```
import java.io.*;
import java.util.*;
import java.time.LocalDate;
import java.time.temporal.ChronoUnit;

// Custom Exception Classes
class BookNotFoundException extends Exception {
    public BookNotFoundException(String message) {
        super(message);
    }
}

class NotEnoughBooksException extends Exception {
    public NotEnoughBooksException(String message) {
        super(message);
    }
}

class MemberNotFoundException extends Exception {
    public MemberNotFoundException(String message) {
        super(message);
    }
}

// Abstract Base Class for Library Items
abstract class LibraryItem implements Serializable {
    protected String itemId;
    protected String title;
    protected boolean isAvailable;
    protected LocalDate dueDate;

    public LibraryItem(String itemId, String title) {
        this.itemId = itemId;
        this.title = title;
        this.isAvailable = true;
    }

    // Abstract methods to be implemented by derived classes
    public abstract String getItemType();
    public abstract void displayDetails();

    // Common methods
    public String getItemId() { return itemId; }
    public String getTitle() { return title; }
    public boolean isAvailable() { return isAvailable; }
    public void setAvailable(boolean available) { this.isAvailable = available; }
    public LocalDate getDueDate() { return dueDate; }
    public void setDueDate(LocalDate dueDate) { this.dueDate = dueDate; }

    public boolean isOverdue() {
        return dueDate != null && LocalDate.now().isAfter(dueDate);
    }

    public long getDaysOverdue() {
        if (dueDate != null && LocalDate.now().isAfter(dueDate)) {
            return ChronoUnit.DAYS.between(dueDate, LocalDate.now());
        }
        return 0;
    }
}
```

```
// Book class inheriting from LibraryItem
class Book extends LibraryItem {
    private String isbn;
    private String author;
    private String genre;
    private double price;
    private int totalCopies;
    private int availableCopies;
    private String publisher;
    private int publicationYear;

    public Book(String isbn, String title, String author, String genre, double price,
                int totalCopies, String publisher, int publicationYear) {
        super(isbn, title); // Using ISBN as itemId
        this.isbn = isbn;
        this.author = author;
        this.genre = genre;
        this.price = price;
        this.totalCopies = totalCopies;
        this.availableCopies = totalCopies;
        this.publisher = publisher;
        this.publicationYear = publicationYear;
    }

    // Implementation of abstract methods
    @Override
    public String getItemType() {
        return "Book";
    }

    @Override
    public void displayDetails() {
        System.out.println("ISBN: " + isbn);
        System.out.println("Title: " + title);
        System.out.println("Author: " + author);
        System.out.println("Genre: " + genre);
        System.out.println("Price: $" + price);
        System.out.println("Available Copies: " + availableCopies + "/" + totalCopies);
        System.out.println("Publisher: " + publisher);
        System.out.println("Year: " + publicationYear);
        System.out.println("Available: " + (isAvailable ? "Yes" : "No"));
        if (dueDate != null) {
            System.out.println("Due Date: " + dueDate);
            if (isOverdue()) {
                System.out.println("Status: OVERDUE by " + getDaysOverdue() + " days");
            }
        }
    }

    // Getters and setters with encapsulation
    public String getIsbn() { return isbn; }
    public String getAuthor() { return author; }
    public String getGenre() { return genre; }
    public double getPrice() { return price; }
    public int getTotalCopies() { return totalCopies; }
    public int getAvailableCopies() { return availableCopies; }
    public String getPublisher() { return publisher; }
    public int getPublicationYear() { return publicationYear; }

    public void setAvailableCopies(int availableCopies) {
        this.availableCopies = availableCopies;
        this.isAvailable = (availableCopies > 0);
```

```
    }

    public void borrowCopy() throws NotEnoughBooksException {
        if (availableCopies <= 0) {
            throw new NotEnoughBooksException("No copies available for book: " + title);
        }
        availableCopies--;
        isAvailable = (availableCopies > 0);
    }

    public void returnCopy() {
        availableCopies++;
        isAvailable = true;
        dueDate = null;
    }

    @Override
    public String toString() {
        return String.format("Book[ISBN: %s, Title: %s, Author: %s, Available: %d/%d]",
                             isbn, title, author, availableCopies, totalCopies);
    }
}

// Magazine class demonstrating inheritance
class Magazine extends LibraryItem {
    private String issn;
    private int issueNumber;
    private LocalDate publicationDate;
    private String category;

    public Magazine(String issn, String title, int issueNumber, LocalDate publicationDate,
String category) {
        super(issn, title); // Using ISSN as itemId
        this.issn = issn;
        this.issueNumber = issueNumber;
        this.publicationDate = publicationDate;
        this.category = category;
    }

    @Override
    public String getItemType() {
        return "Magazine";
    }

    @Override
    public void displayDetails() {
        System.out.println("ISSN: " + issn);
        System.out.println("Title: " + title);
        System.out.println("Issue: " + issueNumber);
        System.out.println("Publication Date: " + publicationDate);
        System.out.println("Category: " + category);
        System.out.println("Available: " + (isAvailable ? "Yes" : "No"));
        if (dueDate != null) {
            System.out.println("Due Date: " + dueDate);
        }
    }

    // Getters and setters
    public String getIssn() { return issn; }
    public int getIssueNumber() { return issueNumber; }
    public LocalDate getPublicationDate() { return publicationDate; }
    public String getCategory() { return category; }
```

```
    @Override
    public String toString() {
        return String.format("Magazine[ISSN: %s, Title: %s, Issue: %d]", issn, title,
issueNumber);
    }
}

// Member class
class Member implements Serializable {
    private String memberId;
    private String name;
    private String email;
    private String phoneNumber;
    private LocalDate registrationDate;
    private List<LibraryItem> borrowedItems;
    private int maxBorrowLimit;

    public Member(String memberId, String name, String email, String phoneNumber) {
        this.memberId = memberId;
        this.name = name;
        this.email = email;
        this.phoneNumber = phoneNumber;
        this.registrationDate = LocalDate.now();
        this.borrowedItems = new ArrayList<>();
        this.maxBorrowLimit = 5; // Default borrow limit
    }

    // Getters and setters with encapsulation
    public String getMemberId() { return memberId; }
    public String getName() { return name; }
    public String getEmail() { return email; }
    public String getPhoneNumber() { return phoneNumber; }
    public LocalDate getRegistrationDate() { return registrationDate; }
    public int getMaxBorrowLimit() { return maxBorrowLimit; }
    public void setMaxBorrowLimit(int maxBorrowLimit) { this.maxBorrowLimit = maxBorrowLimit; }

    public List<LibraryItem> getBorrowedItems() {
        return new ArrayList<>(borrowedItems);
    }

    public void borrowItem(LibraryItem item) throws NotEnoughBooksException {
        if (borrowedItems.size() >= maxBorrowLimit) {
            throw new NotEnoughBooksException("Borrow limit reached. Maximum " + maxBorrowLimit
+ " items allowed.");
        }

        if (item instanceof Book) {
            Book book = (Book) item;
            book.borrowCopy();
        }

        item.setAvailable(false);
        item.setDueDate(LocalDate.now().plusWeeks(2)); // 2 weeks borrowing period
        borrowedItems.add(item);
    }

    public void returnItem(String itemId) throws BookNotFoundException {
        LibraryItem itemToReturn = null;
        for (LibraryItem item : borrowedItems) {
            if (item.getItemId().equals(itemId)) {
                itemToReturn = item;
```

```
                break;
            }
        }

        if (itemToReturn == null) {
            throw new BookNotFoundException("Item with ID " + itemId + " not found in borrowed
items.");
        }

        if (itemToReturn instanceof Book) {
            Book book = (Book) itemToReturn;
            book.returnCopy();
        } else {
            itemToReturn.setAvailable(true);
            itemToReturn.setDueDate(null);
        }

        borrowedItems.remove(itemToReturn);
    }

    public void displayBorrowedItems() {
        if (borrowedItems.isEmpty()) {
            System.out.println("No items currently borrowed.");
            return;
        }

        System.out.println("Borrowed Items (" + borrowedItems.size() + "/" + maxBorrowLimit +
"):");
        for (int i = 0; i < borrowedItems.size(); i++) {
            LibraryItem item = borrowedItems.get(i);
            System.out.println((i + 1) + ". " + item.getItemType() + ": " + item.getTitle());
            System.out.println("   Due Date: " + item.getDueDate());
            if (item.isOverdue()) {
                System.out.println("   âš ï¸ OVERDUE by " + item.getDaysOverdue() + " days");
            }
        }
    }

    public boolean hasOverdueItems() {
        for (LibraryItem item : borrowedItems) {
            if (item.isOverdue()) {
                return true;
            }
        }
        return false;
    }

    @Override
    public String toString() {
        return String.format("Member[ID: %s, Name: %s, Email: %s, Borrowed: %d/%d]",
                             memberId, name, email, borrowedItems.size(), maxBorrowLimit);
    }
}

// Library class
class Library implements Serializable {
    private String name;
    private String address;
    private Map<String, LibraryItem> items; // Using itemId as key
    private Map<String, Member> members;
    private List<LibraryItem> allItems; // For polymorphism demonstration
```

```
public Library(String name, String address) {
    this.name = name;
    this.address = address;
    this.items = new HashMap<>();
    this.members = new HashMap<>();
    this.allItems = new ArrayList<>();
}

// Item management methods
public void addItem(LibraryItem item) {
    items.put(item.getItemId(), item);
    allItems.add(item);
    System.out.println("âœ… Added: " + item.getItemType() + " - " + item.getTitle());
}

public void removeItem(String itemId) throws BookNotFoundException {
    LibraryItem item = items.get(itemId);
    if (item == null) {
        throw new BookNotFoundException("Item with ID " + itemId + " not found.");
    }
    items.remove(itemId);
    allItems.remove(item);
    System.out.println("âœ… Removed: " + item.getTitle());
}

// Search methods demonstrating polymorphism
public List<LibraryItem> searchByAuthor(String author) {
    List<LibraryItem> results = new ArrayList<>();
    for (LibraryItem item : allItems) {
        if (item instanceof Book) {
            Book book = (Book) item;
            if (book.getAuthor().toLowerCase().contains(author.toLowerCase())) {
                results.add(item);
            }
        }
    }
    return results;
}

public List<LibraryItem> searchByGenre(String genre) {
    List<LibraryItem> results = new ArrayList<>();
    for (LibraryItem item : allItems) {
        if (item instanceof Book) {
            Book book = (Book) item;
            if (book.getGenre().toLowerCase().contains(genre.toLowerCase())) {
                results.add(item);
            }
        }
    }
    return results;
}

public List<LibraryItem> searchByTitle(String title) {
    List<LibraryItem> results = new ArrayList<>();
    for (LibraryItem item : allItems) {
        if (item.getTitle().toLowerCase().contains(title.toLowerCase())) {
            results.add(item);
        }
    }
    return results;
}
```

```
// Method accepting base class type (polymorphism)
public void displayItemDetails(LibraryItem item) {
    System.out.println("\n=== " + item.getItemType().toUpperCase() + " DETAILS ===");
    item.displayDetails();
    System.out.println("==================");
}

public void displayAvailableItems() {
    System.out.println("\n=== AVAILABLE ITEMS ===");
    boolean foundAvailable = false;

    for (LibraryItem item : allItems) {
        if (item.isAvailable()) {
            System.out.println("- " + item.getItemType() + ": " + item.getTitle());
            if (item instanceof Book) {
                Book book = (Book) item;
                System.out.println("  Available Copies: " + book.getAvailableCopies());
            }
            foundAvailable = true;
        }
    }

    if (!foundAvailable) {
        System.out.println("No available items at the moment.");
    }
}

// Member management
public void addMember(Member member) {
    members.put(member.getMemberId(), member);
    System.out.println("âœ… Added member: " + member.getName());
}

public void removeMember(String memberId) throws MemberNotFoundException {
    Member member = members.get(memberId);
    if (member == null) {
        throw new MemberNotFoundException("Member with ID " + memberId + " not found.");
    }

    if (!member.getBorrowedItems().isEmpty()) {
        System.out.println("âš ï¸  Member has borrowed items. Cannot remove.");
        return;
    }

    members.remove(memberId);
    System.out.println("âœ… Removed member: " + member.getName());
}

public Member getMember(String memberId) throws MemberNotFoundException {
    Member member = members.get(memberId);
    if (member == null) {
        throw new MemberNotFoundException("Member with ID " + memberId + " not found.");
    }
    return member;
}

public LibraryItem getItem(String itemId) throws BookNotFoundException {
    LibraryItem item = items.get(itemId);
    if (item == null) {
        throw new BookNotFoundException("Item with ID " + itemId + " not found.");
    }
    return item;
```

```
    }

    // Borrow and return operations
    public void borrowItem(String memberId, String itemId)
            throws MemberNotFoundException, BookNotFoundException, NotEnoughBooksException {
        Member member = getMember(memberId);
        LibraryItem item = getItem(itemId);

        if (member.hasOverdueItems()) {
            System.out.println("âš ï¸  Member has overdue items. Cannot borrow new items.");
            return;
        }

        member.borrowItem(item);
        System.out.println("â€¦ " + member.getName() + " borrowed: " + item.getTitle());
        System.out.println("   Due Date: " + item.getDueDate());
    }

    public void returnItem(String memberId, String itemId)
            throws MemberNotFoundException, BookNotFoundException {
        Member member = getMember(memberId);
        member.returnItem(itemId);

        LibraryItem item = getItem(itemId);
        System.out.println("â€¦ " + member.getName() + " returned: " + item.getTitle());

        if (item.isOverdue()) {
            System.out.println("âš ï¸  This item was overdue by " + item.getDaysOverdue() + "
days");
        }
    }

    // Getters
    public String getName() { return name; }
    public String getAddress() { return address; }
    public Collection<LibraryItem> getAllItems() { return new ArrayList<>(allItems); }
    public Collection<Member> getAllMembers() { return new ArrayList<>(members.values()); }

    public void displayLibraryInfo() {
        System.out.println("\n=== LIBRARY INFORMATION ===");
        System.out.println("Name: " + name);
        System.out.println("Address: " + address);
        System.out.println("Total Items: " + allItems.size());
        System.out.println("Total Members: " + members.size());

        int availableBooks = 0;
        int borrowedBooks = 0;
        for (LibraryItem item : allItems) {
            if (item.isAvailable()) {
                availableBooks++;
            } else {
                borrowedBooks++;
            }
        }

        System.out.println("Available Items: " + availableBooks);
        System.out.println("Borrowed Items: " + borrowedBooks);
    }
}

// Serialization Handler
class SerializationHandler {
```

```
    public static void serializeLibrary(Library library, String fileName) {
        try (FileOutputStream fileOut = new FileOutputStream(fileName);
             ObjectOutputStream objectOut = new ObjectOutputStream(fileOut)) {

            objectOut.writeObject(library);
            System.out.println("âœ… Library serialized successfully to: " + fileName);

        } catch (IOException e) {
            System.err.println("â Error serializing library: " + e.getMessage());
        }
    }

    public static Library deserializeLibrary(String fileName) {
        try (FileInputStream fileIn = new FileInputStream(fileName);
             ObjectInputStream objectIn = new ObjectInputStream(fileIn)) {

            Library library = (Library) objectIn.readObject();
            System.out.println("âœ… Library deserialized successfully from: " + fileName);
            return library;

        } catch (IOException | ClassNotFoundException e) {
            System.err.println("â Error deserializing library: " + e.getMessage());
            return null;
        }
    }
}

// Main class to demonstrate the system
public class LibraryManagementSystem {

    public static void main(String[] args) {
        // Create library
        Library library = new Library("City Central Library", "123 Main Street, Cityville");

        // Add books
        Book book1 = new Book("978-0134685991", "Effective Java", "Joshua Bloch",
                              "Programming", 49.99, 5, "Addison-Wesley", 2018);
        Book book2 = new Book("978-0201633610", "Design Patterns", "Erich Gamma",
                              "Software Engineering", 59.99, 3, "Addison-Wesley", 1994);
        Book book3 = new Book("978-0596007126", "Head First Java", "Kathy Sierra",
                              "Programming", 39.99, 8, "O'Reilly", 2005);

        library.addItem(book1);
        library.addItem(book2);
        library.addItem(book3);

        // Add magazine (demonstrating inheritance)
        Magazine magazine1 = new Magazine("1234-5678", "Tech Monthly", 45,
                                          LocalDate.of(2024, 1, 15), "Technology");
        library.addItem(magazine1);

        // Add members
        Member member1 = new Member("M001", "Alice Johnson", "alice@email.com", "555-0101");
        Member member2 = new Member("M002", "Bob Smith", "bob@email.com", "555-0102");

        library.addMember(member1);
        library.addMember(member2);

        // Demonstrate operations
        System.out.println("\n" + "=".repeat(50));
        library.displayLibraryInfo();
```

```
System.out.println("\n" + "=".repeat(50));
System.out.println("DEMONSTRATING SEARCH FUNCTIONALITY:");

// Search by author
System.out.println("\nSearching for books by 'Joshua Bloch':");
List<LibraryItem> results = library.searchByAuthor("Joshua Bloch");
for (LibraryItem item : results) {
    library.displayItemDetails(item);
}

// Search by genre
System.out.println("\nSearching for 'Programming' books:");
results = library.searchByGenre("Programming");
for (LibraryItem item : results) {
    System.out.println("- " + item.getTitle());
}

System.out.println("\n" + "=".repeat(50));
System.out.println("DEMONSTRATING BORROWING OPERATIONS:");

try {
    // Borrow books
    library.borrowItem("M001", "978-0134685991");
    library.borrowItem("M001", "1234-5678"); // Borrow magazine

    // Try to borrow same book again (should show not available)
    library.borrowItem("M002", "978-0134685991");

} catch (Exception e) {
    System.out.println("â Œ " + e.getMessage());
}

System.out.println("\n" + "=".repeat(50));
System.out.println("DISPLAYING MEMBER BORROWED ITEMS:");

try {
    Member alice = library.getMember("M001");
    alice.displayBorrowedItems();
} catch (MemberNotFoundException e) {
    System.out.println("â Œ " + e.getMessage());
}

System.out.println("\n" + "=".repeat(50));
System.out.println("DEMONSTRATING SERIALIZATION:");

// Serialize library
SerializationHandler.serializeLibrary(library, "library_data.ser");

// Deserialize library
Library loadedLibrary = SerializationHandler.deserializeLibrary("library_data.ser");

if (loadedLibrary != null) {
    System.out.println("\nLoaded Library Information:");
    loadedLibrary.displayLibraryInfo();
    loadedLibrary.displayAvailableItems();
}

System.out.println("\n" + "=".repeat(50));
System.out.println("DEMONSTRATING RETURN OPERATIONS:");

try {
```

```
            // Return items
            library.returnItem("M001", "978-0134685991");
            library.returnItem("M001", "1234-5678");

            // Display member's borrowed items after return
            Member alice = library.getMember("M001");
            alice.displayBorrowedItems();

        } catch (Exception e) {
            System.out.println("â Œ " + e.getMessage());
        }

        System.out.println("\n" + "=".repeat(50));
        System.out.println("DEMONSTRATING ERROR HANDLING:");

        try {
            // Try to borrow non-existent book
            library.borrowItem("M001", "999-9999999999");
        } catch (Exception e) {
            System.out.println("â Œ " + e.getClass().getSimpleName() + ": " + e.getMessage());
        }

        try {
            // Try to borrow from non-existent member
            library.borrowItem("M999", "978-0134685991");
        } catch (Exception e) {
            System.out.println("â Œ " + e.getClass().getSimpleName() + ": " + e.getMessage());
        }

        // Display final library state
        System.out.println("\n" + "=".repeat(50));
        System.out.println("FINAL LIBRARY STATE:");
        library.displayLibraryInfo();
        library.displayAvailableItems();
    }
}
```