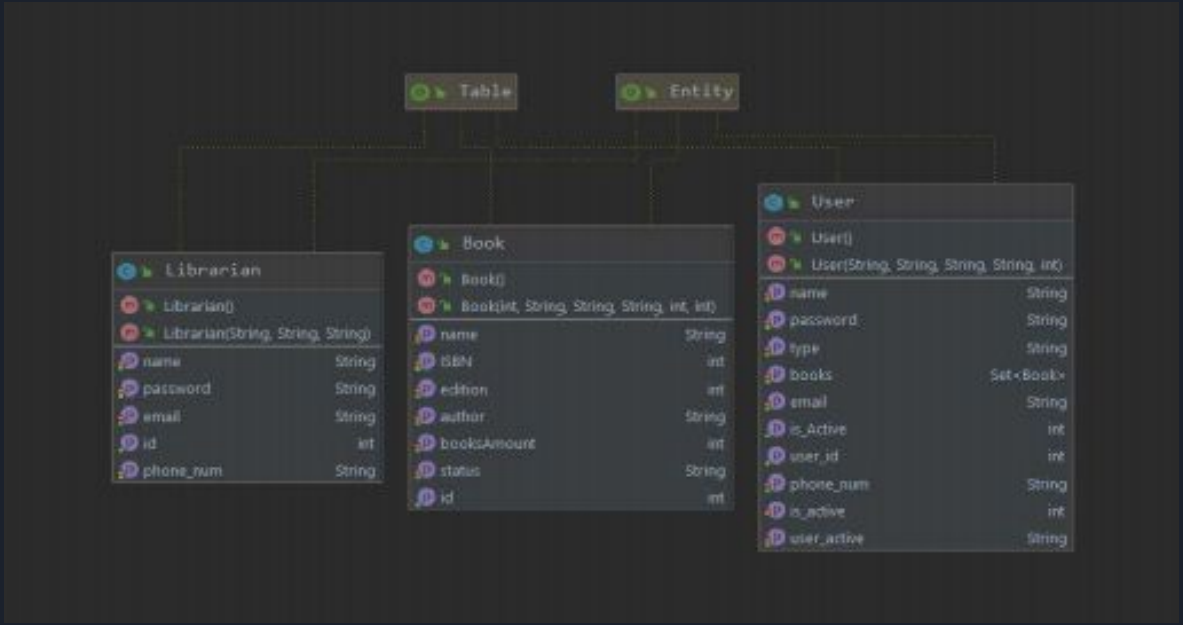
Creation Tutorial and Explanation

Arslan Sultanbek uulu
Suiorkul Abdykaiymov

Database Connection and UML Diagram



Connection to the database is made with Hibernate ORM which allowed to simplify the development process and database design.
Classes in this diagram represents tables in database, because of ORM every entity of table becomes the instance of that class and its very easy to manage these entities.

Main Classes and Methods

In library management system every functionality is divided into separate reusable code.

To give detailed explanation on each function we want to start from the beginning of Execution Procedure or Login Screen and give view of the thing that happening under the hood of Library Management System, this way you will get better understanding how specific functions are related and some nuances that need to covered, which will help for the future users or maintainers of this program.

Login

When user try to login with his credentials how its determined whether user is a librarian or an ordinary user. Login process consist of three main methods authenticate(), check_librarian(), check_user().

authenticate() executes two methods mentioned above if one them return True then users belongs to that category, after it sets User for this session and changes the view to Books view of each user category, if none of them is matching displays an error.

check_user() and check_librarian() are almost the same function but with difference in table that they make query and that user can be active or inactive.

```
private void authenticate(Event e) {
    // Check for User
    if (check_user()) {
        // Setting session user
        GUI.Session.user = sUser;
        // Changing to User Book views
        try {...} catch (Exception ex) {
            ex.printStackTrace();
        }
    // Check for Librarian
    } else if (check_librarian()) {
        // Setting session user
        GUI.Session.librarian = lLibrarian;
        // Changing to Librarian Book views
        try {...} catch (Exception ex) {
            ex.printStackTrace();
        }
    }
    //
    else {
        errorText.setText("Incorrect Email or Password");
    }

}
```

Check user

```java
private boolean check_librarian() {
    SessionFactory factory = new Configuration()
            .configure("hibernate.cfg.xml")
            .addAnnotatedClass(Librarian.class)
            .buildSessionFactory();

    Session session = factory.getCurrentSession();

    List<Librarian> librarians = new ArrayList<>();

    try {
        session.beginTransaction();
        // Taking data from emailField and Making Query in table librarians
        librarians = session.createQuery( = "from Librarian l where l.email=" + String.format("'%s'",emailField.getText()).list();

        session.getTransaction().commit();
        session.close();

    } catch (Exception e) {
        e.printStackTrace();
    }
    System.out.println("Checking librarian");
    for (Librarian librarian : librarians) {
        Pbkdf2PasswordEncoder pbkdf2PasswordEncoder = new Pbkdf2PasswordEncoder();

        if (pbkdf2PasswordEncoder.matches(passwordField.getText(), librarian.getPassword())){
            // If password matches declaring session user and returning True
            lLibrarian = librarian;
            return true;
        }
        return false;
    }

    return false;
}
}
```

Check Librarian

```java
private boolean check_librarian() {
    SessionFactory factory = new Configuration()
            .configure("hibernate.cfg.xml")
            .addAnnotatedClass(Librarian.class)
            .buildSessionFactory();

    Session session = factory.getCurrentSession();

    List<Librarian> librarians = new ArrayList<>();

    try {
        session.beginTransaction();
        // Taking data from emailField and Making Query in table librarians
        librarians = session.createQuery("from Librarian l where l.email='" + String.format("%s",emailField.getText()).list();

        session.getTransaction().commit();
        session.close();

    } catch (Exception e) {
        e.printStackTrace();
    }
    System.out.println("Checking Librarian");
    for (Librarian librarian : librarians) {
        Pbkdf2PasswordEncoder pbkdf2PasswordEncoder = new Pbkdf2PasswordEncoder();

        if (pbkdf2PasswordEncoder.matches(passwordField.getText(), librarian.getPassword())){
            // If password matches declaring session user and returning True
            lLibrarian = librarian;
            return true;
        }
        return false;
    }

    return false;
}
```

Signup

```
signUpBtn.setOnAction(e -> {
    if (nameField.getText().equals("") || typeField.getText().equals("")
            || phoneField.getText().equals("") || passField.getText().equals("")
            || pass2Field.getText().equals("") || emailField.getText().equals("")
    ) {
        raise_error();
    } else {
        if (!pass2Field.getText().equals(passField.getText())) {
            errorText.setText("Password are not matching");
        } else {
            SessionFactory factory = new Configuration()
                    .configure("hibernate.cfg.xml")
                    .addAnnotatedClass(User.class)
                    .addAnnotatedClass(Book.class)
                    .buildSessionFactory();

            Session session = factory.getCurrentSession();

            try {
                session.beginTransaction();
                // Setting empty book set
                Set<Book> bookSet = new HashSet<>();
                // Getting data from signup form and creating new user instance
                User user = new User(emailField.getText(), nameField.getText(), phoneField.getText(), typeField.getText(), IS_ACTIVE: 1);
                user.setBooks(bookSet);

                // Setting hashed password for user
                user.setPassword(passField.getText());

                session.persist(user);

                session.getTransaction().commit();
            } catch (Exception e1) {...}

            try {...} catch (Exception ex) {
                ex.printStackTrace();
            }
        }
```

If user wants to create account in LIS he or she must fill the signup form and submit it. Signup process is bound to Signup Button. Every time user clicks it function take data from fields validates it and if every thing is correct saves user to database.

Books

```
takeBtn.setOnAction(e -> {
    // Getting selected book from table
    Book book = booksTable.getSelectionModel().getSelectedItem();
    Alert alert;

    // If no book was selected showing alert box
    if (book == null) {
        alert = new Alert(Alert.AlertType.INFORMATION, contentText: "Please select one book above", ButtonType.YES);
        alert.showAndWait();
    } else {

        // If book was selected showing confirmation box
        alert = new Alert(Alert.AlertType.CONFIRMATION, contentText: "Do you want ot take this book:" + book + " ?", ButtonType.YES, ButtonType.CANCEL);
        alert.showAndWait();

        // If user agrees to take specific book
        if (alert.getResult() == ButtonType.YES) {

            // Amount of available books equals 0
            if (book.getBooksAmount() == 0) {
                alert = new Alert(Alert.AlertType.INFORMATION, contentText: "Book is unavailable", ButtonType.YES);
                alert.showAndWait(); }

            // Book available
            else {
                SessionFactory factory = new Configuration()
```

Books view for both user and librarian are same, there is table view for all available books in LIS and they both can search by their title, but the functionality is different. Let's first look at user's Book View. In the picture above described process of taking book first by checking whether the book is available or not.

Adding book to users list

After that verifying that user don't has this book already in his books list.

```java
// Book available
else {
    SessionFactory factory = new Configuration()
            .configure("hibernate.cfg.xml")
            .addAnnotatedClass(Book.class)
            .addAnnotatedClass(User.class)
            .buildSessionFactory();

    Session session = factory.getCurrentSession();


    boolean allowed = true;

    try {
        session.beginTransaction();
        Book myBook = session.get(Book.class, book.getId());
        User user = session.get(User.class, GUI.Session.user.getUser_id());

        // Checking if user already has this book
        for (Book b : user.getBooks()) {
            if (b.getISBN().equals(myBook.getISBN())) {
                allowed = false;
            }
        }
        // Checking if user allowed to add this book
        if (allowed) {
            myBook.reduce_amount();
            user.getBooks().add(myBook);
        }
        // If not showing alert box that user already has his book
        else {
            alert = new Alert(Alert.AlertType.INFORMATION, contentText: "You cant add book that you already have", ButtonType.YES);
            alert.showAndWait();
        }
        session.getTransaction().commit();
        session.close();
```

Book Search

```java
public class GeneralDBMethods {

    public static List<Book> get_all_books(String query) {
        SessionFactory factory = new Configuration().configure("hibernate.cfg.xml").addAnnotatedClass(Book.class).buildSessionFactory();
        Session session = factory.getCurrentSession();

        List<Book> books = new ArrayList<>();
        try {
            session.beginTransaction();

            if (!query.equals("")) {
                books = session.createQuery("from Book b where b.name like '%' + query + '%'").list();
            } else {
                books = session.createQuery("from Book").list();
            }
            session.getTransaction().commit();
            session.close();

            return books;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return books;

    }
```

Process of searching is done by get_all_books() static method which returns list of all books if passed arguments is empty string or if other string passed searches for book that contains that string.

Getting Books & Adding

But Librarian has 3 function available which is Edit Book, Add Book, Delete Book. Under the hood first two method are very similar to what we saw above. First Edit Book is similar to Take Book from user view but instead of adding book to users Book List and modifying user instance we just update specific Book instance. Second Add Book is similar to Signup process, but here instead of user we create book instance.

```
// Validating form fields
if (isbnField.getText().equals("") || editionField.getText().equals("") || amountField.getText().equals("")
        || nameField.getText().equals("") || authorField.getText().equals("") || statusField.getText().equals("")) {
    raise_error();
} else {
    // Getting data from form fields
    int ISBN = Integer.parseInt(isbnField.getText());
    int edition = Integer.parseInt(editionField.getText());
    int amount = Integer.parseInt(amountField.getText());

    SessionFactory factory = new Configuration()
            .configure("hibernate.cfg.xml")
            .addAnnotatedClass(Book.class)
            .buildSessionFactory();

    Session session = factory.getCurrentSession();

    try {
        session.beginTransaction();

        Book myBook = session.get(Book.class, book.getId());
        // Updating book fields
        myBook.setName(nameField.getText());
        myBook.setAuthor(authorField.getText());
        myBook.setBooksAmount(amount);
        myBook.setEdition(edition);
        myBook.setISBN(ISBN);
        myBook.setStatus(statusField.getText());

        session.getTransaction().commit();
        session.close();
    }
    catch (Exception e1){
        e1.printStackTrace();
    }
    // Returning to books view
    finally {...}
```

Editing books

```
saveBtn.setOnAction(e -> {
    // Validating Form Fields
    if (isbnField.getText().equals("") || nameField.getText().equals("") || authorField.getText().equals("")
            || editionField.getText().equals("")
            || amountField.getText().equals("")) {
        raise_error();
    }
    else {
            // Getting data from fields
            int ISBN = Integer.parseInt(isbnField.getText());
            int edition = Integer.parseInt(editionField.getText());
            int amount = Integer.parseInt(amountField.getText());

            SessionFactory factory = new Configuration()
                    .configure("hibernate.cfg.cmd")
                    .addAnnotatedClass(Book.class)
                    .buildSessionFactory();

            Session session = factory.getCurrentSession();

            try {
                session.beginTransaction();
                //Creating new book instance
                Book myBook = new  Book(ISBN, authorField.getText(), nameField.getText(), "Available", edition, amour
                // Saving book
                session.save(myBook);
                session.getTransaction().commit();
                session.close();
            }
            catch (Exception e1){
                e1.printStackTrace();
            }
            finally {
                // If no error raised returning to books screen
                try {...}
                catch (Exception ex){...}
            }
```

Deleting Books

But with Delete Book situation is a bit different. When librarian deletes book, it may be linked to users by many to many relations, so database won't allow to delete until they have relation, also there is no Entity class in my program that will allow to access the database with Hibernate ORM. To overcome this problem program run custom sql with execute_sql method that deletes relation between users and specific book is it exists.

```
deleteBtn.setOnAction(e -> {
    Book book = booksTable.getSelectionModel().getSelectedItem();

    Alert alert;

    if (book == null) {
        alert = new Alert(Alert.AlertType.INFORMATION, contentText: "Please select one book above", ButtonType.YES);
        alert.showAndWait();
    } else {
        alert = new Alert(Alert.AlertType.CONFIRMATION, contentText: "Delete " + book + " ?", ButtonType.YES, ButtonType.CANCEL);
        alert.showAndWait();
        if (alert.getResult() == ButtonType.YES) {

            String sql = "DELETE FROM user_book WHERE book_id=" + book.getId();

            execute_sql(sql);

            SessionFactory factory = new Configuration()
                    .configure("hibernate.cfg.xml")
                    .addAnnotatedClass(Book.class)
                    .addAnnotatedClass(User.class)
                    .buildSessionFactory();


            Session session = factory.getCurrentSession();
```

Confirmation and final deletion

```
SessionFactory factory = new Configuration()
        .configure("hibernate.cfg.xml")
        .addAnnotatedClass(Book.class)
        .addAnnotatedClass(User.class)
        .buildSessionFactory();


Session session = factory.getCurrentSession();

try {
    session.beginTransaction();
    Book myBook = session.get(Book.class, book.getId());
    // Deleting Book
    session.delete(myBook);
    session.getTransaction().commit();
    session.close();
} catch (Exception e1) {
    e1.printStackTrace();
}
}
}
booksTable.getItems().setAll(get_all_books( query: ""));

});
```

After deleting all relations, we can easily delete book from database.

Profile

```
if (alert.getResult() == ButtonType.YES) {

    SessionFactory factory = new Configuration()
            .configure("hibernate.cfg.xml")
            .addAnnotatedClass(User.class)
            .addAnnotatedClass(Book.class)
            .buildSessionFactory();

    Session session = factory.getCurrentSession();

    try {
        session.beginTransaction();
        // Getting user insatnce from db
        User myUser = session.get(User.class, user.getUser_id());
        // Setting user's user_active field to 0
        myUser.setIs_active(0);

        session.getTransaction().commit();
        session.close();

    } catch (Exception e1) {
        e1.printStackTrace();
    } finally {
        // Showing updated user in table
        booksTable.getItems().setAll(get_all_users(searchByName.getText()));
```

Profile views for both User and Librarian are exact copy of each other except for the fields that each view contains. As always similar process goes here we query user from database validate fields and then update existing user with new data.

Final thoughts and assumption

Even this program covers most of user management it can’t give GUI for creating librarian accounts, but to solve this problem we have librarian driver.

By running this driver program creates initial migration and create new librarian account.

Thank you for attention!

Good luck in using our application!:)