# Project Summary

We will create a simple Library Management System in Java. This project will cover the main OOP concepts such as encapsulation, inheritance, polymorphism, and abstraction. Additionally, we'll incorporate exception handling to manage errors gracefully. The Library Management System will allow users to add books, issue books to members, return books, and view the list of books. The system will be console-based to keep it straightforward.

## Step-by-Step Instructions

### Step 1: Setup Your Project

Install Java Development Kit (JDK):

Ensure that you have the JDK installed on your computer. You can download it from the Oracle website.

#### Set Up Your Development Environment:

Use an Integrated Development Environment (IDE) like IntelliJ IDEA, Eclipse, or NetBeans. For this example, we'll use IntelliJ IDEA.

Create a new Java project in your IDE.

### Step 2: Create the Basic Structure

#### Create Packages:

model: Contains the classes representing the entities in our system.

service: Contains the classes responsible for the business logic.

exception: Contains custom exception classes.

main: Contains the main class to run the application.

### Step 3: Create the Model Classes

#### Create Book Class:

Add private fields for title, author, ISBN, and isIssued.

Create a constructor to initialize these fields.

Provide getters and setters for each field.

Override the toString method to return a string representation of the book.

#### Create Member Class:

Add private fields for name and memberId.

Create a constructor to initialize these fields.

Provide getters and setters for each field.

Override the toString method to return a string representation of the member.

### Step 4: Create Service Classes

#### Create LibraryService Class:

Create private lists to hold Book and Member objects.

Implement methods to add books and members to these lists.

Implement a method to issue a book to a member, which checks if the book is available and marks it as issued.

Implement a method to return a book, which marks the book as not issued.

Implement a method to list all books.

### Step 5: Create Custom Exception

#### Create BookNotAvailableException Class:

Extend the Exception class.

Create a constructor that takes a message and passes it to the superclass constructor.

### Step 6: Create the Main Class

#### Create LibraryManagementSystem Class:

Instantiate the LibraryService class.

Add some books and members to the library.

List all books to display the current inventory.

Implement the logic to issue a book to a member and handle exceptions if the book is not available.

Implement the logic to return a book and update its status.

List all books again to show the updated inventory.

### Step 7: Run the Application

#### Compile and Run:

Compile the project using your IDE's build tools.

Run the LibraryManagementSystem class to see the output in the console.

Step 8: Test and Expand

Test the Functionality:

Test various scenarios like issuing non-existent books, returning books, etc.

### Expand the Project:

Add more features like searching for books, removing books, etc.

Implement more exception handling for different error cases.

By following these steps, you will have a basic Library Management System in Java that utilizes OOP principles and exception handling. This project can be expanded further to include more advanced features and functionalities.