UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

SOFTWARE DEVELOPMENT II

Course Work

Loganathan Bhavaneetharan
Westminster ID: w1810599
IIT ID: 20201212
Email: loganthan.20201212@iit.ac.lk

Table of contents

Test Cases

Task 1

Test Case	Expected Result	Actual Result	Pass/Fail
(Rooms Initialized correctly) Run the program.	"All Rooms are initialized!" is displayed.	"All Rooms are initialized!" is displayed.	Pass
(Main menu displayed correctly) Run the program	Main menu is displayed in the correct order (A, V, E, D, F, S, L, O, X)	Main menu is displayed in the correct order (A, V, E, D, F, S, L, O, X)	Pass
(Enter your preference works correctly for valid input) Enter "V" or "v"	Displays 'empty' for all rooms	Displays 'empty' for all rooms	Pass
(Enter your preference works correctly for invalid input) Enter "1" or "$@#" or " "	'Invalid Input! Try Again!' is displayed & prompts for choice.	'Invalid Input! Try Again!' is displayed & prompts for choice.	Pass
(Add customer is opened correctly) Enter "A" or "a" for choice.	Program opens add customer method & Displays 'Enter room number (1 – 8)'	Program opens add customer method & Displays 'Enter room number (1 – 8)'	Pass
(Add customer "Bhavan" to room 1) Enter room number: 1, Enter name: "Bhavan".	Displays "Customer Added!" Press 'V' Displays "Bhavan" for room 1	Displays "Customer Added!" Press 'V' Displays "Bhavan" for room 1	Pass
(Add customer with invalid input) Enter "19" or "$@#" or " " or "-6" for room number	"Room number is Invalid!!!" is displayed. Prompts for choice again.	"Room number is Invalid!!!" is displayed. Prompts for choice again.	Pass
(Add customer to an occupied room) Enter "1" for room number (occupied room)	"Room is already occupied! Try Again!!!" is displayed. Prompts for choice again.	"Room is already occupied! Try Again!!!" is displayed. Prompts for choice again.	Pass

(View all rooms) Enter “V” or “v" for choice.	Displays room detail in correct order	Displays room detail in correct order	Pass
(View Empty rooms) Enter “E” or “e" for choice.	Displays only the empty rooms. If there are no empty rooms. "No empty rooms, All Rooms are occupied!!!" is displayed.	Displays only the empty rooms. If there are no empty rooms. "No empty rooms, All Rooms are occupied!!!" is displayed.	Pass
(Delete customer is opened correctly) Enter “D” or “d” for choice.	Program opens delete customer method & Displays ‘Enter room number’	Program opens add customer method & Displays ‘Enter room number’	Pass
(Delete Customer “Bhavan”) Enter room number: 1	“Customer removed" is displayed. Press ‘V' Displays “empty” for room 1	“Customer removed" is displayed. Press ‘V' Displays “empty” for room 1	Pass
(Delete Customer “Bhavan” with invalid input) Enter “19” or “$@#” or “ ” or “-6” for room number	“Room number is Invalid!!!" is displayed. Prompts for choice again.	“Room number is Invalid!!!" is displayed. Prompts for choice again.	Pass
(Delete Customer from “empty” room) Enter “5" for room number (empty room)	“Room is already empty" is displayed	“Room is already empty" is displayed	Pass
(Load Customer from file) Enter “L” or “l” for choice.	"Hotel data Loaded Successfully!" is displayed. If the file does not exist or if any other error happens "Could not load the data!!!" is displayed. Press ‘V', Displays room detail in correct order	"Hotel data Loaded Successfully!" is displayed. (“With no errors") Press ‘V', Displays room detail in correct order	Pass

(Store Customers to a file) Enter “S” or “s” for choice.	"Hotel data Stored Successfully!" is displayed. if any other error happens "Could not store the data!!!" is displayed The details are stored in correct on the text file	"Hotel data Stored Successfully!" is displayed. (“With no errors") The details are stored in correct on the text file	Pass
(Find customer is opened correctly) Enter “F” or “f” for choice.	Program opens find customer method & Displays ‘Enter Customer name’	Program opens find customer method & Displays ‘Enter Customer name’	Pass
(Find Customer room number by giving the name) Enter “F" or “f” for choice. Enter name “James" or “james"	Room number 4 is displayed.	Room number 4 is displayed.	Pass
(Find a non-existing Customer room number by giving the name) Enter “F" or “f" for choice. Enter name “kumar".	"Could not find Customer!!!" is displayed.	"Could not find Customer!!!" is displayed.	Pass
(View customers in alphabetical order) Enter “O" or “o" for choice.	Customer names are displayed in alphabetical order. Press ‘V’ the customer room numbers & names are not changed.	Customer names are displayed in alphabetical order. Press ‘V’ the customer room numbers & names are not changed.	Pass
(Exit from the program) Enter “X" or “x" for choice.	"Task Ended!" is displayed & program ends.	"Task Ended!" is displayed & program ends.	Pass

Task 2

Test Case	Expected Result	Actual Result	Pass/Fail
(Rooms Initialized correctly) Run the program.	"All Rooms are initialized!" is displayed.	"All Rooms are initialized!" is displayed.	Pass
(Main menu displayed correctly) Run the program	Main menu is displayed in the correct order (A, V, E, D, F, S, L, O, X)	Main menu is displayed in the correct order (A, V, E, D, F, S, L, O, X)	Pass
(Enter your preference works correctly for valid input) Enter "V" or "v"	Displays 'empty' for all rooms	Displays 'empty' for all rooms	Pass
(Enter your preference works correctly for invalid input) Enter "1" or "$@#" or " "	'Invalid Input! Try Again!' is displayed & prompts for choice.	'Invalid Input! Try Again!' is displayed & prompts for choice.	Pass
(Add customer is opened correctly) Enter "A" or "a" for choice.	Program opens add customer method & Displays 'Enter room number (1 – 8)'	Program opens add customer method & Displays 'Enter room number (1 – 8)'	Pass
(Add customer "Bhavan" to room 1) Enter room number: 1, Enter name: "Bhavan".	Displays "Customer Added!" Press 'V' Displays "Bhavan" for room 1	Displays "Customer Added!" Press 'V' Displays "Bhavan" for room 1	Pass
(Add customer with invalid input) Enter "19" or "$@#" or " " or "-6" for room number	"Room number is Invalid!!!" is displayed. Prompts for choice again.	"Room number is Invalid!!!" is displayed. Prompts for choice again.	Pass
(Add customer to an occupied room) Enter "1" for room number (occupied room)	"Room is already occupied! Try Again!!!" is displayed. Prompts for choice again.	"Room is already occupied! Try Again!!!" is displayed. Prompts for choice again.	Pass

(View all rooms) Enter “V” or “v" for choice.	Displays room detail in correct order	Displays room detail in correct order	Pass
(View Empty rooms) Enter “E” or “e" for choice.	Displays only the empty rooms. If there are no empty rooms. "No empty rooms, All Rooms are occupied!!!" is displayed.	Displays only the empty rooms. If there are no empty rooms. "No empty rooms, All Rooms are occupied!!!" is displayed.	Pass
(Delete customer is opened correctly) Enter “D” or “d” for choice.	Program opens delete customer method & Displays ‘Enter room number’	Program opens add customer method & Displays ‘Enter room number’	Pass
(Delete Customer “Bhavan”) Enter room number: 1	“Customer removed" is displayed. Press ‘V' Displays “empty” for room 1	“Customer removed" is displayed. Press ‘V' Displays “empty” for room 1	Pass
(Delete Customer “Bhavan” with invalid input) Enter “19” or “$@#” or “ ” or “-6” for room number	“Room number is Invalid!!!" is displayed. Prompts for choice again.	“Room number is Invalid!!!" is displayed. Prompts for choice again.	Pass
(Delete Customer from “empty” room) Enter “5" for room number (empty room)	“Room is already empty" is displayed	“Room is already empty" is displayed	Pass
(Load Customer from file) Enter “L” or “l” for choice.	"Hotel data Loaded Successfully!" is displayed. If the file does not exist or if any other error happens "Could not load the data!!!" is displayed. Press ‘V', Displays room detail in correct order	"Hotel data Loaded Successfully!" is displayed. (“With no errors”) Press ‘V', Displays room detail in correct order	Pass

(Store Customers to a file) Enter “S” or “s” for choice.	"Hotel data Stored Successfully!" is displayed. if any other error happens "Could not store the data!!!" is displayed The details are stored in correct on the text file	"Hotel data Stored Successfully!" is displayed. (“With no errors") The details are stored in correct on the text file	Pass
(Find customer is opened correctly) Enter “F” or “f” for choice.	Program opens find customer method & Displays ‘Enter Customer name’	Program opens find customer method & Displays ‘Enter Customer name’	Pass
(Find Customer room number by giving the name) Enter “F" or “f” for choice. Enter name “James" or “james"	Room number 4 is displayed.	Room number 4 is displayed.	Pass
(Find a non-existing Customer room number by giving the name) Enter “F" or “f" for choice. Enter name “kumar".	"Could not find Customer!!!" is displayed.	"Could not find Customer!!!" is displayed.	Pass
(View customers in alphabetical order) Enter “O" or “o" for choice.	Customer names are displayed in alphabetical order. Press ‘V’ the customer room numbers & names are not changed.	Customer names are displayed in alphabetical order. Press ‘V’ the customer room numbers & names are not changed.	Pass
(Exit from the program) Enter “X" or “x" for choice.	"Task Ended!" is displayed & program ends.	"Task Ended!" is displayed & program ends.	Pass

Task 3 (Class Version)

Test Case	Expected Result	Actual Result	Pass/Fail
(Rooms Initialized correctly) Run the program.	"All Rooms are initialized!" is displayed.	"All Rooms are initialized!" is displayed.	Pass
(Main menu displayed correctly) Run the program	Main menu is displayed in the correct order (A, V, E, D, F, S, L, O, X)	Main menu is displayed in the correct order (A, V, E, D, F, S, L, O, X)	Pass
(Enter your preference works correctly for valid input) Enter "V" or "v"	Displays 'empty' for Firstname, Surname, CreditCard & 0 for Guests	Displays 'empty' for Firstname, Surname, CreditCard & 0 for Guests	Pass
(Enter your preference works correctly for invalid input) Enter "1" or "$@#" or " "	'Invalid Input! Try Again!' is displayed & prompts for choice.	'Invalid Input! Try Again!' is displayed & prompts for choice.	Pass
(Add customer is opened correctly) Enter "A" or "a" for choice.	Program opens add customer method & Displays 'Enter room number (1 – 8)'	Program opens add customer method & Displays 'Enter room number (1 – 8)'	Pass
(Add customer "Bhavan" to room 1) Enter room number: 1. first name: "Bhavan". Sur name: "Loganathan". Credit card: 12345678. Guests in room: 4.	Displays "Customer Added!" Press 'V' Displays all the details for room 1correctly.	Displays "Customer Added!" Press 'V' Displays all the details for room 1correctly.	Pass
(Add customer with invalid input) Enter "19" or "$@#" or " " or "-6" for room number	"Room number is Invalid!!!" is displayed. Prompts for choice again.	"Room number is Invalid!!!" is displayed. Prompts for choice again.	Pass
(Add customer with invalid input) Enter "0" for guests in room.	"Guests in room is invalid Customer is not added" is displayed. Prompts for choice again.	"Guests in room is invalid Customer is not added" is displayed. Prompts for choice again.	Pass

(Add customer with invalid input) Enter “19” or “$@#” or “ ” or “-6” for room number or guests in room	"Room number / Guests in Room is Invalid!!!" is displayed. Prompts for choice again.	"Room number / Guests in Room is Invalid!!!" is displayed. Prompts for choice again.	Pass
(Add customer with invalid input) Enter “-1” or “0” for guests in room.	"Guests in room is invalid Customer is not added" is displayed. Prompts for choice again.	"Guests in room is invalid Customer is not added" is displayed. Prompts for choice again.	Pass
(Add customer to an occupied room) Enter “1” for room number (occupied room)	“Room is already occupied! Try Again!!!" is displayed. Prompts for choice again.	“Room is already occupied! Try Again!!!" is displayed. Prompts for choice again.	Pass
(View all rooms) Enter “V” or “v” for choice.	Displays room detail in correct order	Displays room detail in correct order	Pass
(View Empty rooms) Enter “E” or “e” for choice.	Displays only the empty rooms. If there are no empty rooms. "No empty rooms, All Rooms are occupied!!!" is displayed.	Displays only the empty rooms. If there are no empty rooms. "No empty rooms, All Rooms are occupied!!!" is displayed.	Pass
(Delete customer is opened correctly) Enter “D” or “d” for choice.	Program opens delete customer method & Displays ‘Enter room number’	Program opens add customer method & Displays ‘Enter room number’	Pass
(Delete Customer “Bhavan”) Enter room number: 1	“Customer removed" is displayed. Displays ‘empty’ for Firstname, Surname, CreditCard & 0 for Guests	“Customer removed" is displayed. Displays ‘empty’ for Firstname, Surname, CreditCard & 0 for Guests	Pass
(Delete Customer “Bhavan” with invalid input) Enter “19” or “$@#” or “ ” or “-6” for room number	“Room number is Invalid!!!" is displayed. Prompts for choice again.	“Room number is Invalid!!!" is displayed. Prompts for choice again.	Pass

(Delete Customer from "empty" room) Enter "5" for room number (empty room)	"Room is already empty" is displayed	"Room is already empty" is displayed	Pass
(Load Customer from file) Enter "L" or "l" for choice.	"Hotel data Loaded Successfully!" is displayed. If the file does not exist or if any other error happens "Could not load the data!!!" is displayed. Press 'V', Displays room detail in correct order	"Hotel data Loaded Successfully!" is displayed. ("With no errors") Press 'V', Displays room detail in correct order	Pass
(Store Customers to a file) Enter "S" or "s" for choice.	"Hotel data Stored Successfully!" is displayed. if any other error happens "Could not store the data!!!" is displayed The details are stored in correct on the text file	"Hotel data Stored Successfully!" is displayed. ("With no errors") The details are stored in correct on the text file	Pass
(Find customer is opened correctly) Enter "F" or "f" for choice.	Program opens find customer method & Displays 'Enter Customer First name'	Program opens find customer method & Displays 'Enter Customer First name'	Pass
(Find Customer room number by giving the name) Enter "F" or "f" for choice. Enter name "James" or "james"	Room number 4 is displayed.	Room number 4 is displayed.	Pass
(Find a non-existing Customer room number by giving the name) Enter "F" or "f" for choice. Enter name "kumar".	"Could not find Customer!!!" is displayed.	"Could not find Customer!!!" is displayed.	Pass
(View customers in alphabetical order) Enter "O" or "o" for choice.	Customer names are displayed in alphabetical order.	Customer names are displayed in alphabetical order.	Pass

	Press 'V' the customer room numbers & names are not changed.	Press 'V' the customer room numbers & names are not changed.	
(Exit from the program) Enter "X" or "x" for choice.	"Task Ended!" is displayed & program ends.	"Task Ended!" is displayed & program ends.	Pass

Task 3 (Array Version)

Test Case	Expected Result	Actual Result	Pass/Fail
(Rooms Initialized correctly) Run the program.	"All Rooms are initialized!" is displayed.	"All Rooms are initialized!" is displayed.	Pass
(Main menu displayed correctly) Run the program	Main menu is displayed in the correct order (A, V, E, D, F, S, L, O, X)	Main menu is displayed in the correct order (A, V, E, D, F, S, L, O, X)	Pass
(Enter your preference works correctly for valid input) Enter "V" or "v"	Displays 'empty' for Firstname, Surname, CreditCard & 0 for Guests	Displays 'empty' for Firstname, Surname, CreditCard & 0 for Guests	Pass
(Enter your preference works correctly for invalid input) Enter "1" or "$@#" or " "	'Invalid Input! Try Again!' is displayed & prompts for choice.	'Invalid Input! Try Again!' is displayed & prompts for choice.	Pass
(Add customer is opened correctly) Enter "A" or "a" for choice.	Program opens add customer method & Displays 'Enter room number (1 – 8)'	Program opens add customer method & Displays 'Enter room number (1 – 8)'	Pass
(Add customer "Bhavan" to room 1) Enter room number: 1. first name: "Bhavan". Sur name: "Loganathan". Credit card: 12345678. Guests in room: 4.	Displays "Customer Added!" Press 'V' Displays all the details for room 1correctly.	Displays "Customer Added!" Press 'V' Displays all the details for room 1correctly.	Pass
(Add customer with invalid input) Enter "19" or "$@#" or " " or "-6" for room number	"Room number is Invalid!!!" is displayed. Prompts for choice again.	"Room number is Invalid!!!" is displayed. Prompts for choice again.	Pass
(Add customer with invalid input) Enter "0" for guests in room.	"Guests in room is invalid Customer is not added" is displayed. Prompts for choice again.	"Guests in room is invalid Customer is not added" is displayed. Prompts for choice again.	Pass

(Add customer with invalid input) Enter “19” or “$@#” or “ ” or “-6” for room number or guests in room	"Room number / Guests in Room is Invalid!!!" is displayed. Prompts for choice again.	"Room number / Guests in Room is Invalid!!!" is displayed. Prompts for choice again.	Pass
(Add customer with invalid input) Enter “-1” or “0” for guests in room.	"Guests in room is invalid Customer is not added" is displayed. Prompts for choice again.	"Guests in room is invalid Customer is not added" is displayed. Prompts for choice again.	Pass
(Add customer to an occupied room) Enter “1” for room number (occupied room)	“Room is already occupied! Try Again!!!" is displayed. Prompts for choice again.	“Room is already occupied! Try Again!!!" is displayed. Prompts for choice again.	Pass
(View all rooms) Enter “V” or “v” for choice.	Displays room detail in correct order	Displays room detail in correct order	Pass
(View Empty rooms) Enter “E” or “e” for choice.	Displays only the empty rooms. If there are no empty rooms. "No empty rooms, All Rooms are occupied!!!" is displayed.	Displays only the empty rooms. If there are no empty rooms. "No empty rooms, All Rooms are occupied!!!" is displayed.	Pass
(Delete customer is opened correctly) Enter “D” or “d” for choice.	Program opens delete customer method & Displays ‘Enter room number’	Program opens add customer method & Displays ‘Enter room number’	Pass
(Delete Customer “Bhavan”) Enter room number: 1	“Customer removed" is displayed. Displays ‘empty’ for Firstname, Surname, CreditCard & 0 for Guests	“Customer removed" is displayed. Displays ‘empty’ for Firstname, Surname, CreditCard & 0 for Guests	Pass
(Delete Customer “Bhavan” with invalid input) Enter “19” or “$@#” or “ ” or “-6” for room number	“Room number is Invalid!!!" is displayed. Prompts for choice again.	“Room number is Invalid!!!" is displayed. Prompts for choice again.	Pass

(Delete Customer from "empty" room) Enter "5" for room number (empty room)	"Room is already empty" is displayed	"Room is already empty" is displayed	Pass
(Load Customer from file) Enter "L" or "l" for choice.	"Hotel data Loaded Successfully!" is displayed. If the file does not exist or if any other error happens "Could not load the data!!!" is displayed. Press 'V', Displays room detail in correct order	"Hotel data Loaded Successfully!" is displayed. ("With no errors") Press 'V', Displays room detail in correct order	Pass
(Store Customers to a file) Enter "S" or "s" for choice.	"Hotel data Stored Successfully!" is displayed. if any other error happens "Could not store the data!!!" is displayed The details are stored in correct on the text file	"Hotel data Stored Successfully!" is displayed. ("With no errors") The details are stored in correct on the text file	Pass
(Find customer is opened correctly) Enter "F" or "f" for choice.	Program opens find customer method & Displays 'Enter Customer First name'	Program opens find customer method & Displays 'Enter Customer First name'	Pass
(Find Customer room number by giving the name) Enter "F" or "f" for choice. Enter name "James" or "james"	Room number 4 is displayed.	Room number 4 is displayed.	Pass
(Find a non-existing Customer room number by giving the name) Enter "F" or "f" for choice. Enter name "kumar".	"Could not find Customer!!!" is displayed.	"Could not find Customer!!!" is displayed.	Pass
(View customers in alphabetical order) Enter "O" or "o" for choice.	Customer names are displayed in alphabetical order.	Customer names are displayed in alphabetical order.	Pass

	Press 'V' the customer room numbers & names are not changed.	Press 'V' the customer room numbers & names are not changed.	
(Exit from the program) Enter "X" or "x" for choice.	"Task Ended!" is displayed & program ends.	"Task Ended!" is displayed & program ends.	Pass

Task 4

Test Case	Expected Result	Actual Result	Pass/Fail
(Rooms Initialized correctly) Run the program.	"All Rooms are initialized!" is displayed.	"All Rooms are initialized!" is displayed.	Pass
(Main menu displayed correctly) Run the program	Main menu is displayed in the correct order (A, V, E, D, F, S, L, O, X)	Main menu is displayed in the correct order (A, V, E, D, F, S, L, O, X)	Pass
(Enter your preference works correctly for valid input) Enter "V" or "v"	Displays 'empty' for Firstname, Surname, CreditCard & 0 for Guests	Displays 'empty' for Firstname, Surname, CreditCard & 0 for Guests	Pass
(Enter your preference works correctly for invalid input) Enter "1" or "$@#" or " "	'Invalid Input! Try Again!' is displayed & prompts for choice.	'Invalid Input! Try Again!' is displayed & prompts for choice.	Pass
(Add customer is opened correctly) Enter "A" or "a" for choice.	Program opens add customer method & Displays 'Enter room number (1 – 8)'	Program opens add customer method & Displays 'Enter room number (1 – 8)'	Pass
(Add customer "Bhavan" to room 1) Enter room number: 1. first name: "Bhavan". Sur name: "Loganathan". Credit card: 12345678. Guests in room: 4.	Displays "Customer Added!" Press 'V' Displays all the details for room 1correctly.	Displays "Customer Added!" Press 'V' Displays all the details for room 1correctly.	Pass
(Add customer with invalid input) Enter "19" or "$@#" or " " or "-6" for room number	"Room number is Invalid!!!" is displayed. Prompts for choice again.	"Room number is Invalid!!!" is displayed. Prompts for choice again.	Pass
(Add customer with invalid input) Enter "0" for guests in room.	"Guests in room is invalid Customer is not added" is displayed. Prompts for choice again.	"Guests in room is invalid Customer is not added" is displayed. Prompts for choice again.	Pass

(Add customer with invalid input) Enter “19” or “$@#” or “ ” or “-6” for room number or guests in room	"Room number / Guests in Room is Invalid!!!" is displayed. Prompts for choice again.	"Room number / Guests in Room is Invalid!!!" is displayed. Prompts for choice again.	Pass
(Add customer with invalid input) Enter “-1” or “0” for guests in room.	"Guests in room is invalid Customer is not added" is displayed. Prompts for choice again.	"Guests in room is invalid Customer is not added" is displayed. Prompts for choice again.	Pass
(Add customer to an occupied room) Enter “1” for room number (occupied room)	“Room is already occupied! Try Again!!!" is displayed. Prompts for choice again.	“Room is already occupied! Try Again!!!" is displayed. Prompts for choice again.	Pass
(View all rooms) Enter “V” or “v” for choice.	Displays room detail in correct order	Displays room detail in correct order	Pass
(View Empty rooms) Enter “E” or “e” for choice.	Displays only the empty rooms. If there are no empty rooms. "No empty rooms, All Rooms are occupied!!!" is displayed.	Displays only the empty rooms. If there are no empty rooms. "No empty rooms, All Rooms are occupied!!!" is displayed.	Pass
(Delete customer is opened correctly) Enter “D” or “d” for choice.	Program opens delete customer method & Displays ‘Enter room number’	Program opens add customer method & Displays ‘Enter room number’	Pass
(Delete Customer “Bhavan”) Enter room number: 1	“Customer removed" is displayed. Displays ‘empty’ for Firstname, Surname, CreditCard & 0 for Guests	“Customer removed" is displayed. Displays ‘empty’ for Firstname, Surname, CreditCard & 0 for Guests	Pass
(Delete Customer “Bhavan” with invalid input) Enter “19” or “$@#” or “ ” or “-6” for room number	“Room number is Invalid!!!" is displayed. Prompts for choice again.	“Room number is Invalid!!!" is displayed. Prompts for choice again.	Pass

(Delete Customer from "empty" room) Enter "5" for room number (empty room)	"Room is already empty" is displayed	"Room is already empty" is displayed	Pass
(Load Customer from file) Enter "L" or "l" for choice.	"Hotel data Loaded Successfully!" is displayed. If the file does not exist or if any other error happens "Could not load the data!!!" is displayed. Press 'V', Displays room detail in correct order	"Hotel data Loaded Successfully!" is displayed. ("With no errors") Press 'V', Displays room detail in correct order	Pass
(Store Customers to a file) Enter "S" or "s" for choice.	"Hotel data Stored Successfully!" is displayed. if any other error happens "Could not store the data!!!" is displayed The details are stored in correct on the text file	"Hotel data Stored Successfully!" is displayed. ("With no errors") The details are stored in correct on the text file	Pass
(Find customer is opened correctly) Enter "F" or "f" for choice.	Program opens find customer method & Displays 'Enter Customer First name'	Program opens find customer method & Displays 'Enter Customer First name'	Pass
(Find Customer room number by giving the name) Enter "F" or "f" for choice. Enter name "James" or "james"	Room number 4 is displayed.	Room number 4 is displayed.	Pass
(Find a non-existing Customer room number by giving the name) Enter "F" or "f" for choice. Enter name "kumar".	"Could not find Customer!!!" is displayed.	"Could not find Customer!!!" is displayed.	Pass
(View customers in alphabetical order) Enter "O" or "o" for choice.	Customer names are displayed in alphabetical order.	Customer names are displayed in alphabetical order.	Pass

	Press 'V' the customer room numbers & names are not changed.	Press 'V' the customer room numbers & names are not changed.	
(Waiting list works correctly) Enter "A" or "a" for choice. (when all rooms are occupied)	"Room is already Occupied! You will be Added to Waiting List!" is displayed. Prompts for user inputs.	"Room is already Occupied! You will be Added to Waiting List!" is displayed. Prompts for user inputs.	Pass
(Waiting list works correctly) first name: "John". Sur name: "Wick". Credit card: 12345678. Guests in room: 2.	"Customer Added!" is displayed	"Customer Added!" is displayed	Pass
(Waiting list works correctly) Enter "D" or "d" for choice. Enter room number : 5	"Customer Removed!" & "A Customer from the Waiting List is Added to this Room!" is displayed Press 'V', Displays room detail in correct order with room 5 filled with waiting list customer.	"Customer Removed!" & "A Customer from the Waiting List is Added to this Room!" is displayed Press 'V', Displays room detail in correct order with room 5 filled with waiting list customer.	Pass
(Exit from the program) Enter "X" or "x" for choice.	"Task Ended!" is displayed & program ends.	"Task Ended!" is displayed & program ends.	Pass

Test Case (Description)

I have made separate test case table for each task. I have added test cases for test both the happy and negative path of every scenario in each task. I have arranged the order of the test in way we can easily test all aspects of the program. As per the coursework specifications we must fulfil 8 Main functionalities additionally I have added a Main menu.

1. Main Menu
   The program displays a main menu so the use can select their preference I have added a test to verify this function works correctly for both uppercase and lowercase letter & also another test case to check the invalid inputs.

2. Add customer to room.
   User can go to the add customer option by giving “A” or “a” as input and program asks for the room number, I have added test cases to verify the user inputs and I also verified the entered details.

3. Views All rooms.
   User can view all rooms by giving “V” or “v” as input. I have added test cases to verify the user input and I also verified the displayed details.

4. Display Empty rooms.
   User can view empty rooms by giving “E” or “e” as input. I have added test cases to verify the user input and I also verified the displayed details.

5. Delete customer from room.
   User can go to the delete customer option by giving “D” or “d” as input and program asks for the room number, I have added test cases to verify the user inputs and I also verified the entered details.

6. Find room from customer name.
   User can go to the find customer option by giving “F” or “f” as input and program asks for the customer’s name, I have added test cases to verify the user inputs and I also verified the displayed details.

7. Store program data into file.
   User storeroom details by giving “S” or “s” as input. I have added test cases to verify the user input and I also verified the displayed details on text file.

8. Load program data from file.
   User can load room details by giving “L” or “l” as I have added test cases to verify the user input and I also verified the loaded details.

9. View guests Ordered alphabetically by name.
   User can view customer names in alphabetical order by giving “O” or “o” as I have added test cases to verify the user input and I also verified the displayed details.

   Additionally, I have added test cases according to the functionality of each task. For example, I have added some test cases to verify queue functionality in Task 4.

Code Snippets

Task 1

Hotel.java

```
package CW.Task1;

import java.io.File;
import java.util.Formatter;
import java.util.Scanner;

/**
 * The Hotel program implements an application that
 * does the following operations
 * add/view/delete/find/store/load/viewByOrder
 *
 * @author bhavan
 * @version Task 1 (Array Version)
 */
public class Hotel {
    // static Scanner object to get user input
    static Scanner input = new Scanner(System.in);

    public static void main(String[] args) {
        // String Array hotel to store customer names for 8 rooms
        String[] hotel = new String[8];

        // initializing each room to the default value "empty"
        initialise(hotel);

        // Program Menu
        String menu = (
                "+ - - - - - - - - - - - - - - - - - - - - - - - - - - - +\n" +
                "|                HOTEL  PROGRAM                         |\n" +
                "+ - - - - - - - - - - - - - - - - - - - - - - - - - - - +\n" +
                "|   A.ADD CUSTOMER TO ROOM                              |\n" +
                "|   V.VIEW ALL ROOMS                                    |\n" +
                "|   E.DISPLAY EMPTY ROOMS                               |\n" +
                "|   D.DELETE CUSTOMER FROM ROOM                         |\n" +
                "|   F.FIND ROOM FROM CUSTOMER NAME                      |\n" +
                "|   S.STORE PROGRAM DATA INTO FILE                      |\n" +
                "|   L.LOAD PROGRAM DATA FROM FILE                       |\n" +
                "|   O.VIEW GUESTS IN ALPHABETICAL ORDER                 |\n" +
                "|   X.EXIT PROGRAM                                      |\n" +
                "+ - - - - - - - - - - - - - - - - - - - - - - - - - - - +"
        );
        System.out.println(menu);

        /* Using do-while loop because the user needs
         * to input at least once.
         * using switch case to decide which method to call.
         * loop runs until a user inputs "X".
         * storing the user preference in choice variable. */
        String choice;
```

```
        do {
            System.out.print("\n*** Enter Your Preference -> ");
            // getting the user input & making it uppercase
            choice = input.next().toUpperCase();
            switch (choice) {
                case "A" -> add(hotel);
                case "V" -> view(hotel);
                case "E" -> viewEmpty(hotel);
                case "D" -> delete(hotel);
                case "F" -> find(hotel);
                case "S" -> store(hotel);
                case "L" -> load(hotel);
                case "O" -> viewByOrder(hotel);
                case "X" -> System.out.println("Task Ended!\n");
                default -> System.out.println("Invalid Input! Try Again!");
            }
        } while (!choice.equals("X"));
    }

    /**
     * initialise method to set all rooms in hotel array to "empty"
     * @param hotel - array which holds the room data
     */
    private static void initialise(String[] hotel) {
        for (int i = 0; i < hotel.length; i++) {
            hotel[i] = "empty";
        }
        System.out.println("All Rooms are initialized!\n");
    }

    /**
     * add method which adds a customer to the room
     * @param hotel - array which holds the room data
     */
    public static void add(String[] hotel) {
        try {
            // getting room number from user and storing it to the variable
            System.out.print("Enter Room number(1 - 8) -> ");
            int roomNum = input.nextInt();

            /* Array indexes start from 0 but room numbers start from 1,
             * so we are subtracting 1 from the given room number,
             * this concept is used throughout the program.*/
            int n = roomNum - 1;

            /* if the room is empty we are adding the customer
             * else we are displaying "Room is already occupied!" */
            if (hotel[n].equals("empty")) {
                System.out.print("Enter Customer name for Room number "
                                 + roomNum + " -> ");
                String customerName = input.next();
                hotel[n] = customerName;
                System.out.println("Customer Added!");
            } else {
                System.out.println("Room is already occupied! Try Again!!!");
            }
        } catch (Exception exception) {
            /* If the user enters a invalid value for roomNum
             * we are displaying a error message */
            System.out.println("Room number is Invalid!!!");
        }
    }
```

```java
    /**
     * view method which is to view all room details
     * @param hotel - array which holds the room data
     */
    public static void view(String[] hotel) {
        /* looping through the array to get every room
         * if the room is empty we are displaying "...is empty"
         * if the room is not empty we are displaying "...is occupied by"
         * we are using (i+1) because array index starts from 0 */
        for (int i = 0; i < hotel.length; i++) {
            if (hotel[i].equals("empty")) {
                System.out.println("Room number " + (i + 1) + " is Empty");
            } else {
                System.out.println("Room number " + (i + 1) +
                                   " is Occupied by " + hotel[i]);
            }
        }
    }

    /**
     * viewEmpty method which view only the empty rooms
     * @param hotel - array which holds the room data
     */
    public static void viewEmpty(String[] hotel) {
        // looping through the array to find the room number
        int emptyRooms = 0;
        for (int i = 0; i < hotel.length; i++) {
            if (hotel[i].equals("empty")) {
                System.out.println("Room number " + (i + 1) + " is empty");
                emptyRooms++;
            }
        }
        // if we could not find any empty room we are displaying a message
        if (emptyRooms == 0) {
            System.out.println("No empty rooms, All Rooms are occupied!!!");
        }
    }
    /**
     * delete method which deletes a customer in the room
     * @param hotel - array which holds the room data
     */
    public static void delete(String[] hotel) {
        try {
            System.out.print("Enter Room number to Remove the Customer -> ");
            int roomNum = input.nextInt();
            int n = roomNum - 1;

            /* checking if the room is already empty
             * else we are making the room "empty" */
            if (hotel[n].equals("empty")) {
                System.out.println("Room is already empty");
            } else {
                hotel[n] = "empty";
                System.out.println("Customer Removed!");
            }

        } catch (Exception exception) {
            /* If the user enters a invalid value for roomNum
             * we are displaying a error message */
            System.out.println("Room number is Invalid!!!");
        }
    }
```

```
    /**
     * find method which finds a customer in the room
     * @param hotel - array which holds the room data
     */
    public static void find(String[] hotel) {
        System.out.print("Enter customer name -> ");
        String customer = input.next();
        int customerCount = 0;
        /* looping through the array & checking whether it is
         * equal to the entered customer name  */
        for (int i = 0; i < hotel.length; i++) {
            if (hotel[i].equalsIgnoreCase(customer)) {
                System.out.println("Room number is " + (i + 1));
                customerCount++;
            }
        }
        // if we could not find the customer we are displaying a message
        if (customerCount == 0) {
            System.out.println("Could not find Customer!!!");
        }
    }

    /**
     * store method to store the current customer data to a text file
     * @param hotel - array which holds the room data
     */
    public static void store(String[] hotel) {
        try {
            // Using Formatter to write data to the text file
            Formatter formatter = new Formatter("src/store1.txt");
            // looping through the array to store all room details
            for (int i = 0; i < hotel.length; i++) {
                formatter.format("Room number %d is Occupied by %s\n", (i +
1), hotel[i]);
            }
            formatter.close();
            System.out.println("Hotel data Stored Successfully!");
        } catch (Exception e) {
            System.out.println("Could not store the data!!!");
        }
    }

    /**
     * load method which loads the customer names to the program
     * @param hotel - array which holds the room data
     */
    public static void load(String[] hotel) {
        try {
            String path = "src/load1.txt"; // file path
            // Scanner to read the data
            Scanner file = new Scanner(new File(path));
            // looping until we reach the end of the file
            while (file.hasNext()) {
                int n = file.nextInt();
                String customerName = file.next();
                hotel[n] = customerName;
            }
            file.close();
            System.out.println("Hotel data Loaded Successfully!");
        } catch (Exception e) {
            System.out.println("Could not load the data!!!");
        }
    }
```

```
    /**
     * viewByOrder to print customer names in alphabetical order
     * @param hotel - array which holds the room data
     */
    public static void viewByOrder(String[] hotel) {
        /* We are copying the content to another array because
         * if we use the same array it will change the array indexes */
        String[] sortedHotel = new String[hotel.length];
        for (int i = 0; i < hotel.length; i++) {
            sortedHotel[i] = hotel[i];
        }

        // Using bubble sort algorithm to sort the customer names
        System.out.println("-----Guests in Alphabetical order-----");

        for (int j = 0; j < sortedHotel.length; j++) {
            for (int i = j + 1; i < sortedHotel.length; i++) {
                if (sortedHotel[i].compareTo(sortedHotel[j]) < 0) {
                    String temp = sortedHotel[j];
                    sortedHotel[j] = sortedHotel[i];
                    sortedHotel[i] = temp;
                }
            }
            // printing all rooms except empty rooms
            if (!sortedHotel[j].equals("empty")) {
                System.out.println(sortedHotel[j]);
            }
        }
    }
}
```

Task 2

Hotel.java

```
package CW.Task2;

import java.io.File;
import java.util.Formatter;
import java.util.Scanner;

/**
 * The Hotel program implements an application that
 * does the following operations (with Room)
 * add/view/delete/find/store/load/viewByOrder
 *
 * @author bhavan
 * @version Task 2 (Class Version)
 */
public class Hotel {
    // static Scanner object to get user input
    static Scanner input = new Scanner(System.in);

    public static void main(String[] args) {
        // Room Array hotel to store customer names for 8 rooms
        Room[] rooms = new Room[8];

        // initializing each room to the default value "empty"
        initialise(rooms);

        // Program Menu
        String menu = (
                "+ - - - - - - - - - - - - - - - - - - - - - - - - - - - +\n" +
                "|                  HOTEL  PROGRAM                       |\n" +
                "+ - - - - - - - - - - - - - - - - - - - - - - - - - - - +\n" +
                "|   A.ADD CUSTOMER TO ROOM                              |\n" +
                "|   V.VIEW ALL ROOMS                                    |\n" +
                "|   E.DISPLAY EMPTY ROOMS                               |\n" +
                "|   D.DELETE CUSTOMER FROM ROOM                         |\n" +
                "|   F.FIND ROOM FROM CUSTOMER NAME                      |\n" +
                "|   S.STORE PROGRAM DATA INTO FILE                      |\n" +
                "|   L.LOAD PROGRAM DATA FROM FILE                       |\n" +
                "|   O.VIEW GUESTS IN ALPHABETICAL ORDER                 |\n" +
                "|   X.EXIT PROGRAM                                      |\n" +
                "+ - - - - - - - - - - - - - - - - - - - - - - - - - - - +"
        );
        System.out.println(menu);

        /* Using do-while loop because the user needs to input at least once.
         * using switch case to decide which method to call.
         * loop runs until a user inputs "X".
         * storing the user preference in choice variable. */
        String choice;
```

```
        do {
            System.out.print("\n*** Enter Your Preference -> ");
            // getting the user input & making it uppercase
            choice = input.next().toUpperCase();
            switch (choice) {
                case "A" -> add(rooms);
                case "V" -> view(rooms);
                case "E" -> viewEmpty(rooms);
                case "D" -> delete(rooms);
                case "F" -> find(rooms);
                case "S" -> store(rooms);
                case "L" -> load(rooms);
                case "O" -> viewByOrder(rooms);
                case "X" -> System.out.println("Task Ended!\n");
                default -> System.out.println("Invalid Input! Try Again!");
            }
        } while(!choice.equals("X"));
    }

    /**
     * initialise method to set all rooms in hotel array to "empty"
     * @param hotel - array which holds the room data
     */
    private static void initialise(Room[] hotel) {
        for (int i = 0; i < hotel.length; i++) {
            hotel[i] = new Room("empty");
        }
        System.out.println("All Rooms are initialized!\n");
    }
    /**
     * add method which adds a customer to the room
     * @param hotel - array which holds the room data
     */
    public static void add(Room[] hotel) {
        try{
            // getting room number from user and storing it to the variable
            System.out.print("Enter Room number(1 - 8) -> ");
            int roomNum = input.nextInt();

            /* Array indexes start from 0 but room numbers start from 1,
             * so we are subtracting 1 from the given room number,
             * this concept is used throughout the program.*/
            int n = roomNum - 1;

            /* if the room is empty we are adding the customer
             * else we are displaying "Room is already occupied!" */
            if(hotel[n].getCustomerName().equals("empty")){
                System.out.print("Enter Customer name for Room number "
                                 + roomNum + " -> ");
                String customerName = input.next();
                hotel[n].setCustomerName(customerName);
                System.out.println("Customer Added!");
            } else {
                System.out.println("Room is already occupied! Try Again!!!");
            }
        } catch (Exception exception){
            /* If the user enters a invalid value for roomNum
             * we are displaying a error message */
            System.out.println("Room number is Invalid!!!");
        }
    }
```

```
    /**
     * view method which is to view all room details
     * @param hotel - array which holds the room data
     */
    public static void view(Room[] hotel) {
        /* looping through the array to get every room
         * if the room is empty we are displaying "...is empty"
         * if the room is not empty we are displaying "...is occupied by"
         * we are using (i+1) because array index starts from 0 */
        for (int i = 0; i < hotel.length; i++) {
            if(hotel[i].getCustomerName().equals("empty")){
                System.out.println("Room number " + (i + 1) + " is empty");
            } else {
                System.out.println("Room number " + (i + 1) + " is occupied
by " + hotel[i].getCustomerName());
            }
        }
    }
    /**
     * viewEmpty method which view only the empty rooms
     * @param hotel - array which holds the room data
     */
    public static void viewEmpty(Room[] hotel) {
        // looping through the array to find the room number
        int emptyRooms = 0;
        for (int i = 0; i < hotel.length; i++) {
            if (hotel[i].getCustomerName().equals("empty")) {
                System.out.println("Room number " + (i + 1) + " is empty");
                emptyRooms++;
            }
        }
        // if we could not find any empty room we are displaying a message
        if(emptyRooms == 0){
            System.out.println("No empty rooms, All Rooms are occupied!!!");
        }
    }
    /**
     * delete method which deletes a customer in the room
     * @param hotel - array which holds the room data
     */
    public static void delete(Room[] hotel) {
        try{
            System.out.print("Enter Room number to Remove the Customer -> ");
            int roomNum = input.nextInt();
            int n = roomNum - 1;

            /* checking if the room is already empty
             * else we are making the room "empty" */
            if (hotel[n].getCustomerName().equals("empty")) {
                System.out.println("Room is already empty");
            } else {
                hotel[n].setCustomerName("empty");
                System.out.println("Customer Removed!");
            }

        } catch (Exception exception){
            /* If the user enters a invalid value for roomNum
             * we are displaying a error message */
            System.out.println("Room number is Invalid!!!");
        }
    }
```

```java
    /**
     * find method which finds a customer in the room
     * @param hotel - array which holds the room data
     */
    public static void find(Room[] hotel) {
        System.out.print("Enter customer name -> ");
        String customer = input.next();
        int customerCount = 0;
        /* looping through the array & checking whether it is
         * equal to the entered customer name  */
        for (int i = 0; i < hotel.length; i++) {
            if(hotel[i].getCustomerName().equalsIgnoreCase(customer)){
                System.out.println("Room number is " + (i + 1));
                customerCount++;
            }
        }
        // if we could not find the customer we are displaying a message
        if(customerCount == 0){
            System.out.println("Could not find Customer!!!");
        }
    }

    /**
     * store method to store the current customer data to a text file
     * @param hotel - array which holds the room data
     */
    public static void store(Room[] hotel) {
        try {
            // Using Formatter to write data to the text file
            Formatter formatter = new Formatter("src/store1.txt");
            // looping through the array to store all room details
            for (int i = 0; i < hotel.length; i++) {
                formatter.format("Room number %d is Occupied by %s\n",
                                 (i + 1), hotel[i].getCustomerName());
            }
            formatter.close();
            System.out.println("Hotel data Stored Successfully!");
        } catch (Exception e){
            System.out.println("Could not store the data!!!");
        }
    }
    /**
     * load method which loads the customer names to the program
     * @param hotel - array which holds the room data
     */
    public static void load(Room[] hotel) {
        try{
            String path = "src/load1.txt"; // file path
            // Scanner to read the data
            Scanner file = new Scanner(new File(path));
            // looping until we reach the end of the file
            while (file.hasNext()){
                int n = file.nextInt();
                String customerName = file.next();
                hotel[n].setCustomerName(customerName);
            }
            file.close();
            System.out.println("Hotel data Loaded Successfully!");
        } catch (Exception e){
            System.out.println("Could not load the data!!!");
        }
    }
```

```java
    /**
     * viewByOrder to print customer names in alphabetical order
     * @param hotel - array which holds the room data
     */
    public static void viewByOrder(Room[] hotel) {
        /* We are copying the content to a string array because
         * if we use the same array it will change the array indexes */
        String[] sortedHotel = new String[hotel.length];
        for (int i = 0; i < hotel.length; i++) {
            sortedHotel[i] = hotel[i].getCustomerName();
        }

        // Using bubble sort algorithm to sort the customer names
        System.out.println("-----Guests in Alphabetical order-----");

        for (int j = 0; j < sortedHotel.length; j++) {
            for (int i = j + 1; i < sortedHotel.length; i++) {
                if (sortedHotel[i].compareTo(sortedHotel[j]) < 0) {
                    String temp = sortedHotel[j];
                    sortedHotel[j] = sortedHotel[i];
                    sortedHotel[i] = temp;
                }
            }
            // printing all rooms except empty rooms
            if (!sortedHotel[j].equals("empty")) {
                System.out.println(sortedHotel[j]);
            }
        }
    }
}
```

Room.java

```java
package CW.Task2;
/**
 * The Room class to store room data
 *
 * @author bhavan
 * @version Task 2 (Class Version)
 */
public class Room {
    private String customerName;

    public Room(String customerName) {
        this.customerName = customerName;
    }

    public String getCustomerName() {
        return customerName;
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }
}
```

Task 3 (Class)

Hotel.java

```
package CW.Task3.Class;

import java.io.File;
import java.util.Formatter;
import java.util.Scanner;

/**
 * The Hotel program implements an application that
 * does the following operations (with Room & Person)
 * add/view/delete/find/store/load/viewByOrder
 *
 * @author bhavan
 * @version Task 3 (Class Version)
 */
public class Hotel {
    // static Scanner object to get user input
    static Scanner input = new Scanner(System.in);

    public static void main(String[] args) {
        // Room Array hotel to store customer names for 8 rooms
        Room[] rooms = new Room[8];

        // initializing each room to it's default value "empty"
        initialise(rooms);

        // Program Menu
        String menu = (
                "+ - - - - - - - - - - - - - - - - - - - - - - - - - - - +\n" +
                "|                HOTEL  PROGRAM                         |\n" +
                "+ - - - - - - - - - - - - - - - - - - - - - - - - - - - +\n" +
                "|   A.ADD CUSTOMER TO ROOM                              |\n" +
                "|   V.VIEW ALL ROOMS                                    |\n" +
                "|   E.DISPLAY EMPTY ROOMS                               |\n" +
                "|   D.DELETE CUSTOMER FROM ROOM                         |\n" +
                "|   F.FIND ROOM FROM CUSTOMER NAME                      |\n" +
                "|   S.STORE PROGRAM DATA INTO FILE                      |\n" +
                "|   L.LOAD PROGRAM DATA FROM FILE                       |\n" +
                "|   O.VIEW GUESTS IN ALPHABETICAL ORDER                 |\n" +
                "|   X.EXIT PROGRAM                                      |\n" +
                "+ - - - - - - - - - - - - - - - - - - - - - - - - - - - +"
        );
        System.out.println(menu);

        /* Using do-while loop because the user needs to input at least once.
         * using switch case to decide which method to call.
         * loop runs until a user inputs "X".
         * storing the user preference in choice variable. */
        String choice;
```

```
        do {
            System.out.print("\n*** Enter Your Preference -> ");
            // getting the user input & making it uppercase
            choice = input.next().toUpperCase();
            switch (choice) {
                case "A" -> add(rooms);
                case "V" -> view(rooms);
                case "E" -> viewEmpty(rooms);
                case "D" -> delete(rooms);
                case "F" -> find(rooms);
                case "S" -> store(rooms);
                case "L" -> load(rooms);
                case "O" -> viewByOrder(rooms);
                case "X" -> System.out.println("Task Ended!\n");
                default -> System.out.println("Invalid Input! Try Again!");
            }
        } while(!choice.equals("X"));
    }

    /**
     * initialise method to initialise all rooms
     * @param rooms - array which holds the room data
     */
    private static void initialise(Room[] rooms) {
        for (int i = 0; i < rooms.length; i++) {
            rooms[i] = new Room();
        }
        System.out.println("All Rooms are initialized!\n");
    }

    /**
     * add method which adds a customer to the room
     * @param rooms - array which holds the room data
     */
    public static void add(Room[] rooms) {
        try{
            // getting room number from user and storing it to the variable
            System.out.print("Enter Room number(1 - 8) -> ");
            int roomNum = input.nextInt();

            /* Array indexes start from 0 but room numbers start from 1,
             * so we are subtracting 1 from the given room number,
             * this concept is used throughout the program.*/
            int n = roomNum - 1;

            /* if the room is empty we are adding the customer
             * else we are displaying "Room is already occupied!" */
            if(rooms[n].getGuestsInRoom() == 0){
                addCustomer(rooms[n]);
            } else {
                System.out.println("Room is already occupied! Try Again!!!");
            }
        } catch (Exception exception){
            /* If the user enters a invalid value for roomNum
             * we are displaying a error message */
            System.out.println("Room number / Guests in Room is Invalid!!!");
        }
    }
```

```java
    /**
     * view method which is to view all room details
     * @param rooms - array which holds the room data
     */
    public static void view(Room[] rooms) {
        /* looping through the array to get every room
         * then displaying the details like a table */
        System.out.println("Firstname\tSurname\t\tCreditCard\tGuests");
        for (Room room : rooms) {
            System.out.println(
                room.getPayingGuest().getFirstName() + "\t\t" +
                room.getPayingGuest().getSurName() + "\t\t" +
                room.getPayingGuest().getCreditCardNumber()+ "\t\t" +
                room.getGuestsInRoom()
            );
        }
    }
    /**
     * viewEmpty method which view only the empty rooms
     * @param rooms - array which holds the room data
     */
    public static void viewEmpty(Room[] rooms) {
        // looping through the array to find the room number
        int emptyRooms = 0;
        for (int i = 0; i < rooms.length; i++) {
            if (rooms[i].getGuestsInRoom() == 0)  {
                System.out.println("Room number " + (i + 1) + " is empty");
                emptyRooms++;
            }
        }
        // if we could not find any empty room we are displaying a message
        if(emptyRooms == 0){
            System.out.println("No empty rooms, All Rooms are occupied!!!");
        }
    }
    /**
     * delete method which deletes a customer in the room
     * @param rooms - array which holds the room data
     */
    public static void delete(Room[] rooms) {
        try{
            System.out.print("Enter Room number to Remove the Customer -> ");
            int roomNum = input.nextInt();
            int n = roomNum - 1;

            /* checking if the room is already empty
             * else we are making the room "empty" */
            if (rooms[n].getGuestsInRoom() == 0) {
                System.out.println("Room is already empty");
            } else {
                setRoomData(rooms[n],0,"empty","empty","empty");
                System.out.println("Customer Removed!");
            }
        } catch (Exception exception){
            /* If the user enters a invalid value for roomNum
             * we are displaying a error message */
            System.out.println("Room number is Invalid!!!");
        }
    }
```

```
    /**
     * find method which finds a customer in the room
     * @param rooms - array which holds the room data
     */
    public static void find(Room[] rooms) {
        System.out.print("Enter Customer First name -> ");
        String customer = input.next();
        int customerCount = 0;
        /* looping through the array & checking whether it is
         * equal to the entered customer name  */
        for (int i = 0; i < rooms.length; i++) {

if(rooms[i].getPayingGuest().getFirstName().equalsIgnoreCase(customer)){
                System.out.println("Room number is " + (i + 1));
                customerCount++;
            }
        }
        // if we could not find the customer we are displaying a message
        if(customerCount == 0){
            System.out.println("Could not find Customer!!!");
        }
    }

    /**
     * store method to store the current customer data to a text file
     * @param rooms - array which holds the room data
     */
    public static void store(Room[] rooms) {
        try {
            // Using Formatter to write data to the text file
            Formatter formatter = new Formatter("src/store2.txt");
            // looping through the array to store all room details
            for (int i = 0; i < rooms.length; i++) {
                formatter.format("Room number %d is Occupied by %s %s [%s]
with %d guests\n",
                        (i + 1),
                        rooms[i].getPayingGuest().getFirstName(),
                        rooms[i].getPayingGuest().getSurName(),
                        rooms[i].getPayingGuest().getCreditCardNumber(),
                        rooms[i].getGuestsInRoom()
                );
            }
            formatter.close();
            System.out.println("Hotel data Stored Successfully!");
        } catch (Exception e){
            System.out.println("Could not store the data!!!");
        }
    }
```

```
    /**
     * load method which loads the customer names to the program
     * @param rooms - array which holds the room data
     */
    public static void load(Room[] rooms) {
        try{
            String path = "src/load2.txt"; // file path
            // Scanner to read the data
            Scanner file = new Scanner(new File(path));
            // looping until we reach the end of the file
            while (file.hasNext()){
                int n = file.nextInt();
                String firstName = file.next();
                String surName  = file.next();
                String creditCardNumber = file.next();
                int guestsInRoom = file.nextInt();


setRoomData(rooms[n],guestsInRoom,firstName,surName,creditCardNumber);
            }
            file.close();
            System.out.println("Hotel data Loaded Successfully!");
        } catch (Exception e){
            System.out.println("Could not load the data!!!");
        }
    }

    /**
     * viewByOrder to print customer names in alphabetical order
     * @param rooms - array which holds the room data
     */
    public static void viewByOrder(Room[] rooms) {
        /* We are copying the content to a string array because
         * if we use the same array it will change the array indexes */
        String[] sortedCustomers = new String[rooms.length];
        for (int i = 0; i < rooms.length; i++) {
            sortedCustomers[i] = rooms[i].getPayingGuest().getFirstName() + "
" + rooms[i].getPayingGuest().getSurName();
        }

        // Using bubble sort algorithm to sort the customer names
        System.out.println("-----Guests in Alphabetical order-----");

        for (int j = 0; j < sortedCustomers.length; j++) {
            for (int i = j + 1; i < sortedCustomers.length; i++) {
                if (sortedCustomers[i].compareTo(sortedCustomers[j]) < 0) {
                    String temp = sortedCustomers[j];
                    sortedCustomers[j] = sortedCustomers[i];
                    sortedCustomers[i] = temp;
                }
            }
            // printing all rooms except empty rooms
            if (!sortedCustomers[j].equals("empty empty")) {
                System.out.println(sortedCustomers[j]);
            }
        }
    }
```

```
    /**
     * addCustomer to prompt & get user input
     * @param room - array which holds the room data
     */
    public static void addCustomer(Room room) {
        // getting all data from user using prompt
        System.out.print("Enter First name of the Paying Guest -> ");
        String firstName = input.next();

        System.out.print("Enter Surname of the Paying Guest -> ");
        String surName = input.next();

        System.out.print("Enter Credit Card number -> ");
        String creditCardNumber = input.next();

        System.out.print("Enter Guests In Room ( > 0) -> ");
        int guestsInRoom = input.nextInt();

        // checking if the Guests In Room > 0
        if (guestsInRoom > 0) {

setRoomData(room,guestsInRoom,firstName,surName,creditCardNumber);
            System.out.println("Customer Added!");
        } else {
            System.out.println("Guests in room is invalid!!!\nCustomer is not
added!");
        }
    }

    /**
     * setRoomData to set the room
     * @param room - array which holds the room data
     */
    public static void setRoomData(Room room, int guestsInRoom,
                                   String firstName, String surName, String
creditCardNumber) {

        room.setGuestsInRoom(guestsInRoom);
        Person person = new Person(firstName,surName,creditCardNumber);
        room.setPayingGuest(person);
    }
}
```

Room.java

```
package CW.Task3.Class;

/**
 * The Room class to store room data
 *
 * @author bhavan
 * @version Task 3 (Class Version)
 */
public class Room {
    int guestsInRoom;
    Person payingGuest;

    // Room constructor to initialize room data
    public Room() {
        this.guestsInRoom = 0;
        this.payingGuest = new Person("empty","empty","empty");
    }

    public int getGuestsInRoom() {
        return guestsInRoom;
    }

    public void setGuestsInRoom(int guestsInRoom) {
        this.guestsInRoom = guestsInRoom;
    }

    public Person getPayingGuest() {
        return payingGuest;
    }

    public void setPayingGuest(Person payingGuest) {
        this.payingGuest = payingGuest;
    }
}
```

Person.java

```java
package CW.Task3.Class;

/**
 * The Person class to store person data
 *
 * @author bhavan
 * @version Task 3 (Class Version)
 */
public class Person {
    private String firstName;
    private String surName;
    private String creditCardNumber;

    public Person(String firstName, String surName, String creditCardNumber)
{
        this.firstName = firstName;
        this.surName = surName;
        this.creditCardNumber = creditCardNumber;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getSurName() {
        return surName;
    }

    public void setSurName(String surName) {
        this.surName = surName;
    }

    public String getCreditCardNumber() {
        return creditCardNumber;
    }

    public void setCreditCardNumber(String creditCardNumber) {
        this.creditCardNumber = creditCardNumber;
    }
}
```

Task 3 (Array)

Hotel.java

```
package CW.Task3.Array;

import java.io.File;
import java.util.Formatter;
import java.util.Scanner;

/**
 * The Hotel program implements an application that
 * does the following operations
 * add/view/delete/find/store/load/viewByOrder
 *
 * @author bhavan
 * @version Task 3 (Array Version)
 */
public class Hotel {
    // static Scanner object to get user input
    static Scanner input = new Scanner(System.in);

    public static void main(String[] args) {
        // int Array guestsInRoom to store guests in rooms
        int[] guestsInRoom = new int[8];
        /* String 2D Array hotel to store
         * firstname, surname, creditCardNumber for 8 rooms */
        String[][] person = new String[8][3];

        // initializing each room to the default value "empty"
        initialise(guestsInRoom, person);

        // Program Menu
        String menu = (
                "+ - - - - - - - - - - - - - - - - - - - - - - - - - +\n" +
                "|                HOTEL  PROGRAM                     |\n" +
                "+ - - - - - - - - - - - - - - - - - - - - - - - - - +\n" +
                "|   A.ADD CUSTOMER TO ROOM                          |\n" +
                "|   V.VIEW ALL ROOMS                                |\n" +
                "|   E.DISPLAY EMPTY ROOMS                           |\n" +
                "|   D.DELETE CUSTOMER FROM ROOM                     |\n" +
                "|   F.FIND ROOM FROM CUSTOMER NAME                  |\n" +
                "|   S.STORE PROGRAM DATA INTO FILE                  |\n" +
                "|   L.LOAD PROGRAM DATA FROM FILE                   |\n" +
                "|   O.VIEW GUESTS IN ALPHABETICAL ORDER             |\n" +
                "|   X.EXIT PROGRAM                                  |\n" +
                "+ - - - - - - - - - - - - - - - - - - - - - - - - - +"
        );
        System.out.println(menu);

        /* Using do-while loop because the user needs to input at least once.
         * using switch case to decide which method to call.
         * loop runs until a user inputs "X".
         * storing the user preference in choice variable. */
        String choice;
```

```
        do {
            System.out.print("\n*** Enter Your Preference -> ");
            // getting the user input & making it uppercase
            choice = input.next().toUpperCase();
            switch (choice) {
                case "A" -> add(guestsInRoom, person);
                case "V" -> view(guestsInRoom, person);
                case "E" -> viewEmpty(guestsInRoom);
                case "D" -> delete(guestsInRoom, person);
                case "F" -> find(person);
                case "S" -> store(guestsInRoom, person);
                case "L" -> load(guestsInRoom, person);
                case "O" -> viewByOrder(person);
                case "X" -> System.out.println("Task Ended!\n");
                default -> System.out.println("Invalid Input! Try Again!\n");
            }
        } while(!choice.equals("X"));
    }

    /**
     * initialise method to set all rooms in hotel array to "empty"
     * @param guests - array which holds the guests in room
     * @param person - 2D array to hold firstname, surname, creditCardNumber
     */
    private static void initialise(int[] guests, String[][] person) {
        for (int i = 0; i < person.length; i++) {
            guests[i] = 0;
            for (int j = 0; j < person[i].length; j++) {
                person[i][j] = "empty";
            }
        }
        System.out.println("All Rooms are initialized!\n");
    }

    /**
     * add method which adds a customer to the room
     * @param guests - array which holds the guests in room
     * @param person - 2D array to hold firstname, surname, creditCardNumber
     */
    public static void add(int[] guests, String[][] person) {
        try{
            // getting room number from user and storing it to the variable
            System.out.print("Enter Room number(1 - 8) -> ");
            int roomNum = input.nextInt();

            /* Array indexes start from 0 but room numbers start from 1,
             * so we are subtracting 1 from the given room number,
             * this concept is used throughout the program.*/
            int n = roomNum - 1;
```

```
/* if the room is empty we are adding the customer
             * else we are displaying "Room is already occupied!" */
            if(person[n][0].equals("empty") || guests[n] == 0){
                // getting all data from user using prompt
                System.out.print("Enter First name of the Paying Guest -> ");
                String firstName = input.next();

                System.out.print("Enter Surname of the Paying Guest -> ");
                String surName = input.next();

                System.out.print("Enter Credit Card number -> ");
                String creditCardNumber = input.next();

                System.out.print("Enter Guests In Room ( > 0) -> ");
                int guestsInRoom = input.nextInt();

                // checking if the Guests In Room > 0
                if (guestsInRoom > 0){
                    guests[n] = guestsInRoom;
                    person[n][0] = firstName;
                    person[n][1] = surName;
                    person[n][2] = creditCardNumber;
                    System.out.println("Customer Added!");
                } else {
                    System.out.println("Guests in room is
invalid!!!\nCustomer is not added!");
                }
            } else {
                System.out.println("Room is already occupied! Try Again!!!");
            }
        } catch (Exception exception){
            /* If the user enters a invalid value for roomNum
             * we are displaying a error message */
            System.out.println("Room number / Guests in Room is Invalid!!!");
        }
    }

   /**
     * view method which is to view all room details
     * @param guests - array which holds the guests in room
     * @param person - 2D array to hold firstname, surname, creditCardNumber
     */
    public static void view(int[] guests, String[][] person) {
        /* looping through the array to get every room
         * then displaying the details like a table */
        System.out.println("Firstname\tSurname\t\tCreditCard\tGuests");
        for (int i = 0; i < guests.length; i++) {
            System.out.println(
                person[i][0]  + "\t\t" +
                person[i][1]  + "\t\t" +
                person[i][2]  + "\t\t" +
                guests[i]
            );
        }
    }
```

```java
/**
 * viewEmpty method which view only the empty rooms
 * @param guests - array which holds the guests in room
 */
public static void viewEmpty(int[] guests) {
    // looping through the array to find the room number
    int emptyRooms = 0;
    for (int i = 0; i < guests.length; i++) {
        if (guests[i] == 0) {
            System.out.println("Room number " + (i + 1) + " is empty");
            emptyRooms++;
        }
    }
    // if we could not find any empty room we are displaying a message
    if(emptyRooms == 0){
        System.out.println("No empty rooms, All Rooms are occupied!!!");
    }
}

/**
 * delete method which deletes a customer in the room
 * @param guests - array which holds the guests in room
 * @param person - 2D array to hold firstname, surname, creditCardNumber
 */
public static void delete(int[] guests, String[][] person){
    try{
        System.out.print("Enter Room number to Remove the Customer -> ");
        int roomNum = input.nextInt();
        int n = roomNum - 1;

        /* checking if the room is already empty
         * else we are making the room "empty" */
        if (guests[n] == 0) {
            System.out.println("Room is already empty");
        } else {
            guests[n] = 0;
            person[n][0] = "empty";
            person[n][1] = "empty";
            person[n][2] = "empty";
            System.out.println("Customer Removed!");
        }
    } catch (Exception exception){
        /* If the user enters a invalid value for roomNum
         * we are displaying a error message */
        System.out.println("Room number is Invalid!!!");
    }
}
```

```
    /**
     * find method which finds a customer in the room
     * @param person - 2D array to hold firstname, surname, creditCardNumber
     */
    public static void find(String[][] person) {
        System.out.print("Enter Customer First Name -> ");
        String customer = input.next();
        int customerCount = 0;
        /* looping through the array & checking whether it is
         * equal to the entered customer name  */
        for (int i = 0; i < person.length; i++) {
            if(person[i][0].equalsIgnoreCase(customer)){
                System.out.println("Room number is " + (i + 1));
                customerCount++;
            }
        }
        // if we could not find the customer we are displaying a message
        if(customerCount == 0){
            System.out.println("Could not find Customer!!!");
        }
    }

    /**
     * store method to store the current customer data to a text file
     * @param guests - array which holds the guests in room
     * @param person - 2D array to hold firstname, surname, creditCardNumber
     */
    public static void store(int[] guests, String[][] person){
        try {
            // Using Formatter to write data to the text file
            Formatter formatter = new Formatter("src/store2.txt");
            // looping through the array to store all room details
            for (int i = 0; i < guests.length; i++) {
                formatter.format("Room number %d is Occupied by %s %s [%s]
with %d guests\n",
                        (i + 1),
                        person[i][0],
                        person[i][1],
                        person[i][2],
                        guests[i]);
            }
            formatter.close();
            System.out.println("Hotel data Stored Successfully!");
        } catch (Exception e){
            System.out.println("Could not store the data!!!");
        }
    }
```

```java
    /**
     * load method which loads the customer names to the program
     * @param guests - array which holds the guests in room
     * @param person - 2D array to hold firstname, surname, creditCardNumber
     */
    public static void load(int[] guests, String[][] person) {
        try{
            String path = "src/load2.txt"; // file path
            // Scanner to read the data
            Scanner file = new Scanner(new File(path));
            // looping until we reach the end of the file
            while (file.hasNext()){
                int n = file.nextInt();
                String firstName = file.next();
                String surName  = file.next();
                String creditCardNumber = file.next();
                int guestsInRoom = file.nextInt();

                person[n][0] = firstName;
                person[n][1] = surName;
                person[n][2] = creditCardNumber;
                guests[n] = guestsInRoom;
            }
            file.close();
            System.out.println("Hotel data Loaded Successfully!");
        } catch (Exception e){
            System.out.println("Could not load the data!!!");
        }
    }
    /**
     * viewByOrder to print customer names in alphabetical order
     * @param person - 2D array to hold firstname, surname, creditCardNumber
     */
    public static void viewByOrder(String[][] person) {
        /* We are copying the content to another array because
         * if we use the same array it will change the array indexes */
        String[] sortedHotel = new String[person.length];
        for (int i = 0; i < person.length; i++) {
            sortedHotel[i] = person[i][0] + " " + person[i][1];
        }

        // Using bubble sort algorithm to sort the customer names
        System.out.println("-----Guests in Alphabetical order-----");

        for (int j = 0; j < sortedHotel.length; j++) {
            for (int i = j + 1; i < sortedHotel.length; i++) {
                if (sortedHotel[i].compareTo(sortedHotel[j]) < 0) {
                    String temp = sortedHotel[j];
                    sortedHotel[j] = sortedHotel[i];
                    sortedHotel[i] = temp;
                }
            }
            // printing all rooms except empty rooms
            if (!sortedHotel[j].equals("empty empty")){
                System.out.println(sortedHotel[j]);
            }
        }
    }
}
```

Task 4

Hotel.java

```
package CW.Task4;

import java.io.File;
import java.util.Formatter;
import java.util.Scanner;

/**
 * The Hotel program implements an application that
 * does the following operations (with Room & Person & HotelQueue)
 * add/view/delete/find/store/load/viewByOrder
 *
 * @author bhavan
 * @version Task 4 (Class Version)
 */
public class Hotel {
    // static Scanner object to get user input
    static Scanner input = new Scanner(System.in);
    // static HotelQueue object to store waiting list
    static HotelQueue waitingList = new HotelQueue();

    public static void main(String[] args) {
        // Room Array hotel to store customer names for 8 rooms
        Room[] rooms = new Room[8];

        // initializing each room to it's default value "empty"
        initialise(rooms);

        // Program Menu
        String menu = (
                "+ - - - - - - - - - - - - - - - - - - - - - - - - - - - +\n" +
                "|                HOTEL  PROGRAM                         |\n" +
                "+ - - - - - - - - - - - - - - - - - - - - - - - - - - - +\n" +
                "|   A.ADD CUSTOMER TO ROOM                              |\n" +
                "|   V.VIEW ALL ROOMS                                    |\n" +
                "|   E.DISPLAY EMPTY ROOMS                               |\n" +
                "|   D.DELETE CUSTOMER FROM ROOM                         |\n" +
                "|   F.FIND ROOM FROM CUSTOMER NAME                      |\n" +
                "|   S.STORE PROGRAM DATA INTO FILE                      |\n" +
                "|   L.LOAD PROGRAM DATA FROM FILE                       |\n" +
                "|   O.VIEW GUESTS IN ALPHABETICAL ORDER                 |\n" +
                "|   X.EXIT PROGRAM                                      |\n" +
                "+ - - - - - - - - - - - - - - - - - - - - - - - - - - - +"
        );
        System.out.println(menu);

        /* Using do-while loop because the user needs to input at least once.
         * using switch case to decide which method to call.
         * loop runs until a user inputs "X".
         * storing the user preference in choice variable. */
        String choice;
```

```
        do {
            System.out.print("\n*** Enter Your Preference -> ");
            // getting the user input & making it uppercase
            choice = input.next().toUpperCase();
            switch (choice) {
                case "A" -> add(rooms);
                case "V" -> view(rooms);
                case "E" -> viewEmpty(rooms);
                case "D" -> delete(rooms);
                case "F" -> find(rooms);
                case "S" -> store(rooms);
                case "L" -> load(rooms);
                case "O" -> viewByOrder(rooms);
                case "X" -> System.out.println("Task Ended!\n");
                default -> System.out.println("Invalid Input! Try Again!");
            }
        } while(!choice.equals("X"));
    }

    /**
     * initialise method to initialise all rooms
     * @param rooms - array which holds the room data
     */
    private static void initialise(Room[] rooms) {
        for (int i = 0; i < rooms.length; i++) {
            rooms[i] = new Room();
        }
        System.out.println("All Rooms are initialized!\n");
    }

    /**
     * isFull method to check the rooms are full
     * @param rooms - array which holds the room data
     * @return if room is full true else false
     */
    public static boolean isFull(Room[] rooms){
        for (Room room : rooms) {
            if (room.getGuestsInRoom() == 0) {
                return false;
            }
        }
        return true;
    }
```

```
    /**
     * add method which adds a customer to the room
     * @param rooms - array which holds the room data
     */
    public static void add(Room[] rooms) {
        try{
            // getting room number from user and storing it to the variable
            System.out.print("Enter Room number(1 - 8) -> ");
            int roomNum = input.nextInt();

            /* Array indexes start from 0 but room numbers start from 1,
             * so we are subtracting 1 from the given room number,
             * this concept is used throughout the program.*/
            int n = roomNum - 1;

            // if the room is empty we are adding the customer
            if(rooms[n].getGuestsInRoom() == 0){
                addCustomer(rooms[n]);
            } else {
                /* if rooms are full we are adding the customer
                 * to the waiting list, else we are prompting the user
                 * to enter another room */
                if (isFull(rooms)){
                    System.out.println("Room is already Occupied! You will be
Added to Waiting List!\n");
                    Room room = new Room();
                    addCustomer(room);
                    waitingList.enQueue(room);
                } else {
                    System.out.println("Room is already occupied! Try another
Room!");
                }
            }
        } catch (Exception exception){
            /* If the user enters a invalid value for roomNum
             * we are displaying a error message */
            System.out.println("Room number / Guests in Room is Invalid!!!");
        }
    }

    /**
     * view method which is to view all room details
     * @param rooms - array which holds the room data
     */
    public static void view(Room[] rooms) {
        /* looping through the array to get every room
         * then displaying the details like a table */
        System.out.println("Firstname\tSurname\t\tCreditCard\tGuests");
        for (Room room : rooms) {
            System.out.println(
                room.getPayingGuest().getFirstName() + "\t\t" +
                room.getPayingGuest().getSurName() + "\t\t" +
                room.getPayingGuest().getCreditCardNumber()+ "\t\t" +
                room.getGuestsInRoom()
            );
        }
    }
```

```
    /**
     * viewEmpty method which view only the empty rooms
     * @param rooms - array which holds the room data
     */
    public static void viewEmpty(Room[] rooms) {
        // looping through the array to find the room number
        int emptyRooms = 0;
        for (int i = 0; i < rooms.length; i++) {
            if (rooms[i].getGuestsInRoom() == 0)  {
                System.out.println("Room number " + (i + 1) + " is empty");
                emptyRooms++;
            }
        }
        // if we could not find any empty room we are displaying a message
        if(emptyRooms == 0){
            System.out.println("No empty rooms, All Rooms are occupied!!!");
        }
    }
    /**
     * delete method which deletes a customer in the room
     * @param rooms - array which holds the room data
     */
    public static void delete(Room[] rooms) {
        try{
            System.out.print("Enter Room number to Remove the Customer -> ");
            int roomNum = input.nextInt();
            int n = roomNum - 1;

            /* checking if the room is already empty
             * else we are making the room "empty" */
            if (rooms[n].getGuestsInRoom() == 0) {
                System.out.println("Room is already empty");
            } else {
                // resetting all room data
                setRoomData(rooms[n],0,"empty","empty","empty");
                System.out.println("Customer Removed!");

                /* if the waiting list is not empty we are
                 * setting the deleted room with waiting list customer */
                if(!waitingList.isEmpty()){
                    Room waitingCustomer = waitingList.deQueue();

                    setRoomData(rooms[n],
                            waitingCustomer.getGuestsInRoom(),
                            waitingCustomer.getPayingGuest().getFirstName(),
                            waitingCustomer.getPayingGuest().getSurName(),
                           waitingCustomer.getPayingGuest().
                           getCreditCardNumber());

                    System.out.println("A Customer from the Waiting List is
Added to this Room!");
                }
            }
        } catch (Exception exception){
            /* If the user enters a invalid value for roomNum
             * we are displaying a error message */
            System.out.println("Room number is Invalid!!!");
        }
    }
```

```
    /**
     * find method which finds a customer in the room
     * @param rooms - array which holds the room data
     */
    public static void find(Room[] rooms) {
        System.out.print("Enter Customer first name -> ");
        String customer = input.next();
        int customerCount = 0;
        for (int i = 0; i < rooms.length; i++) {

if(rooms[i].getPayingGuest().getFirstName().equalsIgnoreCase(customer)){
                System.out.println("Room number is " + (i + 1));
                customerCount++;
            }
        }

        if(customerCount == 0){
            System.out.println("Could not find Customer!!!");
        }
    }

    /**
     * store method to store the current customer data to a text file
     * @param rooms - array which holds the room data
     */
    public static void store(Room[] rooms) {
        try {
            // Using Formatter to write data to the text file
            Formatter formatter = new Formatter("src/store2.txt");
            // looping through the array to store all room details
            for (int i = 0; i < rooms.length; i++) {
                formatter.format("Room number %d is Occupied by %s %s [%s]
with %d guests\n",
                        (i + 1),
                        rooms[i].getPayingGuest().getFirstName(),
                        rooms[i].getPayingGuest().getSurName(),
                        rooms[i].getPayingGuest().getCreditCardNumber(),
                        rooms[i].getGuestsInRoom()
                );
            }
            formatter.close();
            System.out.println("Hotel data Stored Successfully!");
        } catch (Exception e){
            System.out.println("Could not store the data!!!");
        }
    }
```

```
    /**
     * load method which loads the customer names to the program
     * @param rooms - array which holds the room data
     */
    public static void load(Room[] rooms) {
        try{
            String path = "src/load2.txt"; // file path
            // Scanner to read the data
            Scanner file = new Scanner(new File(path));
            // looping until we reach the end of the file
            while (file.hasNext()){
                int n = file.nextInt();
                String firstName = file.next();
                String surName  = file.next();
                String creditCardNumber = file.next();
                int guestsInRoom = file.nextInt();


setRoomData(rooms[n],guestsInRoom,firstName,surName,creditCardNumber);
            }
            file.close();
            System.out.println("Hotel data Loaded Successfully!");
        } catch (Exception e){
            System.out.println("Could not load the data!!!");
        }
    }

    public static void viewByOrder(Room[] rooms) {
        /* We are copying the content to a string array because
         * if we use the same array it will change the array indexes */
        String[] sortedCustomers = new String[rooms.length];
        for (int i = 0; i < rooms.length; i++) {
            sortedCustomers[i] = rooms[i].getPayingGuest().getFirstName() + "
" + rooms[i].getPayingGuest().getSurName();
        }

        // Using bubble sort algorithm to sort the customer names
        System.out.println("-----Guests in Alphabetical order-----");

        for (int j = 0; j < sortedCustomers.length; j++) {
            for (int i = j + 1; i < sortedCustomers.length; i++) {
                if (sortedCustomers[i].compareTo(sortedCustomers[j]) < 0) {
                    String temp = sortedCustomers[j];
                    sortedCustomers[j] = sortedCustomers[i];
                    sortedCustomers[i] = temp;
                }
            }
            // printing all rooms except empty rooms
            if (!sortedCustomers[j].equals("empty empty")) {
                System.out.println(sortedCustomers[j]);
            }
        }
    }
```

```
    /**
     * addCustomer to prompt & get user input
     * @param room - array which holds the room data
     */
    public static void addCustomer(Room room) {
        // getting all data from user using prompt
        System.out.print("Enter First name of the Paying Guest -> ");
        String firstName = input.next();

        System.out.print("Enter Surname of the Paying Guest -> ");
        String surName = input.next();

        System.out.print("Enter Credit Card number -> ");
        String creditCardNumber = input.next();

        System.out.print("Enter Guests In Room ( > 0) -> ");
        int guestsInRoom = input.nextInt();

        // checking if the Guests In Room > 0
        if (guestsInRoom > 0) {

setRoomData(room,guestsInRoom,firstName,surName,creditCardNumber);
            System.out.println("Customer Added!");
        } else {
            System.out.println("Guests in room is invalid!!!\nCustomer is not
added!");
        }
    }

    /**
     * setRoomData to set the room
     * @param room - array which holds the room data
     */
    public static void setRoomData(Room room, int guestsInRoom,
                                   String firstName, String surName,
                                    String creditCardNumber) {

        room.setGuestsInRoom(guestsInRoom);
        Person person = new Person(firstName,surName,creditCardNumber);
        room.setPayingGuest(person);
    }
}
```

HotelQueue.java

```java
package CW.Task4;

/**
 * The Hotel program implements an application that
 * does the following operations (with Room & Person & HotelQueue)
 * add/view/delete/find/store/load/viewByOrder
 *
 * @author bhavan
 * @version Task 4 (Class Version)
 */
public class HotelQueue {
    int size;
    int rear;
    int front;
    Room[] circularQueueArray;

    /* constructor to set circular array size
     * front & rear are element po*/
    public HotelQueue() {
        this.size = 10;
        this.circularQueueArray = new Room[size];
        this.rear = -1;
        this.front = -1;
    }

    // add item to queue
    public void enQueue(Room item) {
        if (rear == front && front == -1) {
            front = 0;
        }
        rear = (rear + 1) % size;
        circularQueueArray[rear] = item;
    }

    // remove item from queue and return the removed item
    public Room deQueue() {
        Room room = circularQueueArray[front];
        circularQueueArray[front] = new Room();//reset data
        if (rear == front) {
            rear = -1;
            front = -1;
        } else {
            front = (front + 1) % size;
        }
        return room;
    }

    // check if the queue is empty
    public boolean isEmpty() {
        return (rear == front && rear == -1);
    }
}
```

Person.java

```java
package CW.Task4;

/**
 * The Person class to store person data
 *
 * @author bhavan
 * @version Task 4
 */
public class Person {
    private String firstName;
    private String surName;
    private String creditCardNumber;

    public Person(String firstName, String surName, String creditCardNumber)
{
        this.firstName = firstName;
        this.surName = surName;
        this.creditCardNumber = creditCardNumber;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getSurName() {
        return surName;
    }

    public void setSurName(String surName) {
        this.surName = surName;
    }

    public String getCreditCardNumber() {
        return creditCardNumber;
    }

    public void setCreditCardNumber(String creditCardNumber) {
        this.creditCardNumber = creditCardNumber;
    }
}
```

Room.java

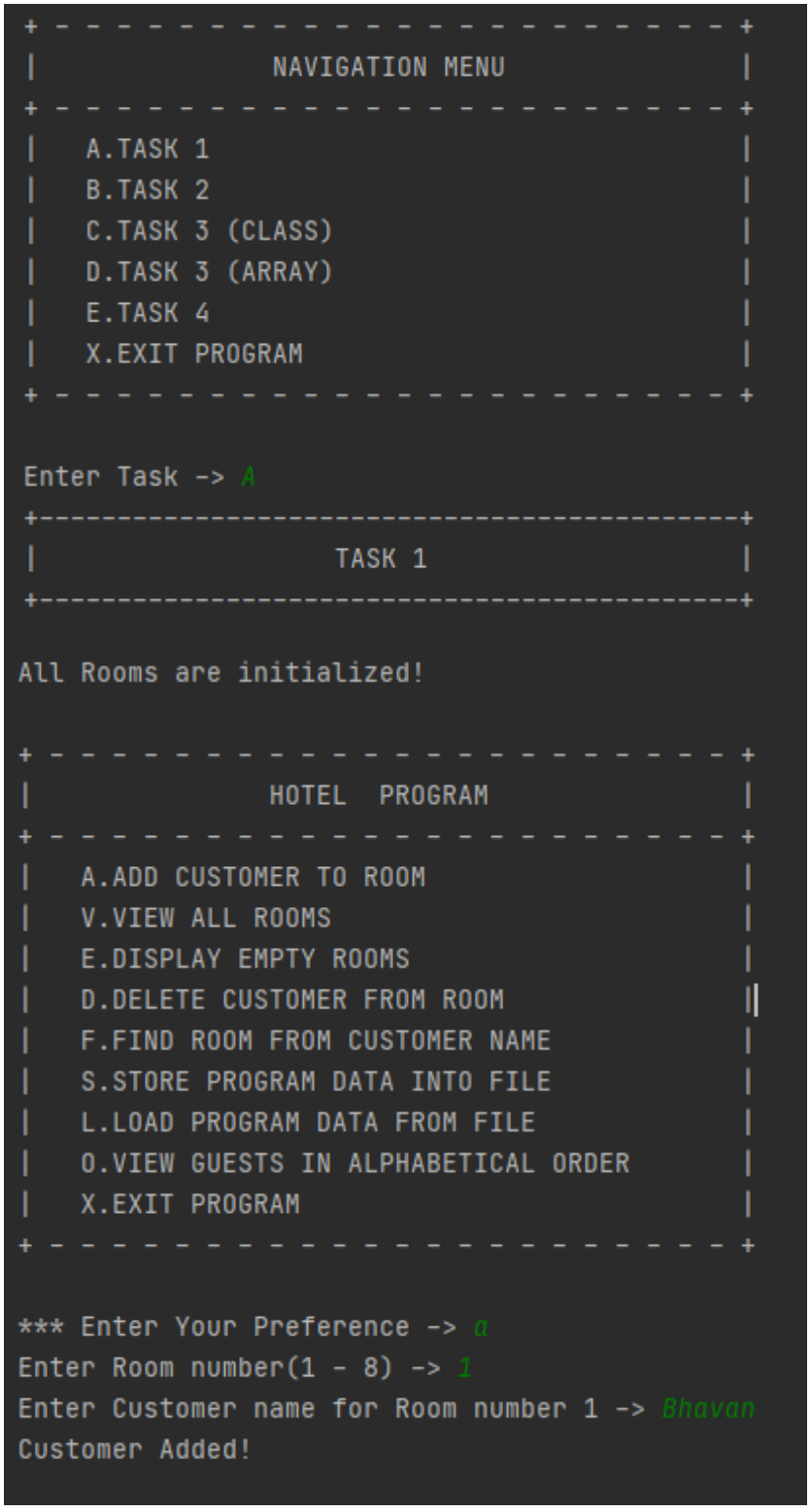
```java
package CW.Task4;

/**
 * The Room class to store room data
 *
 * @author bhavan
 * @version Task 4
 */
public class Room {
    int guestsInRoom;
    Person payingGuest;

    // Room constructor to initialize room data
    public Room() {
        this.guestsInRoom = 0;
        this.payingGuest = new Person("empty","empty","empty");
    }

    public int getGuestsInRoom() {
        return guestsInRoom;
    }

    public void setGuestsInRoom(int guestsInRoom) {
        this.guestsInRoom = guestsInRoom;
    }

    public Person getPayingGuest() {
        return payingGuest;
    }

    public void setPayingGuest(Person payingGuest) {
        this.payingGuest = payingGuest;
    }
}
```

Main.java (navigation menu for tasks).

```
package CW;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        String menu = (
            "+ - - - - - - - - - - - - - - - - - - - - - - - - - - - +\n" +
            "|                NAVIGATION MENU                          |\n" +
            "+ - - - - - - - - - - - - - - - - - - - - - - - - - - - +\n" +
            "|   A.TASK 1                                              |\n" +
            "|   B.TASK 2                                              |\n" +
            "|   C.TASK 3 (CLASS)                                      |\n" +
            "|   D.TASK 3 (ARRAY)                                      |\n" +
            "|   E.TASK 4                                              |\n" +
            "|   X.EXIT PROGRAM                                        |\n" +
            "+ - - - - - - - - - - - - - - - - - - - - - - - - - - - +"
        );
        System.out.println(menu);

        String t1 = (
            "+-----------------------------------------------+\n" +
            "|                    TASK 1                     |\n" +
            "+-----------------------------------------------+\n"
        );
        String t2 = (
            "+-----------------------------------------------+\n" +
            "|                    TASK 2                     |\n" +
            "+-----------------------------------------------+\n"
        );
        String t3C = (
            "+-----------------------------------------------+\n" +
            "|                TASK 3 (CLASS)                 |\n" +
            "+-----------------------------------------------+\n"
        );
        String t3A = (
            "+-----------------------------------------------+\n" +
            "|                TASK 3 (ARRAY)                 |\n" +
            "+-----------------------------------------------+\n"
        );
        String t4 = (
            "+-----------------------------------------------+\n" +
            "|                    TASK 4                     |\n" +
            "+-----------------------------------------------+\n"
        );
```

```
        Scanner input = new Scanner(System.in);
          String choice;
          do {
              System.out.print("\nEnter Task -> ");
              choice = input.next().toUpperCase();
              switch (choice) {
                  case "A" -> {
                      System.out.println(t1);
                      CW.Task1.Hotel.main(args);
                      System.out.println("---------------------------------");
                  }
                  case "B" -> {
                      System.out.println(t2);
                      CW.Task2.Hotel.main(args);
                      System.out.println("---------------------------------");
                  }
                  case "C" -> {
                      System.out.println(t3C);
                      CW.Task3.Class.Hotel.main(args);
                      System.out.println("---------------------------------");
                  }
                  case "D" -> {
                      System.out.println(t3A);
                      CW.Task3.Array.Hotel.main(args);
                      System.out.println("---------------------------------");
                  }
                  case "E" -> {
                      System.out.println(t4);
                      CW.Task4.Hotel.main(args);
                      System.out.println("---------------------------------");
                  }
                  case "X" -> System.out.println("Program Ended! \nThank you!");
                  default -> System.out.println("Invalid Input! Try Again!");
              }
          } while(!choice.equals("X"));
      }
  }
```

Program Output (Screenshots)



```
*** Enter Your Preference -> v
Room number 1 is Occupied by Bhavan
Room number 2 is Empty
Room number 3 is Empty
Room number 4 is Empty
Room number 5 is Empty
Room number 6 is Empty
Room number 7 is Empty
Room number 8 is Empty

*** Enter Your Preference -> %32
Invalid Input! Try Again!

*** Enter Your Preference -> A
Enter Room number(1 - 8) -> 55
Room number is Invalid!!!

*** Enter Your Preference -> A
Enter Room number(1 - 8) -> 1
Room is already occupied! Try Again!!!

*** Enter Your Preference -> E
Room number 2 is empty
Room number 3 is empty
Room number 4 is empty
Room number 5 is empty
Room number 6 is empty
Room number 7 is empty
Room number 8 is empty

*** Enter Your Preference -> A
Enter Room number(1 - 8) -> 2
Enter Customer name for Room number 2 -> Kumar
Customer Added!

*** Enter Your Preference -> d
Enter Room number to Remove the Customer -> 2
Customer Removed!

*** Enter Your Preference -> d
Enter Room number to Remove the Customer -> 2
Room is already empty

*** Enter Your Preference -> d
Enter Room number to Remove the Customer -> 55
Room number is Invalid!!!
```

```
*** Enter Your Preference -> v
Room number 1 is Occupied by Bhavan
Room number 2 is Empty
Room number 3 is Empty
Room number 4 is Empty
Room number 5 is Empty
Room number 6 is Empty
Room number 7 is Empty
Room number 8 is Empty

*** Enter Your Preference -> l
Hotel data Loaded Successfully!

*** Enter Your Preference -> v
Room number 1 is Occupied by Bhavan
Room number 2 is Occupied by Emma
Room number 3 is Occupied by Sophia
Room number 4 is Occupied by James
Room number 5 is Occupied by William
Room number 6 is Occupied by Benjamin
Room number 7 is Occupied by Mason
Room number 8 is Occupied by Isabella

*** Enter Your Preference -> s
Hotel data Stored Successfully!

*** Enter Your Preference -> F
Enter customer name -> james
Room number is 4

*** Enter Your Preference -> F
Enter customer name -> kumar
Could not find Customer!!!

*** Enter Your Preference -> o
-----Guests in Alphabetical order-----
Benjamin
Bhavan
Emma
Isabella
James
Mason
Sophia
William
```

```
*** Enter Your Preference -> v
Room number 1 is Occupied by Bhavan
Room number 2 is Occupied by Emma
Room number 3 is Occupied by Sophia
Room number 4 is Occupied by James
Room number 5 is Occupied by William
Room number 6 is Occupied by Benjamin
Room number 7 is Occupied by Mason
Room number 8 is Occupied by Isabella

*** Enter Your Preference -> x
Task Ended!

------------------------------------------------
```

```
Enter Task -> B
+---------------------------------------------+
|                   TASK 2                    |
+---------------------------------------------+

All Rooms are initialized!

+ - - - - - - - - - - - - - - - - - - - - - - +
|               HOTEL  PROGRAM                |
+ - - - - - - - - - - - - - - - - - - - - - - +
|   A.ADD CUSTOMER TO ROOM                    |
|   V.VIEW ALL ROOMS                          |
|   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 IN ALPHABETICAL ORDER       |
|   X.EXIT PROGRAM                            |
+ - - - - - - - - - - - - - - - - - - - - - - +
```

```
*** Enter Your Preference -> a
Enter Room number(1 - 8) -> 1
Enter Customer name for Room number 1 -> Bhavan
Customer Added!

*** Enter Your Preference -> v
Room number 1 is occupied by Bhavan
Room number 2 is empty
Room number 3 is empty
Room number 4 is empty
Room number 5 is empty
Room number 6 is empty
Room number 7 is empty
Room number 8 is empty

*** Enter Your Preference -> $J
Invalid Input! Try Again!

*** Enter Your Preference -> A
Enter Room number(1 - 8) -> 2
Enter Customer name for Room number 2 -> kumar
Customer Added!
```

```
*** Enter Your Preference -> v
Room number 1 is occupied by Bhavan
Room number 2 is occupied by kumar
Room number 3 is empty
Room number 4 is empty
Room number 5 is empty
Room number 6 is empty
Room number 7 is empty
Room number 8 is empty

*** Enter Your Preference -> d
Enter Room number to Remove the Customer -> 2
Customer Removed!

*** Enter Your Preference -> v
Room number 1 is occupied by Bhavan
Room number 2 is empty
Room number 3 is empty
Room number 4 is empty
Room number 5 is empty
Room number 6 is empty
Room number 7 is empty
Room number 8 is empty

*** Enter Your Preference -> d
Enter Room number to Remove the Customer -> 2
Room is already empty

*** Enter Your Preference -> d
Enter Room number to Remove the Customer -> 55
Room number is Invalid!!!

*** Enter Your Preference -> e
Room number 2 is empty
Room number 3 is empty
Room number 4 is empty
Room number 5 is empty
Room number 6 is empty
Room number 7 is empty
Room number 8 is empty

*** Enter Your Preference -> l
Hotel data Loaded Successfully!
```

```
*** Enter Your Preference -> v
Room number 1 is occupied by Bhavan
Room number 2 is occupied by Emma
Room number 3 is occupied by Sophia
Room number 4 is occupied by James
Room number 5 is occupied by William
Room number 6 is occupied by Benjamin
Room number 7 is occupied by Mason
Room number 8 is occupied by Isabella

*** Enter Your Preference -> s
Hotel data Stored Successfully!

*** Enter Your Preference -> f
Enter customer name -> james
Room number is 4

*** Enter Your Preference -> f
Enter customer name -> kumar
Could not find Customer!!!
*** Enter Your Preference -> o
-----Guests in Alphabetical order-----
Benjamin
Bhavan
Emma
Isabella
James
Mason
Sophia
William

*** Enter Your Preference -> v
Room number 1 is occupied by Bhavan
Room number 2 is occupied by Emma
Room number 3 is occupied by Sophia
Room number 4 is occupied by James
Room number 5 is occupied by William
Room number 6 is occupied by Benjamin
Room number 7 is occupied by Mason
Room number 8 is occupied by Isabella

*** Enter Your Preference -> x
Task Ended!

------------------------------------------------
```

```
Enter Task -> c
+-------------------------------------------+
|               TASK 3 (CLASS)             |
+-------------------------------------------+

All Rooms are initialized!

+ - - - - - - - - - - - - - - - - - - - - - +
|               HOTEL  PROGRAM             |
+ - - - - - - - - - - - - - - - - - - - - - +
|   A.ADD CUSTOMER TO ROOM                  |
|   V.VIEW ALL ROOMS                        |
|   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 IN ALPHABETICAL ORDER     |
|   X.EXIT PROGRAM                          |
+ - - - - - - - - - - - - - - - - - - - - - +

*** Enter Your Preference -> a
Enter Room number(1 - 8) -> 1
Enter First name of the Paying Guest -> Bruce
Enter Surname of the Paying Guest -> Wayne
Enter Credit Card number -> 12345678
Enter Guests In Room ( > 0) -> 4
Customer Added!

*** Enter Your Preference -> v
Firstname   Surname     CreditCard  Guests
Bruce       Wayne       12345678        4
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0

*** Enter Your Preference -> $%
Invalid Input! Try Again!
```

```
*** Enter Your Preference -> a
Enter Room number(1 - 8) -> 2
Enter First name of the Paying Guest -> Hendry
Enter Surname of the Paying Guest -> Clark
Enter Credit Card number -> 65432178
Enter Guests In Room ( > 0) -> 7
Customer Added!

*** Enter Your Preference -> v
Firstname   Surname     CreditCard  Guests
Bruce       Wayne       12345678        4
Hendry      Clark       65432178        7
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0

*** Enter Your Preference -> a
Enter Room number(1 - 8) -> 44
Room number / Guests in Room is Invalid!!!

*** Enter Your Preference -> a
Enter Room number(1 - 8) -> 1
Room is already occupied! Try Again!!!

*** Enter Your Preference -> a
Enter Room number(1 - 8) -> 3
Enter First name of the Paying Guest -> sample
Enter Surname of the Paying Guest -> sample
Enter Credit Card number -> 123448
Enter Guests In Room ( > 0) -> 0
Guests in room is invalid!!!
Customer is not added!

*** Enter Your Preference -> e
Room number 3 is empty
Room number 4 is empty
Room number 5 is empty
Room number 6 is empty
Room number 7 is empty
Room number 8 is empty

*** Enter Your Preference -> d
Enter Room number to Remove the Customer -> 2
Customer Removed!
```

```
*** Enter Your Preference -> v
Firstname   Surname     CreditCard  Guests
Bruce       Wayne       12345678        4
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0

*** Enter Your Preference -> d
Enter Room number to Remove the Customer -> 2
Room is already empty

*** Enter Your Preference -> d
Enter Room number to Remove the Customer -> 54
Room number is Invalid!!!

*** Enter Your Preference -> l
Hotel data Loaded Successfully!
*** Enter Your Preference -> v
Firstname   Surname     CreditCard  Guests
Ruby        Dawson      5535013815      8
Rahim       Andrews     5415319124      1
Martha      Francis     5551238017      8
James       Henson      5452091120      6
Slade       Chang       5106001301      9
Joshua      Parker      5474725319      2
Breanna     Blevins     5337911745      7
Aspen       Whitley     5141155105      8

*** Enter Your Preference -> s
Hotel data Stored Successfully!

*** Enter Your Preference -> f
Enter Customer First name -> aspen
Room number is 8

*** Enter Your Preference -> f
Enter Customer First name -> kkk
Could not find Customer!!!
```

```
*** Enter Your Preference -> o
-----Guests in Alphabetical order-----
Aspen Whitley
Breanna Blevins
James Henson
Joshua Parker
Martha Francis
Rahim Andrews
Ruby Dawson
Slade Chang

*** Enter Your Preference -> v
Firstname   Surname     CreditCard  Guests
Ruby        Dawson      5535013815      8
Rahim       Andrews     5415319124      1
Martha      Francis     5551238017      8
James       Henson      5452091120      6
Slade       Chang       5106001301      9
Joshua      Parker      5474725319      2
Breanna     Blevins     5337911745      7
Aspen       Whitley     5141155105      8

*** Enter Your Preference -> x
Task Ended!
```

```
Enter Task -> D
+---------------------------------------------+
|               TASK 3 (ARRAY)                |
+---------------------------------------------+

All Rooms are initialized!

+ - - - - - - - - - - - - - - - - - - - - - - +
|               HOTEL  PROGRAM                |
+ - - - - - - - - - - - - - - - - - - - - - - +
|   A.ADD CUSTOMER TO ROOM                    |
|   V.VIEW ALL ROOMS                          |
|   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 IN ALPHABETICAL ORDER       |
|   X.EXIT PROGRAM                            |
+ - - - - - - - - - - - - - - - - - - - - - - +
*** Enter Your Preference -> a
Enter Room number(1 - 8) -> 1
Enter First name of the Paying Guest -> Bruce
Enter Surname of the Paying Guest -> Wayne
Enter Credit Card number -> 12345678
Enter Guests In Room ( > 0) -> 4
Customer Added!

*** Enter Your Preference -> v
Firstname   Surname     CreditCard  Guests
Bruce       Wayne       12345678        4
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0

*** Enter Your Preference -> $%
Invalid Input! Try Again!
```

```
*** Enter Your Preference -> a
Enter Room number(1 - 8) -> 2
Enter First name of the Paying Guest -> Hendry
Enter Surname of the Paying Guest -> Clark
Enter Credit Card number -> 65432178
Enter Guests In Room ( > 0) -> 7
Customer Added!

*** Enter Your Preference -> v
Firstname   Surname     CreditCard  Guests
Bruce       Wayne       12345678        4
Hendry      Clark       65432178        7
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0

*** Enter Your Preference -> a
Enter Room number(1 - 8) -> 44
Room number / Guests in Room is Invalid!!!

*** Enter Your Preference -> a
Enter Room number(1 - 8) -> 1
Room is already occupied! Try Again!!!

*** Enter Your Preference -> a
Enter Room number(1 - 8) -> 3
Enter First name of the Paying Guest -> sample
Enter Surname of the Paying Guest -> sample
Enter Credit Card number -> 123448
Enter Guests In Room ( > 0) -> 0
Guests in room is invalid!!!
Customer is not added!

*** Enter Your Preference -> e
Room number 3 is empty
Room number 4 is empty
Room number 5 is empty
Room number 6 is empty
Room number 7 is empty
Room number 8 is empty

*** Enter Your Preference -> d
Enter Room number to Remove the Customer -> 2
Customer Removed!
```

```
*** Enter Your Preference -> v
Firstname   Surname     CreditCard  Guests
Bruce       Wayne       12345678        4
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0

*** Enter Your Preference -> d
Enter Room number to Remove the Customer -> 2
Room is already empty

*** Enter Your Preference -> d
Enter Room number to Remove the Customer -> 54
Room number is Invalid!!!

*** Enter Your Preference -> l
Hotel data Loaded Successfully!
*** Enter Your Preference -> v
Firstname   Surname     CreditCard  Guests
Ruby        Dawson      5535013815      8
Rahim       Andrews     5415319124      1
Martha      Francis     5551238017      8
James       Henson      5452091120      6
Slade       Chang       5106001301      9
Joshua      Parker      5474725319      2
Breanna     Blevins     5337911745      7
Aspen       Whitley     5141155105      8

*** Enter Your Preference -> s
Hotel data Stored Successfully!

*** Enter Your Preference -> f
Enter Customer First name -> aspen
Room number is 8

*** Enter Your Preference -> f
Enter Customer First name -> kkk
Could not find Customer!!!
```

```
*** Enter Your Preference -> o
-----Guests in Alphabetical order-----
Aspen Whitley
Breanna Blevins
James Henson
Joshua Parker
Martha Francis
Rahim Andrews
Ruby Dawson
Slade Chang

*** Enter Your Preference -> v
Firstname   Surname     CreditCard  Guests
Ruby        Dawson      5535013815      8
Rahim       Andrews     5415319124      1
Martha      Francis     5551238017      8
James       Henson      5452091120      6
Slade       Chang       5106001301      9
Joshua      Parker      5474725319      2
Breanna     Blevins     5337911745      7
Aspen       Whitley     5141155105      8

*** Enter Your Preference -> x
Task Ended!
```

```
Enter Task -> E
+---------------------------------------------+
|                   TASK 4                    |
+---------------------------------------------+

All Rooms are initialized!

+ - - - - - - - - - - - - - - - - - - - - - - +
|               HOTEL  PROGRAM                |
+ - - - - - - - - - - - - - - - - - - - - - - +
|   A.ADD CUSTOMER TO ROOM                    |
|   V.VIEW ALL ROOMS                          |
|   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 IN ALPHABETICAL ORDER       |
|   X.EXIT PROGRAM                            |
+ - - - - - - - - - - - - - - - - - - - - - - +

*** Enter Your Preference -> a
Enter Room number(1 - 8) -> 1
Enter First name of the Paying Guest -> Bruce
Enter Surname of the Paying Guest -> Wayne
Enter Credit Card number -> 12345678
Enter Guests In Room ( > 0) -> 4
Customer Added!

*** Enter Your Preference -> v
Firstname   Surname     CreditCard  Guests
Bruce       Wayne       12345678        4
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0

*** Enter Your Preference -> $%
Invalid Input! Try Again!
```

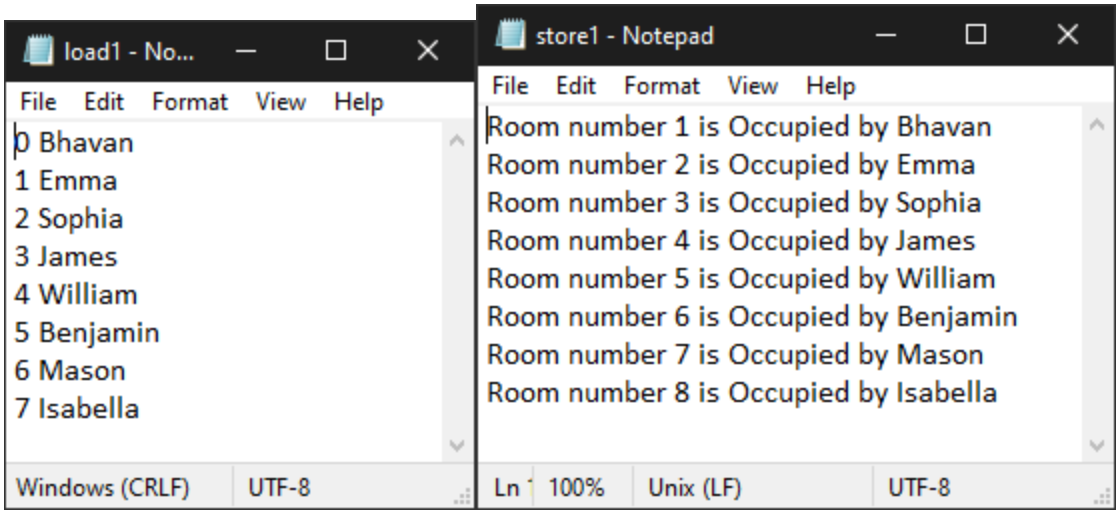
```
*** Enter Your Preference -> a
Enter Room number(1 - 8) -> 2
Enter First name of the Paying Guest -> Hendry
Enter Surname of the Paying Guest -> Clark
Enter Credit Card number -> 65432178
Enter Guests In Room ( > 0) -> 7
Customer Added!

*** Enter Your Preference -> v
Firstname   Surname     CreditCard  Guests
Bruce       Wayne       12345678        4
Hendry      Clark       65432178        7
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0

*** Enter Your Preference -> a
Enter Room number(1 - 8) -> 44
Room number / Guests in Room is Invalid!!!
*** Enter Your Preference -> a
Enter Room number(1 - 8) -> 1
Room is already occupied! Try another Room!

*** Enter Your Preference -> a
Enter Room number(1 - 8) -> 3
Enter First name of the Paying Guest -> sample
Enter Surname of the Paying Guest -> sample
Enter Credit Card number -> 123456
Enter Guests In Room ( > 0) -> 0
Guests in room is invalid!!!
Customer is not added!

*** Enter Your Preference -> e
Room number 3 is empty
Room number 4 is empty
Room number 5 is empty
Room number 6 is empty
Room number 7 is empty
Room number 8 is empty

*** Enter Your Preference -> d
Enter Room number to Remove the Customer -> 2
Customer Removed!
```

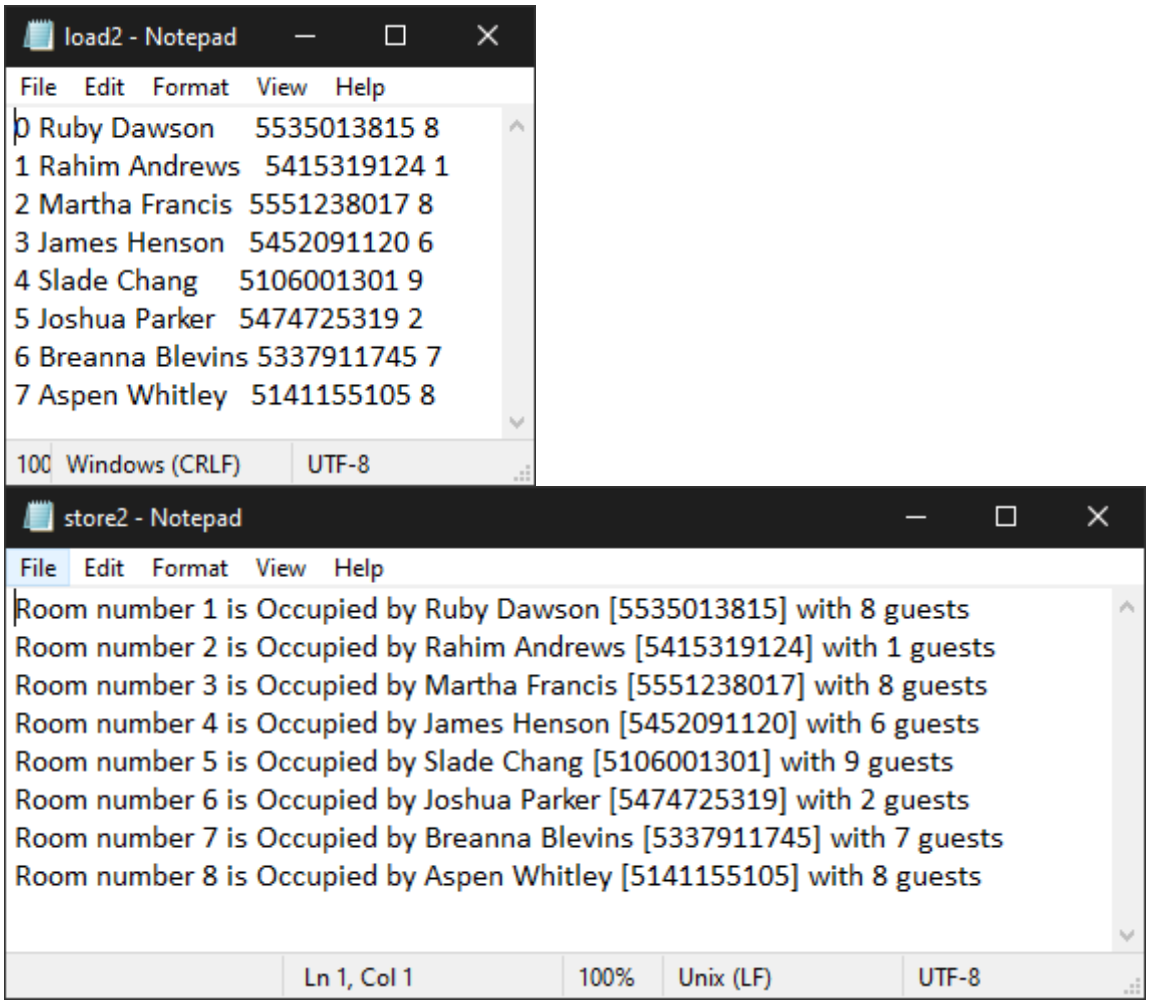
```
*** Enter Your Preference -> v
Firstname   Surname     CreditCard  Guests
Bruce       Wayne       12345678        4
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0
empty       empty       empty       0

*** Enter Your Preference -> d
Enter Room number to Remove the Customer -> 2
Room is already empty

*** Enter Your Preference -> d
Enter Room number to Remove the Customer -> 54
Room number is Invalid!!!

*** Enter Your Preference -> l
Hotel data Loaded Successfully!
*** Enter Your Preference -> v
Firstname   Surname     CreditCard  Guests
Ruby        Dawson      5535013815      8
Rahim       Andrews     5415319124      1
Martha      Francis     5551238017      8
James       Henson      5452091120      6
Slade       Chang       5106001301      9
Joshua      Parker      5474725319      2
Breanna     Blevins     5337911745      7
Aspen       Whitley     5141155105      8

*** Enter Your Preference -> s
Hotel data Stored Successfully!

*** Enter Your Preference -> f
Enter Customer First name -> aspen
Room number is 8

*** Enter Your Preference -> f
Enter Customer First name -> kkk
Could not find Customer!!!
```

```
*** Enter Your Preference -> o
-----Guests in Alphabetical order-----
Aspen Whitley
Breanna Blevins
James Henson
Joshua Parker
Martha Francis
Rahim Andrews
Ruby Dawson
Slade Chang

*** Enter Your Preference -> v
Firstname   Surname     CreditCard  Guests
Ruby        Dawson      5535013815      8
Rahim       Andrews     5415319124      1
Martha      Francis     5551238017      8
James       Henson      5452091120      6
Slade       Chang       5106001301      9
Joshua      Parker      5474725319      2
Breanna     Blevins     5337911745      7
Aspen       Whitley     5141155105      8

*** Enter Your Preference -> a
Enter Room number(1 - 8) -> 4
Room is already Occupied! You will be Added to Waiting List!

Enter First name of the Paying Guest -> Steven
Enter Surname of the Paying Guest -> steve
Enter Credit Card number -> 5337911745
Enter Guests In Room ( > 0) -> 4
Customer Added!

*** Enter Your Preference -> v
Firstname   Surname     CreditCard  Guests
Ruby        Dawson      5535013815      8
Rahim       Andrews     5415319124      1
Martha      Francis     5551238017      8
James       Henson      5452091120      6
Slade       Chang       5106001301      9
Joshua      Parker      5474725319      2
Breanna     Blevins     5337911745      7
Aspen       Whitley     5141155105      8
```

```
*** Enter Your Preference -> a
Enter Room number(1 - 8) -> 5
Room is already Occupied! You will be Added to Waiting List!

Enter First name of the Paying Guest -> bobi
Enter Surname of the Paying Guest -> marley
Enter Credit Card number -> 148168941
Enter Guests In Room ( > 0) -> 2
Customer Added!

*** Enter Your Preference -> v
Firstname   Surname     CreditCard  Guests
Ruby        Dawson      5535013815      8
Rahim       Andrews     5415319124      1
Martha      Francis     5551238017      8
James       Henson      5452091120      6
Slade       Chang       5106001301      9
Joshua      Parker      5474725319      2
Breanna     Blevins     5337911745      7
Aspen       Whitley     5141155105      8

*** Enter Your Preference -> d
Enter Room number to Remove the Customer -> 3
Customer Removed!
A Customer from the Waiting List is Added to this Room!

*** Enter Your Preference -> v
Firstname   Surname     CreditCard  Guests
Ruby        Dawson      5535013815      8
Rahim       Andrews     5415319124      1
Steven      steve       5337911745      4
James       Henson      5452091120      6
Slade       Chang       5106001301      9
Joshua      Parker      5474725319      2
Breanna     Blevins     5337911745      7
Aspen       Whitley     5141155105      8

*** Enter Your Preference -> d
Enter Room number to Remove the Customer -> 8
Customer Removed!
A Customer from the Waiting List is Added to this Room!
```

```
*** Enter Your Preference -> v
Firstname   Surname     CreditCard  Guests
Ruby        Dawson      5535013815      8
Rahim       Andrews     5415319124      1
Steven      steve       5337911745      4
James       Henson      5452091120      6
Slade       Chang       5106001301      9
Joshua      Parker      5474725319      2
Breanna     Blevins     5337911745      7
bobi        marley      148168941       2

*** Enter Your Preference -> x
Task Ended!

------------------------------------------------

Enter Task -> x
Program Ended!
Thank you!

Process finished with exit code 0
```


load1 - No...
File Edit Format View Help
0 Bhavan
1 Emma
2 Sophia
3 James
4 William
5 Benjamin
6 Mason
7 Isabella
Windows (CRLF) UTF-8
store1 - Notepad
File Edit Format View Help
Room number 1 is Occupied by Bhavan
Room number 2 is Occupied by Emma
Room number 3 is Occupied by Sophia
Room number 4 is Occupied by James
Room number 5 is Occupied by William
Room number 6 is Occupied by Benjamin
Room number 7 is Occupied by Mason
Room number 8 is Occupied by Isabella
Ln 100% Unix (LF) UTF-8


load2 - Notepad
File Edit Format View Help
0 Ruby Dawson 5535013815 8
1 Rahim Andrews 5415319124 1
2 Martha Francis 5551238017 8
3 James Henson 5452091120 6
4 Slade Chang 5106001301 9
5 Joshua Parker 5474725319 2
6 Breanna Blevins 5337911745 7
7 Aspen Whitley 5141155105 8
100 Windows (CRLF) UTF-8
store2 - Notepad
File Edit Format View Help
Room number 1 is Occupied by Ruby Dawson [5535013815] with 8 guests
Room number 2 is Occupied by Rahim Andrews [5415319124] with 1 guests
Room number 3 is Occupied by Martha Francis [5551238017] with 8 guests
Room number 4 is Occupied by James Henson [5452091120] with 6 guests
Room number 5 is Occupied by Slade Chang [5106001301] with 9 guests
Room number 6 is Occupied by Joshua Parker [5474725319] with 2 guests
Room number 7 is Occupied by Breanna Blevins [5337911745] with 7 guests
Room number 8 is Occupied by Aspen Whitley [5141155105] with 8 guests
Ln 1, Col 1 100% Unix (LF) UTF-8