Library Management System - Java OOP Case Study

This document presents a Java-based Library Management System designed using Object-Oriented Programming (OOP) principles. It includes core class designs, inheritance hierarchy, interfaces, polymorphism, exception handling, UML diagram, and source code.

Main Classes and Design

1. LibraryItem (Abstract Class)
   - Attributes: id, title, author, isAvailable
   - Methods: borrowItem(), returnItem(), calculateLateFee(int), getItemType(), matches(String)
   - Relationship: Base class for Book, Magazine, Journal

2. Book, Magazine, Journal (Subclasses)
   - Additional Attributes: genre, issueNumber, etc.
   - Inherit from LibraryItem
   - Implement late fee calculations differently

3. Member
   - Attributes: memberId, name, borrowedItems
   - Methods: borrowItem(), returnItem()

4. Library
   - Attributes: items, members
   - Methods: addItem(), registerMember(), searchItem(), borrowItem()

5. Searchable (Interface)
   - Method: matches(String keyword)

6. Custom Exceptions
   - BookNotAvailableException, MemberNotRegisteredException, LateReturnException

OOP Principles in Design

- Encapsulation: Private variables with getters/setters.

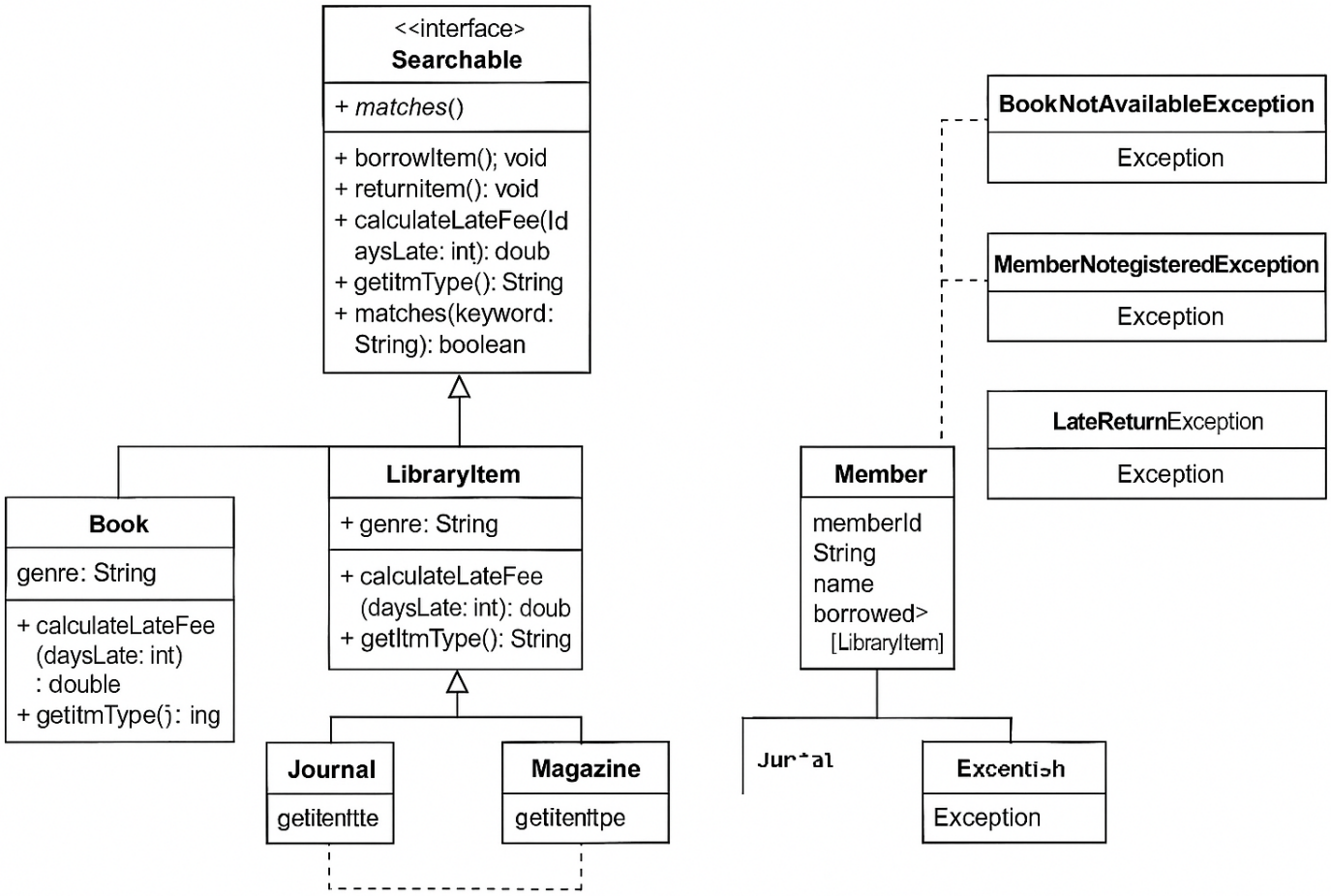
- Inheritance: Book, Magazine, Journal inherit from LibraryItem.

- Polymorphism: calculateLateFee() and borrowItem() use base type.

- Abstraction: LibraryItem is abstract, and Searchable is an interface.

UML Class Diagram

Visual representation of the system's structure.



Book.java

```
public class Book extends LibraryItem {
    private String genre;

    public Book(String id, String title, String author, String genre) {
        super(id, title, author);
        this.genre = genre;
    }

    public String getGenre() {
        return genre;
    }

    @Override
    public double calculateLateFee(int daysLate) {
        return daysLate * 1.5;
    }

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

LibraryItem.java

```
public abstract class LibraryItem implements Searchable {
    private String id;
    private String title;
    private String author;
    private boolean isAvailable = true;

    public LibraryItem(String id, String title, String author) {
        this.id = id;
        this.title = title;
        this.author = author;
    }

    public boolean isAvailable() {
        return isAvailable;
    }

    public void borrowItem() throws BookNotAvailableException {
            if (!isAvailable) throw new BookNotAvailableException("Item is already
borrowed.");
        isAvailable = false;
    }

    public void returnItem() {
        isAvailable = true;
    }

    public abstract double calculateLateFee(int daysLate);
    public abstract String getItemType();

    @Override
    public boolean matches(String keyword) {
        return title.contains(keyword) || author.contains(keyword);
    }
}
```

Magazine.java

```
public class Magazine extends LibraryItem {
    public Magazine(String id, String title, String author) {
        super(id, title, author);
    }

    @Override
    public double calculateLateFee(int daysLate) {
        return daysLate * 1.0;
    }

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

Journal.java

```
public class Journal extends LibraryItem {
    public Journal(String id, String title, String author) {
        super(id, title, author);
    }

    @Override
    public double calculateLateFee(int daysLate) {
        return daysLate * 2.0;
    }

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

Member.java

```
public class Member {
    private String memberId;
    private String name;
    private List<LibraryItem> borrowedItems = new ArrayList<>();

    public Member(String memberId, String name) {
        this.memberId = memberId;
        this.name = name;
    }

    public void borrowItem(LibraryItem item) throws BookNotAvailableException {
        item.borrowItem();
        borrowedItems.add(item);
    }

    public void returnItem(LibraryItem item, int daysLate) throws LateReturnException {
        item.returnItem();
        double fee = item.calculateLateFee(daysLate);
        if (daysLate > 0) throw new LateReturnException("Late return. Fee: Rs." + fee);
        borrowedItems.remove(item);
    }
}
```

Searchable.java

```
public interface Searchable {
    boolean matches(String keyword);
}
```

CustomExceptions.java

```
public class BookNotAvailableException extends Exception {
    public BookNotAvailableException(String msg) {
        super(msg);
    }
}

public class MemberNotRegisteredException extends Exception {
    public MemberNotRegisteredException(String msg) {
        super(msg);
    }
}

public class LateReturnException extends Exception {
    public LateReturnException(String msg) {
        super(msg);
    }
}
```