



SC2002 Object - Oriented Design and Programming

MOBLIMA

Movie Booking and Listing Management System

LAB GROUP SE3 - GROUP 6

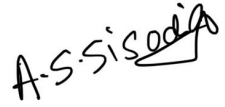

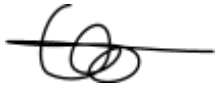
ANG TING FENG
CHEN YI
SISODIA ANUSHKA

Declaration of Original Work for SC/CE/CZ2002 Assignment

We hereby declare that the attached group assignment has been researched, undertaken, completed and submitted as a collective effort by the group members listed below.

We have honored the principles of academic integrity and have upheld Student Code of Academic Conduct in the completion of this work.

We understand that if plagiarism is found in the assignment, then lower marks or no marks will be awarded for the assessed work. In addition, disciplinary actions may be taken.

Name	Course	Lab Group	Signature/Date
Sisodia Anushka	SC2002	SE3	 13/11/2022
Ang Ting Feng	SC2002	SE3	 13/11/2022
Chen Yi	SC2002	SE3	 13/11/2022

Important notes:

1. Name must **EXACTLY MATCH** the one printed on your Matriculation Card.

1. Introduction to Moblima

MOBLIMA is a non-GUI application created to make it easier to reserve and buy movie tickets as well as to serve admin users' administrative needs. One operator with numerous multiplexes located throughout Singapore can use this application.

We will go through a number of design principles in this report, as well as object-oriented ideas we used to build this application. A thorough UML Class diagram is also provided for examination to further highlight the application's functionality.

2. MVC Design Pattern

The Model View Controller (MVC) specifies that an application consist of a data model, presentation information, and control information. We have organised our source code into these categories respectively (Model, View and Controller packages).

- **Model**

Our entity classes were grouped into the Model category. Classes like **Review** and **Showtime** fall under this category. These classes mostly consist of pure data and has simple methods.

- **View**

Each menu in our application is regarded as a view which is managed by the view category. Few examples like **MainMenuView** and **StaffListMoviesView** fall under this category. These classes have methods which presents information to the user.

- **Controller**

These are classes which control most of the logic flow of the application. Our source code contains a total of 4 control classes.

1. DataBaseManager:

Our txt file or database is managed by DataBaseManager. This class mostly consists of static methods that aid in keeping data into text files and retrieving data from databases.

2. IOManager:

Any interaction with the application's user is handled by this class. It is made up of static methods that mostly deal with either obtaining user input or outputting data into the console screen. These methods also assist in handling certain exceptions which may arise when processing the user inputs.

3. PriceManager:

This class is responsible for explaining how prices are calculated. This takes into account both the calculation of the ticket price and the calculation of the order the moviegoer placed. The price of a ticket or order is influenced by a number of variables.

4. ViewsManager:

This class is in charge of controlling the application's flow of the various View classes. This class handles a number of operations, such as moving to the next view and returning to the previous views. This is accomplished by keeping track of a stack that houses the displayed views. Simply pop the current view off the views stack to access the previous view, and push the new view onto the stack to access the subsequent view.

3. SOLID Design Principles

a. **Single Responsibility Principle**

This principle states that “There should never be more than one reason for a class to change”. We would want to have loose/low coupling such that any changes in the one class would have minimal impact on other classes . One example is when the application is presenting the display to user. Instead of implementing the methods in each of the view classes, we implement it in the **DatabaseManager** class. Hence the individual view class is only responsible for presenting the information while **DatabaseManager** is only responsible for handling the user interaction.

b. **Open-Closed Principle**

This principle states that “A module should be open for extension but closed for modification”. We would want to the base class to be closed for modification and its subclass to further extend its methods or attributes. An example would be the **BaseView**, it has very general attributes such as titles. Its subclass such as **StaffEditMovieDetailsView** expands on **BaseView**’s generic attributes and methods to make it more specific.

c. **Liskov Substitution Principle**

This principle states that “Subtypes must be substitutable for their base types”, this means that subclasses should be able to function properly even if a derivation of that class is passed into it. Using back the example of **BaseView** and **StaffEditMovieDetailsView**, the subclass do not expect more or provide less when used and all cases are accounted for in the subclass.

d. Interface Segregation Principle

This principle states that “Many client specific interfaces are better than one general purpose interface”. Specific interfaces were not used in our source code.

e. Dependency Injection Principle

This principle states that “A high level modules should not depend upon low level modules. Both should depend upon abstraction” and “Abstractions should not depend upon details, details should depend on abstraction”. To allow multiple use of high level module, we must ensure that they are independent of the low level, as such we use abstractions to remove the internal dependencies between classes.

4. Object-Oriented Design

a. Abstraction

Abstraction is the process of reducing various specific characteristics to a set of essential characteristics. An example is the abstract method **getFractionalCostOutOfOriginal** in the **Ticket** class. We do not know exactly how to calculate the price of a “generic ticket” due to the lack of information. Hence that method is only implemented in its subclasses such as **AdultTicket**, **ChildTicket** and **SCTicket** where sufficient information is given to calculate the price of the different type of tickets.

b. Encapsulation

Encapsulation is used to protect an object’s data and hide the details or implementation from user. We achieve encapsulation by making all classes’ attributes private. These data can only be accessed and modified using the getter and setter methods which are public.

c. Inheritance

Inheritance describes a “is-a” relationship. We use inheritance as it allows us to write less repetitive codes. **MainMenuView** “is-a” **BaseView** while **AdultTicket** “is-a” **Ticket**. Hence **MainMenuView** and **AdultTicket** are subclasses of each of them respectively and inherits their properties (reuse some of their code).

d. Polymorphism

Polymorphism refers to the ability of an object to take on many forms. In OOP, it means an object/class can take on many different type. For example, the **ticket** class took on the form of **AdultTicket**, **ChildTicket** and **SCTicket**.

5. Further Enhancement

Feature 1:

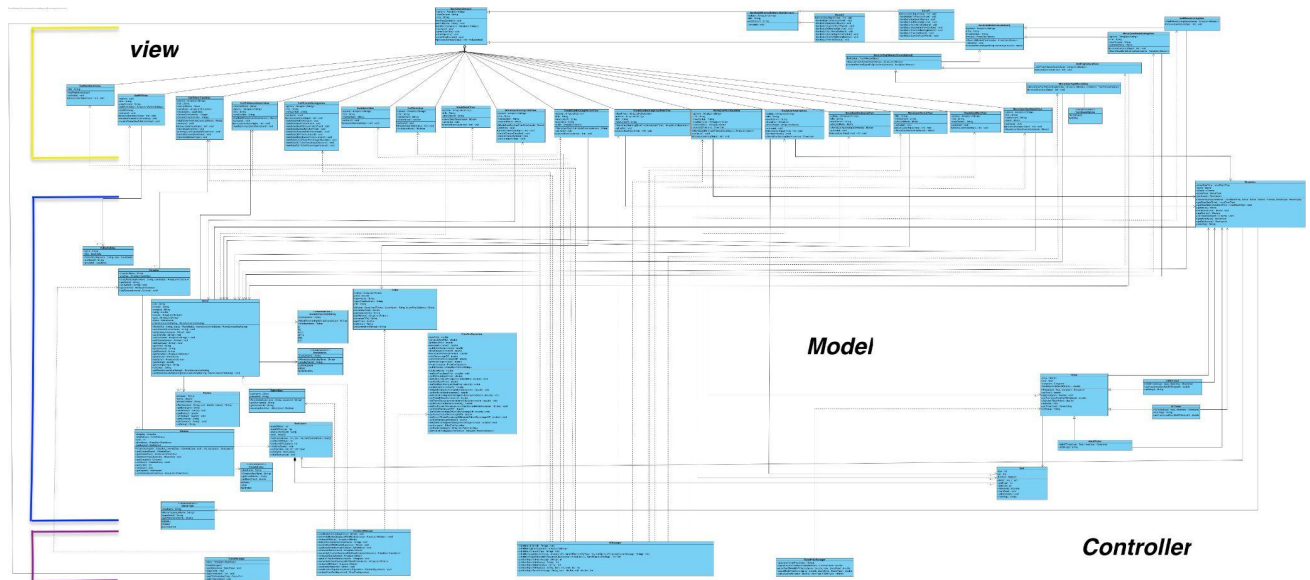
Admins would be able to add new cinema types in their cinema complex. For example, in the event that a Cinema company wants to introduce new cinema classes such as a “Premium Class”, it can be added into the system easily in the **CinemaClass** source code. As this class is loosely coupled, it has a weak association with other classes. Hence addition of this new “Premium Class” would require minimum coding. This is possible as we use abstraction and interfaces to achieve loose coupling and high cohesion. As such any modification to the the **CinemaClass** will not affect the other classes as much.

Feature 2:

Movie-goers can have a display for comparing the tickets prices for different combinations as factors such as the type of movie, the type of cinema class, their age and the date affects pricing. This would allow users to better gauge and book tickets base on what suits them the most. We can create an additional subclass **CompareView** which inherits the **BaseView** root class and customise it by making use existing classes such as the **PriceConfiguartion**, to calculate the different ticket prices, and many other classes to and display them to the user.

6. Class Diagram

Please go to the folder for a clearer image of the UML.



7. Essential Test Case

The following are some of the test cases which were not covered in the demo video.

Price difference between Adult and Senior Citizen

Order	Order
Tickets:	Tickets:
Ticket Type: Adult	Ticket Type: Senior Citizen
Seat ID: G1	Seat ID: G7
Price: \$13.00	Price: \$7.80
Showtime:	Showtime:
Bishan Cineplex	Bishan Cineplex
Cinema Code: 2	Cinema Code: 5
12/11/2022 - 13:00 - Saturday	12/11/2022 - 11:30 - Saturday
Total Price Before GST : \$13.00	Total Price Before GST : \$7.80
GST : \$0.91	GST : \$0.55
Total Price: \$13.91	Total Price: \$8.35
1) Confirm	1) Confirm
2) Reselect seat	2) Reselect seat
3) Cancel and back to main menu	3) Cancel and back to main menu

Price difference between the Normal Class Ticket and Gold Class Ticket

Order	Order
Tickets:	Tickets:
Ticket Type: Adult	Ticket Type: Adult
Seat ID: G1	Seat ID: G7
Price: \$13.00	Price: \$23.00
Showtime:	Showtime:
Bishan Cineplex	Bishan Cineplex [GOLD]
Cinema Code: 2	Cinema Code: 1
12/11/2022 - 13:00 - Saturday	12/11/2022 - 13:30 - Saturday
Total Price Before GST : \$13.00	Total Price Before GST : \$23.00
GST : \$0.91	GST : \$1.61
Total Price: \$13.91	Total Price: \$24.61
1) Confirm	1) Confirm
2) Reselect seat	2) Reselect seat
3) Cancel and back to main menu	3) Cancel and back to main menu

More examples of price variation due to movie type and the day of the week

Order	Order	Order
Tickets:	Tickets:	Tickets:
Ticket Type: Adult	Ticket Type: Adult	Ticket Type: Adult
Seat ID: F7	Seat ID: G7	Seat ID: F6
Price: \$15.00	Price: \$18.00	Price: \$23.00
Showtime:	Showtime:	Showtime:
Jurong East Cineplex [PLATINUM]	Jurong East Cineplex [PLATINUM]	Jurong East Cineplex [PLATINUM]
Cinema Code: 3	Cinema Code: 3	Cinema Code: 3
14/11/2022 - 15:30 - Monday	Movie Type: SD	Movie Type: Blockbuster
	14/11/2022 - 16:45 - Monday	13/11/2022 - 15:25 - Sunday
Total Price Before GST : \$15.00	Total Price Before GST : \$18.00	Total Price Before GST : \$23.00
GST : \$1.05	GST : \$1.26	GST : \$1.61
Total Price: \$16.05	Total Price: \$19.26	Total Price: \$24.61
1) Confirm	1) Confirm	1) Confirm
2) Reselect seat	2) Reselect seat	2) Reselect seat
3) Cancel and back to main menu	3) Cancel and back to main menu	3) Cancel and back to main menu