Programming II

Year 1 (2020/21), Semester 2

SCHOOL OF INFOCOMM TECHNOLOGY

Diploma in Financial Informatics
Diploma in Cybersecurity & Digital Forensics
Diploma in Information Technology

ASSIGNMENT

Duration	: 2 weeks (18 Jan 2021 to 1 Feb 2021)		
Weightage	: 40% of total coursework		
Individual/Team	: Team (2 students)		
Format	: Programming -	Basic Requirements	(50%)
		Advanced Requirements	(20%)
	Presentation		(30%)

Cut-Off Date/Time: **Monday, 1 February 2021, 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.

Last Update: 14/01/2021

COVID-19 Monitoring System

In this assignment, you are to apply Object Oriented Programming to develop a simple ***COVID-19 Monitoring 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.

NOTE: The scenario attempts to mirror as closely as possible to existing measures and initiatives, but with a reduced complexity. Please be advised to refer to official sources for application beyond this assignment.

1. Background

Since first identified in late 2019, COVID-19 has ravaged countries around world, halting economies, grinding international travel, and has affected all walks of lives. Similarly, Singapore as an international hub for multiple sectors such as healthcare, education, and trade, is not spared by the pandemic.

Singapore has recorded more than 58,000 COVID-19 cases as of Jan 2021. The country had to activate a month-long partial lockdown in Apr 2020 as a preventive measure by the government to reduce the community spread of the virus. Since then, border controls have also been in place to reduce the risk of importation of COVID-19 to the nation.

TravelEntry

As the country transitions into a new normal, Singapore's border and health control measures are regularly updated in response to the evolving global COVID-19 situation. For example, travellers from certain countries will need to serve a 14-day Stay-Home Notice (SHN) at SHN dedicated facilities (SDF), while some are allowed to serve a 7-day SHN at their own accommodation, as shown in Table 1.

Country	**Entry Arrangements**
- New Zealand - Vietnam	- No SHN (0 day) - Swab test
- Macao SAR	- 7-day SHN at own accommodation - Swab test
- All other countries	- 14-day SHN at SHN dedicated facility (SDF) - Swab test

Table 1: SHN Requirements based on Country

At the end of the SHN period, or immediately for those not requiring SHN, the following charges are applicable. For all entry, they are subject to a COVID-19 swab test that costs $200 before 7% GST.

Category	SHN Mode	Swab Test (bef GST)	Additional Charges (bef GST)
Residents	None	$200	-
	7-day SHN at own accommodation		Transportation – $20
	14-day SHN at SDF		Transportation – $20 SDF charge - $1,000
Visitors	None		Transportation – (to be calculated)
	7-day SHN at own accommodation		
	14-day SHN at SDF		Transportation – (to be calculated) SDF charge - $2,000

The transportation cost for *visitors* are calculated based on the following:

Base Fare	$50 + Distance from selected checkpoint (Land/Sea/Air) * $0.22
Entry during: **6 am to 8.59 am; or** **6 pm to 11.59 pm**	Additional 25% surcharge of base fare
Entry during: **Midnight to 5.59 am**	Additional 50% surcharge of base fare

SafeEntry & TraceTogether

For quick identification of persons who may have come into close contact with anyone who has tested positive for COVID-19, the SafeEntry and TraceTogether system were introduced. The SafeEntry[1] system requires every individual to "check-in" when visiting a location (e.g., shopping centre, shop, attractions), while TraceTogether [2] promotes community-driven contact tracing to facilitate contract tracing efforts.

The SafeEntry system allows for businesses to register for a location to enable the check-in process. In addition, it ensures that the number of visitors at a location does not exceed the allowable maximum capacity.

For individuals wishing to participate in the TraceTogether programme but do not wish to use their mobile device for this purpose, TraceTogether tokens can be issued to residents. Each TraceTogether token expires 6 months after issuing, and it can be replaced only within 1 month from its expiry. TraceTogether tokens are only issued to residents.

[1] https://www.safeentry.gov.sg/
[2] https://www.tracetogether.gov.sg/

Information for use with the Person (and related sub-classes) and BusinessLocation classes are given in the ***Person.csv*** and ***BusinessLocation.csv*** files respectively. Both files can be found and downloaded from MeL.

The information for use with the **SHNFacility** class is available by calling an API at the following link: https://covidmonitoringapiprg2.azurewebsites.net/facility

Your program is required to load the above information at the start of your program.

The class diagram for the COVID-19 monitoring system is shown in Figure 1 on the next page.

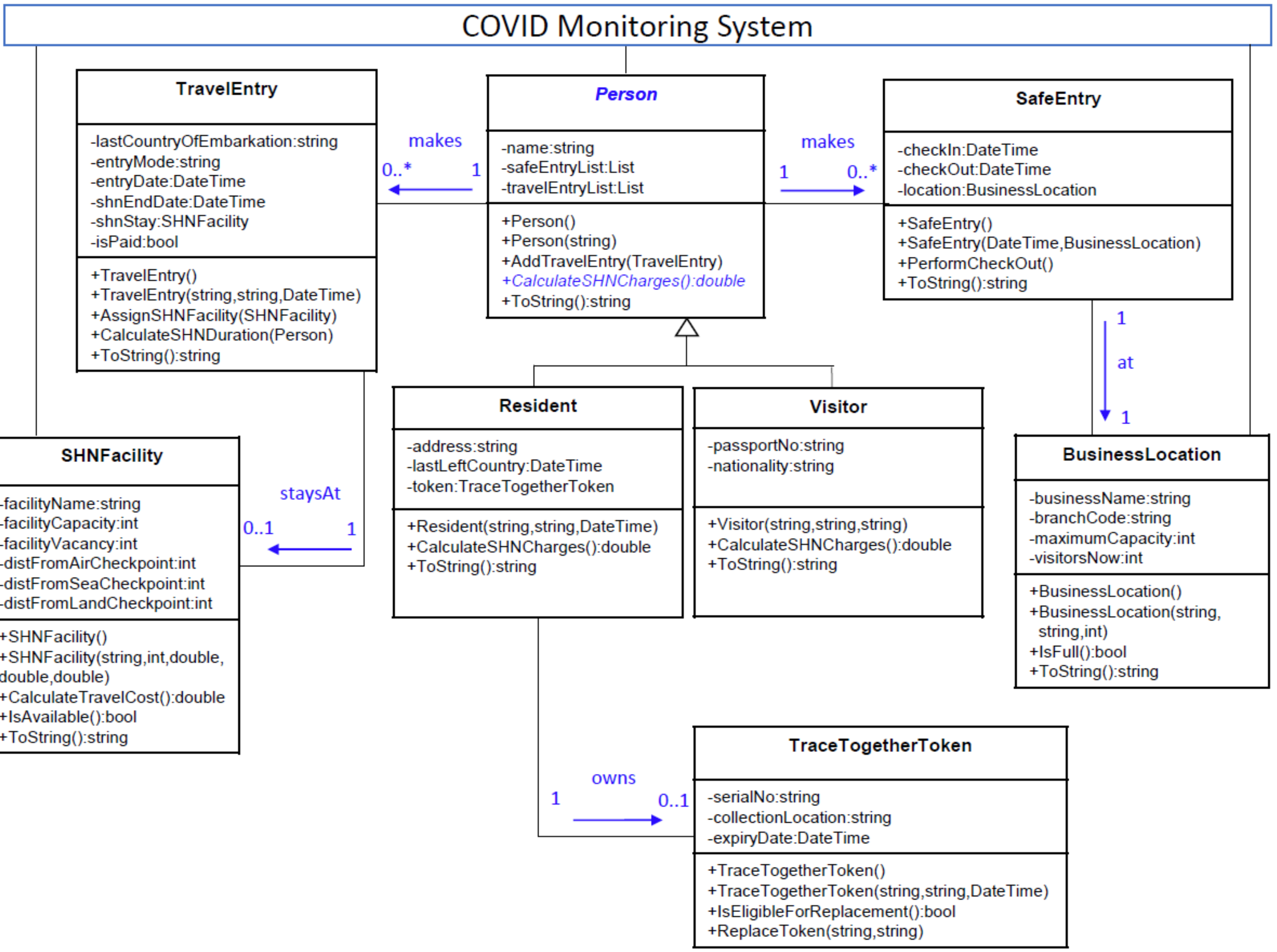

Figure 1: Class Diagram for COVID-19 Monitoring System

2. BASIC FEATURES

===General===

1) Load Person and Business Location Data

1. *load both given CSV and populate two lists*

2) Load SHN Facility Data

1. *call API and populate a list*

3) List all Visitors

4) List Person Details

1. *prompt user for name*
2. *search for person*
3. *list person details including TravelEntry and SafeEntry details*
   i. *if resident, display TraceTogetherToken details*

===SafeEntry/TraceTogether===

5) Assign/Replace TraceTogether Token

1. *prompt user for name*
2. *search for resident name*
3. *create and assign a TraceTogetherToken object if resident has no existing token*
4. *replace token if it meets the criteria stipulated in the background brief*

6) List all Business Locations

7) Edit Business Location Capacity

1. *prompt user to enter details*
2. *search for business location*
3. *prompt user to edit maximum capacity*

8) SafeEntry Check-in

1. *prompt user for name*
2. *search for person*
3. *list all business locations*
4. *prompt user to select for business location to check-in*
5. *create SafeEntry object if the location is not full, and increase visitorsNow count*
6. *add SafeEntry object to person*

9) SafeEntry Check-out

1. *prompt user for name*
2. *search for person*
3. *list SafeEntry records for that person that have not been checked-out*
4. *prompt user to select record to check-out*
5. *call PerformCheckOut() to check-out, and reduce visitorsNow by 1*

===TravelEntry===

10) List all SHN Facilities

11) Create Visitor

1. *prompt user for details*
2. *create Visitor object*

12) Create TravelEntry Record

1. *prompt user for name*
2. *search for person*
3. *prompt user for details (last country of embarkation, entry mode)*
4. *create TravelEntry object*
5. *call CalculateSHNDuration(Person) to calculate SHNEndDate based on criteria given in the background brief*
6. *list SHN facilities if necessary, for user to select*
7. *assign chosen SHN facility if necessary, and reduce the vacancy count*
8. *call AddTravelEntry() in Person to assign the TravelEntry object*

13) Calculate SHN Charges

1. *prompt user for name*
2. *search for person*
3. *retrieve TravelEntry with SHN ended and is unpaid*
4. *call CalculateSHNCharges() to calculate the charges based on the criteria provided in the background brief*
    i. *Note: To add 7% GST*
5. *prompt to make payment*
6. *change the isPaid Boolean value*

- **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*
    - *The program is to display appropriate feedback messages (e.g., successful or not successful)*

3. Advanced Features

You are required to do all the advanced features below.

3.1 Contact Tracing Reporting

- *Given a date/time and business name, generate a list of persons that are checked-in at that location and period.*
- *Export a CSV with details of their visit (e.g., check-in time, check-out time)*

3.2 SHN Status Reporting

- *Given a date, generate a csv report of all travellers serving their SHN, their SHN end date, and where they are serving their SHN.*

3.3 Other Possible Features

- *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 TravelEntry portion, and another for SafeEntry / TraceTogether.*
- *Individual student without a team is required to implement the General part, and either the TravelEntry, or SafeEntry/TraceTogether.*
- *You are expected to create a main menu that is used for navigating through all features of your system (13 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 1 February 2021, 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 before the deadline.

- Demonstrate your application to your tutor during your PRG2 classes right after the submission deadline of 1 February 2021.

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*