Library Management System using OOP, Serialization and Exception Handling  
  
import java.io.\*;

import java.util.\*;

class Book implements Serializable {

    private String ISBN, title, author, genre;

    private double price;

    private boolean available;

    public Book(String ISBN, String title, String author, double price, boolean available, String genre) {

        this.ISBN = ISBN;

        this.title = title;

        this.author = author;

        this.price = price;

        this.available = available;

        this.genre = genre;

    }

    public String getISBN() { return ISBN; }

    public String getAuthor() { return author; }

    public String getGenre() { return genre; }

    public boolean isAvailable() { return available; }

    public void setAvailable(boolean available) { this.available = available; }

    public String toString() {

        return title + " by " + author + " | Genre: " + genre + " | Price: " + price + " | Available: " + available;

    }

}

class BookNotFoundException extends Exception {

    public BookNotFoundException(String msg) { super(msg); }

}

class NotEnoughBooksException extends Exception {

    public NotEnoughBooksException(String msg) { super(msg); }

}

class Library implements Serializable {

    private String name, address;

    private List<Book> books = new ArrayList<>();

    public Library(String name, String address) {

        this.name = name;

        this.address = address;

    }

    public void addBook(Book book) { books.add(book); }

    public void removeBook(String ISBN) throws BookNotFoundException {

        Book book = books.stream().filter(b -> b.getISBN().equals(ISBN)).findFirst().orElse(null);

        if (book == null) throw new BookNotFoundException("Book not found");

        books.remove(book);

    }

    public List<Book> searchByAuthor(String author) {

        return books.stream().filter(b -> b.getAuthor().equalsIgnoreCase(author)).toList();

    }

    public List<Book> searchByGenre(String genre) {

        return books.stream().filter(b -> b.getGenre().equalsIgnoreCase(genre)).toList();

    }

    public List<Book> displayAvailableBooks() {

        return books.stream().filter(Book::isAvailable).toList();

    }

    public Book getBookByISBN(String ISBN) throws BookNotFoundException {

        return books.stream().filter(b -> b.getISBN().equals(ISBN)).findFirst()

                .orElseThrow(() -> new BookNotFoundException("Book not found"));

    }

}

class Member implements Serializable {

    private String memberId, name;

    private List<Book> borrowedBooks = new ArrayList<>();

    public Member(String memberId, String name) {

        this.memberId = memberId;

        this.name = name;

    }

    public void borrowBook(Library library, String ISBN) throws Exception {

        Book book = library.getBookByISBN(ISBN);

        if (!book.isAvailable()) throw new NotEnoughBooksException("Book unavailable");

        book.setAvailable(false);

        borrowedBooks.add(book);

    }

    public void returnBook(Library library, String ISBN) throws BookNotFoundException {

        Book book = borrowedBooks.stream().filter(b -> b.getISBN().equals(ISBN)).findFirst()

                .orElseThrow(() -> new BookNotFoundException("Book not borrowed"));

        book.setAvailable(true);

        borrowedBooks.remove(book);

    }

    public List<Book> displayBorrowedBooks() { return borrowedBooks; }

}

class SerializationHandler {

    public static void serializeLibrary(Library library, String fileName) throws IOException {

        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(fileName))) {

            out.writeObject(library);

        }

    }

    public static Library deserializeLibrary(String fileName) throws IOException, ClassNotFoundException {

        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(fileName))) {

            return (Library) in.readObject();

        }

    }

}

public class LibrarySystem {

    public static void main(String[] args) throws Exception {

        Library lib = new Library("City Library", "Downtown");

        lib.addBook(new Book("B1", "Java Basics", "James", 350, true, "Programming"));

        lib.addBook(new Book("B2", "Python Guide", "Guido", 400, true, "Programming"));

        Member m = new Member("M1", "Sanjay");

        m.borrowBook(lib, "B1");

        SerializationHandler.serializeLibrary(lib, "library.ser");

        Library loadedLib = SerializationHandler.deserializeLibrary("library.ser");

        System.out.println("Available Books: " + loadedLib.displayAvailableBooks());

        System.out.println("Borrowed Books: " + m.displayBorrowedBooks());

    }

}

Output:

Available Books:

[Python Guide by Guido | Genre: Programming | Price: 400.0 | Available: true]

Borrowed Books:

[Java Basics by James | Genre: Programming | Price: 350.0 | Available: false]