



**NANYANG
TECHNOLOGICAL
UNIVERSITY**
SINGAPORE

Semester 1 AY 2022/2023

Declaration of Original Work for SC/CE/CZ2002 Assignment






MOvie Booking and Listing Management Application (MOBLIMA) 22S1

SSP3 Assignment Group 5

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 honoured 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 (CE2002 or CZ2002)	Lab Group	Signature /Date
Chong Yue Hong	SC2002	SSP3 Group 5	 5 Nov 2022
Lee Yen Foong Ernest	SC2002	SSP3 Group 5	 5 Nov 2022
Rayden Teo Wei Xuan	SC2002	SSP3 Group 5	 5 Nov 2022
Tan Meng Hong	SC2002	SSP3 Group 5	 5 Nov 2022
Tey Li Zhang Edmund	SC2002	SSP3 Group 5	 5 Nov 2022

Design Considerations

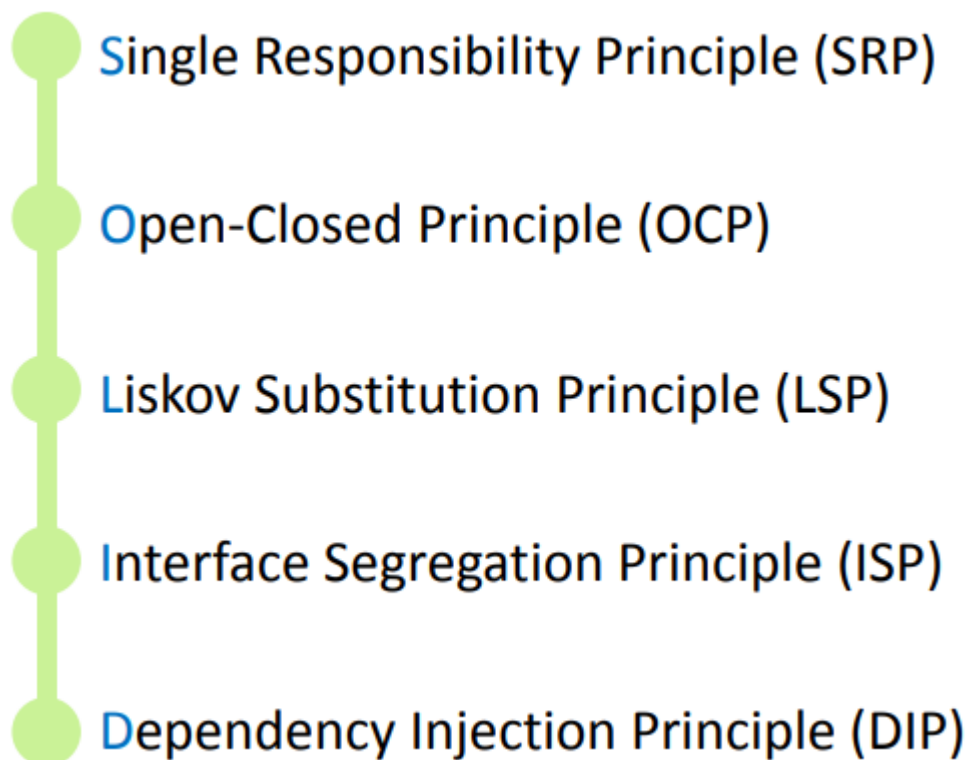
Approach Taken

Our group aims to create a MOBLIMA that is good code. It should not only be understandable and readable, but able to be reused, extended and easily maintained. Hence, we need to ensure that the dependencies between packages and the classes within them are created in such a way that individual changes would lead to minimal impact of change on other parts of the software. This can be achieved using the SOLID Design Principles. In our application, the movie goers are labelled as `Customer` and movie staff as `CinemaStaff`.

Our GitHub Repository: <https://github.com/AmosChong20/SC2002-MOBLIMA>

SOLID Design Principles

Using the SOLID Design Principles, our MOBLIMA is able to achieve loose coupling and high cohesion.



1. Single Responsibility Principle (SRP)

The SRP ensures that there is high cohesion throughout our program, with each class within the 3 packages (Entity, Control and Boundary) having only one responsibility.

We will illustrate this using the signing up process for a customer as an example. When the customer selects “Sign up” in `CustomerControl`, control is passed to `CustomerLoginView` for the collection of the customer’s information. Relevant checks for validity of the customer’s information (such as username) will be done and stored in `DataBase` using `DatabaseManager` if all are valid.

In this example, `CustomerControl` handles only the customer’s choice to log in or sign up. It is unaware of the upcoming processes of collecting and storing the customer’s information done by `CustomerLoginView` and `DatabaseManager` respectively. As such, `CustomerControl` only has the single responsibility of handling the customer’s choice to login or sign up. This same SRP is applied to all the other classes in our MOBLIMA.

2. Open-Closed Principle (OCP)

OCP states that a module should be open for extension but closed for modification. Through OCP, functionalities in a module can be changed without modifying the original code of that module. This is done by extension.

In our program, OCP is exhibited through the use of a `MainControl` interface in our control package. For example, if we need to add a new functionality for `Movie`, (e.g. display movie’s final screening date), a new class that implements `MainControl` can be created in the control package. This new functionality can then be added in `MovieControl` for the new class to be initialised. As such, `Movie` will not be changed directly, maintaining an OCP.

3. Liskov Substitution Principle (LSP)

LSP states that subtypes must be substitutable for their base types. A user of a base class should continue to function properly if a derivative of that base class is passed to it.

In the `NavigateControl` class, we implemented a method `load(MainControl mainController)`. But in the `UserControl` class, we called `NavigateControl.load(new CustomerControl())` and passed in

CustomerControl as a parameter. CustomerControl is a subclass of MainControl. This proves the point that the subclass, CustomerControl, is able to substitute the base class, MainControl, and the method load in NavigateControl is still able to work as expected.

4. *Interface Segregation Principle (ISP)*

ISP states that our code should make use of many client specific interfaces which comprises a single function rather than one general interface with many different functions. Using this principle, our classes do not have to depend on interfaces that are useless to them.

For example, we have an ItemName interface to get the name of the class and another interface called BookMovie which has the methods such as getAvailableSeats, checkAvailability and checkFull. Combining them into one general interface would violate the ISP, as a class that needs the interface to get the name of the class but does not require the BookMovie functionality would have empty methods and vice versa. For instance, in the Movie class, it only inherits the nameToString method and does not inherit the methods it does not need to use.

5. *Dependency Injection Principle (DIP)*

DIP states that a high level module should not depend upon low level modules, both should depend upon abstractions. In the NavigateControl class, we call mainController.begin(). The method begin() belongs to the interface MainControl. We are making mainController.begin() depend on the MainControl interface's begin() method when calling the method rather than a concrete class's begin() method when the program runs. We are linking a concrete class which implements the begin() method with NavigateControl using the MainControl interface. Making changes to the concrete classes will not affect the interface method.

OO Concepts

1. Abstraction

The main goal of Abstraction is to handle complexity by hiding unnecessary details, which will improve the reusability and maintainability of a program. This is achieved in the control package through the use of the interface `MainControl` and its abstract method `begin`. The other classes in the control package implements `MainControl` and achieves their own functionalities by implementing their version of `begin`. As such, the classes only know the details of their own methods, achieving abstraction between classes.

2. Encapsulation/Information Hiding

Encapsulation is a protective barrier that keeps the data and code safe within the class itself. For our MOBLIMA, every classes' attributes are private, only allowing access to them through certain public methods. For example, in `Customer`, the attributes such as the `Customer`'s `mobileNumber` and `emailAddress`, are sensitive information and thus need to be set to private.

Furthermore, we protect the passwords of accounts through the hashing of passwords. During sign up, passwords keyed in by the users are encapsulated using `hashPassword` in `User`. This ensures that the passwords of users are truly protected, and will only be known by the users themselves.

3. Inheritance

Inheritance allows us to derive new classes from existing classes by absorbing their attributes and behaviours and adding new capabilities in the new classes. This enables code to be reused and can greatly reduce the programming effort in implementing new classes. For example, in our program we have a base interface `ItemName` which is inherited by many of our other classes such as `Movie` class. The `ItemName` interface has a `nameToString` method which the `Movie` class would be able to inherit and use.

nameToString method in ItemName interface:

```
public String nameToString();
```

nameToString method inherited by Movie class:

```
@Override  
public String nameToString() {  
    return title;  
}
```

4. Polymorphism

Polymorphism refers to the ability of an object reference being referred to different types; knowing which method to apply depends on where it is in the inheritance hierarchy. This includes overriding of methods. A subclass can override a method in the parent class by defining a method with exactly the same signature and return or replace or refine the method in the parent class. For example, different classes such as `HolidayControl`, and `MovieTimeControl` implement the `begin` method from the `MainControl` interface but have slightly different versions of it.

HolidayControl's implementation of begin method:

```
@Override  
public void begin() {  
    HolidayView.displayHolidays();  
    NavController.popOne();  
}
```

MovieTimeControl's implementation of begin method:

```
@Override  
public void begin() {  
    ShowTimeView.getShowTimeView();  
    NavController.popOne();  
}
```

New Features Proposal – 2 Features

1. Combo Set

Customers can order a Combo set which includes a movie with snacks. Creating a `Food` class which implements enum of the `FoodItems` and their respective prices. `calculatePrice` in `BookingControl` will take the enum class as an additional parameter when calculating the price the customer will have to pay. The `Food` class takes care of all the food items and the price of the item. It ensures Open-Closed Principle as the `Food` class is open for extension but closed for modification. `Food` class can be further extended in the future to further add in newer features while still being able to be used in `calculatePrice` in `BookingControl`. `calculatePrice` does not need to know about the different types of food to be able to calculate the price.

2. Member Discount

Another possible new feature would be to allow customers to become Premium Members of MOBLIMA. Premium Members would enjoy further discounts on movie tickets as an exclusive benefit. Customers can choose to proceed as a regular member or a Premium Member during sign up, having to pay an annual renewal fee if they choose to be a Premium Member. For calculation of the new ticket price, due to our adhering to the Single Responsibility Principle, this can be simply achieved by slightly modifying `getTicketPrice` in `TicketPrice`. We just need an additional parameter to check if the customer is a Premium Member, and decide to apply an additional discount to the final price of the ticket or not.

Testing

Test	Description	✓ / X
Login	Upon launching of MOBLIMA, users are given the option to login as (1) Customer, (2) Cinema Staff, or to (3) exit the application. If (1) is selected, customers can either sign up as a new member or login with an existing account. If (2) is selected, Cinema Staff can only login with existing accounts.	✓
Create/Update/Remove movie listing	After logging in as a Cinema Staff, the Cinema Staff is able to Create/Update/Remove Movie Listing. Cinema Staff can then (1) Display movie details, (2) Add movie, (3) Update movie, (4) Remove movie.	✓
Create/Update/Remove cinema showtimes and the movies to be shown	Regarding cinema showtimes, Cinema Staff can (1) Display movie show times, (2) Add a movie show time, (3) Update a movie show time, (4) Remove a movie show time.	✓
Configure top 5 listing criteria	Cinema staff can configure how the users list the movies (1) Ticket Sales, (2) Overall Ratings, (3) Default (Ticket Sales and Overalls Ratings)	✓
Search/List movie	After logging in as a customer, they can view all available movies in the cinema. The following details about the movie will be displayed: Movie Number, Title, Showing Status (E.g. #1, Black Adam, Now Showing).	✓
View movie details – including reviews and ratings	After selecting a movie, customers can (1) Display movie details, (2) View past reviews & ratings, (3) Input a review.	✓
Single unoccupied seat between selected seats	Customer is unable to leave a single unoccupied seat between selected seats.	✓
Check seat availability and selection of seat/s.	When a customer decides to book a movie ticket, the seat layout will be printed, and booked/unavailable seats are indicated with an 'X'.	✓
Book and purchase ticket	Customers are able to select a movie and time and would receive a transaction ID for their purchase	✓

Booking on a different day of the week or holiday and type of cinema	Customers have the option to book a ticket for different types of cinema and for different screening times	✓
View booking history	Customers are able to view all the bookings that they have made.	✓
List the Top 5 ranking by ticket sales OR by overall reviewers' ratings	Customers and Cinema Staffs are able to view top 5 movies by ticket sales and overall ratings	✓
Configuring a holiday date and the ticket price is shown correctly when booking is done on that date	Cinema Staffs have the option to "Create/Update/Remove Ticket Pricing Scheme & Holiday" in the Cinema Staff main menu	✓
Configuring "End of Showing" date and the movie should not be listed for booking	Cinema Staff have the option "Create/Update/Remove Movie Listing" in the Cinema Staff main menu where they can remove a movie by configuring it to "End of Showing" so that customers would not be able to book tickets for this movie	✓
Booking only allowed for "Preview" and "Now Showing" status.	After a Cinema Staff configures the movie to "End of Showing", Customers are not able to book tickets for this movie under "Book a movie ticket" option choices	✓

Additional Testing

1. Booking the same seat for a movie

When a customer tries to book a seat, the program will check whether the seat selected has already been booked by another customer. If yes, an error will appear asking the current customer to select another seat instead.

```
-----
A [ x ] [ x ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] A
B [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] B
C [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] C
D [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] D
E [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] E
F [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] F
G [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] G
H [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] H
I [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] I
J [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] J
-----
      1      2      3      4      5      6      7      8      9      10
-----
                        ENTRANCE

Legend:
[ ] Normal Seat
[ ] Couple Seat
[ ] Wheelchair Seat
To continue, please press enter.

How many seats would you like to book:
1
Please enter the #1 seat number (eg. G6): a1
Seats selected are unavailable.
Please choose another seat.
Please enter the #1 seat number (eg. G6):
```

2. Customer accidentally selecting the same seat when booking a movie

When a customer is deciding on the seats to purchase for a certain movie at a certain time, the customer is not allowed to choose multiple of the same seat. (Eg. “Seat A2 has already been selected, choose a different seat.”)

```
-----
A [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] A
B [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] B
C [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] C
D [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] D
E [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] E
F [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] F
G [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] G
H [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] H
I [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] I
J [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] J
-----
      1      2      3      4      5      6      7      8      9     10     11     12     13     14     15     16     17     18     19     20     21
-----
                        ENTRANCE

Legend:
[ ] Normal Seat
[ ] Couple Seat
[ ] Wheelchair Seat
To continue, please press enter.

How many seats would you like to book:
5
Please enter the #1 seat number (eg. G6): a1
Please enter the #2 seat number (eg. G6): a2
Please enter the #3 seat number (eg. G6): a2
Seat A2 has already been selected, choose a different seat.
Please enter the #3 seat number (eg. G6):
```

3. *Registering with existing username*

When a customer (or cinema staff) tries to register for an account, they have to key in a unique username that does not exist. Else, they will encounter an error that says the username already exists.

```
Please select an option
-----
1: Sign up
2: Login
3: Exit
Enter option: 1

Username: er
Error: Username already exists
```

4. *Customer choosing more seats than what is available*

The Customer would not be able to book more seats (e.g. 200 seats) when the selected movie only has limited (e.g. 10) seats left. The Customer would be prompted to reselect the number of seats.

```

      SCREEN
-----
  1   2   3   4   5   6   7   8   9  10
-----
A [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] A
B [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] B
C [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] C
D [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] D
E [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] E
F [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] F
G [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] G
H [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] H
I [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] I
J [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] J
-----
  1   2   3   4   5   6   7   8   9  10
-----
      ENTRANCE

Legend:
[ ] Normal Seat
[ ] Couple Seat
[ ] Wheelchair Seat
To continue, please press enter.

How many seats would you like to book:
101
Sorry, there are only 100 seats available
How many seats would you like to book:
```