BIT203 Assignment 1

Release Date:	Friday, 29th October 2021
Due Date:	Saturday, 27th November 2021
Value:	15%
Assessment Mode:	Individual Assignment

Rationale

This assignment has been designed to allow students to test and demonstrate their ability to write a Java program that uses a range of different concepts and facilities. This assessment relates to the following learning outcomes:

Expected Learning Outcomes Assessed

- CLO1 write programs using several classes based on UML class diagrams and other models
- CLO2 apply object-oriented concepts in the design and implementation of the programs

In particular, this assignment tests the students ability to use appropriate class hierarchies and collection classes.

Private Covid-19 Vaccination Scheme

The Covid-19 vaccination rollout in Malaysia and most countries have been underway for a few months. As more and more of the population have been vaccinated, private healthcare centres have been allowed to purchase and administer vaccines, especially to patients who want to choose the type of vaccine that they receive. However, the vaccinations that have been administered by the private healthcare centres have to be recorded in the national vaccination committee database. A system is required for the Private Covid-19 Vaccination Scheme, known as PCVS.

The conceptual model/domain class diagram for PCVS is given in Fig.1.

The controller class PCVS must maintain at least a collection of User objects, a collection of Healthcare Centre, and a collection of Vaccine objects. Note that User is an abstract class, and is the superclass to the two concrete subclasses – Patient and Administrator. A Patient may request vaccination appointment. An Administrator works for Heathcare Centre, and his tasks include records new vaccine batch, confirms vaccination appointment, and records vaccination administered. For all classes, you may include additional methods if you wish to do so, for example the *toString*, and *equals* methods. Two users are considered equal if their *username* are the same. Associations between the classes are to be implemented according to the class diagram and as necessary to realize the use cases described.

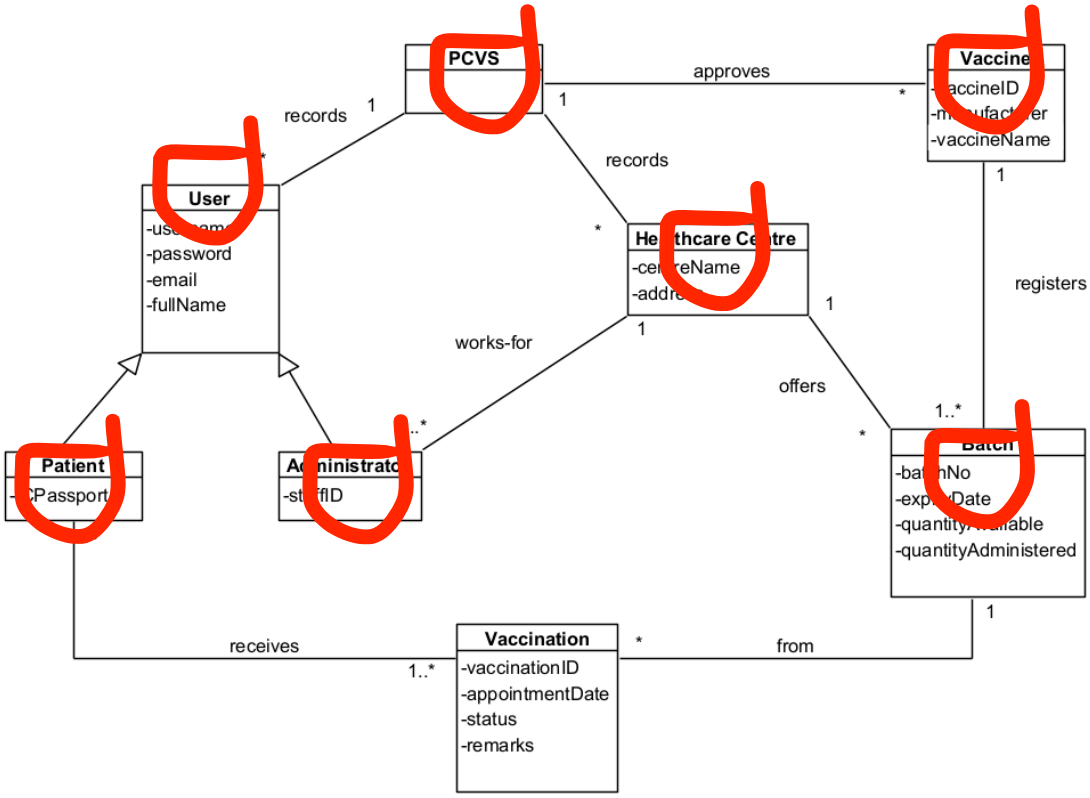

Figure 1: PCVS Class Diagram

The use case diagram is given in Fig. 2.

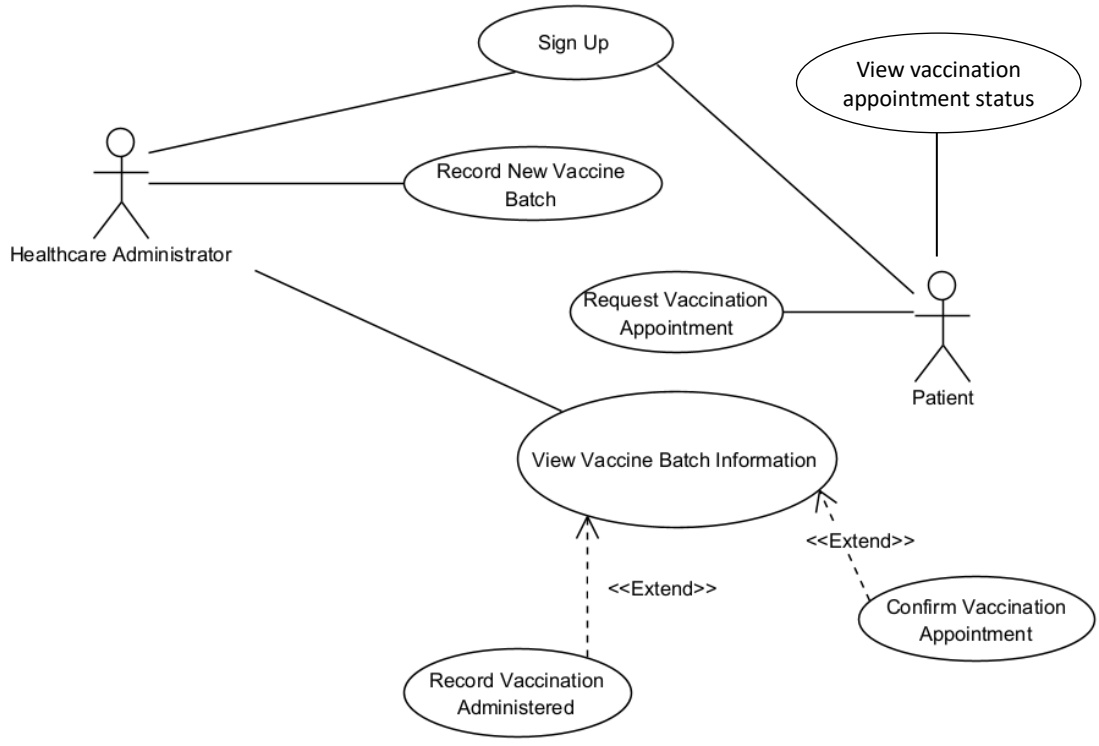

Figure 2: PCVS Use Case Diagram

The use cases are given briefly below:

Use Case 1	Sign Up	
Goal in Context	To allow new users to sign up for accounts in the PCVS	
Primary Actor **Secondary Actor**	Healthcare Administrator Patient	
Trigger	A new user wants to sign up to participate in the private Covid-19 vaccination scheme	
Typical Course of Events **Actor Action**		**System Response**
1. This use case begins when a healthcare administrator wants to sign up to the PCVS as a healthcare administrator.		
2. The healthcare administrator selects the centreName of the healthcare centre.		The centreName and centreAddress of the healthcare centre are displayed.
3. The healthcare administrator enters the username, password, email, and full name to sign up.		A Healthcare Administrator account is created for the healthcare administrator for the healthcare centre. The staffID is automatically generated.
Alternative Course of Events		
Line 2a: If the user is a patient, skip to Line 3a		
Line 3a. The patient enters the username, password, email, full name, IC or passport and a Patient account is created.		

Use Case 2	Record New Vaccine Batch	
Goal in Context	To allow a healthcare administrator to list the available batches of vaccines at a healthcare centre.	
Primary Actor **Secondary Actor**	Healthcare Administrator (None)	
Trigger	A healthcare centre has received a new batch of vaccines.	
Typical Course of Events **Actor Action**		**System Response**
1. This use case begins when a healthcare administrator wants to record that a new batch of vaccines is available.		
2. The healthcare administrator logs in with a valid username and password		The healthcare centre name is shown.
3. The healthcare administrator selects the vaccineID		The manufacturer and the vaccine name is shown.
4. The healthcare administrator enters the batch number, expiry date and the quantity of doses available.		The batch is recorded for the vaccine and the healthcare centre.
Alternative Course of Events		

Use Case 3	Request Vaccination Appointment	
Goal in Context	To allow a patient to request a vaccination appointment.	
Primary Actor **Secondary Actor**	Patient (None)	
Trigger	A patient wants a vaccination appointment.	
Typical Course of Events **Actor Action**		**System Response**
1.This use case begins when a patient wants to request a vaccination appointment.		
2. The patient logs in to request an appointment by entering a username and password		The patient's full name is shown.
3.The patient selects to view available vaccines.		A list of vaccine name and manufacturer is shown.
4. The patient records the vaccine that is required		A list of healthcare centres name and addresses offering this vaccine is shown.
5. The patient selects a healthcare centre to view.		The batches of vaccines that have quantity available and not yet expired is shown.
6. The patient selects a batchNo for a vaccine batch.		The expiry date and quantity available is shown. The quantity available is calculated based on the number of pending and administered vaccinations.
7. The patient selects an upcoming date to request an appointment.		A vaccinationID is generated for a new Vaccination. The vaccination status is recorded as "pending" and the vaccination is created for the patient and the batch.
Alternative Course of Events		
Line 4a: If there are no suitable batches, the patient may repeat lines 3 and 4.		
Line 7a: If the appointment date is after the batch expiry date, let the patient select another date.		

Use Case 4	View Vaccine Batch Information	
Goal in Context	To view vaccination appointments for a healthcare centre	
Primary Actor **Secondary Actor**	Healthcare Administrator (None)	
Trigger	A healthcare administrator wants to confirm the appointment request.	
Typical Course of Events **Actor Action**		**System Response**
1. This use case begins when the healthcare administrator wants to check information about a vaccine batch.		
2. The healthcare administrator logs in with a valid username and password		The healthcare centre name is shown with a list of available vaccine batches by vaccine name and number of pending appointments.

3. The healthcare administrator selects a batchNo.	The information about the batch expiry date, number of available, pending and administered vaccinations is shown. A list of vaccinations is shown with the status and the appointment date.
4. The healthcare administrator selects a vaccinationID.	The information about the vaccine, vaccination batch and patient is shown.
5. The healthcare administrator selects manage the vaccination or logs out.	
Alternative Course of Events	
Line 3a: The healthcare administrator may choose to view a different batch.	
Line 4a: The healthcare administrator may choose to view information about a different vaccination.	
Line 5a: If the healthcare administrator wants to confirm the vaccination appointment, proceed to use case 5: Confirm vaccination appointment.	
Line 5b: If the healthcare administrator wants to record that the vaccination has been administered, proceed to use case 6: Record vaccination administered.	

Use Case 5	Confirm Vaccination Appointment	
Goal in Context	To confirm a patient's appointment request	
Primary Actor **Secondary Actor**	Healthcare Administrator (None)	
Trigger	A healthcare administrator wants to confirm/reject the appointment request.	
Typical Course of Events **Actor Action**		System Response
1. This use case <<extends>> use case 4: View Vaccine Batch Information.		
2. The healthcare administrator selects a vaccinationID.		The full name, IC or passport number of the patient is shown together with the batch no, expiry date, manufacturer and name of the vaccine.
3. The healthcare administrator confirms the vaccination date.		The status is set to 'confirmed' and a confirmation email will be sent to the patient.
Alternative Course of Events		
Line 3a: The healthcare administrator may reject the appointment request and enter remarks. The status will be changed to "rejected" and an email sent to the patient with the remarks.		

Use Case 6	Record Vaccination Administered	
Goal in Context	To record that a vaccination has been administered to a patient.	
Primary Actor **Secondary Actor**	Healthcare Administrator (None)	
Trigger	A patient has received a vaccination.	
Typical Course of Events **Actor Action**		**System Response**
1. This use case <<extends>> use case 4: View Vaccine Batch Information.		
2. The healthcare administrator selects a vaccinationID.		The full name, IC or passport number of the patient is shown together with the batch no, expiry date, manufacturer and name of the vaccine.
3. The healthcare administrator confirms the vaccination has been administered and enters any remarks.		The vaccination status is set to 'administered' and the quantity administered for the batch is updated.
Alternative Course of Events		

Use Case 7	View Vaccination Appointment Status	
Goal in Context	To allow a patient to check his vaccination appointment status.	
Primary Actor **Secondary Actor**	Patient (None)	
Trigger	A patient wants to view his/her vaccination appointment status.	
Typical Course of Events **Actor Action**		**System Response**
1. This use case begins when a Patient wants to check the status of his or her vaccination appointment.		
2. The patient logs in by entering a username and password		The patient's full name is shown.
3.The patient selects to view status of available vaccination appontment.		The status of the vaccination appointment is shown.
Alternative Course of Events		
Line 3a: If there is no available vaccination appointment, an error message is displayed.		

Assignment Requirements:

User interface specifications

A Java application fulfilling the role of a user interface should be initiated in a class called `PCVSConsole`. This class is not shown in the diagram (it is not a problem domain class) but is a view class that interacts with the `PCVS` controller. The class should provide a *console style* user interface. That is, all output for the user should be directed to standard output (and appears on the screen) and all input should be obtained from standard input (read from the keyboard).

This console interface class will provide a menu that allows a user of your program to perform the following operations:

- given a *username*, *password, email, fullname* and ICPassport ADD a Patient to the created PCVS object;
- given a *username*, *password, email,* and *fullname* ADD an Administrator to the created PCVS object;
- allows the Administrator to record new vaccine batch;
- allows a Patient to request a vaccination appointment;
- allows an Administrator to view vaccine batch information;
- allows an Administrator to confirm/reject vaccination appointment;
- allows an Administrator to record vaccination administered;
- allows a Patient to view status of vaccination appointment;
- display detail of all users, either in original sequence, or sorted according to fullname;
- display detail of all vaccination appointments.

NOTE:

i) The details of some tasks are shown in the high level use cases in previous pages.

ii) When the application starts, you should create a 'single' working PCVS object. You then add two or three HealthcareCentres by hard-coding using any valid values. You should add two or three Vaccines of your choices as well.

The `PCVSConsole` class must contain the application's **main method** so that the application can be launched with a command equivalent to

```
java PCVSConsole
```

User interaction and output

It is a specific requirement of this assignment that **none** of the problem domain classes listed above may contain any user interaction code, including the reading of values from a keyboard or interactive input device and none of the problem domain classes may generate any output for the user, such as screen messages or prompts.

It is acceptable (encouraged) to have screen output messages generated by problem domain classes for **debugging** purposes. These can be very useful during the implementation and testing phases of development. Debugging code should be removed or commented out of all problem domain classes in the final version of the application.

Design and implementation

In completing this assignment you should carefully consider the design of the system, one use case at a time. Once you have coded each class, test its functionality completely. You should only make use of 'getters' and 'setters' to access and alter attributes, and provide a toString method for each class.

You will need to design for and include the appropriate collection objects as well. You will also need to ensure that the data that is input by users is valid.

Coding style and comments

Your source code should be clear and readable, using correct indentation, meaningful identifiers and comments. Include javadoc comments and tags as follows:

- for public classes, to indicate their purpose;
- for public methods, to indicate their effect, parameters and return values;

- for public fields, to indicate their purpose.

You will be expected to generate and submit the javadoc documentation for your classes as part of this assignment – included inside the submitted compressed file (together with the Java source code).

SUBMISSION REQUIREMENT

Your assignment has to submit to TurnItIn:

1. All your Java source files, printed in Word document format
2. Printed output (showing your interactivity with your program) is to be included at the end of your Java source files, in Word document created in (1)
3. Generate a runnable jar file of your project, and compress the jar file together with all your java source files into ONE file.
4. Submit your solutions to the Turnitin link created in LMS:
   - The Word document created in (1) is uploaded to Part 1 in Turnitin.
   - The compressed file created in (3) is uploaded to Part 2 in Turnitin.

Marking Scheme

Refer to the Excel file, *203A1MS_SS21.xlsx*, for detailed breakdown of the marks allocated for the requirements.

Note:

If your program does not meet the requirements by the due date you should obtain help from the lecturer and notify the lecturer that you will submit the assignment late (marks will be deducted).

Note about testing and plagiarism

It is very important that you **complete** this assignment **alone**. You may of course obtain general assistance from the lecturing staff in the subject and your peers, but the coding must be carried out yourself. It is normally quite easy to detect when two or more students work together on their coding.

It is also very important that the demonstration of the results of your program using the given test data is produced using the identical version of the program to the printout of your source code. **Students who hand in substantially similar assignments or whose programs do not match their demonstration of testing will fail the assignment.**

Any student suspected of copying, or of not producing the work himself or herself, can be called for **oral examination**, where the student will be expected **to demonstrate sufficient knowledge of the application** to show that it is his or her own original work.

Assignment No.: __1___

Assignment Cover Sheet

Student Information (For group assignment, please state names of all members)		Grade/Marks
Name	**ID**	

Module/Subject Information		Office Acknowledgement
Module/Subject Code	BIT203	
Module/Subject Name	Advanced OO Programming	
Lecturer/Tutor/Facilitator		
Due Date	27 November 2021	
Assignment Title/Topic	Assignment 1	
Intake (where applicable)		
Word Count	n/a	**Date/Time**

Declaration

- I/We have read and understood the Programme Handbook that explains on **plagiarism**, and I/we testify that, unless otherwise acknowledged, the work submitted herein is entirely my/our own.
- I/We declare that no part of this assignment has been written for me/us by any other person(s) except where such collaboration has been authorized by the lecturer concerned.
- I/We authorize the University to test any work submitted by me/us, using text comparison software, for instances of plagiarism. I/We understand this will involve the University or its contractors copying my/our work and storing it on a database to be used in future to test work submitted by others.

Note: 1) The attachment of this statement on any electronically submitted assignments will be deemed to have the same authority as a signed statement.
2) The Group Leader signs the declaration on behalf of all members.

Signature:	Date:
E-mail:	

Feedback/Comments*
Main Strengths
Main Weaknesses
Suggestions for improvement

	Student acknowledge feedback/comments
Grader's signature	Student's signature:
Date:	Date:

Note:

1) A soft and hard copy of the assignment shall be submitted.
2) The signed copy of the assignment cover sheet shall be retained by the marker.
3) If the Turnitin report is required, students have to submit it with the assignment. However, departments may allow students up to **THREE** (3) working days after submission of the assignment to submit the Turnitin report. The assignment shall only be marked upon the submission of the Turnitin report.

*Use additional sheets if required.