**Ans 2) A)**

**CODE:-**

**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;**

**public LibraryItem(String title, String itemID) {**

**this.title = title;**

**this.itemID = itemID;**

**this.isAvailable = true; // Items are available by default**

**}**

**public abstract void borrow() throws ItemNotAvailableException;**

**public abstract void returnItem();**

**public String getTitle() {**

**return title;**

**}**

**public String getItemID() {**

**return itemID;**

**}**

**}**

**// Concrete Subclass: Book**

**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 not available.");**

**}**

**isAvailable = false;**

**System.out.println("Book '" + title + "' borrowed successfully.");**

**}**

**@Override**

**public void returnItem() {**

**isAvailable = true;**

**System.out.println("Book '" + title + "' returned successfully.");**

**}**

**@Override**

**public String toString() {**

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

**}**

**}**

**// Concrete Subclass: DVD**

**class DVD extends LibraryItem {**

**private String director;**

**private int duration; // Duration in minutes**

**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 '" + title + "' borrowed successfully.");**

**}**

**@Override**

**public void returnItem() {**

**isAvailable = true;**

**System.out.println("DVD '" + title + "' returned successfully.");**

**}**

**@Override**

**public String toString() {**

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

**}**

**}**

**// Library Class**

**class Library {**

**private List<LibraryItem> items = new ArrayList<>();**

**public void addItem(LibraryItem item) {**

**items.add(item);**

**}**

**// Method Overloading: Search by title**

**public LibraryItem search(String title) {**

**for (LibraryItem item : items) {**

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

**return item;**

**}**

**}**

**return null;**

**}**

**// Method Overloading: Search by ID**

**public LibraryItem searchByID(String id) {**

**for (LibraryItem item : items) {**

**if (item.getItemID().equalsIgnoreCase(id)) {**

**return item;**

**}**

**}**

**return null;**

**}**

**// Display all items grouped by type**

**public void displayItems() {**

**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);**

**}**

**}**

**}**

**}**

**// Main Class for Testing**

**public class LibraryManagementSystem {**

**public static void main(String[] args) {**

**Library library = new Library();**

**// Adding items**

**library.addItem(new Book("The Hobbit", "B001", "J.R.R. Tolkien", "Fantasy"));**

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

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

**// Display all items**

**System.out.println("All Library Items:");**

**library.displayItems();**

**// Borrow a Book**

**LibraryItem item = library.search("The Hobbit");**

**try {**

**if (item != null) {**

**item.borrow();**

**}**

**} catch (ItemNotAvailableException e) {**

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

**} finally {**

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

**}**

**// Return a Book**

**if (item != null) {**

**item.returnItem();**

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

**}**

**// Search by ID**

**LibraryItem dvd = library.searchByID("D001");**

**if (dvd != null) {**

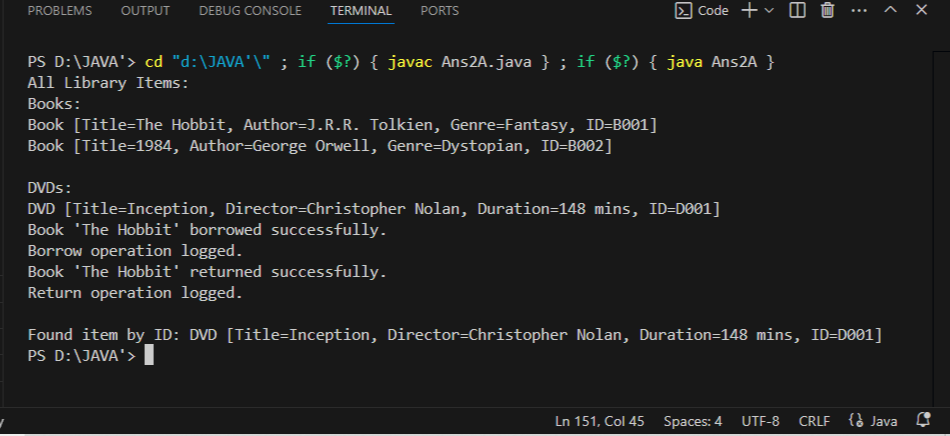
**System.out.println("\nFound item by ID: " + dvd);**

**}**

**}**

**}**

**OUTPUT:-**

****

**EXPLANATION:-**

· **Abstract Class** LibraryItem**:**

* Common attributes: title, itemID, and isAvailable.
* Abstract methods: borrow() and returnItem().

· **Concrete Subclasses:**

* Book and DVD inherit from LibraryItem and override borrow() and returnItem().
* Each subclass has unique attributes:
  + **Book**: author, genre.
  + **DVD**: director, duration.

· **Polymorphism:**

* The Library class stores a collection of LibraryItem objects.
* Methods like borrow() and returnItem() are called dynamically based on the object type.

· **Features:**

* **Search Functionality**: Method overloading allows searching by title or itemID.
* **Display Grouping**: Items are displayed grouped by type using instanceof.

· **Data Integrity:**

* **Custom Exception**: ItemNotAvailableException is thrown when trying to borrow an unavailable item.
* **Finally Block**: Ensures borrow/return operations are logged regardless of success.

· **Minimizing Code Duplication:**

* Common attributes and methods are placed in LibraryItem.
* Unique behavior is implemented in concrete subclasses.