UECS2354 SOFTWARE TESTING
202401 - ASSIGNMENT

<u>Project details:</u>

You are to develop and test a Hotel Room Booking System. Consider the following requirements:

- A user can be a VIP member, normal member or non-member.
- Exclusive reward can be entitled to a VIP or normal member occasionally.
- Default room type allocation is shown in Table 1.
- The rules for room allocation based on member type are given below.

i. **VIP member**
   - If VIP room is fully booked when a VIP member places a booking, Deluxe room will be allocated instead.
   - In case Deluxe room is fully booked, Standard room will be allocated.
   - If Standard room is fully booked too, then the VIP member will be placed into VIP waiting list.
   - A VIP member can book up to 3 rooms at a time. If only one VIP room is available, the other rooms can be both Deluxe or Standard rooms or combination of one Deluxe room and one Standard room.
   - If the number of rooms requested cannot be fulfilled, the member is placed into VIP waiting list.

ii. **Normal member**
   - If Deluxe room is fully booked when a normal member places a booking,
     - o Standard room will be allocated to member without exclusive reward.
     - o a member with exclusive reward will be allocated VIP room subject to availability.
       - ▪ Upon allocation, the exclusive reward is marked redeemed.
       - ▪ If the VIP room is unavailable, Standard room will be allocated and the exclusive reward remains valid for other redemptions.
   - If Standard room is fully booked too, then the member will be placed into member waiting list.
   - A normal member can book up to 2 rooms at a time. The combinations of the rooms allocated can be both Deluxe/Standard rooms, one VIP and one Deluxe, one VIP and one Standard or one Deluxe and one Standard.
   - If the number of rooms requested cannot be fulfilled, the member is placed into member waiting list.

iii. **Non-member**
   - If Standard room is fully booked when a non-member places a booking, he/she will be placed into normal waiting list.
   - A non-member can book for only one room at a time.

- User is allowed to cancel the booking. If the user is in the waiting list, it must be removed from the list.

User	**Default Room type**
VIP member	VIP
Normal member	Deluxe
Non-member	Standard

Table 1: Room type allocation

Assignment Deliverables:

Part A: Test Plan

Based on the project details and requirement, plan for the testing activities. Your test plan must contain scope, objective, test basis, features to be tested and not to be tested, test conditions, test entry and exit criteria. Refer to the test plan template.

Part B: Test Design

Based on the project details, **create decision table**, and **design the test cases** for the testing phase. There must be clear description in every test case. Refer to the test case template.

After analyse the requirement and clearly state the test objectives and test scope, create the decision table based on the condition stated in the test plan. You can use excel to create your decision table. After identify the rules for each conditions, design the test cases accordingly. Put the rule identifier as reference in your test cases.

For ex: if you have 2 rules for a condition #1. You can give any identifiers for the rules (*Rule #1* and *Rule #2*). In the test case template, design test case for each of this rules.

Test Case#	Test Title	Test Summary	Test Steps	Test Data	Expected Result
Test case 001	To verify rule#1	if *condition #1* is true, the action will be *action#1*	[list down the steps]	User = "VIP" VIP Room = "Available"	Room allocated = "VIP"
Test case 002	To verify rule#2	if *condition #2* is true, the action will be *action#2*	[list down the steps]	User = "VIP" VIP Room = "Booked"	Room allocated = "Deluxe"
Test case 003	To verify rule#3	if *condition #3* is false, the action will be *action#3*	[list down the steps]	Type = [other than stated]	Should return error.

****This is just an example, it's depends on how you design your test cases and it will be unique for each individual testers.**

Part C: Application code and Test Code

Jar files: Place all JAR files (jUnitParams & Mockito) in C:\ jar_files

1. Application code:
   Partial class diagram is shown in Figure 1. You are responsible to complete and implement the User, Booking and WaitingList classes in the application.

   The setBooking() method in Booking class will determine the type of room allocated or place the user into a waiting list.

   The cancelBooking() will cancel a booking and remove the user from the waiting list if applicable.

   The **Printer** and **Room** classes are not ready for testing yet. However, the signature of each method is given as follow:

   Room class: method to check for room availability
   public boolean checkRoom(String room_type)

   Printer class: method to print the booking information
   public void printInfo(String name, String member_type, String room_type)

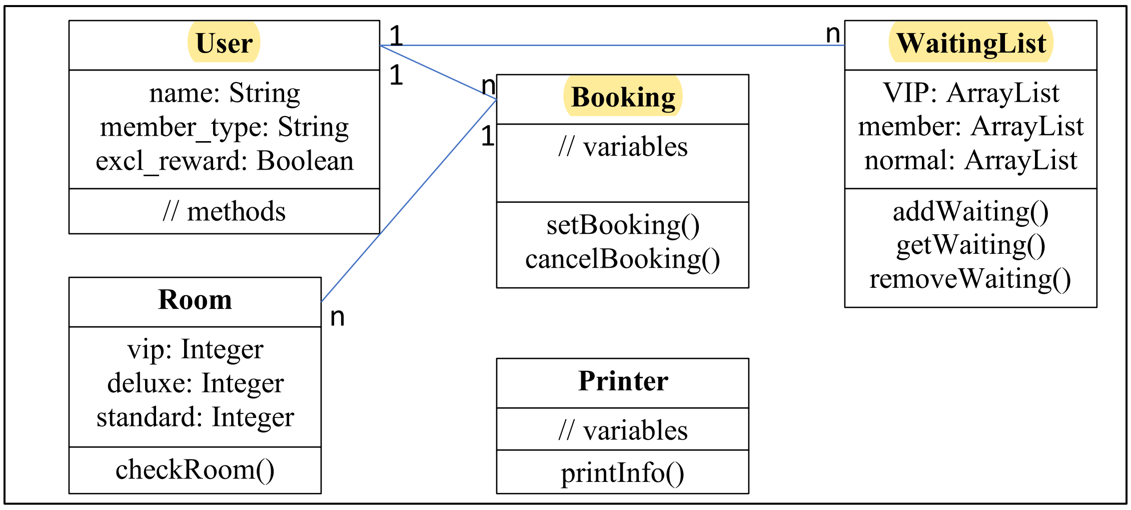

Figure 1: Class diagram

2. Test code:
   Write test code using jUnit Framework and Mockito to test the application code.

Further Information:

The particular testing issues that you need to pay attention to and their associated marks are shown in the marking scheme (202401_*UECS2354_Marking.docx*). Include comments in test code to clearly specify which aspect of these testing issues you are addressing. For example, when creating parameterised tests, include comments to state the approach that you are using (e.g., boundary value analysis, etc.).

The focus is NOT to write as many tests as possible, but rather to create the right number of tests necessary to verify the functionality of the methods. For example, if you are using equivalence partitioning, writing tests that use all the inputs within the same partition that produce identical output from a method is a waste of space.

You may give additional assumptions for your application, state them clearly in the report. To make your program more robust and avoid problems at run time, do as much status/error checking as you could in your program. And, good organization of the code and meaningful variable names would help readability, and liberal use of comments can help the marker understands what the program does and why. In addition, provide class diagram(s) to illustrate the design of your program.

Submission details:

Due date:
22 April 2024 (Monday – W13)

This is a group assignment. Form a group of 4 members in a group. Register your group member's name in the google sheet:
https://docs.google.com/spreadsheets/d/11ibN5zclCzsYR1ctKaT_UAuy1lXDKb1XjTxtnJLE8rg/edit?usp=sharing

Submit the following items through WBLE (link will be created in WBLE course page).

1. **Java project folder**:
   Archive of Eclipse **project folder** that contains source code for both the application code and the test code. The application code and test code should be placed in two separate source directories.

   Note: Do not include jar files as it may be blocked.

2. **Report**:
   Your report should contain the following:
   - Marking sheet (.doc) as the front page of the documentation
   - Test plan (Part A)
   - Test cases design (Part B)
   - Assumptions (if any)
   - Class diagram for the application code *(If you update, add, remove any details for the classes given above, reflect it in the class diagram)*
   - The application code and jUnit test code, should be attached at the end of the documentation.

Total mark

The total mark of this practical assignment is 100. The 100 marks will contribute 20% of your final mark. It's your responsibility to understand the requirements of the tasks and prepare well for your submission. You might be asked questions about the works you submit to ensure that you understand them.

Late Submission

No late submission of assignment is allowed. Assignment received after the due date without valid reasons will be penalized using the following policy: 5 marks will be deducted for every day the assignment is overdue.

Plagiarism

It is important that your solutions to the practical assignment be your own work. It is perfectly acceptable to seek help and advice when completing the practical assignment, but this must not be taken to the point where what is submitted is in part someone else's work. Any group found to have committed plagiarism or cheating by copying program codes from other sources will be given a failing grade.