VIETNAMESE – GERMAN UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER SCIENCE DEPARTMENT

JAVA PROJECT REPORT

BOOK MANAGEMENT SYSTEM

Module 61CSE215: Object Oriented Programming in JAVA

1. Đoàn Lê Gia Bảo - 10423008
2. Nguyễn Văn Công - 10423020

Lecturer: Dr. Tran Hong Ngoc

VGU, WS2025

Abbreviation

OOP	Object-Oriented Programming
MySQL	My Structured Query Language
PC	Personal Computer
CPU	Central Processing Unit
RAM	Random-Access Memory
JDBC	Java Database Connectivity

List of Figures

List of Tables

Table of Contents

I. INTRODUCTION/BUSINESS REQUIREMENT

In this project, we investigate the business functions of the Book Management System, and provide a desktop application supporting users (librarians or bookshop staff) in book management.

The system objective is to simplify the process of managing book information, tracking borrowers, and maintaining a structured and organized database, It helps reduce manual errors, improve accessibility of information, and enhance productivity for book management operations.

The full source code for this project is available at: *OurProjectGithubLink*

This project provides the basic functions for the users as follows:
- Input book information
- Search and view the status of books
- Manage customers
- Manage borrowing and buying

User roles

Role	Description
Admin	Has full control over the system. Can view/purchase/borrow/add/update/remove books, view staff and customers.
Librarian	Can view/purchase/borrow/add/update/remove the book information.
Customer	Can view/purchase/borrow the list of available books

Table 1: Show all the user roles

II. CLASS ANALYSIS

1. Objects

No	Object Name	State	Behaviours
1	A00 – System Admin	username – “admin”	camManageStaff(), canManageBooks(),

		password – “admin123” role – “ADMIN”	viewBooks(), borrow(), paid()
2	L00 – Librarian	username – “library” password – “library123” role – “LIBRARIAN”	canMangeBooks(), viewBooks(), borrow(), paid()
3	M00 - Customer	username – “member” password – “member123” role – “MEMBER”	viewBooks(), paid(), borrow()
4	B00 – Book	title – “Feynmann’s Lectures” author – “Richard Feynmann” numAvailable – 3 price – 3.0	getTitle(), canBorrow(), getAuthor(), getPrice(), getNumAvailable(), setTitle(), setAuthor(), setPrice(), setNumAvailable(), displayInfo()
5	B01 – Book	title – “Principia Mathematica” author – “Issac Newton” numAvailable – 4 price – 3.0	getTitle(), canBorrow(), getAuthor(), getPrice(), getNumAvailable(), setTitle(), setAuthor(), setPrice(),

			setNumAvailable(), displayInfo()
6	B02 – Book	title – Myth of Sisphus author – Albert Camus numAvailable – 5 price – 2.0	getTitle(), canBorrow(), getAuthor(), getPrice(), getNumAvailable(), setTitle(), setAuthor(), setPrice(), setNumAvailable(), displayInfo()

Table 2: Show all the objects

2. Classes

No	Class Name	Objects	Attributes	Methods
1	Person (abstract)	(base for Admin, Librarian, Member)	username, password, role	getUsername(), getPassword(), getRole(), canManageBooks(), canManageStaff(), viewBooks(), borrow(), paid()
2	Admin	A00 - System Admin		canMangeBooks(), canManageStaff()
3	Librarian	L00 – Librarian		canMangeBooks(), canManageStaff(),
4	Member	M00 – Member		canMangeBooks(), canManageStaff(),

5	LibraryItem	(base for Book and Magazine)		getTitle(), getNumAvailable(), setTitle(), setNumAvailable() displayInfo()
6	Book	B00	title, author, numAvailable, price	getTitle(), canBorrow(), getAuthor(), getPrice(), getNumAvailable(), setTitle(), setAuthor(), setPrice(), setNumAvailable(), displayInfo()
7	Magazine	MA00	title, author, numAvailable, price	getTitle(), canBorrow(), getAuthor(), getPrice(), getNumAvailable(), setTitle(), setAuthor(), setPrice(), setNumAvailable(), displayInfo()

Table 3: Show all the classes

III. CLASS DESIGN

1. Classes

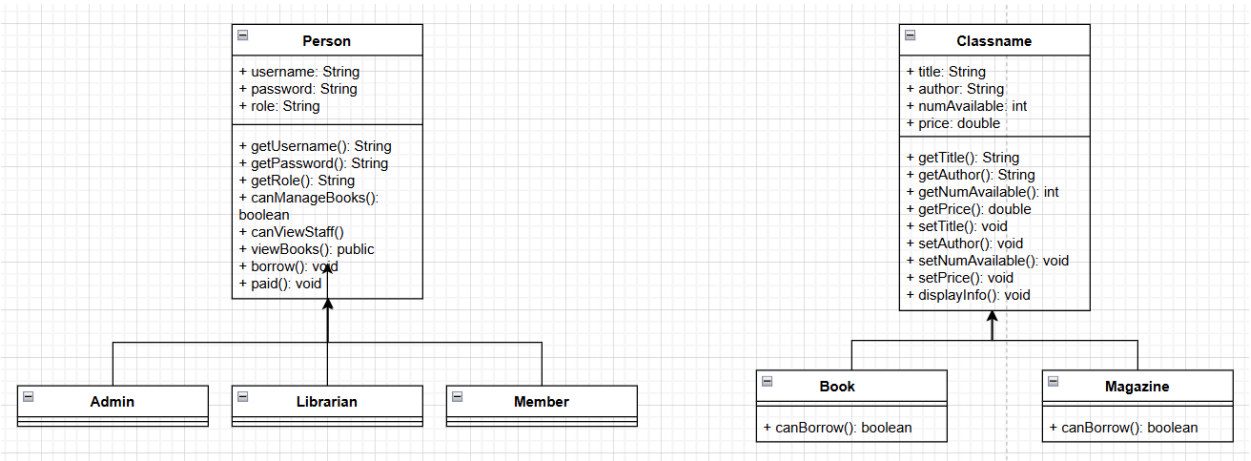

Figure 1: Show the relationship among classes

No	Class	Instance Variable	Methods	Description
1	Person	1. private String username 2. private String password 3. private String role	1. public String getUsername() 2. public String getPassword() 3. public String getRole() 4. public abstract boolean canManageBooks() 5. public abstract boolean canViewStaff() 6. public void viewBooks() 7. public void borrow() 8. public void paid()	This class is used as abstract class and is inherited by Admin, Librarian, Member classes

2	Admin		1. public boolean canManageBooks() 2. public boolean canViewStaff() 	This class is used for viewing staff information and viewing/adding/updating/deleting library items
3	Librarian		1. public boolean canManageBooks() 2. public boolean canViewStaff() 3.	This class is used for viewing/adding/updating/deleting library items
4	Member		1. public boolean canManageBooks() 2. public boolean canViewStaff() 	This class is used for viewing/buying/borrowing the book
5	LibraryItem	1. protected String title 2. protected String author 3. protected int numAvailable 4. protected double price	1. public abstract String getTitle() 2. public abstract String getAuthor() 3. public abstract int getNumAvailable() 4. public abstract int getPrice() 5. abstract void setTitle() 6. abstract void setAuthor() 7. abstract void setAuthor()	This class is used as abstract class and is inherited by Book and Magazine classes

			8. abstract void setPrice() 9. public abstract void displayInfo()	
6	Book		1. public boolean canBorrow() 2. public String getTitle() 3. public String getAuthor() 4. public int getNumAvailable() 5. public int getPrice() 6. void setTitle() 7. void setAuthor() 8. void setAuthor() 9. void setPrice() 10. public void displayInfo()	This class is used to store the information of the book
7	Magazine		1. public boolean canBorrow() 2. public String getTitle()	This class is used to store the information of the magazine

			3. public String getAuthor() 4. public int getNumAvailable() 5. public int getPrice() 6. void setTitle() 7. void setAuthor() 8. void setAuthor() 9. void setPrice() public void displayInfo()	

Table 4: Show the detailed classes

- Abstract classes

No	Abstract Class	Abstract Methods	Concrete Methods	Decription
1	Person	public canManageBooks(), public canManageStaff()	public getUsername(), public getPassword(), public getRole() public viewBooks() public borrow()	This class is an abstract class that is used by subclasses Admin, Librarian, and Member.

			public paid()	
2	LibraryItem	public getTitle(), public getNumAvailable(), public setTitle(), default setNumAvailable(), public displayInfo()		This class is an abstract class that is used by subclasses Book and Magazine.

Table 5: Show the abstract classes

2. Some OOP techniques

Overloading method:

- viewBooks() method in abstract class Person
- Contrucstors in LibraryItem, Book, and Magazine classes

Overriding method:

- canMangeBooks() and canManageStaff() methods in class Admin
- canMangeBooks() and canManageStaff() methods in class Librarian
- canMangeBooks() and canManageStaff() methods in class Member

Inheritance

- We defined Person class as abstract to contain all common fields (username, password, and role) and common behaviors (getUsername(), getPassword(), getRole(), canManageBooks(), canManageStaff(), viewBooks(), borrow(), paid()).
- The Admin, Librarian and Member classes inherit all public and protected attributes and methods from Person class.
- We defined LibraryItem as abstract to contain all common behaviors (getTitle(), getNumAvailable(), setTitle(), setNumAvailable(), displayInfo()).
- The Book and Magazine classes inherit all public and protected methods from LibraryItem class.

IV. Package Design

- We use 2 packages, including models and GUI:
  - Model package contains all classes (Person, Admin, Librarian, Member, LibraryItem, Book, and Magazine).

- GUI package contains GUI class (BookstoreAdminApp, and LoginSystem)

V. Interface Design

- LoginSystem for handling login, validating and determine user type (Admin, Librarian, Member)
- BookstoreAdminApp for book management and staff viewing based on user type

VI. Access Control

Table X1: Data Access Control Table

No	Data	Class	Modifier	Description
1	username	Person	private	The username for log-in process
2	password	Person	private	The password for log-in process
3	role	Person	private	The role for the logged-in account (ADMIN/LIBRARIAN/MEMBER)
4	title	LibraryItem	protected	The title of a book/magazine
5	author	LibraryItem	protected	The author of a book/magazine
6	numAvailable	LibraryItem	protected	The number of the book/magazines are available
7	price	LibraryItem	protected	The price of a book/magazine

Table 6: Show data access control table

Table X2: Method Access Control Table

No	Method	Class	Modifier	Description
1	getUsername()	Person	public	Obtain the username of the account
2	getPassword()	Person	public	Obtain the password of the account

3	getRole()	Person	public	Obtain the role of the account
4	canManageBooks()	Person, Admin, Librarian, Member	public	Be the boolean method to check the account can manage books/magazine or not
5	canViewStaff()	Person, Admin, Librarian, Member	public	Be the boolean method to check the account can view or not
6	viewBooks()	Person, Admin, Librarian, Member	public	View the books/magazine available
7	paid()	Member	public	Customer is paid to the chosen book
8	borrow()	Member	public	Customer borrow the chosen book
9	getTitle()	LibraryItem, Book, Magazine	public	Obtain the title of the book/magazine
10	getAuthor()	LibraryItem, Book, Magazine	public	Obtain the author of the book/magazine
11	getNumAvailable()	LibraryItem, Book, Magazine	public	Obtain the number of books/magazines are available
12	getPrice()	LibraryItem, Book, Magazine	public	Obtain the price of the book/magazine
13	setTitle()	LibraryItem, Book, Magazine	default	Update the title of the book/magazine
14	setAuthor()	LibraryItem, Book, Magazine	default	Update the author of the book/magazine
15	setNumAvailable()	LibraryItem, Book, Magazine	default	Update the number of books/magazines available
16	setPrice()	LibraryItem, Book, Magazine	default	Update the price of the book/magazine
17	displayInfo()	LibraryItem, Book, Magazine	public	Display the information of the book/magazine

18	canBorrow()	Books, Magazine	public	Check whether the book/magazine available or not

Table 7: Show the method access control table

VII. Encapsulation vs Inheritance vs Polymorphism

1. Encapsulation

Encapsulation means hiding internal data and only allowing a few specific methods to access or modify these data.

```
package models;

import javax.swing.*;
import java.sql.*;

public abstract class Person {

    protected static final String DB_URL  =
"jdbc:mysql://localhost:3306/java";
    protected static final String DB_USER = "java";
    protected static final String DB_PASS = "java";

    private String username;
    private String password;
    private String role;

    public Person(String username, String password, String
role) {
        this.username = username;
        this.password = password;
        this.role = role;
    }

    public String getUsername() { return username; }
```

```
    public String getPassword() { return password; }
    public String getRole() { return role; }
```

Figure 2: The Definition of abstract class Person

Important data (username, password, and role) cannot be accessed directly from outside.

2. Inheritance

Inheritance allows one class to inherit attributes and methods from another class.

```
public abstract class Person {

    protected static final String DB_URL  =
"jdbc:mysql://localhost:3306/java";
    protected static final String DB_USER = "java";
    protected static final String DB_PASS = "java";

    private String username;
    private String password;
    private String role;

    public Person(String username, String password, String
role) {
        this.username = username;
        this.password = password;
        this.role = role;
    }

    public String getUsername() { return username; }
    public String getPassword() { return password; }
    public String getRole() { return role; }
```

```
public class Admin extends Person {

    public Admin(String username, String password) {
        super(username, password, "ADMIN");
    }

    @Override
    public boolean canManageBooks() { return true; }

    @Override
    public boolean canViewStaff() { return true; }
}
```

```
package models;

public class Librarian extends Person {

    public Librarian(String username, String password) {
        super(username, password, "LIBRARIAN");
    }

    @Override
    public boolean canManageBooks() { return true; }

    @Override
    public boolean canViewStaff() { return false; }
}
```

```
package models;

import javax.swing.*;
```

```
import java.sql.*;
import models.*;

public class Member extends Person {
    public Member(String username, String password) {
        super(username, password, "MEMBER");
    }

    @Override
    public boolean canManageBooks() { return false; }

    @Override
    public boolean canViewStaff() { return false; }
}
```

Figure 3: The Definition of abstract class Person, and inherited classes (Admin, Librarian, and Member)

Admin, **Librarian**, and **Member** inherit common attributes (username, password, and role) and methods (getUsername(), getPassword(), …)

3. Polymorphism

Polymorphism allows different classes to provide different implementations of the same method.

```
public void viewBooks() {
        String sql = "SELECT * FROM books ORDER BY id ASC";

        try (Connection conn =
DriverManager.getConnection(DB_URL, DB_USER, DB_PASS);
             PreparedStatement stmt =
conn.prepareStatement(sql);
             ResultSet rs = stmt.executeQuery()) {

            StringBuilder sb = new StringBuilder("All
Books:\n\n");
```

```
            while (rs.next()) {
                sb.append(rs.getInt("id")).append(" - ")
                  .append(rs.getString("title")).append(" (")
                  .append(rs.getString("author")).append(") -
$")

.append(rs.getDouble("price")).append("\n");
            }

            JOptionPane.showMessageDialog(null,
sb.toString());

        } catch (Exception ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(null, "Error
retrieving books.");
        }
    }
```

```
public void viewBooks(double maxPrice) {
        String sql = "SELECT * FROM books WHERE price <= ?
ORDER BY price ASC";

        try (Connection conn =
DriverManager.getConnection(DB_URL, DB_USER, DB_PASS);
             PreparedStatement stmt =
conn.prepareStatement(sql)) {

            stmt.setDouble(1, maxPrice);

            try (ResultSet rs = stmt.executeQuery()) {
```

```
                StringBuilder sb = new StringBuilder(
                        "Books priced up to $" + maxPrice +
":\n\n"
                );

                boolean found = false;

                while (rs.next()) {
                    found = true;
                    sb.append(rs.getInt("id")).append(" - ")
                      .append(rs.getString("title")).append("
(")

.append(rs.getString("author")).append(") - $")

.append(rs.getDouble("price")).append("\n");
                }

                if (!found) {
                    JOptionPane.showMessageDialog(null,
                            "No books found under $" +
maxPrice);
                } else {
                    JOptionPane.showMessageDialog(null,
sb.toString());
                }
            }

        } catch (Exception ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(null, "Error
retrieving books.");
        }
    }
```

```
@Override
    public boolean canManageBooks() { return true; }

    @Override
    public boolean canViewStaff() { return true; }
```

```
    @Override
    public boolean canManageBooks() { return true; }

    @Override
    public boolean canViewStaff() { return false; }
```

```
@Override
    public boolean canManageBooks() { return false; }

    @Override
    public boolean canViewStaff() { return false; }
}
```

Figure 4: The Overloading & Overriding methods in Admin, Librarian, and Member classes

The same method name (canMangeBooks() and canViewStaff()), but different outputs

VIII. Experiment

1. Environment and Tools

a. **Environment:**
   - 1 PC (8 CPU cores, 1 RAM)

b. **Tools**:
   - MySQL server
   - JDBC
   - Java Swing library

2. Project functions

- List and describe a bit the functions of the project
- There are 3 types of user (Admin, Librarian, Member):
  + Admin can manage books and view staff without restriction
  + Librarian can only manage books
  + Member can only view books
  + Everyone can view books and filter by price
- User is prompted a login screen and need to enter the correct credentials in order to continue

3. Database

Magazines

ID	Title	Author	Price	Quantity

Books

ID	Title	Author	Price	Quantity

Staff

ID	Name	Role

Member

ID	Name	Items_borrowed	Items_paid

Table 8: MySQL Database

Magazines

Column Name	Data Type	Description
ID	INT	Unique identifier of each magazine
Title	VARCHAR(50)	Title of each magazine
Author	VARCHAR(50)	Name of magazine's author

Price	DOUBLE	Price of each magazine
Quantity	INT	Quantity of each magazine

Books

Column Name	Data Type	Description
ID	INT	Unique identifier of each book
Title	VARCHAR(50)	Title of each book
Author	VARCHAR(50)	Name of book's author
Price	DOUBLE	Price of each book
Quantity	INT	Quantity of each book

Staff

Column Name	Data Type	Description
ID	INT	Unique identifier of each staff
Name	VARCHAR(50)	Name of each staff
Role	VARCHAR(50)	Role of each staff (Admin/Librarian)
Items_managed	INT	ID of item being managed (Book/Magazine)

Member

Column Name	Data Type	Description
ID	INT	Unique identifier of each member
Name	VARCHAR(50)	Name of each member
Items_borrowed	INT	ID of item borrowed

		(Book/Magazine)
Items_paid	INT	ID of item that's been paid

Table 9: Data Dictionary

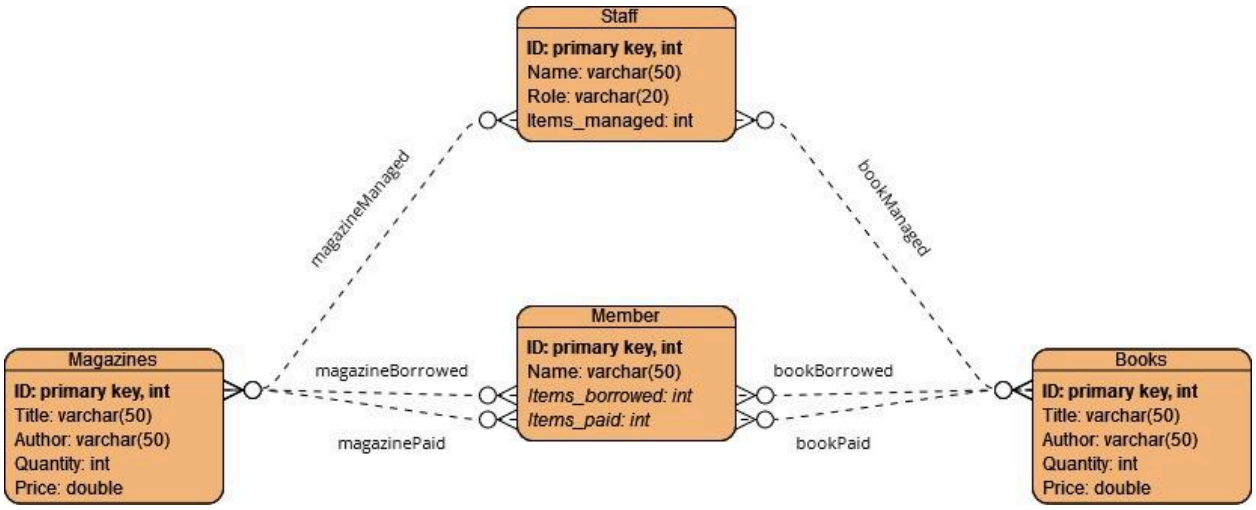

Figure 5: Entity-Relationship Diagram

~~4.~~ GUI

Order	Name
1	LoginSystem
2	BookstoreAdminApp

Figure 6: LoginSystem

User need to enter the right credentials and press the login button in order to transition to BookstoreAdminApp

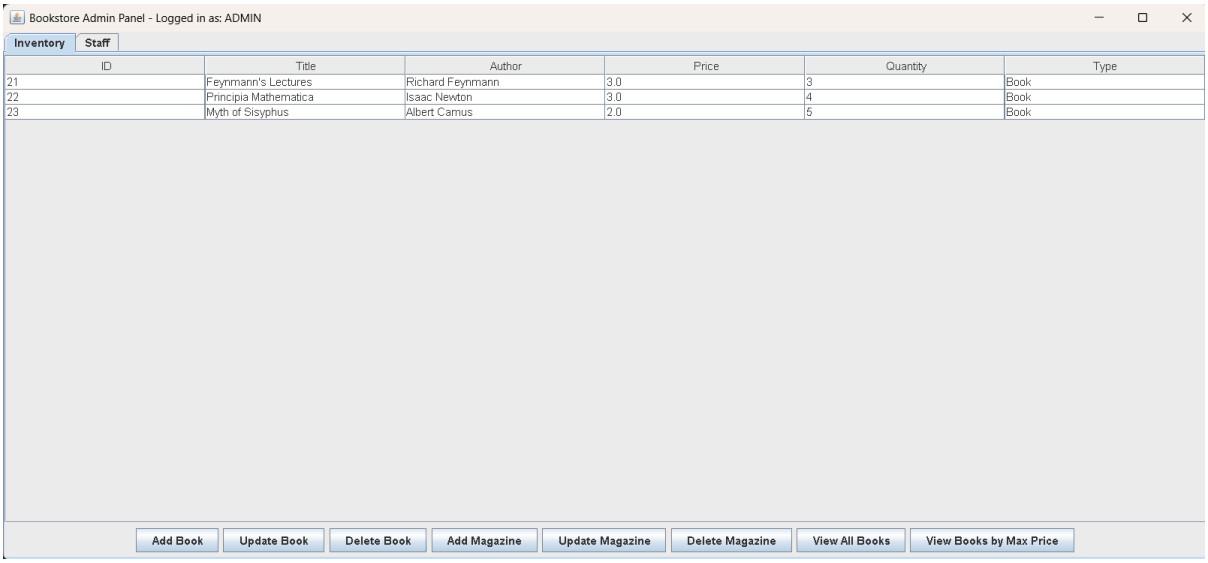

Figure 7: BookstoreAdminApp, Book Panel

On this BookstoreAdminApp screen, you can interact with books or magazines:

- Add, update or delete a book in the by pressing “Add Book”, “Update Book”, “Delete Book” buttons
  Ex: Select “Feynmann’s Lectures” and press “Update Book” to modify its value
- Add, update or delete a magazine in the database by pressing “Add Magazine”, “Update Magazine”, “Delete Magazine” buttons
  Ex: Press “Add Magazine” to add a new entry of type Magazine

- Press “View All Books” or “View Books by Max Price” to view all the items or view all items up to a certain price in the database
  Ex: View all the books up to 2$

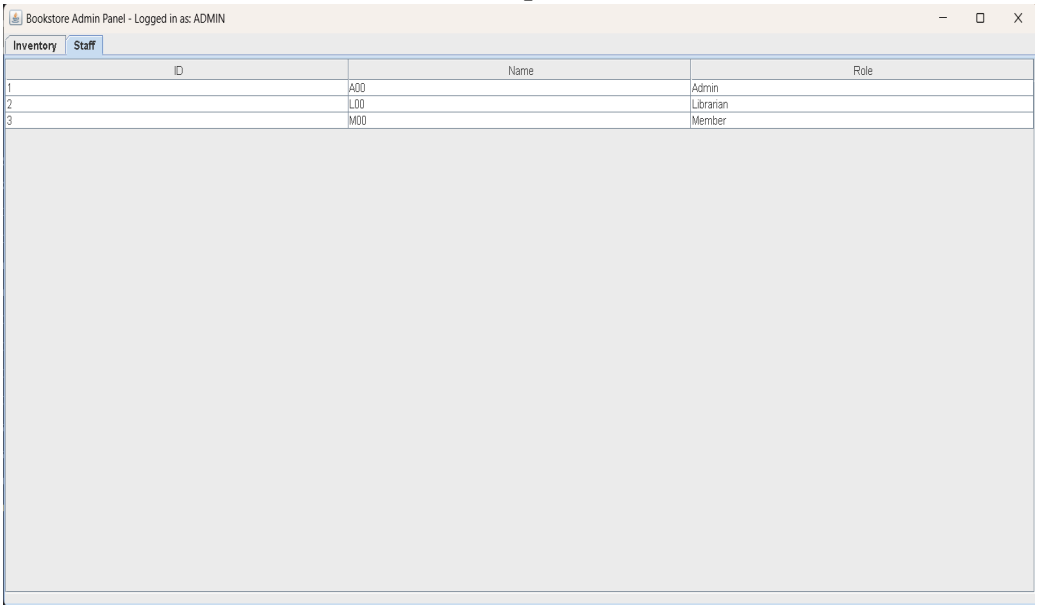

Figure 8: Staff Panel

As admin, you can switch to “Staff” panel in order to view all the staff information

IX. Conclusion

Pros and Cons

Feature	Pros	Cons
Design	Make use of abstract classes (Person, and LibraryItem), which promotes code reusability and maintainability.	less diversity in the attributes and methods of classes
Access Control	Protect sensitive data (username, password, and role) against directly modify from outside.	Cannot track which staff/member view/purchase/add/update/remove

User Interface	Provides a functional and user-friendly LoginSystem and Dashboard application.	Display many buttons or unnecessary buttons which can make user confused

We rate the project ourselves at around 9.0/10.0. Because The project successfully met all requirements and applied all core OOP principles in Java. Deductions are mainly come from the current simplicity of the GUI and the lack of diversity of classes.

DUTY ROSTER

ID	Task	In Charge	Start	End	State	Note
1	Requirements Analysis & Class Design	Bao & Cong	01-Nov	05-Nov	Done	Completed Class Analysis and Design tables
2	MySQL Database Setup	Nguyen Van Cong	06-Nov	08-Nov	Done	Created the Books and Staff tables
3	Core Model Classes	Doan Le Gia Bao	06-Nov	12-Nov	Done	Implemented Person, LibraryItem, Admin, Librarian, Member, Book, Magazine classes
4	GUI Development	Doan Le Gia Bao	13-Nov	15-Nov	Done	Completed the login interface
5	GUI Development: BookstoreAdminApp	Nguyen Van Cong	16-Nov	20-Nov	Done	Completed the main management interface (Book & Staff Panels)
6	JDBC Connection & Data Integration	Nguyen Van Cong	21-Nov	24-Nov	Done	Ensured the application could connect to and manage data in MySQL database

7	Report Writing	Bao & Cong	25-Nov	29-Nov	Done	Writing and final review of the report

REFERENCE

1. Java database connectivity with MySQL, Geeksforgeeks, 2025, https://www.geeksforgeeks.org/java/java-database-connectivity-with-mysql/
2. Java Swing tutorial, TutorialsPoint, https://www.tutorialspoint.com/swing/index.htm