University of Westminster
School of Electronics and Computer Science

4COSC005W Software Development 2 – Coursework	
Module leader	Iresh Bandara
Weighting:	50% of the module
Qualifying mark	30%
Description	Coursework
Learning Outcomes Covered in this Assignment:	LO1, LO3, LO4, LO5.
Handed Out:	10th June 2021
Due Date	Code due on Blackboard coursework upload Monday 26th July 2021 11.00pm.
Expected deliverables	a) Zip the project directory of each implementation and upload as w1234557_arrays_only.zip, and w1234567_classes.zip. b) Submit test results (pdf) c) Online demo
Method of Submission:	Blackboard
Type of Feedback and Due Date:	Written feedback and marks 15 working days (3 weeks) after the submission deadline. All marks will remain provisional until formally agreed by an Assessment Board.

Assessment regulations

Refer to section 4 of the "How you study" guide for undergraduate students for a clarification of how you are assessed, penalties and late submissions, what constitutes plagiarism etc.

Penalty for Late Submission

If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand.

Coursework Description

COVID-19 VACCINATION CENTER Program.

Task 1. Arrays version.
Design a program for a **COVID-19 VACCINATION CENTER** which can vaccinate six people simultaneously using similar code to the code given below (see attachment 1). Start by checking that the code works. Once the basic code runs, implement the following functionalities as sperate procedures. You can build up your test cases as you develop your program (see testing below). Vaccination Center will have exactly 150 vaccinations in their stock. For each Patient added to the booth, stock should be updated (reduce 1 for each person), and a warning message should be displayed when the stock reaches a value of 20. Operator should be able to perform the following tasks by selecting from a console menu.

100 or VVB:	**V**iew all **V**accination **B**ooths
101 or VEB:	**V**iew all **E**mpty **B**ooths
102 or APB:	**A**dd **P**atient to a **B**ooth
103 or RPB:	**R**emove **P**atient from a **B**ooth
104 or VPS:	**V**iew **P**atients **S**orted in alphabetical order (Do not use library sort routine)
105 or SPD:	**S**tore **P**rogram **D**ata into file
106 or LPD:	**L**oad **P**rogram **D**ata from file
107 or VRV:	**V**iew **R**emaining **V**accinations
108 or AVS:	**A**dd **V**accinations to the **S**tock
999 or EXT:	**E**xit the Program

Display all the menu options to the operator, When the operator types 102 or APB, it should do the add method. When the operator types 103 or RPB, it should do the remove method.

Task 2. Classes version.
Create a second version of the **COVID-19 VACCINATION CENTER PROGRAM** using an *array of Booth objects*. Create a class called **VacinationCenter** and another class called **Booth**. The program should function as in Task 1.

Task 3. Extend your programs from Task 1 and Task2. Modify both programs so that each booth can now hold the following additional information. (Hint: you will need a Patient class for the class version)

a. Additional information of the Patient.
   i. First Name.
   ii. Surname.
   iii. Age (Class version only)
   iv. City (Class version only)
   v. NIC or Passport Number (Class version only)
   vi. Vaccination Requested
   (AstraZeneca,Sinopharm,Pfizer)

b. There are three types of vaccinations given by the center based on the request of the Patient.

    Booth 0 & 1: AstraZeneca
    Booth 2 & 3: Sinopharm
    Booth 4 & 5: Pfizer

    Place the Patient into the correct booth depending on the vaccination request.

(This task will familiarise you with what we mean by "**maintainability**" of a program. If you do not use classes, you will need to use parallel arrays! While you are doing this task think about which of the programs was easier to extend and why)

Task 4. Linked List version. Add a waiting list to your **VACCINATION CENTER** class version.

Modify
102 or APB:	Add Patient to a Booth and
103 or RPB:	Remove Patient from a Booth as follows:
When adding a new Patient, if the VACCINATION CENTER is full, the Patient should be added to a Waiting List (a Linked List).
When releasing a Patient from a booth, the next Patient in the waiting list should be automatically placed in the booth.
Extra marks will be awarded if you implement the waiting list considering the vaccination request.

Task 5. JavaFX. Create a GUI for the operator to enter the Patient details (details taken in the task 3) along with the vaccination type given, Booth number issued etc. there should be a "**Generate Receipt**" button which should generate a receipt for the Patient to be printed with the date-time stamp. Design of the UI and the receipt is up to you.

Task 6. **Testing**. Create a table of test cases showing how you tested your program (see below for example). Write a brief (no more than one page) discussion of how you chose your test cases to ensure that your tests cover all aspects of your program.

Test Case	**Expected Result**	**Actual Result**	**Pass/Fail**
(Booths Initialised correctly) After program starts, 100 or VVB	Displays 'empty' for all booths	Displays 'empty' for all booths	Pass
(Add Patient "Scott" to Booth 5) 102 or APB enter "Scott"	100 or VVB Displays "Scott" for booth 5	Displays "Scott" for booth 4	Fail

Note: Solutions should be java console applications up to task 4, javaFX application for task 5

Marking scheme

The coursework will be marked based on the following marking criteria:

Criteria	**Max for Subcomponent**	**Max Subtot**
Task 1 One mark for each option (10 options) Menu works correctly	5 2	**(7)**
Task 2 Booth class correctly implemented. Options implemented and methods work correctly (1 mark each option)	10 5	**(15)**
Task 3 Arrays version implementation (7) Placing patient in the correct booth Class Version: Patient class implementation (7) Placing patient in the correct booth	7 8 7 8	**(30)**
Task 4 Waiting list as Linked List implementation 102 or APB: Add Patient to a Booth works correctly 103 or RPB: Remove Patient works correctly Placing patient in the correct booth based on the vaccination request	10 3 3 4	**(20)**
Task 5 JavaFX GUI for entering Details GUI for receipt	 5 5	**(10)**
Task 6 Test case coverage and reasons	10	**(10)**
Coding Style (Comments, indentation, style) **Total**	8	**(8)** **(100)**
Demo: Marks allocated for your ability to answer questions and demonstrate understanding of your solutio ○ **Each Task has a demo component of 50%. If you cannot explain your code and are unable to point a reference within your code of where this code was found (i.e., in a textbook or on the internet) th no marks will be given for the demo of that component.**		
NOTE: If you do not attend your online demo only Part 1 and Part 2 will be marked.		

Attachment 1

```java
import java.util.Scanner;

public class ServiceCenterExample {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        String customerName;
        int tokenNum;
        String agentName;
        int boothNum =0;
        String[] ServiceCenter = new String[7];
        //for (int x = 0; x < 6; x++ ) hotel[x] = ""; //initialise
        initialise(ServiceCenter); //better to initialise in a procedure
        while ( boothNum < 6 )
        {
            for (int x = 0; x < 6; x++ )
            {
                if (ServiceCenter[x].equals("e"))System.out.println("booth
" + x + " is empty");
            }
            System.out.println("Enter booth number (0-5) or 6 to stop:" );
            boothNum = input.nextInt();
            System.out.println("Enter customer name for booth " + boothNum
+" :" ) ;
            customerName = input.next();
            ServiceCenter[boothNum] = customerName ;

            for (int x = 0; x < 6; x++ )
            {
                System.out.println("booth " + x + " occupied by " +
ServiceCenter[x]);
            }
        }
    }
    private static void initialise( String hotelRef[] ) {
        for (int x = 0; x < 6; x++ ) hotelRef[x] = "e";
        System.out.println( "initilise ");
    }

}
```