import java.util.\*;

class ItemNotAvailableException extends Exception {

public ItemNotAvailableException(String message) {

super(message);

}

}

abstract class LibraryItem {

protected String title;

protected String itemID;

protected boolean isAvailable;

public LibraryItem(String title, String itemID) {

this.title = title;

this.itemID = itemID;

this.isAvailable = true;

}

public abstract void borrow() throws ItemNotAvailableException;

public abstract void returnItem();

public String getTitle() {

return title;

}

public String getItemID() {

return itemID;

}

public boolean isAvailable() {

return isAvailable;

}

}

class Book extends LibraryItem {

private String author;

private String genre;

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

super(title, itemID);

this.author = author;

this.genre = genre;

}

@Override

public void borrow() throws ItemNotAvailableException {

if (!isAvailable) {

throw new ItemNotAvailableException("Book '" + title + "' is currently unavailable.");

}

isAvailable = false;

System.out.println("Book borrowed: " + title);

}

@Override

public void returnItem() {

isAvailable = true;

System.out.println("Book returned: " + title);

}

@Override

public String toString() {

return "Book [Title: " + title + ", Author: " + author + ", Genre: " + genre + ", ID: " + itemID + "]";

}

}

class DVD extends LibraryItem {

private String director;

private int duration;

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

super(title, itemID);

this.director = director;

this.duration = duration;

}

@Override

public void borrow() throws ItemNotAvailableException {

if (!isAvailable) {

throw new ItemNotAvailableException("DVD '" + title + "' is currently unavailable.");

}

isAvailable = false;

System.out.println("DVD borrowed: " + title);

}

@Override

public void returnItem() {

isAvailable = true;

System.out.println("DVD returned: " + title);

}

@Override

public String toString() {

return "DVD [Title: " + title + ", Director: " + director + ", Duration: " + duration + " mins, ID: " + itemID + "]";

}

}

class Library {

private List<LibraryItem> items;

public Library() {

items = new ArrayList<>();

}

public void addItem(LibraryItem item) {

items.add(item);

}

public void borrowItem(String itemID) {

try {

LibraryItem item = searchByID(itemID);

if (item != null) {

item.borrow();

} else {

System.out.println("Item with ID " + itemID + " not found.");

}

} catch (ItemNotAvailableException e) {

System.out.println(e.getMessage());

} finally {

System.out.println("Borrow operation completed.");

}

}

public void returnItem(String itemID) {

LibraryItem item = searchByID(itemID);

if (item != null) {

item.returnItem();

} else {

System.out.println("Item with ID " + itemID + " not found.");

}

System.out.println("Return operation completed.");

}

public LibraryItem searchByTitle(String title) {

for (LibraryItem item : items) {

if (item.getTitle().equalsIgnoreCase(title)) {

return item;

}

}

return null;

}

public LibraryItem searchByID(String itemID) {

for (LibraryItem item : items) {

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

return item;

}

}

return null;

}

public void displayAllItems() {

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

for (LibraryItem item : items) {

if (item instanceof Book) {

System.out.println(item);

}

}

System.out.println("\nDVDs:");

for (LibraryItem item : items) {

if (item instanceof DVD) {

System.out.println(item);

}

}

}

}

public class Main {

public static void main(String[] args) {

Library library = new Library();

library.addItem(new Book("The Great Gatsby", "B001", "F. Scott Fitzgerald", "Fiction"));

library.addItem(new Book("1984", "B002", "George Orwell", "Dystopian"));

library.addItem(new DVD("Inception", "D001", "Christopher Nolan", 148));

library.addItem(new DVD("The Matrix", "D002", "The Wachowskis", 136));

library.borrowItem("B001");

library.returnItem("B001");

library.borrowItem("D001");

library.returnItem("D001");

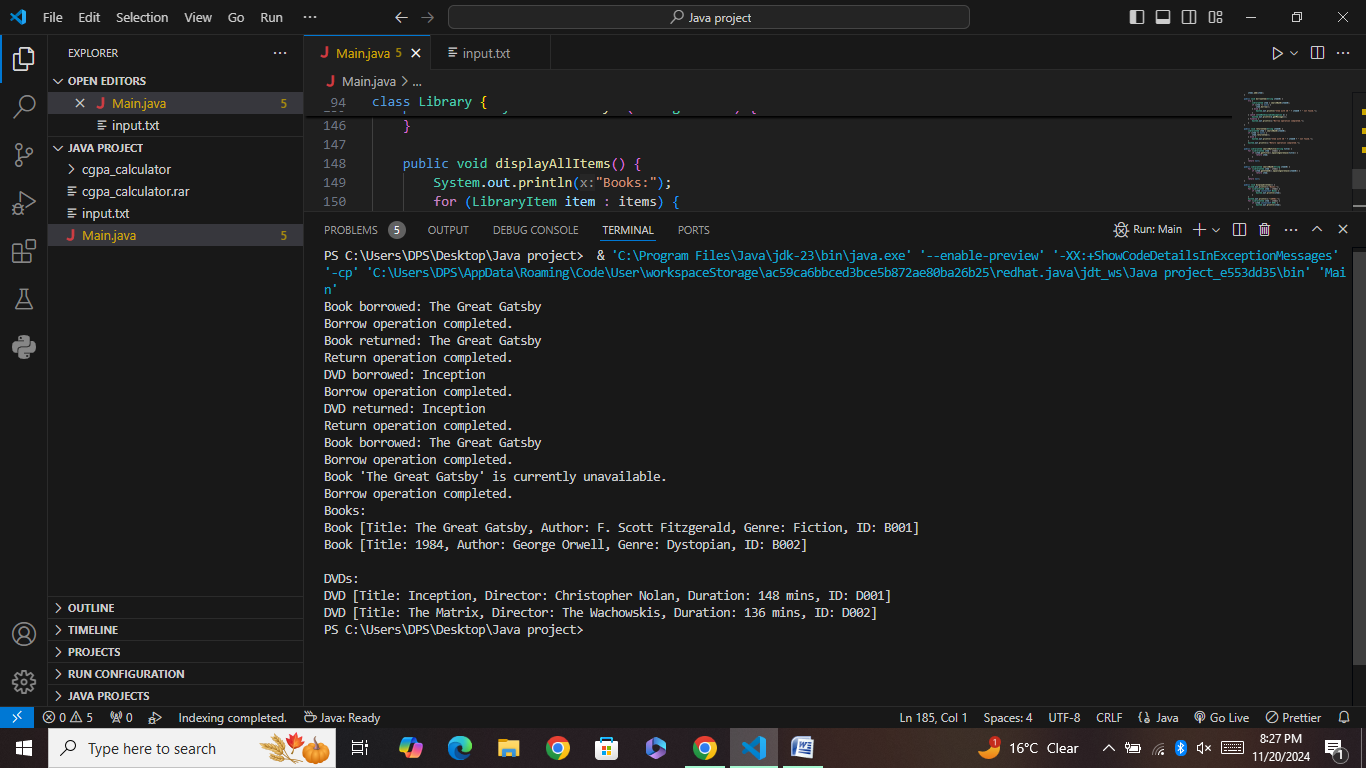
library.borrowItem("B001");

library.borrowItem("B001");

library.displayAllItems();

}

}



Here's the Java implementation for the given library management system:

---

### \*\*Library Management System Code\*\*

```java

import java.util.\*;

// Custom Exception

class ItemNotAvailableException extends Exception {

public ItemNotAvailableException(String message) {

super(message);

}

}

// Abstract Class LibraryItem

abstract class LibraryItem {

protected String title;

protected String itemID;

protected boolean isAvailable;

// Constructor

public LibraryItem(String title, String itemID) {

this.title = title;

this.itemID = itemID;

this.isAvailable = true; // Initially available

}

// Abstract Methods

public abstract void borrow() throws ItemNotAvailableException;

public abstract void returnItem();

// Common Method

public String getTitle() {

return title;

}

public String getItemID() {

return itemID;

}

public boolean isAvailable() {

return isAvailable;

}

}

// Book Subclass

class Book extends LibraryItem {

private String author;

private String genre;

// Constructor

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

super(title, itemID);

this.author = author;

this.genre = genre;

}

@Override

public void borrow() throws ItemNotAvailableException {

if (!isAvailable) {

throw new ItemNotAvailableException("Book '" + title + "' is not available.");

}

isAvailable = false;

System.out.println("Book borrowed: " + title);

}

@Override

public void returnItem() {

isAvailable = true;

System.out.println("Book returned: " + title);

}

@Override

public String toString() {

return "Book[Title: " + title + ", Author: " + author + ", Genre: " + genre + ", ID: " + itemID + "]";

}

}

// DVD Subclass

class DVD extends LibraryItem {

private String director;

private int duration; // In minutes

// Constructor

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

super(title, itemID);

this.director = director;

this.duration = duration;

}

@Override

public void borrow() throws ItemNotAvailableException {

if (!isAvailable) {

throw new ItemNotAvailableException("DVD '" + title + "' is not available.");

}

isAvailable = false;

System.out.println("DVD borrowed: " + title);

}

@Override

public void returnItem() {

isAvailable = true;

System.out.println("DVD returned: " + title);

}

@Override

public String toString() {

return "DVD[Title: " + title + ", Director: " + director + ", Duration: " + duration + " mins, ID: " + itemID + "]";

}

}

// Library Class

class Library {

private List<LibraryItem> items;

public Library() {

items = new ArrayList<>();

}

// Add Item

public void addItem(LibraryItem item) {

items.add(item);

}

// Search by Title (Overloaded Method)

public void search(String title) {

for (LibraryItem item : items) {

if (item.getTitle().equalsIgnoreCase(title)) {

System.out.println(item);

return;

}

}

System.out.println("Item with title '" + title + "' not found.");

}

// Search by ID (Overloaded Method)

public void searchByID(String itemID) {

for (LibraryItem item : items) {

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

System.out.println(item);

return;

}

}

System.out.println("Item with ID '" + itemID + "' not found.");

}

// Display All Items Grouped by Type

public void displayAllItems() {

System.out.println("\nBooks:");

for (LibraryItem item : items) {

if (item instanceof Book) {

System.out.println(item);

}

}

System.out.println("\nDVDs:");

for (LibraryItem item : items) {

if (item instanceof DVD) {

System.out.println(item);

}

}

}

// Borrow Item

public void borrowItem(String itemID) {

try {

for (LibraryItem item : items) {

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

item.borrow();

return;

}

}

System.out.println("Item with ID '" + itemID + "' not found.");

} catch (ItemNotAvailableException e) {

System.out.println(e.getMessage());

} finally {

System.out.println("Borrow operation logged.");

}

}

// Return Item

public void returnItem(String itemID) {

for (LibraryItem item : items) {

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

item.returnItem();

System.out.println("Return operation logged.");

return;

}

}

System.out.println("Item with ID '" + itemID + "' not found.");

}

}

// Main Class

public class LibraryManagementSystem {

public static void main(String[] args) {

Library library = new Library();

// Adding items to the library

library.addItem(new Book("The Great Gatsby", "B001", "F. Scott Fitzgerald", "Fiction"));

library.addItem(new Book("1984", "B002", "George Orwell", "Dystopian"));

library.addItem(new DVD("Inception", "D001", "Christopher Nolan", 148));

library.addItem(new DVD("The Matrix", "D002", "The Wachowskis", 136));

// Display all items

library.displayAllItems();

// Search for items

System.out.println("\nSearching for '1984':");

library.search("1984");

System.out.println("\nSearching for item ID 'D002':");

library.searchByID("D002");

// Borrow and return items

System.out.println("\nBorrowing 'B001':");

library.borrowItem("B001");

System.out.println("\nBorrowing 'B001' again:");

library.borrowItem("B001");

System.out.println("\nReturning 'B001':");

library.returnItem("B001");

}

}

```

---

### Explanation:

1. \*\*Abstract Class and Subclasses\*\*:

- `LibraryItem` defines shared attributes and abstract methods.

- `Book` and `DVD` inherit from `LibraryItem` and add specific attributes with unique `borrow()` and `returnItem()` logic.

2. \*\*Polymorphism\*\*:

- `borrow()` and `returnItem()` methods are called dynamically based on the object type (e.g., `Book` or `DVD`).

3. \*\*Method Overloading\*\*:

- `search()` is overloaded to search by title or ID.

4. \*\*Custom Exception\*\*:

- `ItemNotAvailableException` ensures items cannot be borrowed if unavailable.

5. \*\*Thread Safety\*\*:

- Demonstrated using `finally` for operation logging.

6. \*\*Output Grouping\*\*:

- Items are displayed grouped by type (Books or DVDs).