



Programming II

Year 1 (2021/22), Semester 2

SCHOOL OF INFOCOMM TECHNOLOGY

Diploma in Data Science
Diploma in Cybersecurity & Digital Forensics
Diploma in Information Technology

ASSIGNMENT

Duration	: 2 weeks (17 Jan 2022 to 31 Jan 2022)		
Weightage	: 30% of total coursework		
Individual/Team	: Team (2 students)		
Format	: Programming - Presentation	Basic Requirements	(50%)
		Advanced Requirements	(20%)
			(30%)

Cut-Off Date/Time: **Monday, 31 January 2022, 8:30 AM**

There is a total of 10 pages (including this page) in this hand-out.

WARNING

If a student is found to have submitted work not done by him/her, he/she will not be awarded any marks for this assignment. Disciplinary action will also be taken. Similar action will be taken for the student who allows other student(s) to copy his/her work.

Movie Ticketing System

In this assignment, you are to apply Object Oriented Programming to develop a simple **Movie Ticketing System**. The assignment requirements described below are broken down into 2 stages of development, described in this document as '**Basic Features**' and '**Advanced Features**'. You are advised to do your programming progressively in these stages. Refer to the '**Grading Criteria**' to have an idea of how the different components are graded.

1. BACKGROUND

Singa Cineplexes, a leading cinema exhibitor on our island country state, has engaged your team to develop a movie ticketing system to computerize the movie ticket sales process. For a start, its management has requested for a simple prototype of the system to better establish the effectiveness of this solution as a system for ticket counter staff and as a self-service kiosk.

The company has a total of 20 cinema halls located at 5 locations, and is home to the most number of movie screenings in the country. Every movie scheduled for screening has been approved by the relevant authority and given a classification as follows:






	General Suitable for all ages.
	Parental Guidance 13 Restricted to persons aged 13 and above
	No Children Under 16 Restricted to persons aged 16 and above.
	Mature 18 Restricted to persons aged 18 and above.
	Restricted 21 Restricted to persons aged 21 and above

Table 1 – Movie classification

A list of cinema hall and movie information is centrally managed by headquarters. Staff can then schedule and manage a movie screening session based on the given details i.e., Cinema Hall and Movie. Each screening of a movie can either be in a 2D or 3D format. A 30 mins cleaning time is allocated after the end of every movie screening session.

During the ticket purchase process, the counter staff will require the purchaser to provide the age of the ticket holder should the movie fall under the PG13, NC16, M18, or R21 categories. The ticket will only be sold if the age requirement is met.

To support price discrimination strategies, tickets are sold at different prices depending on several factors such as days of the week, type of screening (e.g., 3D, 2D), opening date of the movie, and selected concession holders. The ticket prices are given in Table 2.

	Screening Type			
	3D Movies		2D Movies	
Days of Week	Monday to Thursday	Friday to Sunday	Monday to Thursday	Friday to Sunday
Adult Price	\$11.00	\$14.00	\$8.50	\$12.50
Student Price	\$8.00		\$7.00	
Senior Citizen Price	\$6.00		\$5.00	
First 7 days of movie opening date	Student and senior citizen tickets are to be charged at Adult ticket price			

Table 2 – Information of ticket prices

If a customer wishes to purchase a student ticket, the level of study must be provided (e.g., Primary, Secondary, Tertiary). For a senior citizen ticket, the ticket holder's age must be 55 and above. By default, an adult ticket is to be purchased, even for children who do not fall into the student category. For customers buying adult tickets, they will also be entitled to purchase a popcorn set at a discounted rate of \$3.00.

Information for use with the Movie, Cinema and Screening classes are given in the **Movie.csv**, **Cinema.csv** and **Screening.csv** files respectively. All files can be found and downloaded from MeL.

The class diagram for the movie ticketing system is shown in Figure 1 on the next page.

The different "status" in the Order class are as follows:

1. Unpaid (When a new order is created)
2. Paid (After payment is made)
3. Cancelled

The order number in the Order class is a running number starting from 1.

The screening number in the Screening class is a running number starting from 1001.

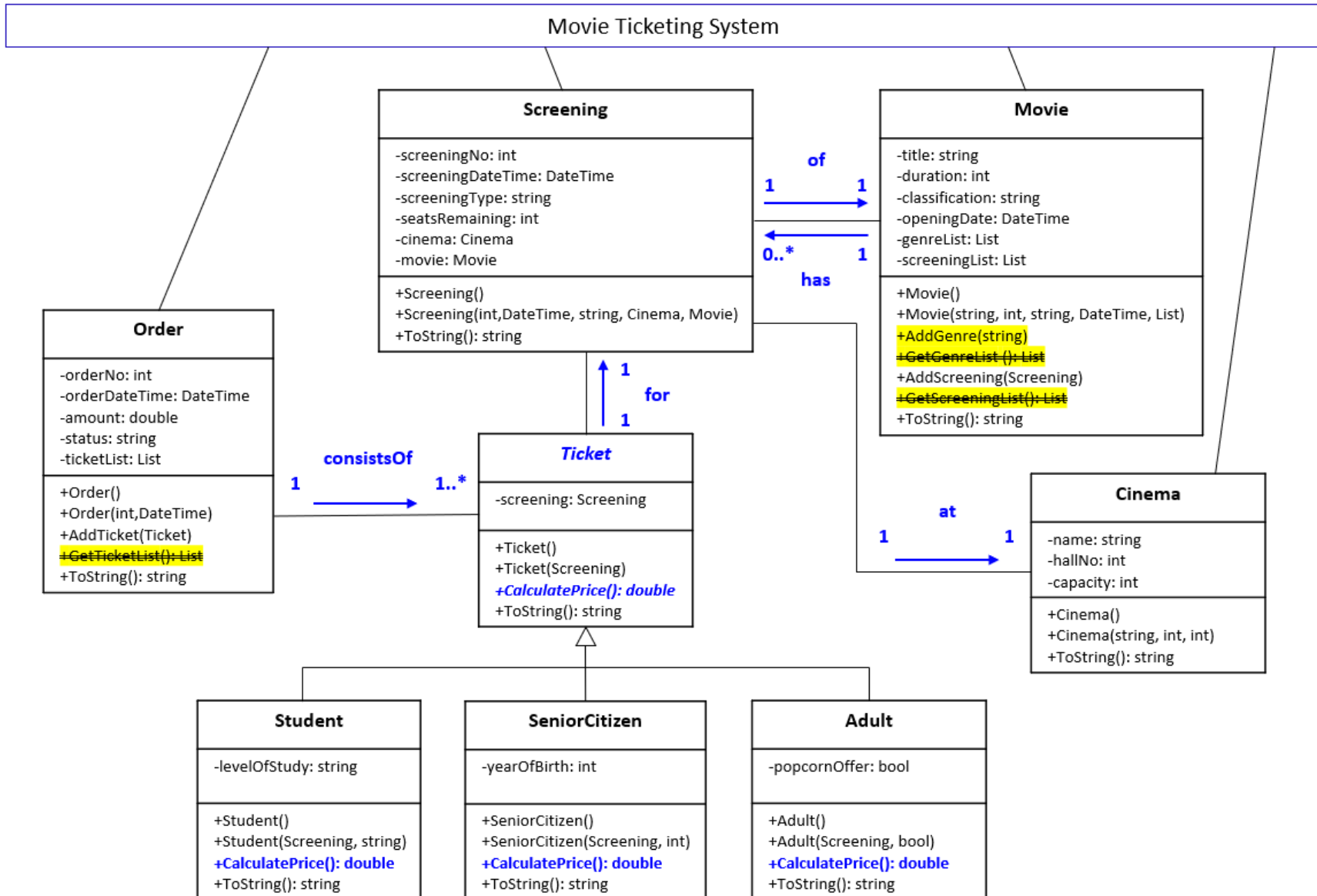


Figure 1: Class Diagram for Movie Ticketing System

2. BASIC FEATURES

=== General ===

1) Load Movie and Cinema Data

1. load data from given CSV files and populate into two lists

2) Load Screening Data

1. load data from given csv file and populate into a list

3) List all movies

1. display the information of all movies

4) List movie screenings

1. list all movies
2. prompt user to select a movie
3. retrieve movie object
4. retrieve and display screening sessions for that movie

=== Screening ===

5) Add a movie screening session

1. list all movies
2. prompt user to select a movie
3. prompt user to enter a screening type [2D/3D]
4. prompt user to enter a screening date and time (check to see if the datetime entered is after the opening date of the movie)
5. list all cinema halls
6. prompt user to select a cinema hall (check to see if the cinema hall is available at the datetime entered in point 4) [need to consider the movie duration and cleaning time]
7. create a Screening object with the information given and add to the relevant screening list
8. display the status of the movie screening session creation (i.e. successful or unsuccessful)

6) Delete a movie screening session

1. list all movie screening sessions that have not sold any tickets
2. prompt user to select a session
3. remove the movie screening from all screening lists
4. display the status of the removal (i.e. successful or unsuccessful)

=== Order ===**7) Order movie ticket/s**

1. *list all movies*
2. *prompt user to select a movie*
3. *list all movie screenings of the selected movie*
4. *prompt user to select movie screening*
5. *retrieve the selected movie screening*
6. *prompt user to enter the total number of tickets to order (check if figure entered is more than the available seats for the screening)*
7. *prompt user if all ticket holders meet the movie classification requirements (except movies classified as G)*
8. *create an Order object with the status "Unpaid"*
9. *for each ticket,*
 - a. *prompt user for a response depending on the type of ticket ordered:*
 - i. *Student: level of study [Primary, Secondary, Tertiary]*
 - ii. *Senior Citizen: year of birth (must be 55 years and above)*
 - iii. *Adult: popcorn for \$3 [Y/N]*
 - b. *create a Ticket object (Student, SeniorCitizen or Adult) with the information given*
 - c. *add the ticket object to the ticket list of the order*
 - d. *update seats remaining for the movie screening*
10. *list amount payable*
11. *prompt user to press any key to make payment*
12. *fill in the necessary details to the new order (e.g amount)*
13. *change order status to "Paid"*

8) Cancel order of ticket

1. *prompt user for order number*
2. *retrieve the selected order*
3. *check if the screening in the selected order is screened*
4. *update seat remaining for the movie screening based on the selected order*
5. *change order status to "Cancelled"*
6. *display a message indicating that the amount is refunded*
7. *display the status of the cancelation (i.e. successful or unsuccessful)*

▪ Validations (and feedback)

- *The program should handle all invalid entries by the user e.g. invalid option, invalid year, invalid month, invalid day, etc.*
- *If user made a mistake in the entry, the program should inform the user via appropriate feedback*

3. ADVANCED FEATURES

Each of you is required to do at least one advanced feature below.

3.1 Recommend movie based on sale of tickets sold

3.2 Display available seats of screening session in descending order

3.3 Other Possible Features

- Possible features:
 - Top 3 movies based on tickets sold
 - Top sales chart of the Cinema Name
 - Add seat number to each Ticket sold
 - Use Web API to enhance the system
- You may gain up to 5 bonus marks if you propose and successfully implement an additional feature. Check with your tutor with your idea before implementing.

IMPORTANT INSTRUCTIONS:

- The team has to divide the work when implementing the classes and the General part.
- A student is to implement the Screening portion, and another for Order.
- Individual student without a team is required to implement the General part, and other parts to be discussed with your tutor.
- You are expected to create a main menu that is used for navigating through all features of your system (8 basic features + 3 advanced features [2 compulsory + 1 optional]).
- Please note that you should implement the advanced features only AFTER all the basic features have been fully implemented and working.
- NO MARKS will be awarded for the advanced features if the basic features have NOT been fully implemented and working.
- Marks will be deducted if you are not able to show your understanding of the program, both basic and advanced features (if applicable), during the presentation.

4. ACTIVITY PLAN

Suggestions for Getting Started

a) Analysis

1. Understand the program specification and the requirements before attempting the assignment.
e.g. the relationships between the classes
the use of the attributes in each class

b) Program Design

2. Work out the User Interface required for user input and suitable output.
3. Work out the main logic of the program using Object-Oriented programming techniques;
i.e. use inheritance and association of the classes properly.
4. You are required to use suitable classes appropriately for this assignment.
Marks will be deducted for inefficient use of the classes or improper use of classes

c) Implementation & Testing

5. Determine the order in which the classes are to be implemented (certain classes need to be implemented before other classes can be implemented).
6. Implement the classes **ONE** at a time.
7. Test your program logic to make sure that it works as expected.
You must prepare test data to see that your program works correctly. All data entry should be validated and illegal data entry should be highlighted to the user so that the user can enter correct data.

5. DELIVERABLES

- Name your BitBucket repository "PRG2_TXX_TeamY" where "TXX" is your tutorial group, and "TeamY" is your team number, e.g. PRG2_T01_Team1.
- In each of your .cs file, you MUST include a blocked comment at the top stating your student number(s), name(s) and group as shown below:

```
//=====
// Student Number : S12345678, S87654321
// Student Name   : John Tan, Johnny Yeo
// Module Group   : T01
//=====
```

- Ensure all classes (source files) that you have written for the whole assignment are pushed to your BitBucket repository by 31 January 2022, 8.30 am.

- In addition, submit the whole project folder, including all the classes (source files) that you have written, to your PRG2 Assignment network folder (\\ictspace.ict.np.edu.sg\PRG2Assignment) before the deadline.
- Demonstrate your application to your tutor during your PRG2 classes right after the submission deadline of 31 January 2022.

7. GRADING CRITERIA

This assignment constitutes 40% of this module. Performance Criteria for grading the assignment is as described below. Marks awarded will be based on **program code** as well as student's degree of understanding of work done as assessed during the **presentation**.

Grading criteria for the program is given below.

A Grade

- ◆ Program implements the *Basic Features* successfully
- ◆ Program implements all the basic *input validations* successfully
- ◆ Program implements the *Advanced Features* successfully
- ◆ Program demonstrates good design with the correct use of methods
- ◆ Program provides strong evidence of good programming practice
- ◆ Program has been tested adequately
- ◆ Demonstrates good use of Git (Such as meaningful commit messages, regular commits and push)

B Grade

- ◆ Program implements the *Basic Features* successfully
- ◆ Program implements some basic *input validations* successfully
- ◆ Program attempts to use methods
- ◆ Program provides sufficient evidence of good programming practice
- ◆ Program has been tested adequately
- ◆ Adequate use of Git

C Grade

- ◆ Program implements the *Basic Features* successfully
- ◆ Program provides some evidence of good programming practice
- ◆ Program has been tested adequately
- ◆ Demonstrates some evidence of usage of Git

D Grade

- ◆ Program implements the *Basic Features* successfully
- ◆ Program has been tested adequately

NOTE

- *Evidence of good programming practice include the use of meaningful variable names, proper indentation of code, appropriate and useful comments, adoption of standard naming conventions etc.*
- *Basic Input validation refers to the checking of the inputs entered by the user. e.g. invalid option, invalid date*