University of Westminster
School of Electronics and Computer Science

4COSC005/W Software Development 2 – Ref/Def Coursework

Module leader	Rajitha Jayasinghe
Weighting:	50% of the module
Qualifying mark	30%
Description	Coursework
Learning Outcomes Covered in this Assignment:	LO1, LO3, LO4, LO5.
Handed out:	18th June 2021
Due Date	Code due on Blackboard coursework upload Monday 5th July 2021 / 1pm (LK)
Expected deliverables	a) Zip the project directory of each implementation and upload as w1234557_arrays_only.zip, and w1234567_classes.zip to code submission link b) Report to report submission link. Copy and paste the code to the report and attach outputs as screenshots. Do not attach screenshots of your code. c) Online viva
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. For more detailed information regarding University Assessment Regulations, please refer to the following website:**http://www.westminster.ac.uk/study/current-students/resources/academic-regulations**

Coursework Description

Hotel Program.

Note for the ref/def version of the coursework you must implement a different sort algorithm to Bubble sort. Suggestions are Insertion sort or Selection sort, but you may choose any sorting algorithm other than Bubble sort. You will also be required to explain how the sort algorithm works in your demonstration.

Task 1. **Arrays version.** Design a program for a hotel with **twelve** rooms using code similar to the code given here. Start by checking that the code works.

```
package arrays;
import java.util.*;
public class HotelExample {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        String roomName;
        int roomNum = 0;
        String[] hotel = new String[7];
        //for (int x = 0; x < 6; x++ ) hotel[x] = ""; //initialise
        initialise(hotel); //better to initialise in a procedure
        while ( roomNum < 6 )
        {
            for (int x = 0; x < 6; x++ )
            {
                if (hotel[x].equals("e"))System.out.println("room " + x + " is
                        empty");
            }
            System.out.println("Enter room number (0-5) or 6 to stop:" );
            roomNum = input.nextInt();
            System.out.println("Enter name for room " + roomNum +" :" ) ;
            roomName = input.next();
            hotel[roomNum] = roomName ;
            for (int x = 0; x < 6; x++ )
            {
                System.out.println("room " + x + " occupied by " + hotel[x]);
            }
        }
    }
    private static void initialise( String hotelRef[] ) {
        for (int x = 0; x < 6; x++ ) hotelRef[x] = "e";
        System.out.println( "initilise ");
    }
}
```

Once the basic code runs, put the code that 'Views All rooms' and 'Adds customer to room', into separate procedures, and test it works. You can build up your test cases as you develop your program (see testing below).

Then add a menu system which will allow the user to choose what they want to select. Enter an 'A' to add a customer to a room, and a 'V' to view all rooms. Implement each as a method. When an 'A' is pressed, it should do the Add method; a 'V' should do the View method.

One by one, add extra methods to do each of the following. The user should be able to choose from the menu what the program does.

E:	Display Empty rooms
D:	Delete customer from room
F:	Find room from customer name
S:	Store program data into file

L: Load program data from file
O: View guests Ordered alphabetically by name. (**Implement insertion sort algorithm**)

Task 2. **Classes version.** Create a second version of the Hotel program using an *array of Room objects*. Create a class called Hotel and another class called Room. The program should function as in Task 1.

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

a. The number of guests in a room.
b. Additional information for the paying guest.
    i. First Name.
    ii. Surname.
    iii. Credit Card number.

(This task will familiarise you with what we mean by "**maintainability**" of a program. If you don't 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. Queue version. Add a waiting list to your Hotel class version. Modify your 'A: Add' and 'D: Delete' as follows:
When you press 'A' to add a new customer, if the hotel is full, the customer should be added to a Waiting List (a queue).
When you press 'D' to delete a customer from a room, the next customer in the waiting queue should be automatically placed in the room.
Extra marks will be awarded if you implement the waiting list as a circular queue.

Task 5. **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**
(Rooms Initialised correctly) After program starts, Press 'V'	Displays 'e' for all rooms	Displays 'e' for all rooms	Pass
(Add customer "Bob" to room 5) Select A, enter "Bob"	Press 'v' Displays "Bob" for room 5	Displays "Bob" for room 4	X

Note: Solutions should be java console applications (not windows).

Marking scheme

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

Criteria	Max for Subcomponent	Max Subtotal
Task 1 One mark for each option (A,V,E,D,F,S,L,O) Menu works correctly	8 2	**(10)**
Task 2 Room class correctly implemented. Options implemented as methods work correctly (1 mark each option)	12 8	**(20)**
Task 3 Arrays version implementation (7) (1 Mark for each option that works correctly) Hotel Room class implementation (7) (1 mark for each option that works correctly)	7 8 7 8	**(30)**
Task 4 Waiting list as queue implementation "A: Add"works correctly "D: Delete"works correctly Circular queue	10 3 3 4	**(20)**
Task 5 Test case coverage and reasons	10	**(10)**
Coding Style (Comments, indentation, style) **Totals**	10	**(10)** **(100)**
Demo: Marks allocated for your ability to answer questions and demonstrate understanding of your solutions. ○ **Each Task has a demo component of 50%. If you cannot explain your code and are unable to point to a reference within your code of where this code was found (i.e., in a textbook or on the internet) then 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.		