INFORMATICS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF WESTMINSTER

Informatics Institute of Technology

Department of Computing

BEng(Hons) Software Engineering

Module: Software Development 2

4COSC005/W

Individual Coursework Report

Module Leader : Rajitha Jayasinghe

Name: A.E.W Jayatilake
Student ID:2019530
Student UOW ID: w1761374

Group Number: Group H

Test Case	Expected Result	Actual Result	Pass/Fail
Press “A”	Add customers	Add customer to Rooms	Pass
After the Click “A” Press “1 to 9”	Enter the Customer’s Details	Enter the Customer’s Details	Pass
Press “V”	View all rooms	Displays all rooms	Pass
Press “E”	Display empty rooms	Displays all empty rooms	Pass
Press “D”	Delete customer from room	Delete customer from selected room	Pass
After the Click “D”Press “1 to 9”	Delete Room	Delete Room	Pass
Press ”F”	Find room from customer name	Displays the Room number	Fail
After the Enter “F” Then Search Using Name	Display the Room	Display the Room	Pass
Press “S”	Store To File	Displays Please add a correct value. Try again	Fail

Arrays Version

```
"C:\Program Files\Java\jdk1.8.0_281\bin\java.exe" ...
Enter the the menu number that you want to select
A - Add customers
V - View all rooms
E - Display empty rooms
D - Delete customer from room
F - Find room from customer name
S-  Store To File :
```

Select add customer and enter data.

```
a
Enter room number (1-8) or 9 to stop:
1
Enter name for room 1 :
abcd
Enter number of guests
6
Enter payer first name
achintha
Enter payer last name
jay
Enter payer card number
1234567
Enter room number (1-8) or 9 to stop:
2
Enter name for room 2 :
12345678
Enter number of guests
9
Enter payer first name
Alpha
Enter payer last name
King
Enter payer card number
123456780
Enter room number (1-8) or 9 to stop:
9
```

Viewing all rooms

```
Enter the the menu number that you want to select
A - Add customers
V - View all rooms
E - Display empty rooms
D - Delete customer from room
F - Find room from customer name
S-  Store To File :
v
room 0 occupied by abcd
room 1 occupied by 12345678
room 2 is empty
room 3 is empty
room 4 is empty
room 5 is empty
room 6 is empty
room 7 is empty
Enter the the menu number that you want to select
A - Add customers
V - View all rooms
E - Display empty rooms
D - Delete customer from room
F - Find room from customer name
S-  Store To File :
```

Display only empty Rooms

```
Enter the the menu number that you want to select
A - Add customers
V - View all rooms
E - Display empty rooms
D - Delete customer from room
F - Find room from customer name
S-  Store To File :
e
room 2 is empty
room 3 is empty
room 4 is empty
room 5 is empty
room 6 is empty
room 7 is empty
Enter the the menu number that you want to select
A - Add customers
V - View all rooms
E - Display empty rooms
D - Delete customer from room
F - Find room from customer name
S-  Store To File :
```

Deleting customer from Room.

```
Enter the the menu number that you want to select
A - Add customers
V - View all rooms
E - Display empty rooms
D - Delete customer from room
F - Find room from customer name
S-  Store To File :
d
Enter room number (1-8) or 9 to stop:
1
Enter room number (1-8) or 9 to stop:
9
```

Output

```
Enter the the menu number that you want to select
A - Add customers
V - View all rooms
E - Display empty rooms
D - Delete customer from room
F - Find room from customer name
S-  Store To File :
v
room 0 is empty
room 1 occupied by 12345678
room 2 is empty
room 3 is empty
room 4 is empty
room 5 is empty
room 6 is empty
room 7 is empty
```

Finding Room Number By name

```
Enter the the menu number that you want to select
A - Add customers
V - View all rooms
E - Display empty rooms
D - Delete customer from room
F - Find room from customer name
S-  Store To File :
f
Enter customer name
Achin
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException Create breakpoint : 5
    at Hotel.findRoomForCustomer(Hotel.java:183)
    at Hotel.menuSection(Hotel.java:121)
    at Hotel.main(Hotel.java:14)

Process finished with exit code 1
```

Store to File

```
"C:\Program Files\Java\jdk1.8.0_281\bin\java.exe" ...
Enter the the menu number that you want to select
A - Add customers
V - View all rooms
E - Display empty rooms
D - Delete customer from room
F - Find room from customer name
S-  Store To File :
s
Please add a correct value. Try again
```

Classes

```
"C:\Program Files\Java\jdk1.8.0_281\bin\java.exe" ...
Enter the the menu number that you want to select
A - Add customers
V - View all rooms
E - Display empty rooms
D - Delete customer from room
F - Find room from customer name
S-  Store To File :
```

Select add customer and enter data.

```
a
Enter room number (1-8) or 9 to stop:
1
Enter name for room 1 :
abcd
Enter number of guests
6
Enter payer first name
achintha
Enter payer last name
jay
Enter payer card number
1234567
Enter room number (1-8) or 9 to stop:
2
Enter name for room 2 :
12345678
Enter number of guests
9
Enter payer first name
Alpha
Enter payer last name
King
Enter payer card number
123456780
Enter room number (1-8) or 9 to stop:
9
```

Viewing all rooms

```
Enter the the menu number that you want to select
A - Add customers
V - View all rooms
E - Display empty rooms
D - Delete customer from room
F - Find room from customer name
S-  Store To File :
v
room 0 occupied by abcd
room 1 occupied by 12345678
room 2 is empty
room 3 is empty
room 4 is empty
room 5 is empty
room 6 is empty
room 7 is empty
Enter the the menu number that you want to select
A - Add customers
V - View all rooms
E - Display empty rooms
D - Delete customer from room
F - Find room from customer name
S-  Store To File :
```

Display only empty Rooms

```
Enter the the menu number that you want to select
A - Add customers
V - View all rooms
E - Display empty rooms
D - Delete customer from room
F - Find room from customer name
S-  Store To File :
e
room 2 is empty
room 3 is empty
room 4 is empty
room 5 is empty
room 6 is empty
room 7 is empty
Enter the the menu number that you want to select
A - Add customers
V - View all rooms
E - Display empty rooms
D - Delete customer from room
F - Find room from customer name
S-  Store To File :
```

Deleting customer from Room.

```
Enter the the menu number that you want to select
A - Add customers
V - View all rooms
E - Display empty rooms
D - Delete customer from room
F - Find room from customer name
S-  Store To File :
d
Enter room number (1-8) or 9 to stop:
1
Enter room number (1-8) or 9 to stop:
9
```

Output

```
Enter the the menu number that you want to select
A - Add customers
V - View all rooms
E - Display empty rooms
D - Delete customer from room
F - Find room from customer name
S-  Store To File :
v
room 0 is empty
room 1 occupied by 12345678
room 2 is empty
room 3 is empty
room 4 is empty
room 5 is empty
room 6 is empty
room 7 is empty
```

Finding Room Number By name

```
Enter the the menu number that you want to select
A - Add customers
V - View all rooms
E - Display empty rooms
D - Delete customer from room
F - Find room from customer name
S-  Store To File :
f
Enter customer name
Achin
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException Create breakpoint : 5
    at Hotel.findRoomForCustomer(Hotel.java:183)
    at Hotel.menuSection(Hotel.java:121)
    at Hotel.main(Hotel.java:14)

Process finished with exit code 1
```

Store to File

```
"C:\Program Files\Java\jdk1.8.0_281\bin\java.exe" ...
Enter the the menu number that you want to select
A - Add customers
V - View all rooms
E - Display empty rooms
D - Delete customer from room
F - Find room from customer name
S-  Store To File :
s
Please add a correct value. Try again
```

Arrays Version

- **Hotel.java**

```
//Importing the libraries
import java.util.Scanner;
import java.util.*;
import java.io.*;

public class Hotel {


    public static void main(String[] args) {
//creating 2 arrays
        String[][] hotel = new String[5][8];
        HotelWaiting q = new HotelWaiting();

        initialise(hotel);
        menuSection(hotel,q);


    }
    //creating private methods for initialise
```

```
    private static void initialise(String hotelRef[][]) {

        //initialaise
        for (int y = 0; y < 5; y++) {
            for (int x = 0; x < 8; x++) {
                hotelRef[y][x] = "e";
            }
        }
    }

    //creating private methods to addCustomers
    private static void addCustomers(String[][] hotel,
HotelWaiting q) {
        //for the user input create a scanner
        Scanner input = new Scanner(System.in);
        int roomNum = 0;
        String guest;
        String fisrtName;
        String lastName;
        String cardNumber;
        String roomName;

        int full = 0;

        //addCustomers
        for (int x = 0; x < 8; x++) {
            if (!hotel[0][x].equals("e")) {
                full += 1;
            }
        }


        while (roomNum < 9 ) {

            System.out.println("Enter room number (1-8) or 9 to
stop:");
            roomNum = input.nextInt();

            if (roomNum > 8){
                break;
            }
            System.out.println("Enter name for room " + roomNum
+ " :");
            roomName = input.next();
            if (roomNum > 0 && roomNum < 9) {
                System.out.println("Enter number of guests ");
                guest = input.next();
                System.out.println("Enter payer first name ");
                fisrtName = input.next();
                System.out.println("Enter payer last name ");
                lastName = input.next();
                System.out.println("Enter payer card number ");
                cardNumber = input.next();

                if (full < 8) {
                    hotel[0][roomNum - 1] = roomName;
                    hotel[1][roomNum - 1] = guest;
                    hotel[2][roomNum - 1] = fisrtName;
                    hotel[3][roomNum - 1] = lastName;
                    hotel[4][roomNum - 1] = cardNumber;
                }
                else {

q.enQueue(roomName,guest,fisrtName,lastName,cardNumber);
                }
            }
        }
```

```
    }

    //creating private methods for viewRooms
    private static void viewRooms(String[][] hotel) {
        for (int x = 0; x < 8; x++) {
            if (hotel[0][x].equals("e")) {
                System.out.println("room " + x + " is empty");
            } else {
                System.out.println("room " + x + " occupied by "
+ hotel[0][x]);
            }
        }
    }

    //creating private methods for menuSection
    private static void menuSection(String[][] hotel,
HotelWaiting q) {

        Scanner input = new Scanner(System.in);
        String selection;

        boolean state = true;

        while (state) {
            System.out.println("Enter the the menu number that
you want to select");
            System.out.println("A - Add customers");
            System.out.println("V - View all rooms");
            System.out.println("E - Display empty rooms");
            System.out.println("D - Delete customer from room");
            System.out.println("F - Find room from customer
name");
            System.out.println("S-  Store To File :");
            selection = input.nextLine();



            if (selection.equalsIgnoreCase("A")) {
                addCustomers(hotel,q);
            } else if (selection.equalsIgnoreCase("V")) {
                viewRooms(hotel);
            } else if (selection.equalsIgnoreCase("E")) {
                displayEmptyrooms(hotel);
            } else if (selection.equalsIgnoreCase("D")) {
                deleteCutomer(hotel,q);
            } else if (selection.equalsIgnoreCase("F")) {
                findRoomForCustomer(hotel);
            }
//              else if (selection.equalsIgnoreCase("S")) {
//                StoreProgramDataInToFile(hotel);
//            }
            else {
                System.out.println("Please add a correct value.
Try again");
            }
        }
    }

    //creating private methods for displayEmptyrooms
    private static void displayEmptyrooms(String[][] hotel) {

        for (int x = 0; x < 8; x++) {
            if (hotel[0][x].equals("e")) {
                System.out.println("room " + x + " is empty");
            }
```

```
        }

    }

    //creating private methods for deleteCutomer
    private static void deleteCutomer(String[][] hotel,
HotelWaiting q) {

        Scanner input = new Scanner(System.in);
        int roomNum = 0;

        while (roomNum < 9) {

            System.out.println("Enter room number (1-8) or 9 to
stop:");
            roomNum = input.nextInt();

            if (roomNum > 0 && roomNum < 9) {
                if (!q.isEmpty()) {
                    String[][] obj = q.deQueue();
                    hotel[0][roomNum - 1] = obj[0][0];
                    hotel[1][roomNum - 1] = obj[1][0];
                    hotel[2][roomNum - 1] = obj[2][0];
                    hotel[3][roomNum - 1] = obj[3][0];
                    hotel[4][roomNum - 1] = obj[4][0];
                }
                else {
                    hotel[0][roomNum - 1] = "e";
                    hotel[1][roomNum - 1] = "e";
                    hotel[2][roomNum - 1] = "e";
                    hotel[3][roomNum - 1] = "e";
                    hotel[4][roomNum - 1] = "e";
                }
            }

        }

    }

    //creating private methods for find room for customer
    private static void findRoomForCustomer(String[][] hotel) {

        Scanner input = new Scanner(System.in);
        String name;

        System.out.println("Enter customer name");
        name = input.nextLine();

        for (int x = 0; x < 6; x++) {
            if (hotel[x].equals(name)) {
                System.out.println("Room " + x + 1 + " was
occupied by " + name);
            }
        }

    }

    //creating private methods for store program in to file path
    private static void StoreProgramDataInToFile(String[][]
hotel) throws IOException {
        try (PrintWriter out = new PrintWriter(new
FileWriter("C:/Users/Hp -
Pavilion/Desktop/CW/w1761374_ArrayOnly/outputfile.txt"))) {

            for (int x = 0; x < 8; x++) {
                out.println("Name and Room number is: " +
hotel[0][x] + "at: " + x);
```

```
                }
            }

            System.out.println("All Room Names have been Saved.");
        }


    }
```

- **HotelWaiting.java**

```
public class HotelWaiting {
    int SIZE = 5;
    int front, rear;
    String items[][] = new String[5][SIZE];

    HotelWaiting() {
        front = -1;
        rear = -1;
    }

    //check Queue is full
    boolean isFull() {
        if (front == 0 && rear == SIZE - 1) {
            return true;
        }
        if (front == rear + 1) {
            return true;
        }
        return false;
    }


    //check Queue is empty
    boolean isEmpty() {
        if (front == -1)
            return true;
        else
            return false;
    }

    //enter value for add circular queve
    void enQueue(String name,String guest,String fisrt,String last,String
card) {
        if (isFull()) {
            System.out.println("Queue is full");
        } else {
            if (front == -1)
                front = 0;
            rear = (rear + 1) % SIZE;
            items[0][rear] = name;
            items[1][rear] = guest;
            items[2][rear] = fisrt;
            items[3][rear] = last;
            items[4][rear] = card;
            System.out.println("Inserted " + name);
        }
    }
```

```
    //removing queue
    String[][] deQueue() {
        String[][] element = new String[5][1];
        if (isEmpty()) {
            System.out.println("Queue is empty");
            return null;
        } else {
            element[0][0] = items[0][front];
            element[1][0] = items[1][front];
            element[2][0] = items[2][front];
            element[3][0] = items[3][front];
            element[4][0] = items[4][front];
            if (front == rear) {
                front = -1;
                rear = -1;
            } else {
                front = (front + 1) % SIZE;
            }
            return (element);
        }
    }

}
```

Classes Version

- **Hotel2**

```
•   //Importing the libraries
    import java.io.FileWriter;
    import java.io.IOException;
    import java.io.PrintWriter;
    import java.util.Scanner;

    public class Hotel2 {


        public static void main(String[] args) {

            Room[] hotel = new Room[8];
            HotelWaiting2 q = new HotelWaiting2();

            initialise(hotel);
            menuSection(hotel,q);
        }

        //creating private methods for initialise
        private static void initialise(Room[] hotelRef) {

            for (int y = 0; y < 8; y++) {
                hotelRef[y] = new Room("k","k",new
    Person("k","k","k"));
            }
        }

        //creating private methods to addCustomers
        private static void addCustomers(Room[] hotel,HotelWaiting2
    q) {

            Scanner input = new Scanner(System.in);
            int roomNum = 0;
```

```
        String guest;
        String fisrtName;
        String lastName;
        String cardNumber;
        String roomName;

        int full = 0;


        //addCustomers
        for (int x = 0; x < 8; x++) {
            if (!hotel[x].getRoomName().equalsIgnoreCase("e")) {
                full += 1;
            }
        }


        while (roomNum < 9) {

            System.out.println("Enter room number (1-8) or 9 to
stop:");
            roomNum = input.nextInt();
            if (roomNum > 8) {
                break;
            }
            System.out.println("Enter name for room " + roomNum
+ " :");
            roomName = input.next();
            if (roomNum > 0 && roomNum < 9) {
                System.out.println("Enter number of guests ");
                guest = input.nextLine();
                System.out.println("Enter payer first name ");
                fisrtName = input.nextLine();
                System.out.println("Enter payer last name ");
                lastName = input.nextLine();
                System.out.println("Enter payer card number ");
                cardNumber = input.nextLine();

                if (full < 8) {
                    hotel[roomNum - 1] = new
Room(roomName,guest,new Person(fisrtName,lastName,cardNumber));
                } else {
                    q.enQueue(new Room(roomName,guest,new
Person(fisrtName,lastName,cardNumber)));
                }
            }
        }

    }

    //creating private methods for viewRooms
    private static void viewRooms(Room[] hotel) {
        for (int x = 0; x < 8; x++) {
            if (hotel[x].getRoomName().equals("e")) {
                System.out.println("room " + x + " is empty");
            } else {
                System.out.println("room " + x + " occupied by "
+ hotel[x].getRoomName());
            }
        }
    }

    //creating private methods for menuSection
    private static void menuSection(Room[] hotel,HotelWaiting2
q) {
```

```
        Scanner input = new Scanner(System.in);
        String selection;

        boolean state = true;

        while (state) {
            System.out.println("Enter the the menu number that
you want to select");
            System.out.println("A - Add customers");
            System.out.println("V - View all rooms");
            System.out.println("E - Display empty rooms");
            System.out.println("D - Delete customer from room");
            System.out.println("F - Find room from customer
name");
            selection = input.nextLine();

            if (selection.equalsIgnoreCase("A")) {
                addCustomers(hotel,q);
            } else if (selection.equalsIgnoreCase("V")) {
                viewRooms(hotel);
            } else if (selection.equalsIgnoreCase("E")) {
                displayEmptyrooms(hotel);
            } else if (selection.equalsIgnoreCase("D")) {
                deleteCutomer(hotel,q);
            } else if (selection.equalsIgnoreCase("F")) {
                findRoomForCustomer(hotel);
            } else {
                System.out.println("Please add a correct value.
Try again");
            }
        }
    }

    //creating private methods for displayEmptyrooms
    private static void displayEmptyrooms(Room[] hotel) {

        for (int x = 0; x < 8; x++) {
            if (hotel[x].getRoomName().equals("e")) {
                System.out.println("room " + x + " is empty");
            }
        }

    }

    //creating private methods for deleteCutomer
    private static void deleteCutomer(Room[] hotel,HotelWaiting2
q) {

        Scanner input = new Scanner(System.in);
        int roomNum = 0;

        while (roomNum < 9) {

            System.out.println("Enter room number (1-8) or 9 to
stop:");
            roomNum = input.nextInt();

            if (roomNum > 0 && roomNum < 9) {
                if (!q.isEmpty()) {
                    Room obj = q.deQueue();
                    hotel[roomNum - 1] = obj;
                } else {
                    hotel[roomNum - 1] = new Room("e","e",new
Person("e","e","e"));
                }
            }
```

```
            }

        }

        //creating private methods for find room for customer
        private static void findRoomForCustomer(Room[] hotel) {

            Scanner input = new Scanner(System.in);
            String name;

            System.out.println("Enter customer name");
            name = input.nextLine();

            for (int x = 0; x < 8; x++) {
                if (hotel[x].getRoomName().equals(name)) {
                    System.out.println("Room " + x + 1 + " was
    occupied by " + name);
                }
            }

        }

        //creating private methods for store program in to file path
        private static void StoreProgramDataInToFile(Room[] hotel)
    throws IOException {
            try (PrintWriter out = new PrintWriter(new
    FileWriter("C:/Users/Hp -
    Pavilion/Desktop/CW/w1761374_Classes/outputfile.txt"))) {

                for (int x = 0; x < 8; x++) {
                    out.println("Name and Room number is: " +
    hotel[x].getRoomName() + "at: " + x);

                }
            }
            System.out.println("All Room Names have been Saved.");

        }

    }
```

- **HotelWaiting2**

```
public class HotelWaiting2 {
    int SIZE = 5;
    int front, rear;
    Room items[] = new Room[SIZE];

    HotelWaiting2() {
        front = -1;
        rear = -1;
    }


    //check Queue is full
    boolean isFull() {
        if (front == 0 && rear == SIZE - 1) {
            return true;
        }
        if (front == rear + 1) {
            return true;
```

```
        }
        return false;
    }


    //check Queue is empty
    boolean isEmpty() {
        if (front == -1)
            return true;
        else
            return false;
    }


    //enter value for add circular queve
    void enQueue(Room room) {
        if (isFull()) {
            System.out.println("Queue is full");
        } else {
            if (front == -1)
                front = 0;
            rear = (rear + 1) % SIZE;
            items[rear] = room;
            System.out.println("Inserted " + room.getRoomName());
        }
    }


    //removing queue
    Room deQueue() {

        if (isEmpty()) {
            System.out.println("Queue is empty");
            return null;
        } else {

            Room item = items[front];

            if (front == rear) {
                front = -1;
                rear = -1;
            } else {
                front = (front + 1) % SIZE;
            }
            return item;
        }
    }

}
```

- **Room**

```
public class Room {

    //creating private veriables
    private String roomName;
    private String guest;
    private Person person;

    public Room(String roomName,String  guest,Person person) {

        this.roomName = roomName;
        this.guest = guest;
        this.person = person;
    }
```

```
        //create a getter and setters for room name
        public String getRoomName() {
            return roomName;
        }

        public Room setRoomName(String roomName) {
            this.roomName = roomName;
            return this;
        }
        //create a getter and setters for Guest
        public String getGuest() {
            return guest;
        }

        public Room setGuest(String guest) {
            this.guest = guest;
            return this;
        }

        //create a getter and setters for person
        public Person getPerson() {
            return person;
        }

        public Room setPerson(Person person) {
            this.person = person;
            return this;
        }
    }
```

- **Person**

```
public class Person {

    //creating private veriables
    private String firstName;
    private String lastName;
    private String cardNumber;

    public Person(String firstName,String lastName,String cardNumber) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.cardNumber = cardNumber;
    }

    //create a getter and setters for First name
    public String getFirstName() {
        return firstName;
    }

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

    //create a getter and setters for Last name
    public String getLastName() {
        return lastName;
    }

    public Person setLastName(String lastName) {
```

```
        this.lastName = lastName;
        return this;
    }

    //create a getter and setters for card number
    public String getCardNumber() {
        return cardNumber;
    }

    public Person setCardNumber(String cardNumber) {
        this.cardNumber = cardNumber;
        return this;
    }
}
```