

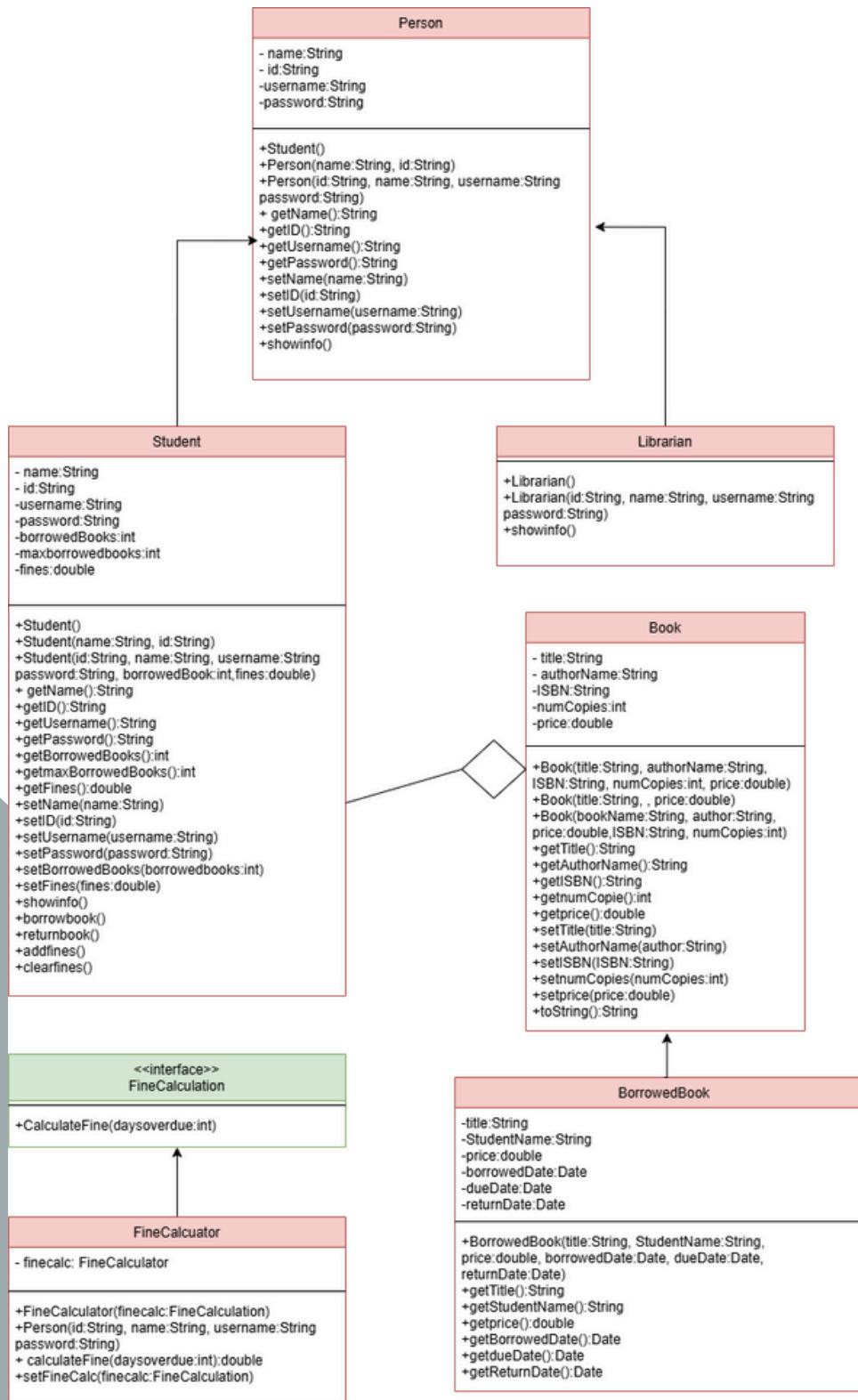
# **Library Management System**

**Class 5**

Abdelrahman Ahmed Mohamed	221000777
Mohamed badee	221000627
Judy Hossam	221000495
Rimas Emad Eldib	221001067

# Overview and UML

The aim of this project is to create a Library Management System that stores and manages student and librarian information, we have linked it to a database to make this possible. It also deals with the maintenance, management and organization of all the data relating to the books and the users.



# Design Pattern

This is a Factory Design Pattern, how does it work? The class (BookFactory) contains a static method (createBook) which acts as the factory method. Its purpose is to create and return Book objects. The (loadBooksFromDatabase) method retrieves book data from our database and uses the (createBook) method to make Book objects.

```
public class BookFactory {  
  
    public static Book createBook(String title, double price, int numCopies) { no usages  
        return new Book(title, author: null, price, isbn: null, numCopies);  
    }  
}
```

```
private void loadBooksFromDatabase() { 1 usage  
    String sql = "SELECT title, price, number_of_copies FROM books";  
    try (Connection connection = DatabaseConnection.connect();  
         PreparedStatement preparedStatement = connection.prepareStatement(sql);  
         ResultSet resultSet = preparedStatement.executeQuery()) {  
  
        while (resultSet.next()) {  
            String title = resultSet.getString(columnLabel: "title");  
            double price = resultSet.getDouble(columnLabel: "price");  
            int numCopies = resultSet.getInt(columnLabel: "number_of_copies");  
  
            // Use the Factory to create the Book object  
            Book book = BookFactory.createBook(title, price, numCopies);  
            books.add(book);  
        }  
    } catch (SQLException ex) {  
        showError("Database error: " + ex.getMessage());  
    }  
}
```

# Exceptions, Overloading,Overriding

```
private void loadBorrowedBooksFromDatabase() { 1 usage
    String sql = "SELECT title, student_name, price, borrowed_date, due_date FROM b
    try (Connection connection = DatabaseConnection.connect();
        PreparedStatement preparedStatement = connection.prepareStatement(sql);
        ResultSet resultSet = preparedStatement.executeQuery()) {

        while (resultSet.next()) {
            String title = resultSet.getString( columnLabel: "title");
            String studentName = resultSet.getString( columnLabel: "student_name");
            double price = resultSet.getDouble( columnLabel: "price");
            Date borrowedDate = resultSet.getDate( columnLabel: "borrowed_date");
            Date dueDate = resultSet.getDate( columnLabel: "due_date");

            System.out.println("Title: " + title + ", Student: " + studentName);

            BorrowedBook borrowedBook = new BorrowedBook(title, studentName, price,
                borrowedBooks.add(borrowedBook);
        }
    } catch (SQLException ex) {
        showError("Database error: " + ex.getMessage());
    }
}
```

```
@FXML 1 usage
public void handleAddBook() {
    try {
        FXMLLoader loader = new FXMLLoader(this.getClass().getResource( name: "/com,
        Stage stage = (Stage)this.addnewbookButton.getScene().getWindow();
        stage.setScene(new Scene((Parent)loader.load()));
        stage.setTitle("Add Book");
    } catch (IOException var3) {
        IOException ex = var3;
        this.showError("Failed to load the page: " + ex.getMessage());
    }
}

@FXML 1 usage
public void handleBorrowedBooks() {
    try {
        FXMLLoader loader = new FXMLLoader(this.getClass().getResource( name: "/com,
        Stage stage = (Stage)this.x.getScene().getWindow();
        stage.setScene(new Scene((Parent)loader.load()));
        stage.setTitle("Borrowed Books List");
    } catch (IOException var3) {
        IOException ex = var3;
        this.showError("Failed to load the page: " + ex.getMessage());
    }
}
```

```
public Book(String title, String authorName, String ISBN, int numCopies, double price
    this.title = title;
    this.authorName = authorName;
    this.ISBN = ISBN;
    this.numCopies = numCopies;
    this.price = price;
}

public Book(String title, double price) { 5 usages
    this.title = title;
    this.price = price;
}

public Book(String bookName, String author, double price, String isbn, int numberofCopies
    title=bookName;
    this.authorName=author;
    this.price=price;
    this.numCopies=numberofCopies;
}
```

```
public void showInfo() { no usages 1 override
    System.out.println("Name: " + getName());
    System.out.println("ID: " + getId());
}

}
```

```
7  >    public Librarian(String name, String username, String password, String id) { super(name,
10
11
12 @Override  no usages
13     public void showInfo() {
14         System.out.println("Name: " + getName());
15         System.out.println("ID: " + getId());
16         System.out.println("Username: " + getUsername());
17     }
18 }
```

# FXML

Library Management System

This screenshot shows the login interface for the Library Management System. It features a blue header with the system name. Below it is a white input field with a dropdown arrow. The next section is labeled "Username" with a text input field. Below that is a section labeled "Password" with a text input field. At the bottom are two orange rounded rectangular buttons: "Login" on the left and "Sign Up" on the right.

Available Books

This screenshot shows a list of available books. The title "Available Books" is at the top in bold black font. Below it is a large, empty rectangular area representing the book list. At the bottom right is a prominent orange rounded rectangular button labeled "Buy Book".

Welcome Librarian!!

This screenshot shows the librarian's dashboard. The title "Welcome Librarian!!" is at the top. Below it are three orange rounded rectangular buttons: "All Books" on the left, "Add New Book" in the center, and "Borrowed Books" on the right.

Add New Book

This screenshot shows the form for adding a new book. It includes fields for "Title" (with a "No of Copies" field to its right), "Author Name" (with a "Price" field to its right), and "ISBN". At the bottom is an orange rounded rectangular button labeled "Add Book".

Borrowed Books Log

This screenshot shows a table titled "Borrowed Books Log". The table has columns: Book, Student Name, Price, Borrowed Date, and Due Date. A message "No content in table" is displayed below the table.

Book	Student Name	Price	Borrowed Date	Due Date
No content in table				

All Books

This screenshot shows a table titled "All Books". The table has columns: Books, Price, and No of copies. A message "No content in table" is displayed below the table.

Books	Price	No of copies
No content in table		