SHIELD TECHNOLOGIES(PVT)LTD

SHIELD TECHNOLOGIES
RAVANA LABS DEPARTMENT

REQURITEMENT EXAMINATION 2021

Instructor	**Sheron Don**
Description	**REQURITEMENT EXAMINATION 2021**
Learning Outcomes Covered in this test :	**Arrays version implementation & java programing knowledge**
Handed Out:	**01st April 2021**
Due Date	**03rd April 2021 1pm.**
Expected deliverables	**a) Zip the project directory of each implementation and upload as name_arrays_only.zip, and name_classes.zip. b) Submit test results (pdf or doc) c) Online demo**
Method of Submission:	**Mail us : shieldtec2020@gmail.com Mention as heading your name and test details**
Type of Feedback and Due Date:	**Written feedback and marks 02 working days after the submission deadline.**

Description

Hotel Program.

Task 1. Arrays version. Design a program for a hotel with eight rooms using the code given below. 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. (Do not use library sort routine)

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 familiarize 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 ‘E’	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 test 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	**(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. o **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.**		