Readme

This readme is mainly about a rental system framework

The framework used 13 different design patterns namely

1.Template – at Service layer (Database layer)

2.Prototype – address

3.Builder – custom alert – book entity

4.Mediator – ui validation

5.Chain of responsibility – login window

6.Factory -- data access layer

7.Façade – db helper

8.Adapter – checkout record

9.Singleton – login window

10. momento – Add member controller UI

11.Iterator – member UI model

12.State – application Layer

13.Strategy – user authentication – member ui

    Our team have developed rental system framework. This framework is high level framework and provides abstract service layer for extending. Framework is UI, persistency layer and platform free, it means you are free to use this anywhere. For example:

    1. Developing Server side application which provides api for its clients

    2. Developing standalone applications where Java is applicable(Desktop, JavaMe, Android etc)

    Framework provides common business logic for any rental system in it abstract layer. It provides basic domain layer - Entity classes. framework provides bellow functionality:

    1. Tools for creating persistence layer like DBMS, in memory db etc;

    2. Entity layer;

    3. Abstract services for every domain object;

    4. Auth subsystem;

    During developing framework our team tried to keep up SOLID approach. Because of using GOD design patterns properly we could achieve a framework which is open for extensions and closed for changes. Our team tried to cover project with Java doc properly. Because of that entry level for framework is really small. From the beginning we created conception for how to collaborate efficiently. And choose git as a version control system. We followed «best practices» of using git and you can see whole history of the project in the github as a commit history. From the beginning, we had a choose Maven as a build system for our project. But we decided not to use maven for below reasons:

    1. Maven well known but it is not flexible in term of build system.

    2. Maven doesn't have imperative language for writing complex build scenarios

    3. Maven barely provides working with remote dependencies

    We choosed gradle as a build system. It is new era for build systems.

    Based on created framework we builded up our library system. We separated our system into 3 major layer.

1. Ui layer — Ui controller, View(Fxml files)

2. Service layer

3. Persistency layer -  Dao, ORM layer