

**NANYANG
TECHNOLOGICAL
UNIVERSITY**
SINGAPORE

SC2002: MOvie Booking and LIsting Management Application (MOBLIMA)

Date of Submission: 12 NOV 2022

Tutorial Group: SS3 Group 3

Group Members:

Kee Ruitao	U2122845E
Lim Ke En	U2122069L
Ong Tsien Jin	U2120567H
Ong Zhen Yu	U2122064A
Phoen Yu Heng, Ryan	U2121283H

Table Of Contents






Declaration of Original Work for SC/CE/CZ2002 Assignment	3
1. Design Considerations	4
a. Builder Design	4
b. SOLID Design Principles	5
1. Single Responsibility Principle	5
2. Open-Closed Principle	5
3. Interface Segregation Principle	6
c. Object-Oriented Design Concept	7
1. Encapsulation	7
2. Abstraction	7
2. Additional Assumptions made	7
3. Additional features for further enhancement	8
1. Promo Code for users	8
2. Foreign Key Constraints	8
4. UML Class Diagram	9
5. Testing	9
1. Customer Booking System	9
2. Booking History	11
3. Movie Menu	11
4. Staff Login	12
5. Staff Configuration Settings	12

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
Kee Ruitao	SCSE	SS3	 <u>12/11/22</u>
Lim Ke En	SCSE	SS3	 <u>12/11/22</u>
Ong Tsien Jin	SCSE	SS3	 <u>12/11/22</u>
Ong Zhen Yu	SCSE	SS3	 <u>12/11/22</u>
Phoen Yu Heng, Ryan	SCSE	SS3	 <u>12/11/22</u>

Important notes: 1. Name must EXACTLY MATCH the one printed on your Matriculation Card.

1. Design Considerations

a. Builder Design

The primary design approach used is the builder design. It is a creational design pattern that allows us to construct complex objects step-by-step. We can use the same constructors to produce objects with varying representations. The builder pattern helps in reducing the number of parameters in the constructors. Calling an object with a large number of parameters is usually inconvenient; hence the builder pattern comes into play where we build objects step by step, using only those steps that we require. This gives us more flexibility when constructing new objects and prevents the need to create multiple subclasses to cover all possible combinations.

For instance, in our code, movie is a complex object where it contains variables such as movie ID, movie title etc. Hence, instead of bloating the Movie class with a huge list of constructors, we extract the assembly of Movie into a separate class called the MovieBuilder class. If the client code needs to “assemble” a movie class, it can work with the builder directly. The builder will help to construct a particular representation of the product and in our example, the StaffCSVDriver will pass in the MovieBuilder to construct the Movie object.

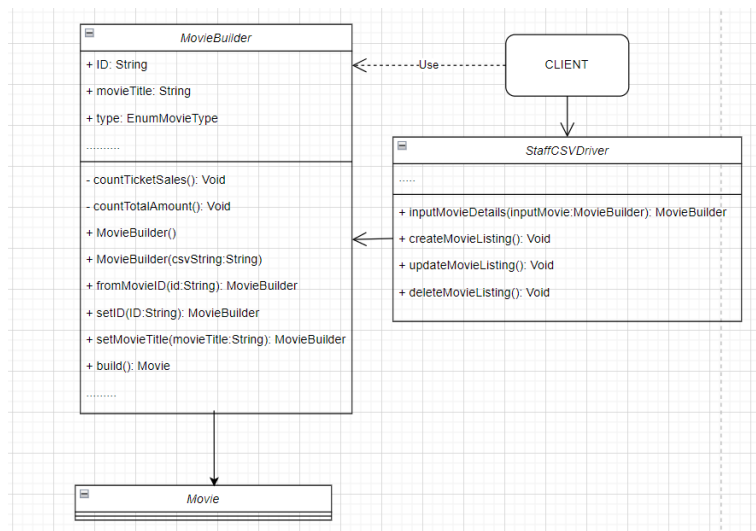


Figure 1: MovieBuilder

With the implementation of the builder pattern, we therefore are able to construct new objects step-by-step and reuse the same constructor when building various representations of products.

Furthermore, our implementation of *builder* objects includes a method to parse data retrieved from a Comma-Separated-Value (CSV) string as seen in Figure 2.

```
/** Private method to update attributes from CSV line
 * @param csvString
 */
private void updateFromCsvString(String csvString){
    ArrayList<String> csvArr = new ArrayList<>(Arrays.asList(csvString.split(mainDelimiter)));

    this.ID = csvArr.get(0);
    this.movieTitle = csvArr.get(1);
    this.status = EnumShowingStatus.valueOf(csvArr.get(2));
    this.type = EnumMovieType.valueOf(csvArr.get(3));
    this.synopsis = csvArr.get(4);
    this.director = csvArr.get(5);
    this.cast = new ArrayList<>(Arrays.asList(csvArr.get(6).split(subDelimiter)));

    this.reviews = new ReviewContainer(this.ID);
    this.countTicketSales();
    this.countTotalAmount();
}
```

Figure 2: Code snippet of the method to parse CSV strings in MovieBuilder.

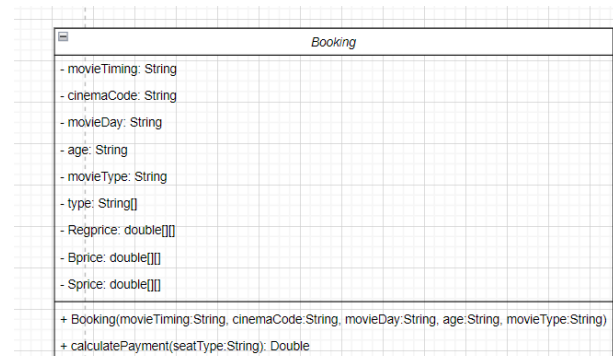
This reinforces the Single Responsibility Principle where we can isolate complex construction of code from other functionalities.

b. SOLID Design Principles

1. Single Responsibility Principle

A class should only have just one reason to change. Specific methods would be contained in its own class and would only contain a small amount of methods. For example, Booking class in which its only function is to calculate the price of the ticket. Other functions such as printing of menus will be completed in other classes.

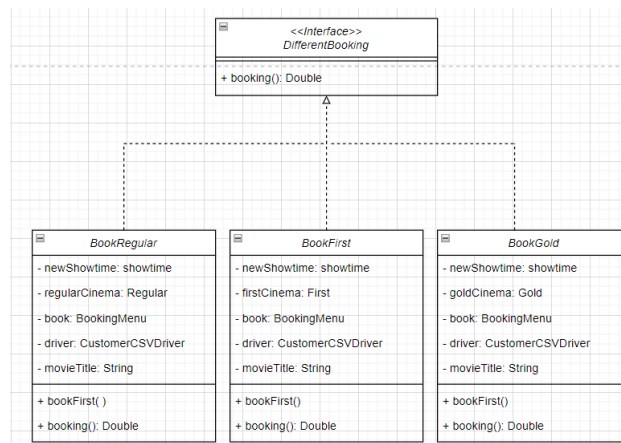
This ensures that we reduce the complexity of our code and it will be easier to follow when the program extends or changes.



2. Open-Closed Principle

Our classes are designed in such a way that it is open for extension but closed for modification. For the process of booking the seats for different cinemas. In order to prevent breaking of code, we have added an interface class that contains the abstract method of booking. With this, if in the

future we would like to include a new cinema type, we are able to implement the new booking method without touching the order codes. Therefore, creating a subclass and allowing for overriding of parts will ensure that the addition of new code will not break the existing code.



3. Interface Segregation Principle

In order to achieve consistency in adopting the *Builder* design principle, we used an array of interfaces that dictate the necessary methods. Furthermore, interfaces allow us to utilise repeated values (e.g. *mainDelimiter* and *subDelimiter* in *InterfaceCsvDelimiter*) so that important values can be shared across classes without being tightly coupled – *Movie* objects share the same delimiter as *Showtime* objects, maintaining consistency when writing to our CSV files.

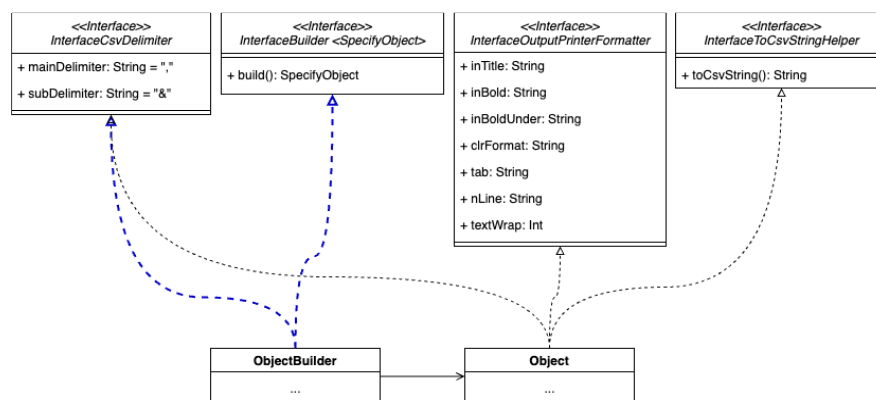


Figure 3. Interface implementation of MovieBuilder and Movie classes.

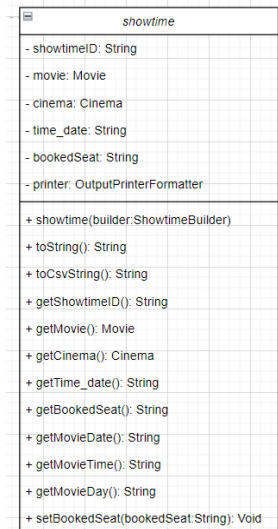
To standardise our naming convention for methods with the same function objectives, we created an interface named "*InterfaceToCsvStringHelper*" to specify the toCsvString method and an interface named "*InterfaceCsvDelimiter*" to specify the delimiters for our CSV file.

c. Object-Oriented Design Concept

Object-oriented concepts are being used to allow our code to be extensible and reusable.

1. Encapsulation

Encapsulation will build a barrier to protect an object's private data. This will help in isolating the parts in the program into independent modules, protecting the codes from changes. For instance, the showtime class where it hides the private details from other classes which prevents other classes from changing those variables. Additionally, for every class variable, we implement the accessor and mutator method to aid in obtaining and changing the variable.



2. Abstraction

By separating different types of methods needed into different classes, the inner workings of each class are not of concern to other classes. An example of this is the *FileIO* class, where it handles all the CRUD operations. Classes that use *FileIO* therefore do not have to create their own methods to handle CRUD operations. Furthermore, this allows us to reuse the same methods in our code base.

2. Additional Assumptions made

We have made some additional assumptions and implementation to our MOBLIMA application, excluding those that were already mentioned in the assignment brief.

- We have implemented 3 types of cinema classes, differentiating each cinema class by the size: Gold (6x6 Seats), Regular (10x10 seats) and First Class (8 x 8 seats).

- In each class, there are 9 types of tickets, ticket cost depends on age (Student, Adult, Senior Citizen), time (non-peak weekday, peak weekday, weekend, public holiday) and type of seating (Normal, Ultimate, Elite, Platinum).
- Ticket cost also varies according to these 3 types of movies: Regular, Blockbuster and 3D/IMax.

3. Additional features for further enhancement

Since we are practising the SOLID design principle, new features can be implemented easily. Adhering to the Single Responsibility Principle, we can easily extend or change any new or existing functionality.

1. Promo Code for users

An additional feature we can add is to introduce promotional codes to incentivise customers. During booking, we can provide customers with the option of applying a promocode to reduce the price of the movie ticket. We can also limit the number of promotional codes available.

2. Foreign Key Constraints

In our database, some attributes are foreign keys that references a primary key in another table. This suggests that when we are performing updates or deletions to the database, we have to be careful not to delete a record from a table that contains attributes required in another table.

For example, review.csv contains the movieID attribute which is in reference to the movieID in movie.csv. Updating or deleting a record from movie.csv may result in a conflict. As we are practising SOLID design principle, our CRUD operations are in separate classes and hence, we only need to add error handling to the appropriate classes to prevent such issues.

4. UML Class Diagram

Separate attachments of the UML are also included outside of the report.

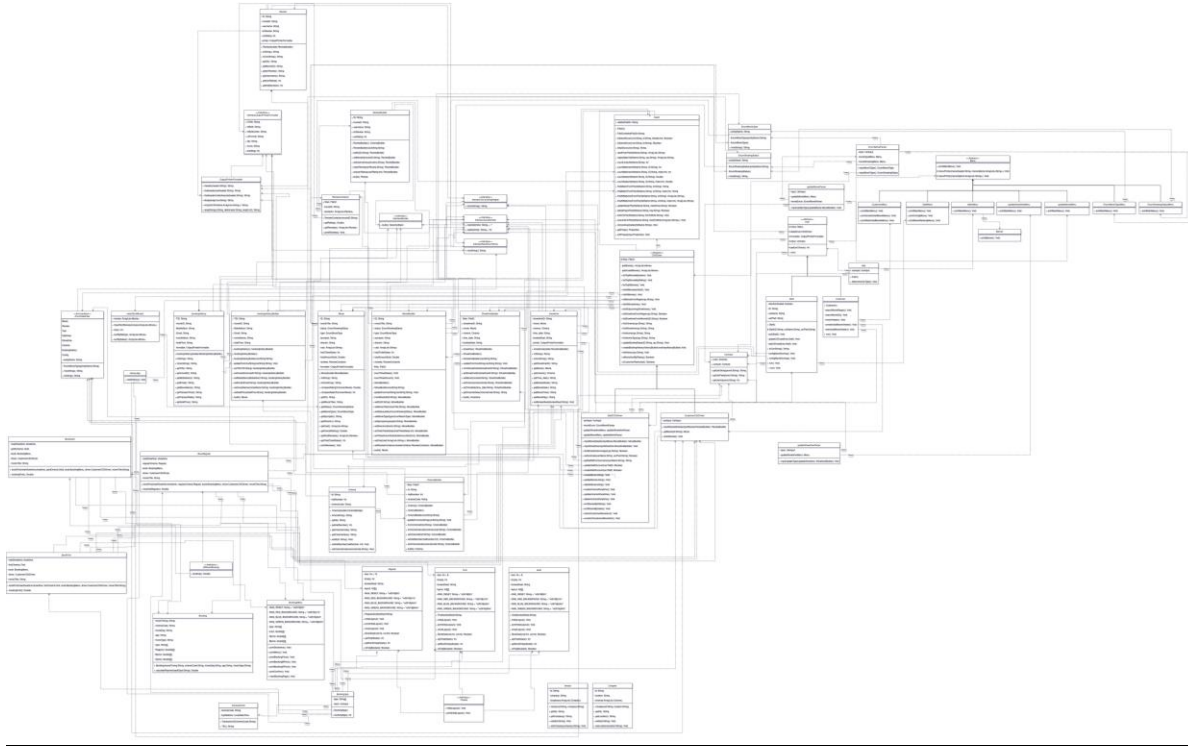
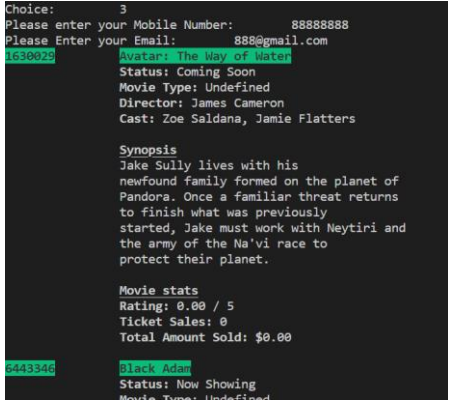
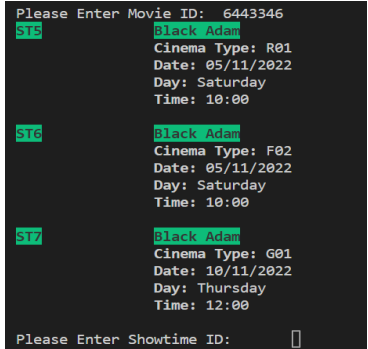
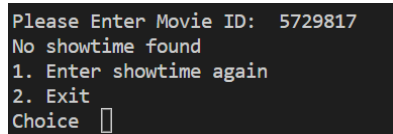
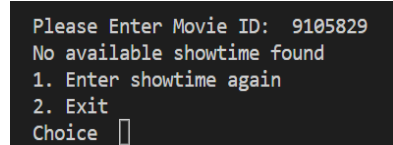
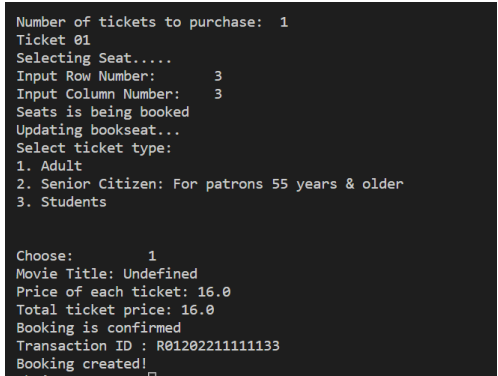
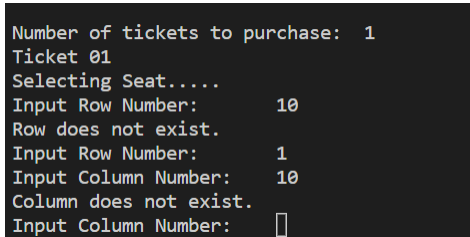


Figure 4. Interface implementation of MovieBuilder and Movie classes.

5. Testing

1. Customer Booking System

	Test Scenario	Expected Results	Actual Outcome
1.	Customers will select option 4 from the Customer Main Menu which will direct them to the booking page.	Upon selecting option 4, users are able to select which cineplexes they want to visit then the booking menu page would be shown where customers are able to either view ticket pricing or to start booking their tickets.	<pre>Enter Option: 4 --Welcome to the Booking App-- Location of Cineplexes: 1. Jurong Point 2. Paya Lebar 3. Bugis 4. Punggol Please choose the cineplex 1 1. Print this menu 2. View Ticket Pricing 3. Start Booking 4. Exit Choice: </pre>

3	Customers can select option 3, to begin the process of booking a movie ticket.	Customers will need to enter their mobile number and email address, and only movies whose status are “Now Showing” and “Preview” will be listed out for customers to choose from. Customers will then enter the movie they want to book by the movie ID.	 <pre> Choice: 3 Please enter your Mobile Number: 88888888 Please Enter your Email: 888@gmail.com 6430029 Avatar: The Way of Water Status: Coming Soon Movie Type: Undefined Director: James Cameron Cast: Zoe Saldana, Jamie Flatters Synopsis Jake Sully lives with his newfound family formed on the planet of Pandora. Once a familiar threat returns to finish what was previously started, Jake must work with Neytiri and the army of the Na'vi race to protect their planet. Movie stats Rating: 0.00 / 5 Ticket Sales: 0 Total Amount Sold: \$0.00 6443346 Black Adam Status: Now Showing Movie Type: Undefined </pre>
4	After the customer enters the desired movie ID	<p>If show times are available, then the list of showtime will be shown.</p> <p>If there are no upcoming showtime, then users will be prompted to enter the show time again.</p> <p>Else, “No showtime found” will be printed and users will be prompted to enter show time again.</p>	 <pre> Please Enter Movie ID: 6443346 STS Black Adam Cinema Type: R01 Date: 05/11/2022 Day: Saturday Time: 10:00 STG Black Adam Cinema Type: F02 Date: 05/11/2022 Day: Saturday Time: 10:00 STV Black Adam Cinema Type: G01 Date: 10/11/2022 Day: Thursday Time: 12:00 Please Enter Showtime ID: </pre>  <pre> Please Enter Movie ID: 5729817 No showtime found 1. Enter showtime again 2. Exit Choice </pre>  <pre> Please Enter Movie ID: 9105829 No available showtime found 1. Enter showtime again 2. Exit Choice </pre>
6	Users will then be prompted to enter the number of tickets to purchase, then the row and columns selection and the which ticket category.	Upon correct inputs of the number of tickets, row number as well as column number then the booking will confirm at the end in which the price of total ticket will be shown and transaction ID will be created.	 <pre> Number of tickets to purchase: 1 Ticket 01 Selecting Seat..... Input Row Number: 3 Input Column Number: 3 Seats is being booked Updating bookseat... Select ticket type: 1. Adult 2. Senior Citizen: For patrons 55 years & older 3. Students Choose: 1 Movie Title: Undefined Price of each ticket: 16.0 Total ticket price: 16.0 Booking is confirmed Transaction ID : R01202211111133 Booking created! </pre>
7	User inputs invalid row/column number	Error validation. System will ask the user to input the row/column number again.	 <pre> Number of tickets to purchase: 1 Ticket 01 Selecting Seat..... Input Row Number: 10 Row does not exist. Input Row Number: 1 Input Column Number: 10 Column does not exist. Input Column Number: </pre>

8	When a user inputs a seat that is already booked.	System will ask the user to input another seat number and prints "Seats has already been booked".	<pre> Number of tickets to purchase: 1 Ticket 01 Selecting Seat..... Input Row Number: 2 Input Column Number: 0 Seat has already been booked Input Row Number: [</pre>
---	---	---	--

2. Booking History

	Test Scenario	Expected Results	Actual Outcome
1.	To view booking history, users can search by mobile number or Email address.	<p>If the mobile number/email address is not found, "No booking history found!" will be printed.</p> <p>Else it will print the list of booking the user had made.</p>	<pre> Mobile Number: 87661029 77661029 Transaction ID: G02202211040107 Movie ID: 0068646 Movie Name: The Godfather Total Price: 34.0 Transact Time: 01 : 07 Transact Date: 04 - 11 - 2022 77661029 Transaction ID: R01202211042058 Movie ID: 0068646 Movie Name: The Godfather Total Price: 18.0 Transact Time: 20 : 58 Transact Date: 04 - 11 - 2022 </pre> <pre> Choice: 1 Enter your mobile number: Mobile Number: 89898989 No booking history found! Enter Option: [</pre>

3. Movie Menu

	Test Scenario	Expected Results	Actual Outcome
1.	When user enter the keyword of the movie User key "Deleted"	Show all the movies with the title/keyword.	<pre> Enter choice: 4 Enter regex or string: Deleted 7194827 Deleted Status: Now Showing Movie Type: Undefined Director: Ken Ng Lai Huat Cast: Zheng Ge Ping, Vincent Ng, Fattah Amin, Dato Rosyam Nor Synopsis A Malaysian police detective search for his daughter who was kidnapped by child traffickers. Movie stats Rating: 0.00 / 5 Ticket Sales: 0 Total Amount Sold: \$16.00 </pre>
2.	Enter a movie ID to view the details of a movie	<p>If movie ID exists, show the movie details.</p> <p>Else, prints "Movie not found"</p>	<pre> Enter choice: 5 Enter movie ID: 7194827 7194827 Deleted Status: Now Showing Movie Type: Undefined Director: Ken Ng Lai Huat Cast: Zheng Ge Ping, Vincent Ng, Fattah Amin, Dato Rosyam Nor Synopsis A Malaysian police detective search for his daughter who was kidnapped by child traffickers. Movie stats Rating: 0.00 / 5 Ticket Sales: 0 Total Amount Sold: \$16.00 </pre> <pre> Enter choice: 5 Enter movie ID: 123 Movie not found! </pre>

4. Staff Login

	Test Scenario	Expected Results	Actual Outcome
1.	Staff login - Enters the correct username and password	Staff will be able to login and sees the admin menu	<pre> LOGIN AS STAFF Enter Username: admin Enter Password: Staff Menu: 1) Print this menu </pre>
2.	Staff login - Enters the wrong username and password	Unable to login. Asked staff to try again.	<pre> LOGIN AS STAFF Enter Username: a Enter Password: Username not found! Wrong login credentials! Please try again! </pre>

5. Staff Configuration Settings

	Test Scenario	Expected Results	Actual Outcome
1.	Create a new admin profile	Ask to enter username and password, then the user will be created.	<pre> Enter choice: 4 Enter ID for new user: HII Enter username for new user: hi Enter password for new user: </pre>
2.	Ticket ranking choice	Allow the staff to set if they want to show the ticket choice by tickets sold, rating or allowing the customer to view both.	<pre> Movie Ranking Sorting: 1) Tickets Sold 2) Rating 3) Enable Customer Sorting Choice 4) Disable Customer Sorting Choice 5) Back Enter choice: 1 </pre>