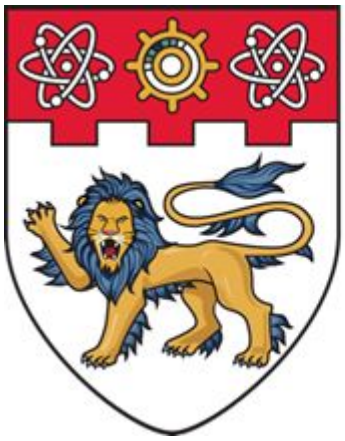


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

ASSIGNMENT CE/CZ2002: Object-Oriented Design & Programming

Building an OO Application

Lab: SE2
Group: 12

Members: Cao Liu, Chen Guanyu, Liu Mingyu, Ma Yuqian

2017/2018 SEMESTER 1

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY

CZ2002 Object Oriented Programming & Design
MOBLIMA Project Report

The following contents are included in this report:

1. Analysis of the program on design consideration, design principle and OO concept.
2. Assumptions we made during developing this application.
3. UML Class diagram to illustrate the relationships among classes.
4. UML Sequence diagrams (a. Search movie, b. List movie, c. View movie details, d. reviews and ratings) to illustrate each of these processes.
5. Screenshots of functionalities that are not demonstrated in the video. (Most of the functionalities have been shown in the video)

Analysis of program

Design Considerations

Since this application is a highly-interacted application. In this design, MVC design pattern is adopted, which means all the java classes in this application are classified into the following three packages:

1. Model
Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes. Hence, all the objects created, such as *cinema*, *seat* and *showtime*, are under this package. Moreover, most of the methods in this part are constructors, mutators and accessors.

2. View
View represents the visualization of the data that model contains. Therefore, all the user interface used to interact with customers to acquire and display information, such as display menu, display movie information and display reviews, are under this package.

3. Controller
Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate. Hence, all the logical control functions, such as get input, print output and exception handling, are under this package. Moreover, most of the methods in this package are static.

Design Principles

The design of this application strictly follows the SOLID principles. The analysis will be performed to show how our design fulfils the SOLID principles.

1. Single Responsibility Principle (SRP) --"There should never be more than ONE reason for a class to change".

In this application, multiple classes are created to fulfil different functions. For instance, since this is a highly-interacted programme, input and output are used at an extremely high frequency. Therefore, *printHeader*, *printMenu* etc., such kind of output methods, *readChoice*, *readString* etc., such kind of input methods and many exception handling methods are independently created in *IOController* class for repeatedly uses in the following models or views.

2. Open-Closed Principle (OCP) -- "A module should be open for extension but closed for modification".

In this application, this principle has been fulfilled by creating a *View* class as the base class. All other classes (e.g. *MoviegoerView* class, *StaffView* class) under *View* package are subclasses which extend the *View* class. In this way, all its subclasses have extended the behavior of their base class without having to modify the source code of the *View* class. Furthermore, protected and private methods are adopted in most classes under *View* package as well as *Controller* package to restrict access and modification.

3. Liskov Substitution Principle (LSP) --"Subtypes must be substitutable for their base types".

In fact, this principle is very closely related to OCP, such that if LSP is violated, it affects the OCP as well. In this application, the principle is applied in a way that in the controller package, *DataController* is an abstract class which allows being overridden by other Controller classes, such as *CineplexManager*.

4. Interface Segregation Principle (ISP) --"Many client specific interfaces are better than one general purpose interface".

In this application, although interfaces are not widely applied, some abstract classes are adopted in a decomposite manner to fulfill this principle. For instance, in the *View* package, a lot of abstract views are created for the more specific implementation views. Each of the view fulfill some of the abstract methods. This Segregation helps a lot when writing code to implement those view classes and methods.

5. Dependency Injection Principle (DIP) --“A high level modules should not depend upon low level modules. Both should depend upon abstractions. Abstractions should not depend upon details. Details should depend upon abstractions.”

In this application, *Model* classes and *View* classes are separated into two different packages. The *Model* classes do not know about *View* classes and they depend on the classes under *Controller* to interact with each other. In this way, *Model* classes can be designed independently without taking the usage in *View* classes into consideration and direct interaction between *Model* and *View* is avoided.

Use of OOP Concepts

1. Abstraction

Abstraction means using simple things to represent complexity. In Java, abstraction works by creating useful, reusable tools. To implement this, various variables, methods and classes are created. For instance, we have a *Movie* class to represent a movie object. Inside the *Movie* class, the information of the movie is stored in its attributes, such as movie title, director, synopsis and ticket sales. By using abstraction, we can avoid repeating the same work and it can represent complex system by using multiple classes.

2. Encapsulation

Encapsulation is the practice of keeping certain fields within a class private and provide public method to access. This can keep the data and code safe within the class itself. In this application, encapsulation is well implemented. For classes under *Model*, all attributes are set to be private to restrict access from other classes. To modify or access private fields from other classes, public methods are added to achieve this. For example, in *Movie* class:

```
public class Movie implements Serializable {
	private String title;
	private AgeRestriction ageRestriction;
	private String director;
	…

	public String getTitle() { return title; }
	public AgeRestriction getAgeRestriction() {
		return ageRestriction;
	}
	public String getDirector() { return director; }
	…
```

}
Public methods are used to access private fields.

3. Inheritance

Inheritance allows programmers to create new classes that share some of existing attributes and methods to reuse existing classes. In this application, inheritance is widely used. For example, the *DataManager* class contains protected methods that can read and write from files. *CineplexManager* is a subclass of *DataManager*, hence *CineplexManager* inherit the protected methods from DataManager. In addtion to the methods and attributes inherited from its base class, *CineplexManager* has its own fields and methods. This helps to avoid duplicated codes and improve code reusability.

4. Polymorphism

Polymorphism means the ability to take more than one form. An operation may exhibit different behaviors in different instances. Polymorphism is extensively used in implementing Inheritance and overriding is a necessary tool for polymorphism. In the view package, all of the other view classes such as *MovieListingView*, *ReviewView* and some other views overrides the *start()* method which is implemented in *View* (abstract class) as an abstract method. Moreover, *DataController* is an abstract class which allows being overridden by other Controller classes, such as *CineplexManager.* Furthermore, all of the classes inherit *Object*, and in the *Model* package, all of the classes overide the method *equal*, *hashCode*, *toString*

Assumptions

1. Only one user will be using the application. Hence, the application will read all data from local files at the beginning and will not re-read while using the application.

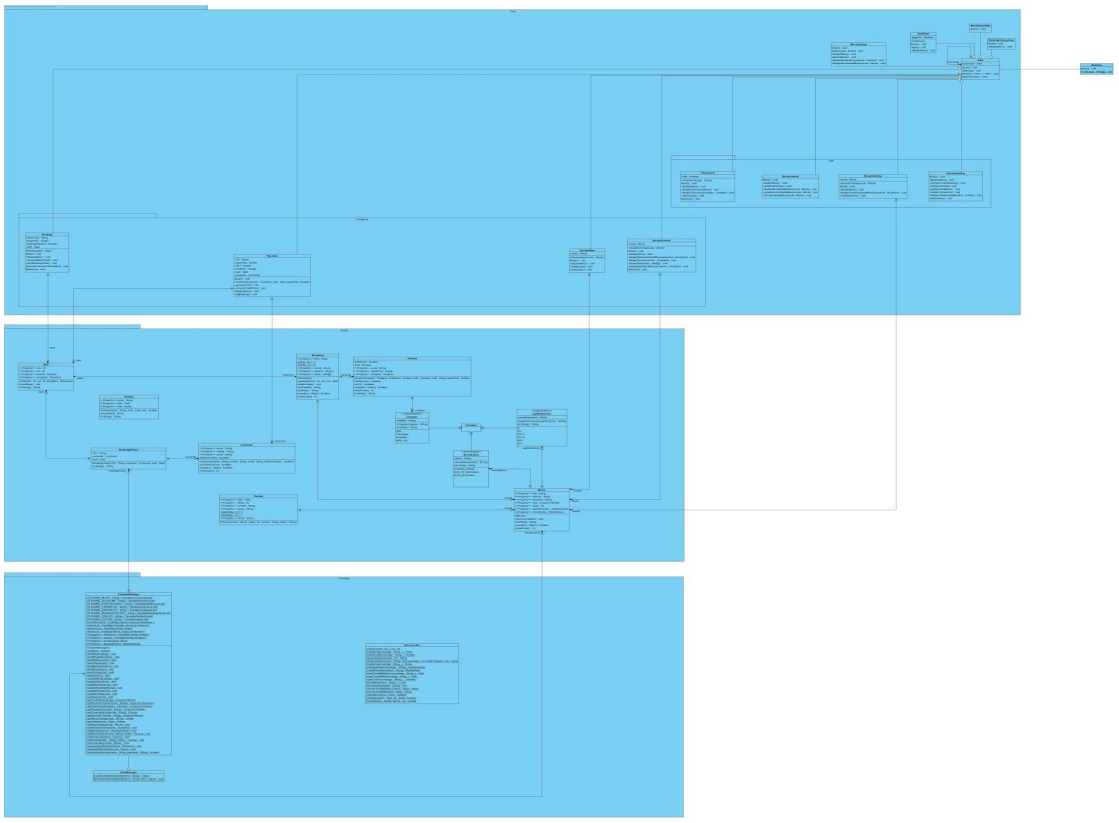
2. There will only be one holiday on one day. Adding another holiday on the same date will overwrite the existing holiday on that day.

3. Platinum cinema and normal cinema have the same layout but different prices.

4. The type of movie (digital / 3D) is determined by the cinema where the showtime is assigned to.

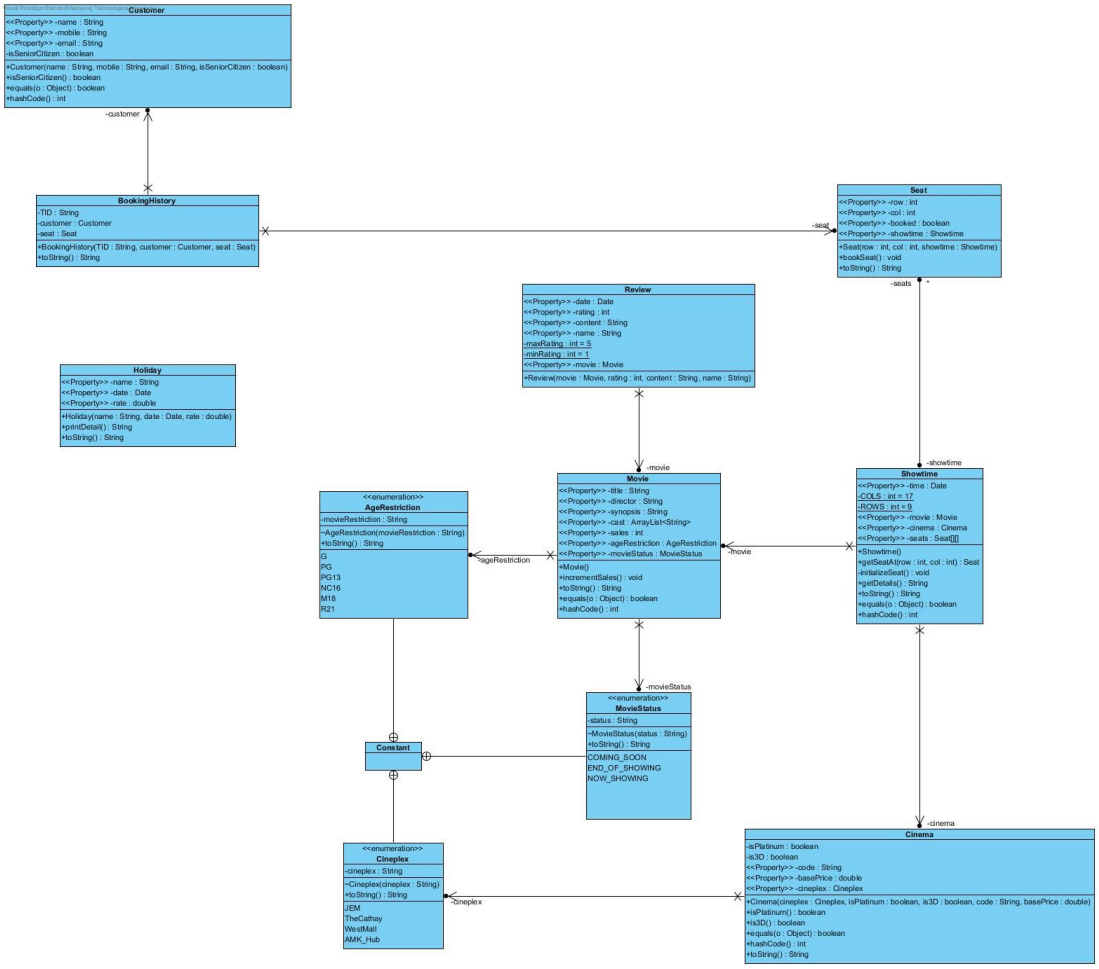
5. For the movie of status “Now Showing”, only tickets within three days which are of today, tomorrow and the day after are allowed to be booked. For the movie of status “Coming Soon”, only tickets of tomorrow and the day after are allowed to be booked.

UML Class Diagram



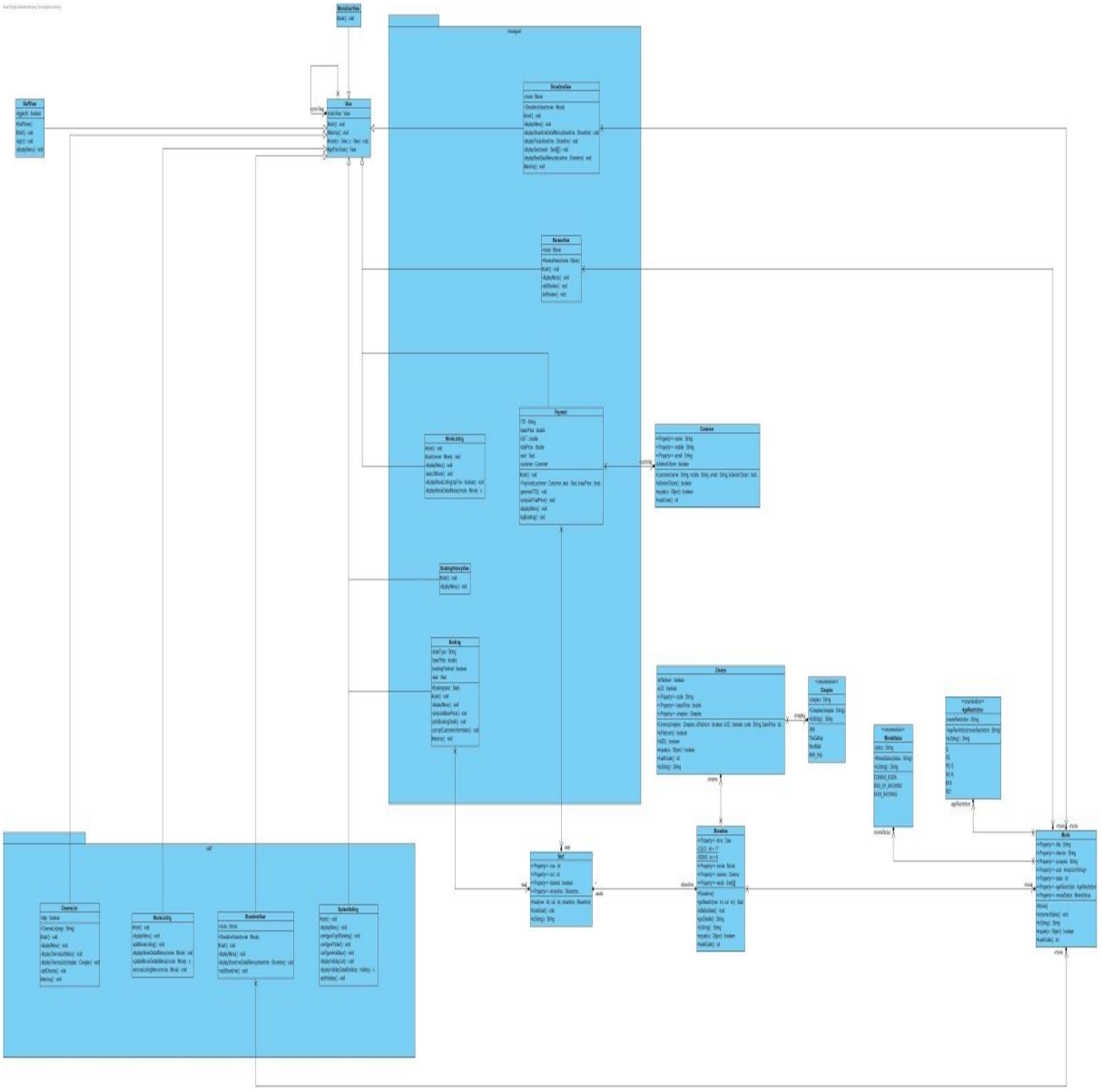
There are three package, the first one is *View* package, the second one is *Model* package, and the third one is the *Controller* package. In the *View* package, there are also two small packages, which are *staff* and *moviegoer*. Please refer to attached file (ClassDiagram.jpg) for higher resolution.

And the following are the Class diagram for each part:



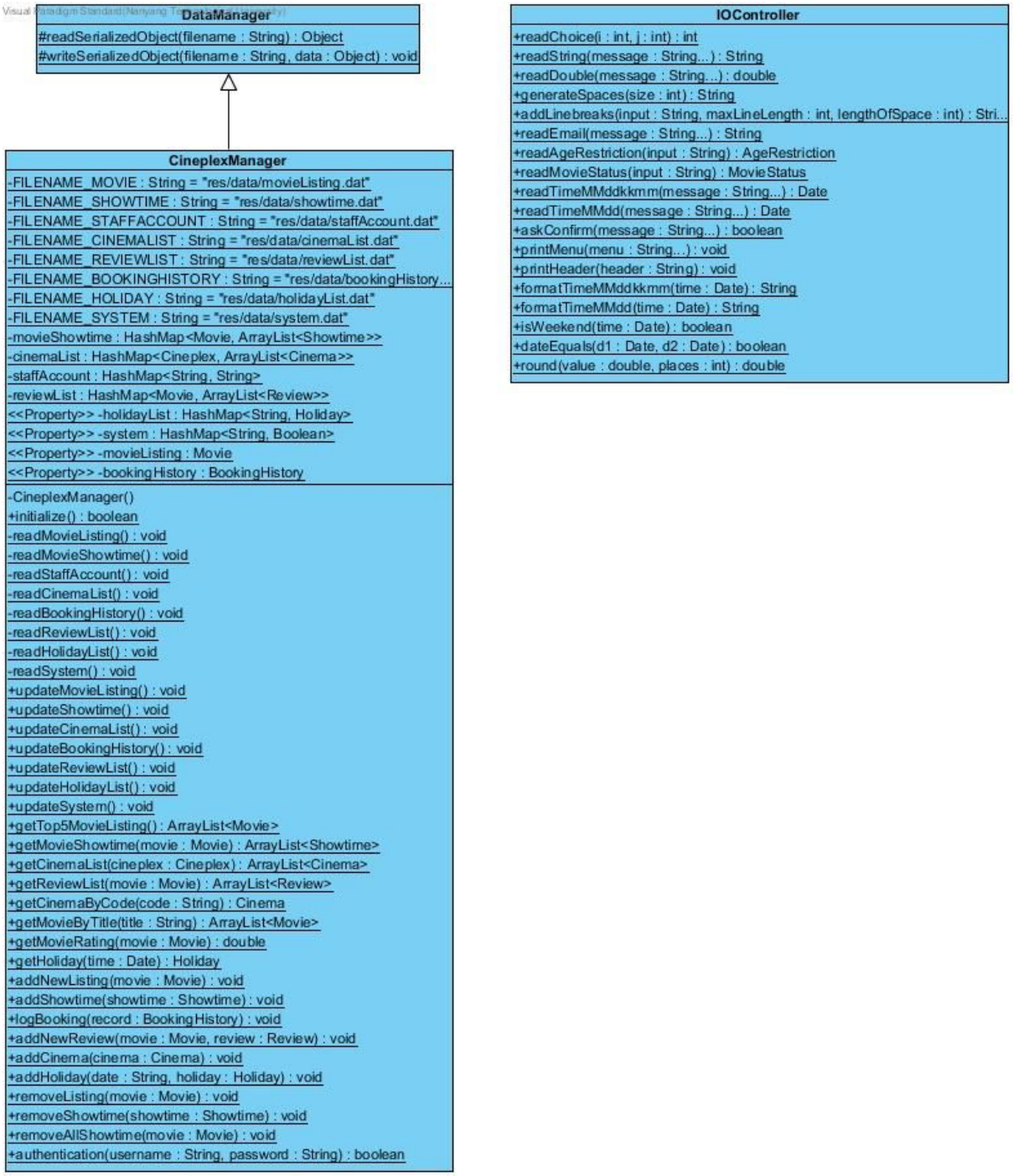
Model package

Please refer to attached file (*Model.jpg*) for higher resolution.



View package

Please refer to attached file (*View.jpg*) for higher resolution.

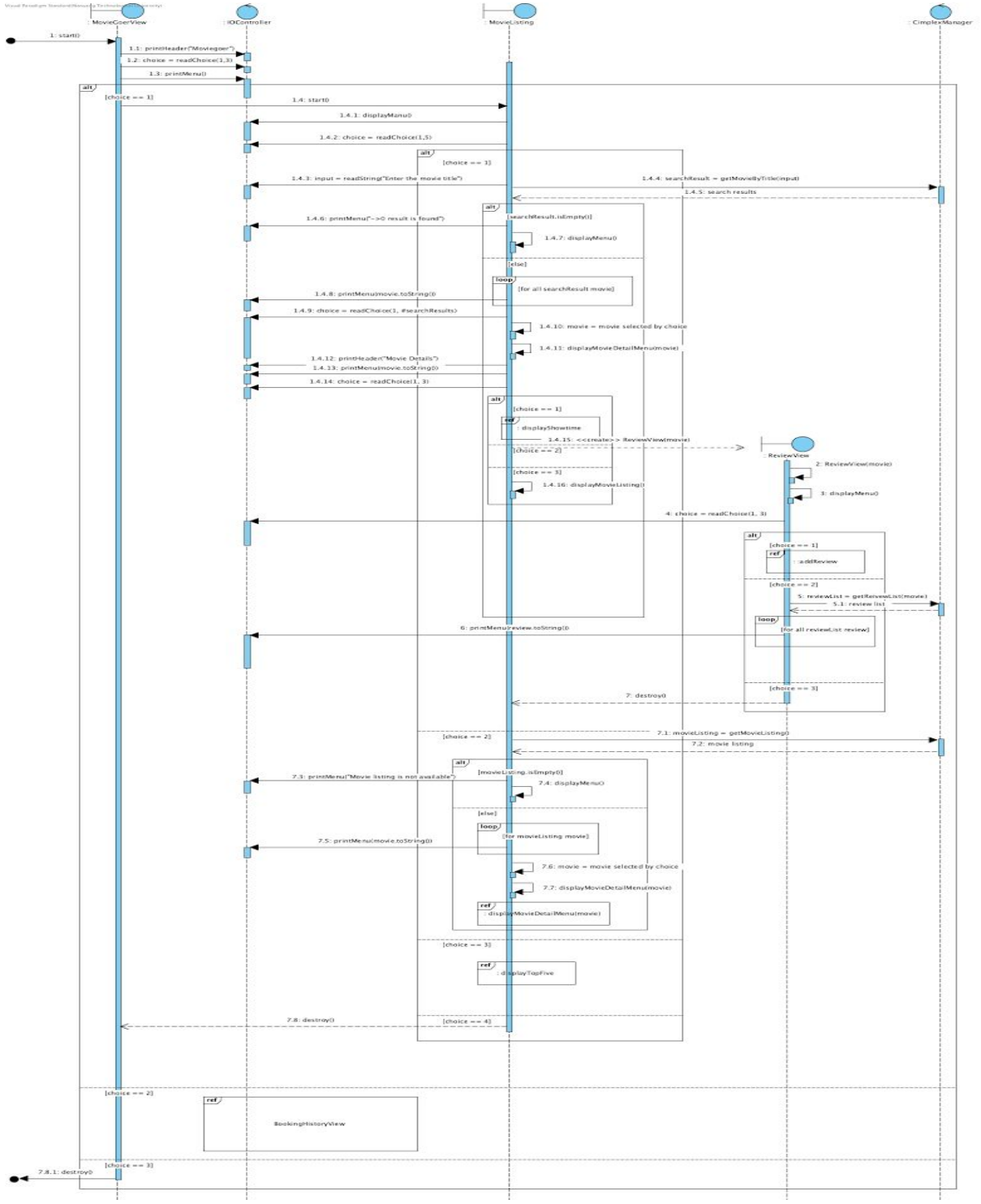


Controller package

Please refer to attached file (*Controller.jpg*) for higher resolution.

UML Sequence Diagram

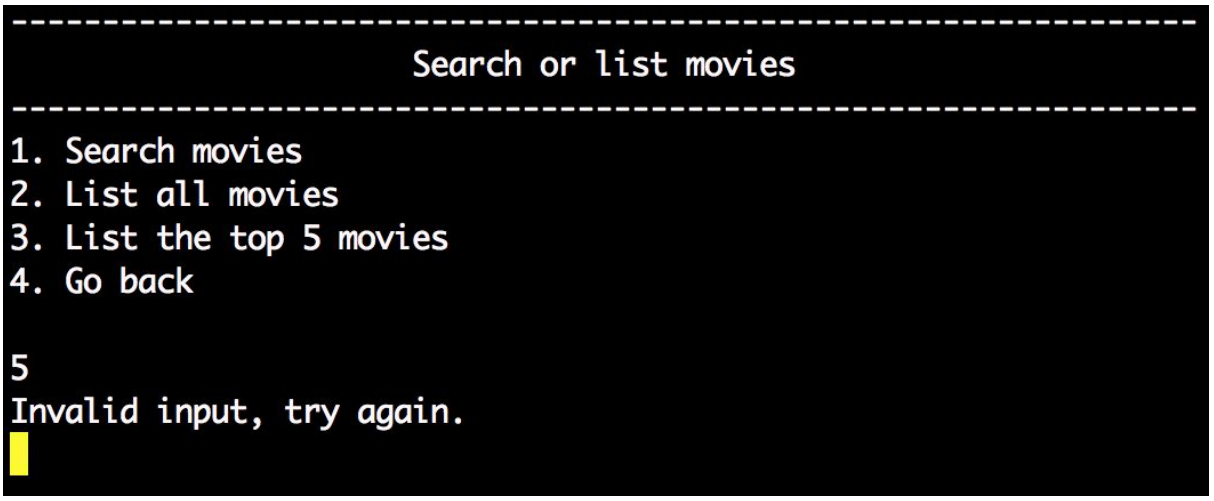
The sequence diagram illustrates the process of “Search/List movies & View movie details - including reviews and ratings



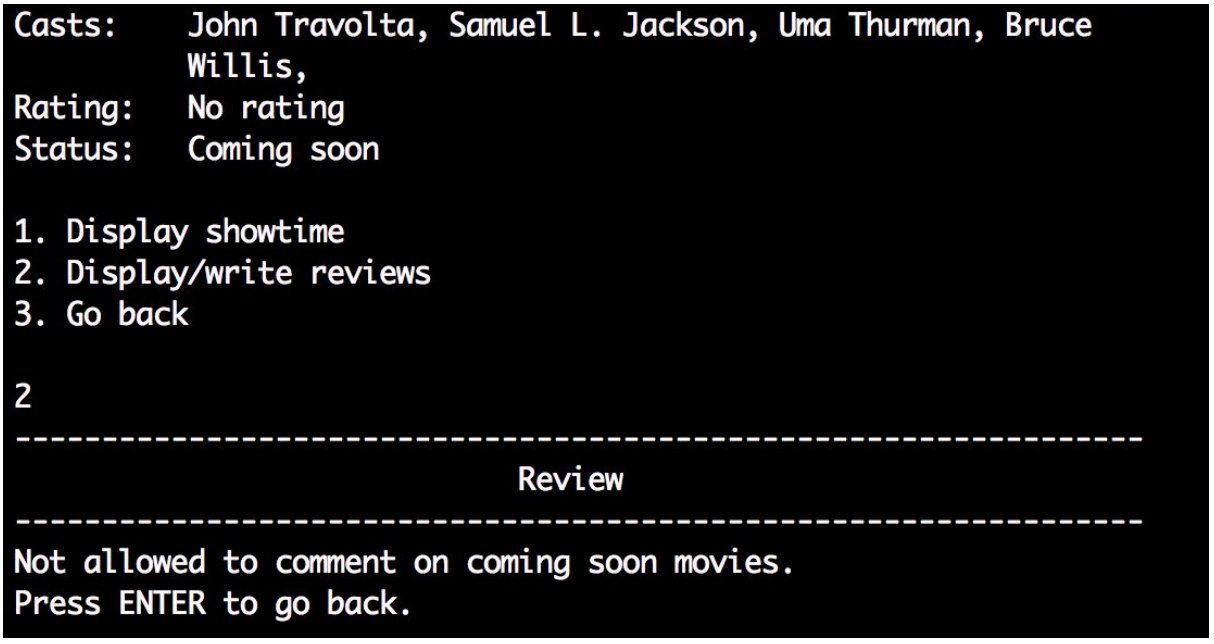
Please refer to attached file (SequenceDiagram.jpg) for higher resolution.

Screen Captures of Testings

1. Invalid input for *readChoice* method:



2. Reviewing a Coming Soon movie is not allowed:



3. Booking for a Coming Soon movie is only allowed on tomorrow or the day after tomorrow (not allowed on today):

```
Casts:    John Travolta, Samuel L. Jackson, Uma Thurman, Bruce
          Willis,
Rating:   No rating
Status:   Coming soon

1. Display showtime
2. Display/write reviews
3. Go back

1
1. November, 18
2. November, 19
Please choose a date:
```

4. Drop an existing showtime

```
------------------------------------------------------------------
                The Cathay: November 17, 16:30
------------------------------------------------------------------
Cineplex: The Cathay
Cinema: JPN
Time: Fri Nov 17 16:30:00 SGT 2017

1. Modify cineplex/cinema
2. Modify time
3. Remove the showtime
4. Go back

3
Are you sure to remove the showtime?
Enter Y to confirm, N to cancel:
y
The showtime has been removed.
------------------------------------------------------------------
                            Show time
------------------------------------------------------------------
No showtime on that day.

1. Add a show time
2. Go back
```

5. Change the base price of a cinema:

```
------------------------------------------------------------
                        CHN (Platinum)
------------------------------------------------------------
The ticket price of the cinema is 10.0.
Proceed to change?
Enter Y to confirm, N to cancel:
y
Enter the new ticket price:
50
Ticket price has been successfully changed.
```

```
------------------------------------------------------------
                Ticket price for Dunkirk (3D)
------------------------------------------------------------
                    Weekdays        Weekends
Regular Citizens    50.00           60.00
Senior Citizens     25.00           30.00
```

6. Change top five movie ranking schema:

```
------------------------------------------------------------
              Configure top 5 ranking schema
------------------------------------------------------------
Current top 5 ranking is ordered by overall rating,
Change to order by ticket sales?
Enter Y to confirm, N to cancel:
y
Successfully changed the setting.
```

```
------------------------------------------------------------
                          Movies
------------------------------------------------------------
1. Dunkirk                              (Now showing) [2]
2. Train to Busan                       (Now showing) [1]
3. Pulp Fiction                         (Coming soon) [No sale]
4. Avengers: Infinite War               (Coming soon) [No sale]
5. Brooklyn                             (Now showing) [No sale]
6. Go back
```

7. Add a new cinema:

```
------------------------------------------------------------
                          AMK Hub
------------------------------------------------------------
SGP(3D) MYS(3D) THA(Digital)
```

```
------------------------------------------------------------
                     Configure cinemas
------------------------------------------------------------
1. List cinemas
2. Add cinemas
3. Go back

2
1: JEM
2: The Cathay
3: West Mall
4: AMK Hub
Choose a cineplex to add the cinema:
4
Is this a platinum cinema?
Enter Y for yes, N for no:
y
Is this cinema for 3D movies?
Enter Y for yes, N for no
y
What's the base price for the cinema?
(Weekday price = base price * 1.2, senior citizens enjoy 50% off)
You are advised to set a higher base price for 3D and platinum cinemas
1.8
Enter the cinema code
e.g. ABC (do not enter the same cinema code for two different cinemas)
IND
Successfully added the cinema.
```

```
------------------------------------------------------------
                          AMK Hub
------------------------------------------------------------
SGP(3D) MYS(3D) THA(Digital) IND(3D)
```

Declaration of Original Work for CE/CZ2002 Assignment

APPENDIX B:

Declaration of Original Work for 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 (CE2002 or CZ2002)	Lab Group	Signature /Date
Liu Mingyu	CZ2002	SE2	刘明宇 / 17 Nov. 2017
Gao Liu	CZ2002	SE2	曹流 / 17 NOV. 2017.
Ma Yuqian	CZ2002	SE2	马钰茜 / 17 Nov 2017
Chen Guanyu	CZ2002	SE2	[illegible] / 17 Nov 2017

Important notes:

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