Director1", 201), 201);

catalog.addItem(book1);

catalog.addItem(dvd1);

// Display catalog

catalog.displayCatalog();

// Removing an item

catalog.removeItem(book1);

// Display catalog after removal

catalog.displayCatalog();

}

}

## **Instruction for running the program**

* Open your CLI and navigate to the file path, saved mine with “LibrarySystem”
* Ensure to have Java installed on your computer
* **Javac LibrarySystem.java** - to compile the code
* **java LibrarySystem** - to run the code

## **Output Screenshot**

