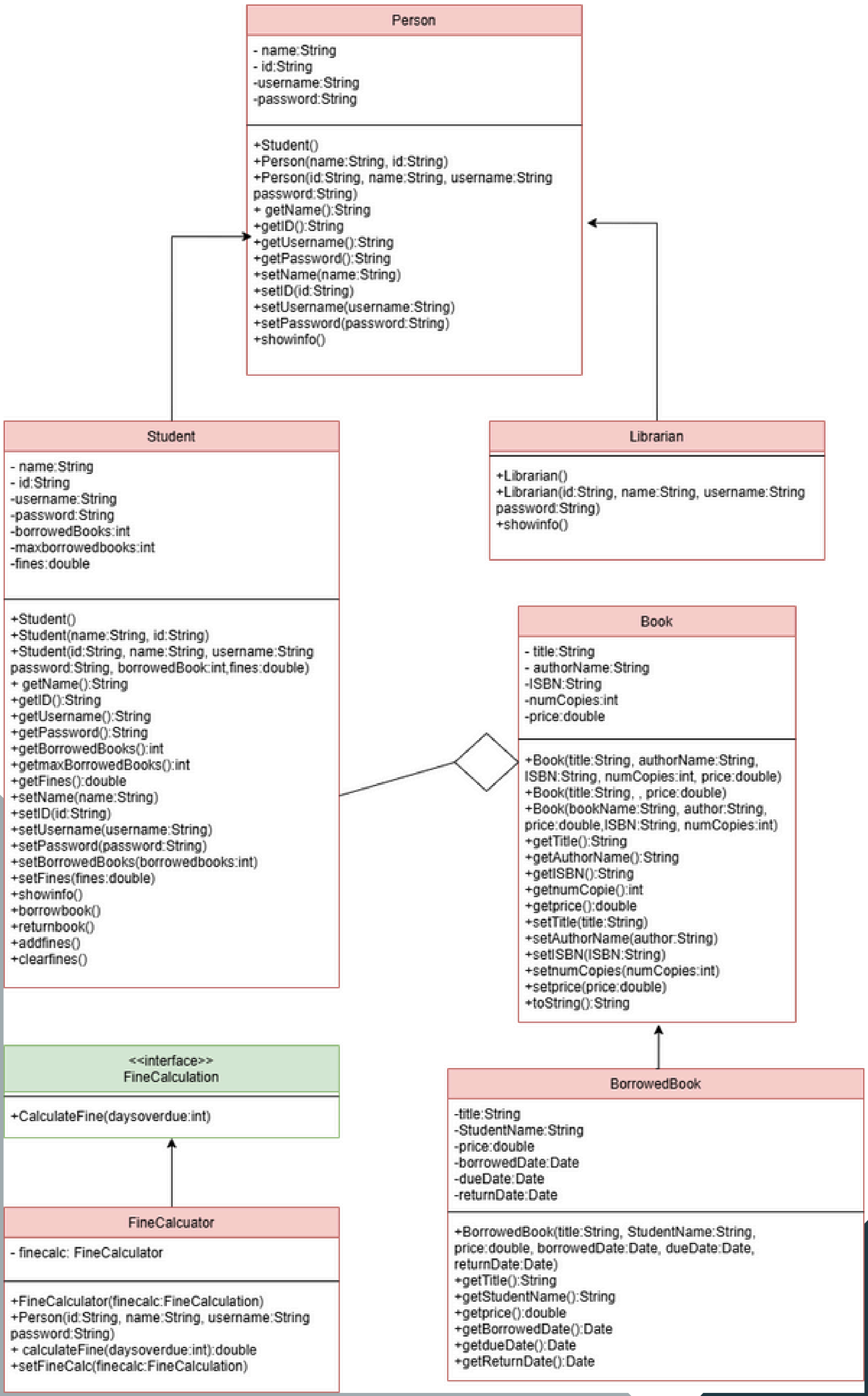
Library Management System

Class 5

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

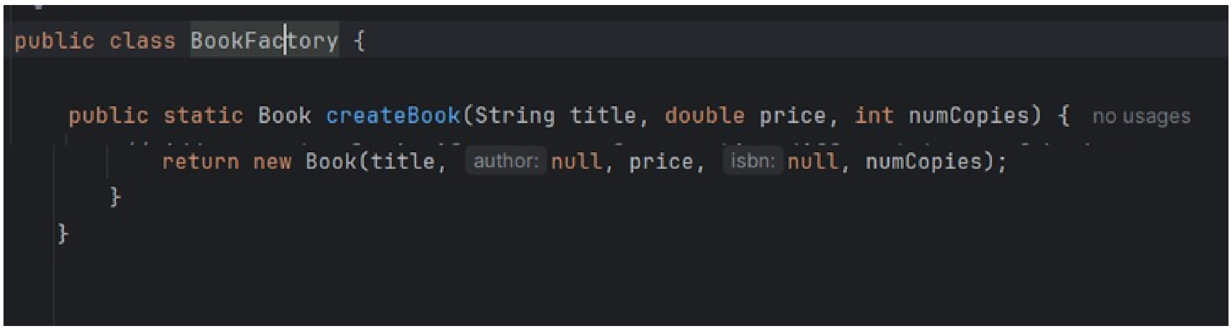
Overview and UML

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.

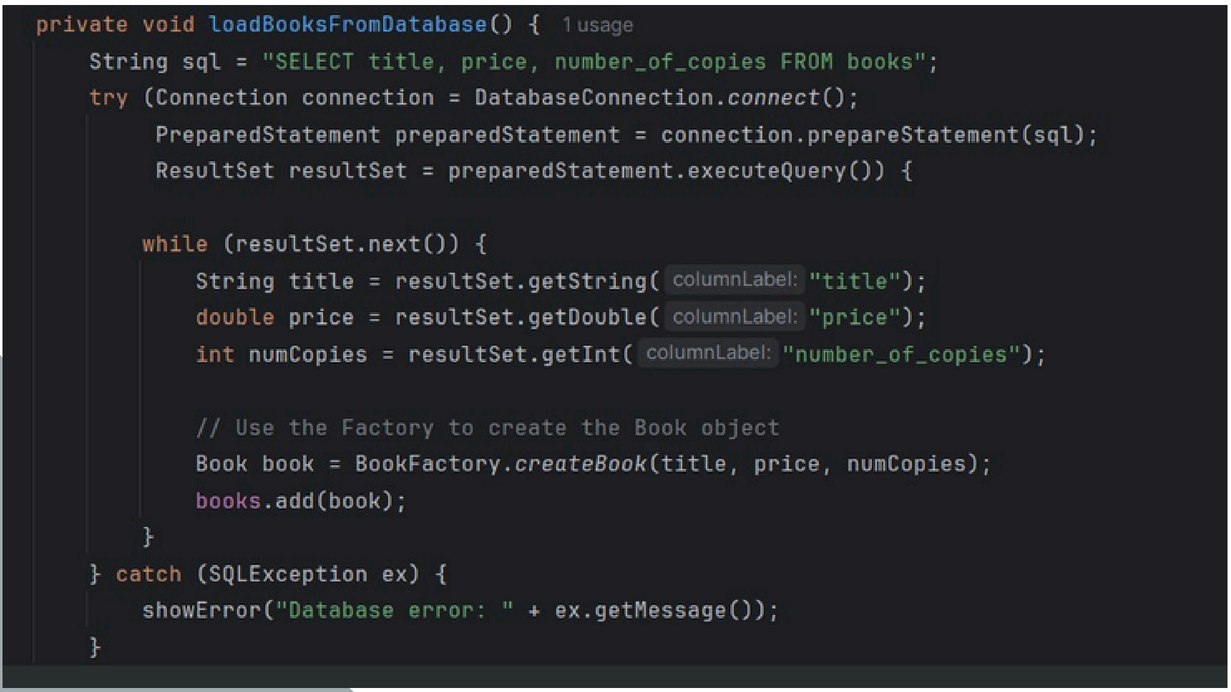


Design Pattern

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.

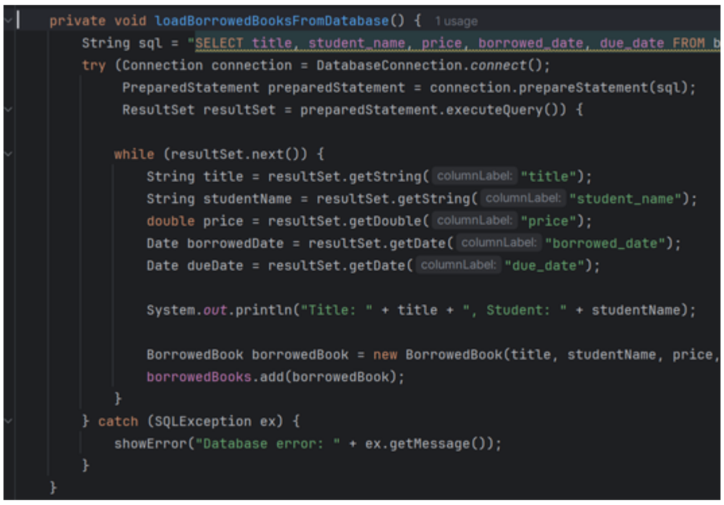


```
public class BookFactory {

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

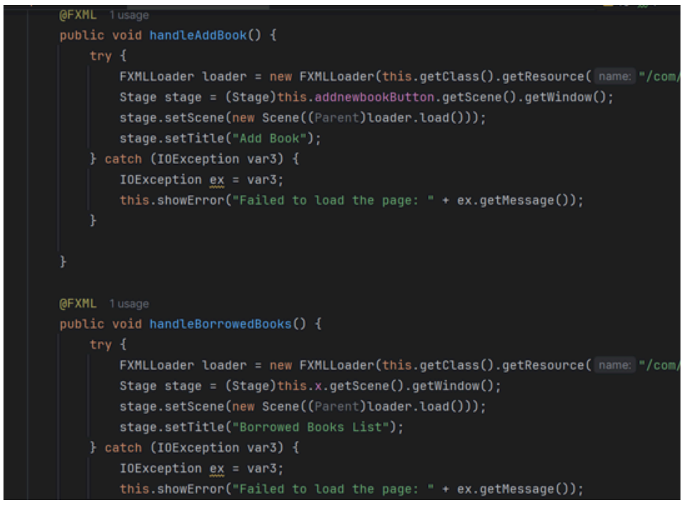


```
private void loadBooksFromDatabase() {  1 usage
    String sql = "SELECT title, price, number_of_copies FROM books";
    try (Connection connection = DatabaseConnection.connect();
         PreparedStatement preparedStatement = connection.prepareStatement(sql);
         ResultSet resultSet = preparedStatement.executeQuery()) {

        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());
    }
```

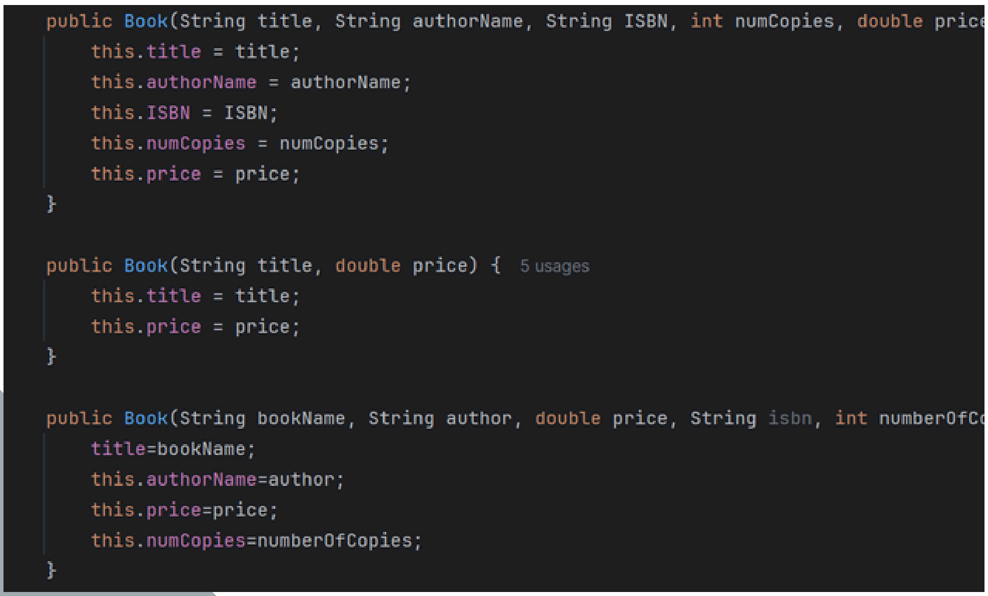
Exceptions, Overloading,Overriding



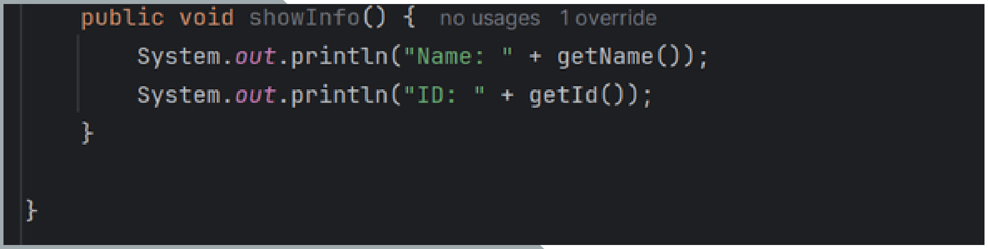
```java
private void loadBorrowedBooksFromDatabase() {  1 usage
    String sql = "SELECT title, student_name, price, borrowed_date, due_date FROM b
    try (Connection connection = DatabaseConnection.connect();
         PreparedStatement preparedStatement = connection.prepareStatement(sql);
        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



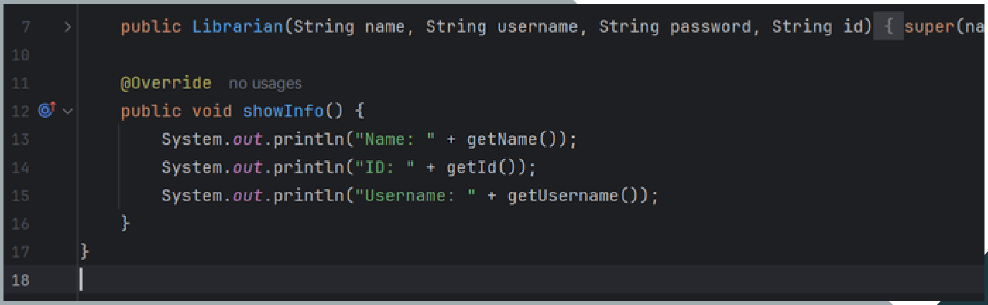
```java
@FXML  1 usage
public void handleAddBook() {
    try {
        FXMLLoader loader = new FXMLLoader(this.getClass().getResource( name: "/com/
        Stage stage = (Stage)this.addnewbookButton.getScene().getWindow();
        stage.setScene(new Scene((Parent)loader.load()));
        stage.setTitle("Add 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());
```



```java
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 numberOfCo
    title=bookName;
    this.authorName=author;
    this.price=price;
    this.numCopies=numberOfCopies;
}
```

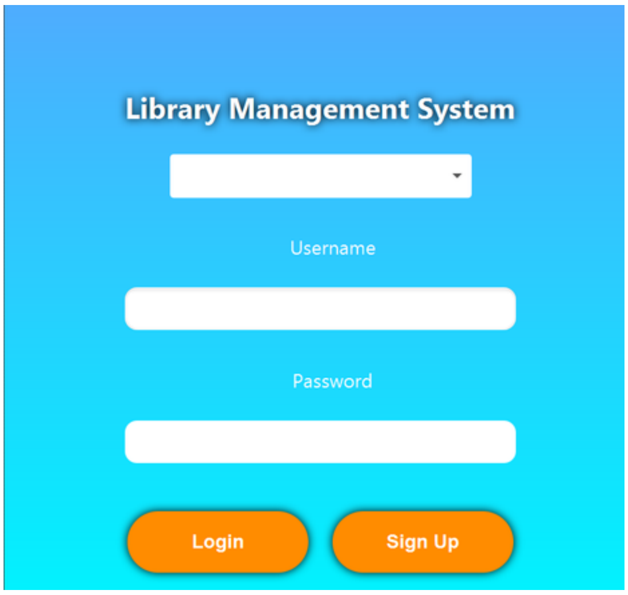


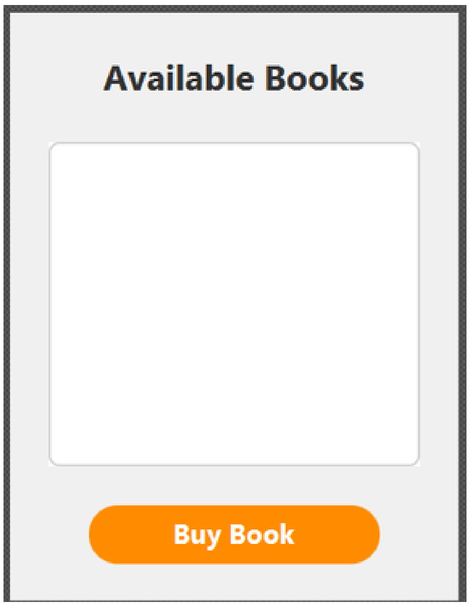
```java
    public void showInfo() {  no usages   1 override
        System.out.println("Name: " + getName());
        System.out.println("ID: " + getId());
    }


}
```

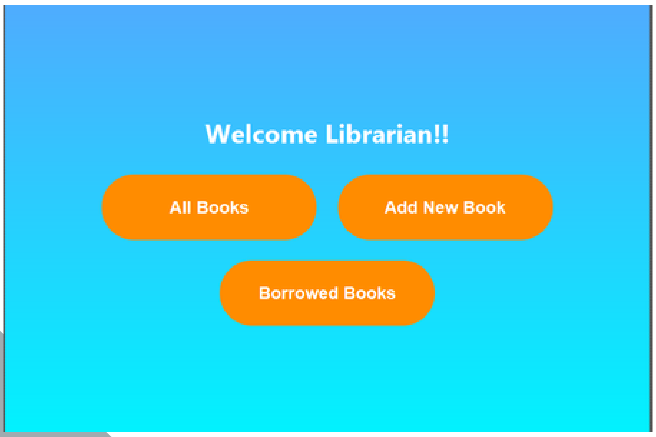


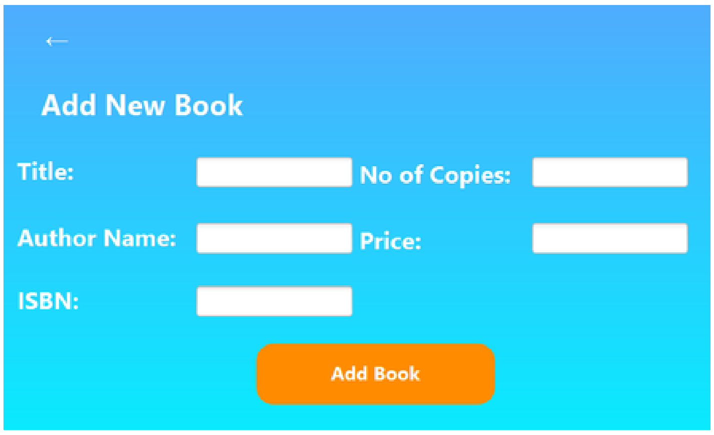
```java
    public Librarian(String name, String username, String password, String id) { super(na

    @Override  no usages
    public void showInfo() {
        System.out.println("Name: " + getName());
        System.out.println("ID: " + getId());
        System.out.println("Username: " + getUsername());
    }
}
```

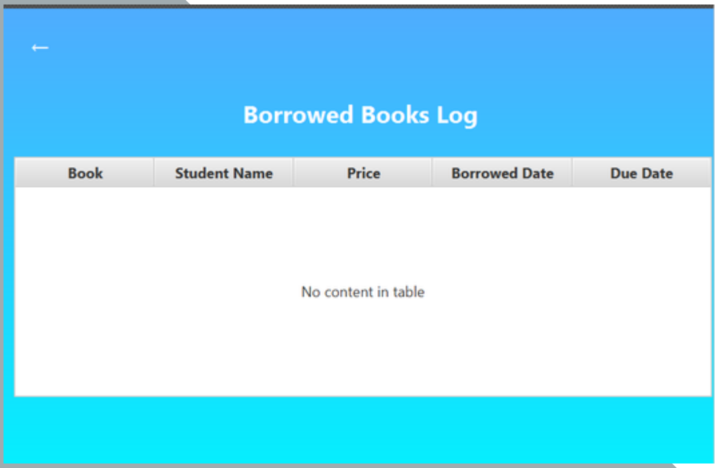
FXML





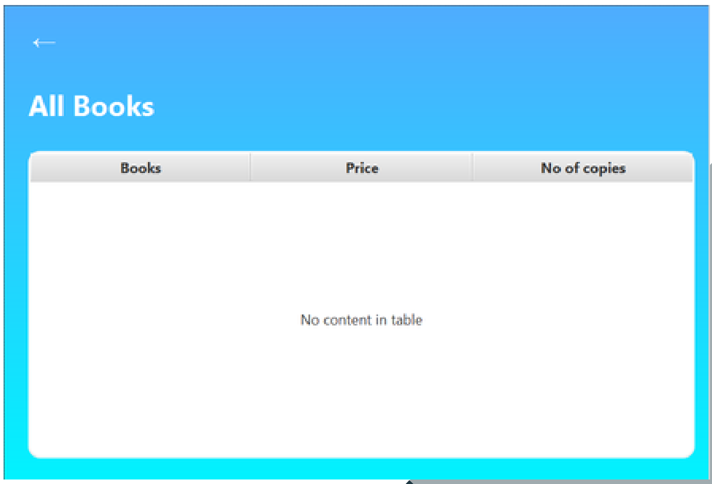






Book	Student Name	Price	Borrowed Date	Due Date

No content in table



Books	Price	No of copies

No content in table