<Assignment 1>
Analysis and Design Document

Student: Oltean Victor
Group: 30431

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

Use JAVA/C# API to design and implement an application for the National Theater of Cluj. The application should have two types of users (a cashier user represented and an administrator) which must provide a username and a password to use the application.

1.2 Functional Requirements

The administrator user can perform the following operations:

- *CRUD on cashiers' information.*
- *CRUD on the list of shows that are performed at the theater. Keep track of the Genre (Opera, Ballet), Title, Distribution list (a long string is enough), Date of the show and the Number of tickets per show.*
- *From time to time he can export all the tickets that were sold for a certain show (either in a csv or xml file).*

The cashier can perform the following operations:

- *Sell tickets to a show. A ticket should hold information about the seat row and seat number.*
- *The system should notify the cashier that the number of tickets per show was not exceeded.*
- *A cashier can see all the tickets that were sold for a show, cancel a reservation or edit the seat.*

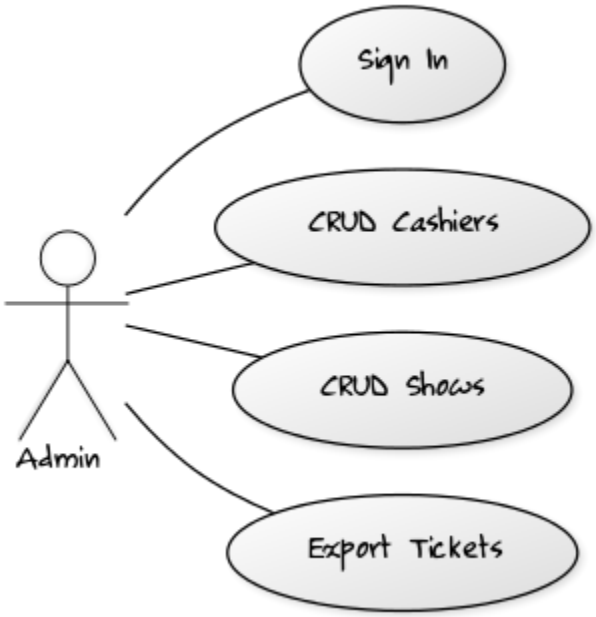
- *The data will be stored in a database.*
- *Use the Layers architectural pattern to organize your application.*
- *Passwords are encrypted when stored to the database with a one-way encryption algorithm.*
- *Provide unit tests for the number of tickets for show exceeded validation and the encryption algorithm.*
- *Use factory method (not factory) for export to csv/xml.*

1.3 Non-functional Requirements

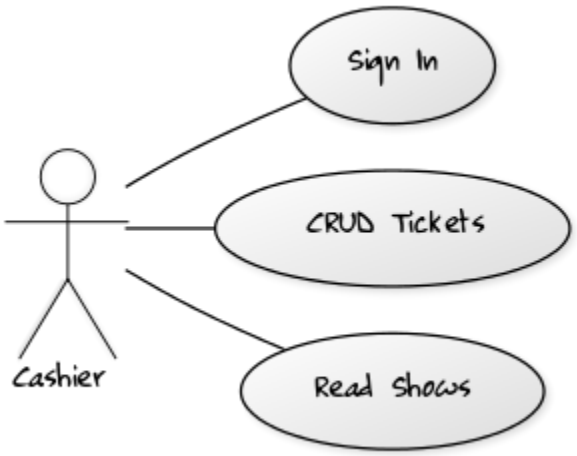
All requests should load in under 200 ms.

2. Use-Case Model

Use case: administrator
Level: user-goal level
Primary actor: administrator
Main success scenario: The Admin signs in and performs CRUD on the Cashiers
Extensions: CRUD fails because the connection to the database was not properly made



Use case: cashier
Level: user-goal level
Primary actor: cashier
Main success scenario: The cashier signs in and performs CRUD on the tickets
Extensions: CRUD fails because the connection to the database was not properly made

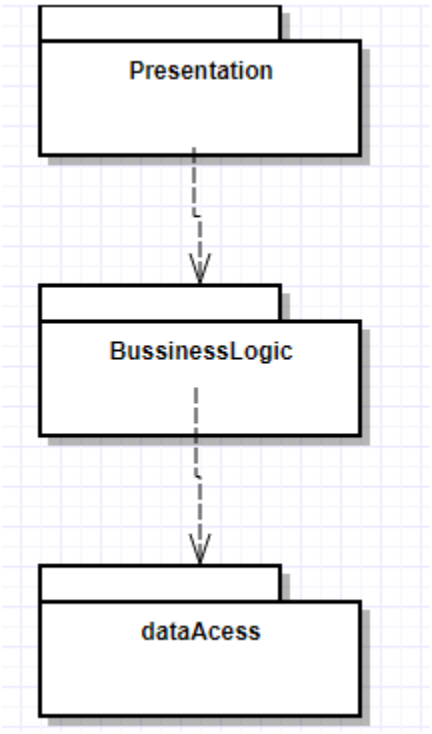


3. System Architectural Design

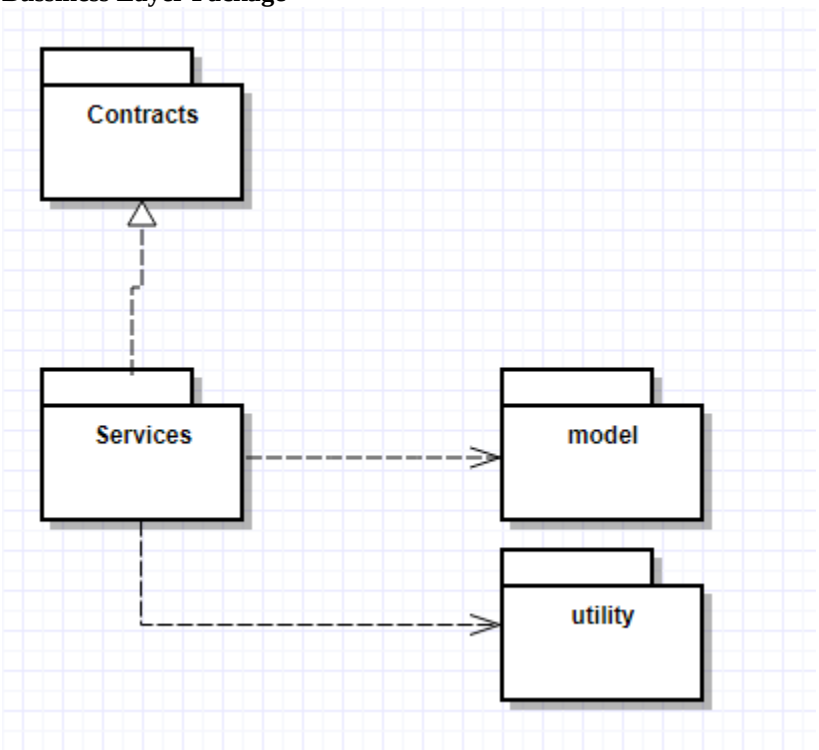
3.1 Architectural Pattern Description

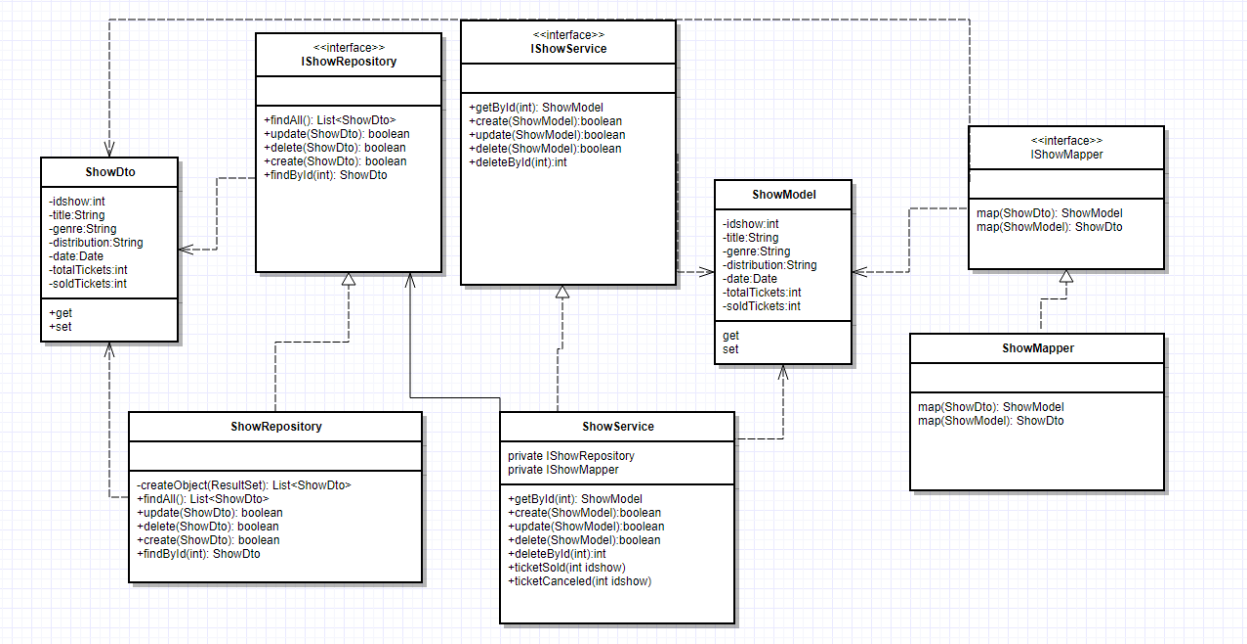
The used architectural pattern is called Layers. In this pattern, the system is organized in three layers: Data Access, Business Layer, Presentation Layer. Each layer communicates only with the layer below it, such that The Presentation Layer shouldn't have any connection to the Data Access Layer.

3.2 Diagrams



Bussiness Layer Package



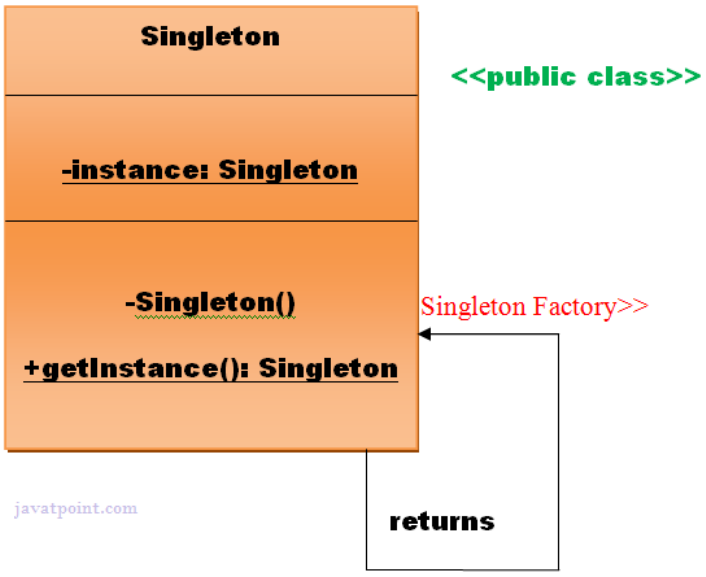


5. Class Design

5.1 Design Patterns Description

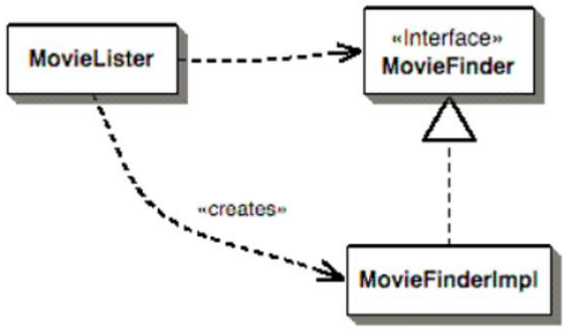
1.1.1 SINGLETON

Singleton has been used in order to make the connection to the database.
The Singleton class produces a single instance of an object, which can then be reused.

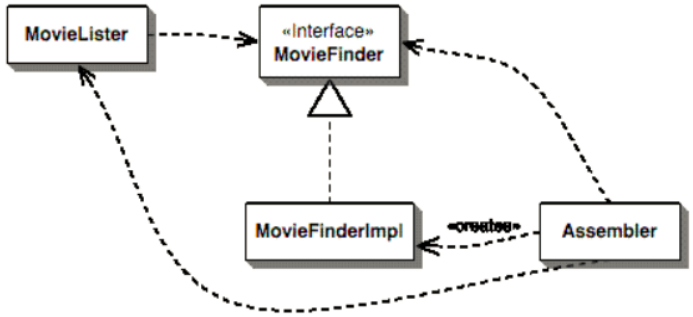


1.1.2 Dependency Injection and Inversion

*According to the SOLI**D** principles, higher level modules shouldn't depend on lower level modules, and they should depend on abstractions.*

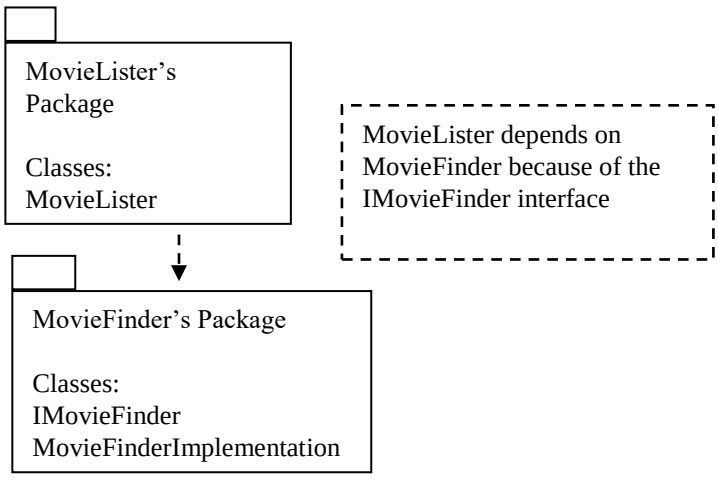


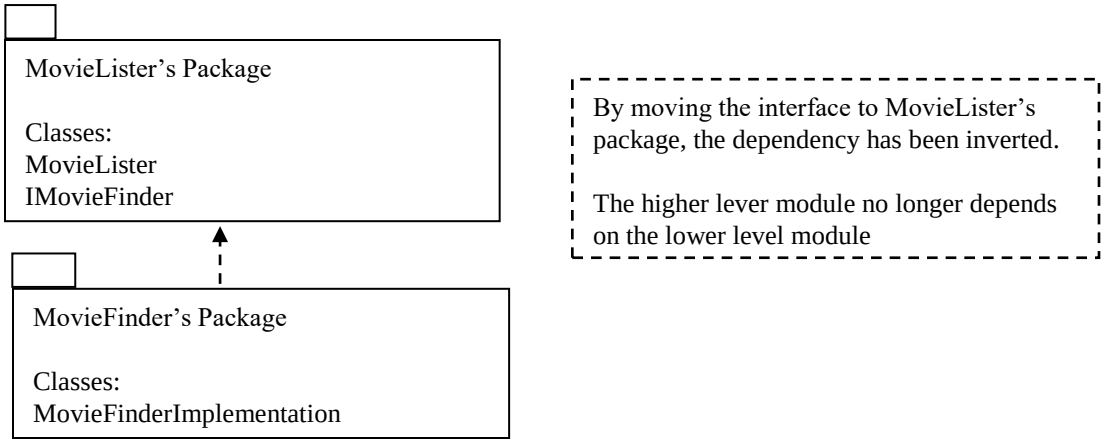
In this photo, MovieLister depends both on the abstraction(the interface), and also on the implementation. This can be resolved through dependency Injection.



Right now, the MovieFinderImplementation is created by the assembler, which is then fed to the MovieLister, in our case through a setter.

If MovieLister and MovieFinder were in different packages, the MovieLister package would still depend on the MovieFinder package. However, if we move the MovieFinder Interface to Movie Finder's package, there is no dependency anymore from high level to low level. This is dependency inversion.





6. Data Model

Show

int idshow
String title
String genre
String distribution
Date date
int totalTickets
int soldTickets

Ticket

int idticket
int idshow
int seatNb
int rowNb
String onName

User

int idUser
String username
String Password
String permission
String name

7. System Testing

Testing has been done using JUNIT, testing whether a new ticket can be put on a seat which is already taken. There is also a JUNIT test for testing wether

8. Bibliography

www.stackoverflow.com
SD Lectures from UTCN