Task 5-Testing.

Name: U.A.Pragathi Sithmini

UoW id: w1810216

IIT id: 20200351

Group-E

4COSC005/W Software Development 2 – Coursework

Table of Contents

Discussion

Test cases had to be used to verify their accuracy in the following tasks.

Test cases had to be used to make certain of their act of having no error in the coming of their act of having no error in the coming here-after tasks. Therefore I have ordered experiments for every one of the four undertakings freely. While frame for events up the experiments one at a time, it takes care of to be done so the complete program of work is covered. In the awake of changing over what it was dependent on upon of us as per a given undertaking into a program, we get the important yield when it is carried out. At that point need to check if this is the right yield.

Here I got the yields by running the codes set one up by one errand and giving undertaking and giving the information comparing to them. The information needed to get the yields were furnished as per the program.

Here the information that should be given by the errand in run all through the program. That information is shown through the program yields. Experiments are made from those yields. So from the experiments here I have covered what anticipate from the entire program.

Table of test cases.

Task3

Arrays Version

Test Case	Expected Result	Actual Result	Pass/Fail
(Rooms Initialised correctly) After program starts, Press 'E'	Displays 'empty' for all rooms	Displays 'empty' for all rooms	Pass
(Add customer "Bob" to room 0) Select 'A', enter "Bob" Again, press enter and type, "Martin" (surname) Again, press enter and type credit Card number (658974152) Finally press enter and type number of people in the room (3)	The details should be as follows. Add Customer to a Room: Enter room number (0-8) or 8 to stop: 0 Enter name for room 0: Bob Enter First Name: Bob Enter Surname: Martin Enter Credit card Number: 658974152 Enter number of guests in room: 3 After Press "V"	It was displayed as follows. room 0 occupied by Bob room 1 is empty room 2 is empty room 3 is empty room 4 is empty room 5 is empty room 6 is empty room 7 is empty	Pass
(Add customer "Bob" to room 0) Select A, enter "Bob" and other details Press 'E'	Display "empty" for room 1, 2, 3, 4, 5, 6, 7	Display "empty" for room 1, 2, 3, 4, 5, 6, 7	Pass
Select 'D' for delete customer from room, and Enter room number 0 Press 'V'	Displays "empty" for room 0 (Displays "empty" for all rooms)	Displays "empty" for room 0 (Displays "empty" for all rooms)	Pass

Select ‘A’ for add customer to a room, and enter name for room (customer name), (Jone) room number 5, Select F for find name from customer name	Display John is in room 5	Display “ * Jone's room number is 5.”	Pass
Select ‘S’ for store program data into a file	Make an information document in the folder and show the information remembered for the program up until now.	room 0 occupied by e room 1 occupied by e room 2 occupied by e room 3 occupied by e room 4 occupied by e room 5 occupied by Jone room 6 occupied by e room 7 occupied by e	Pass
Select ‘L’ for load program data from above file.	Displays “e” for room 0, 1, 2, 3,4,6,7 and displays “Bob” for room 3.	---LOAD PROGRAM DATA INTO FILE--- room 0 occupied by e room 1 occupied by e room 2 occupied by e room 3 occupied by e room 4 occupied by e room 5 occupied by Jone room 6 occupied by e room 7 occupied by e	Pass
Select ‘O’ for view guests ordered alphabetically by name. (Enter ‘A’ several times and add 8 names to 6 rooms) 0- Bob 1- Sam 2- Oliver 3- Cooper 4- Luke 5- Robert 6- Dennis 7- Paulo	Enter ‘V’ for view all rooms and after that select ‘O’ for ordered alphabetically by name. Bob Cooper Dennis Luke Oliver Paulo Robert Sam	It was displayed as follows. Bob Cooper Dennis Luke Oliver Paulo Robert Sam	Pass

Task 4

Class Version

Test Case	**Expected Result**	**Actual Result**	**Pass/Fail**
(Rooms Initialised correctly) After program starts, Press ‘E’	Displays “empty” for all rooms (empty)	---DISPLAY EMPTY ROOMS--- room 0 is empty room 1 is empty room 2 is empty room 3 is empty room 4 is empty room 5 is empty room 6 is empty room 7 is empty	Pass
Select ‘A’ and add customer to a room with details	Displays Room 0 occupied by Jone Martin.	room 0 occupied by Jone room 1 occupied by e room 2 occupied by e room 3 occupied by e room 4 occupied by e room 5 occupied by e room 6 occupied by e room 7 occupied by e	Pass
Add Names with details for all rooms and press ‘V’	Displays Room Numbers with names	room 0 occupied by Jone room 1 occupied by Bob room 2 occupied by Cooper room 3 occupied by Dennis room 4 occupied by Luke room 5 occupied by Oliver room 6 occupied by Paulo room 7 occupied by Sam	Pass
Again add name(Robbert) for room number 3 ,Then Enter ‘D’ for delete customer from Room3	Displays Room3 is occupied by “Robbert”(auto displayed)	room 0 occupied by Jone room 1 occupied by Bob room 2 occupied by Cooper	Pass

		room 3 occupied by Robbert room 4 occupied by Luke room 5 occupied by Oliver room 6 occupied by Paulo room 7 occupied by Sam	

***Should be considered: Every one of the alternatives made in the class variant of assignment 3 dynamic (work) similarly here. Here a holding up list has been added to keep the custom booking got when the lodging is full and it has been set up so that when a client is deleted it is set to automatically add somebody from that waiting list.

Codes and Screenshots

1.Task 3

Arrays Version

1.1Code Snippet

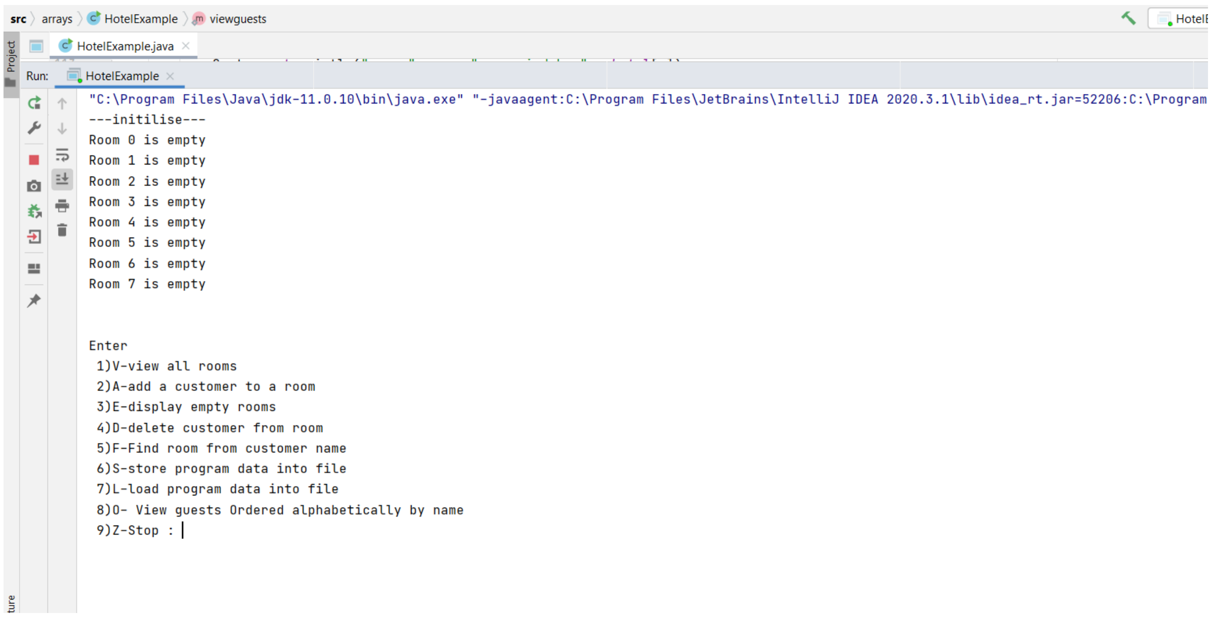
```
package arrays;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.*;
import java.util.*;
public class HotelExample {
   static int roomNum;
   static String roomName;
   static String findperson;                    //created the variables as global variables.
   static int roomdelete;
   static String[] hotel = new String[9];           //created the global  arraylist.
   static Integer[] noGuests = new Integer[9];           //created the global  arraylist.
   static String[] firstNames = new String[9];
   static String[] sirnames = new String[9];
```

```
    static Integer[] creditCardNumbers = new Integer[9];
    static Scanner input = new Scanner(System.in);
    static ArrayList sortedRoomNames;
    static String name;


    public static void main(String[] args) throws IOException {        //main class
        initialise();                                              //main class  for (int x = 0; x < 6; x++ ) hotel[x] = "";
//better to initialise in a procedure
        menu();                                //execute the menu method.
    }
    static void initialise() {
        System.out.println( "---initilise---");
        for (int x = 0; x < 8; x++ ) {
            hotel[x] = "e";
            System.out.println("Room " + x + " is empty");
        }
    }
    static void menu() throws IOException {//created the method called menu for stored the methods
which prompt from the user.

        while (true) {
            System.out.println("\n");
            System.out.print("Enter \n 1)V-view all rooms \n 2)A-add a customer to a room \n 3)E-display
empty rooms" +
                    " \n 4)D-delete customer from room \n 5)F-Find room from customer name \n 6)S-store
program data into file \n " +
                    "7)L-load program data into file \n 8)O- View guests Ordered alphabetically by name \n
9)Z-Stop : ");
            String choice = input.next();
            System.out.println("\n");
            if (choice.equals("Z")){
                break;
            }
            switch (choice) {
                case "V":
                    System.out.println("---VIEW ALL ROOMS---");
                    viewsallrooms();
                    break;
                case "A":
                    System.out.println("---ADD A CUSTOMER TO A ROOM---");
                    addscustomertoroom();
                    break;
                case "E":
                    System.out.println("---DISPLAY EMPTY ROOMS---");
                    displayEmptyRooms();
                    break;
                case "D":
```

```
                System.out.println("---DELETE CUSTOMER FROM ROOM---");
                deleteCustomer();
                break;
             case "F":
                System.out.println("---FIND ROOM FROM CUSTOMER NAME---");
                findcustomer();
                break;
             case "S":
                storedata();
                break;
             case "L":
                System.out.println("---LOAD PROGRAM DATA INTO FILE---");
                loaddata();
                break;
             case "O":
                System.out.println("---VIEW GUESTS ORDERED ALPHABETICALLY BY NAME---");
//method for bview guests ordered alphabetically by name.
                viewguests();
                break;
             default:
                System.out.println("Invalid input");
                break;
          }

        }
    }
    static void addscustomertoroom(){                                    //method for add a customer to a
room.
        System.out.print("Enter room number (0-8) or 8 to stop: " );
        roomNum = input.nextInt();
        System.out.print("Enter name for room " + roomNum +" : " ) ;
        roomName = input.next();
        hotel[roomNum] = roomName ;


        System.out.print("Enter FirstNAme : " ) ;
        String firstName = input.next();
        firstNames[roomNum] = firstName;

        System.out.print("Enter Sirname : " ) ;
        String sirname = input.next();
        sirnames[roomNum] = firstName;

        System.out.print("Enter Credit card Number : " ) ;
        int CreditCardNo = input.nextInt();
        creditCardNumbers[roomNum] = CreditCardNo;

        System.out.print("Enter number of guests in  room : " ) ;
        int guestCount = input.nextInt();
        noGuests[roomNum] = guestCount ;

    }
    static void viewsallrooms(){                                    //method for view all rooms.
        for (int x = 0; x < 8; x++ ){
            System.out.println("room " + x + " occupied by " + hotel[x]);
        }
    }

    static void displayEmptyRooms() {                                    //method for display empty rooms.
        for (int x = 0; x < 8; x++ ){
            if (hotel[x].equals("e")){
                System.out.println("room " + x + " is empty");
            }
        }
    }

    static void deleteCustomer(){                                    //method for delete customer from
room.
        System.out.print("Enter the room number which want to delete the customer : ");
        roomdelete = input.nextInt();
        hotel[roomdelete]="e";
        noGuests[roomdelete] = 0;
        firstNames[roomdelete] = "";
        sirnames[roomdelete] = "";
        creditCardNumbers[roomdelete] = 0;
    }
    static void findcustomer(){                                    //method for find room from customer
name.
        System.out.print("Enter the name of the person who want owns the room number to be find :
");
        findperson=input.next();
        for(int x=0;x<8;x++){
            if(hotel[x].equals(findperson)){
                System.out.println("* "+findperson+"'s room number is " +x+".");
            }
        }
    }

    static void storedata() throws IOException {                              //method for store the data into
file.
        FileWriter fw=new FileWriter ("store.txt");
        for (int x = 0; x < 8; x++ ){
            fw.write("room " + x + " occupied by " + hotel[x]+"\n");
        }
        fw.close();
    }
```

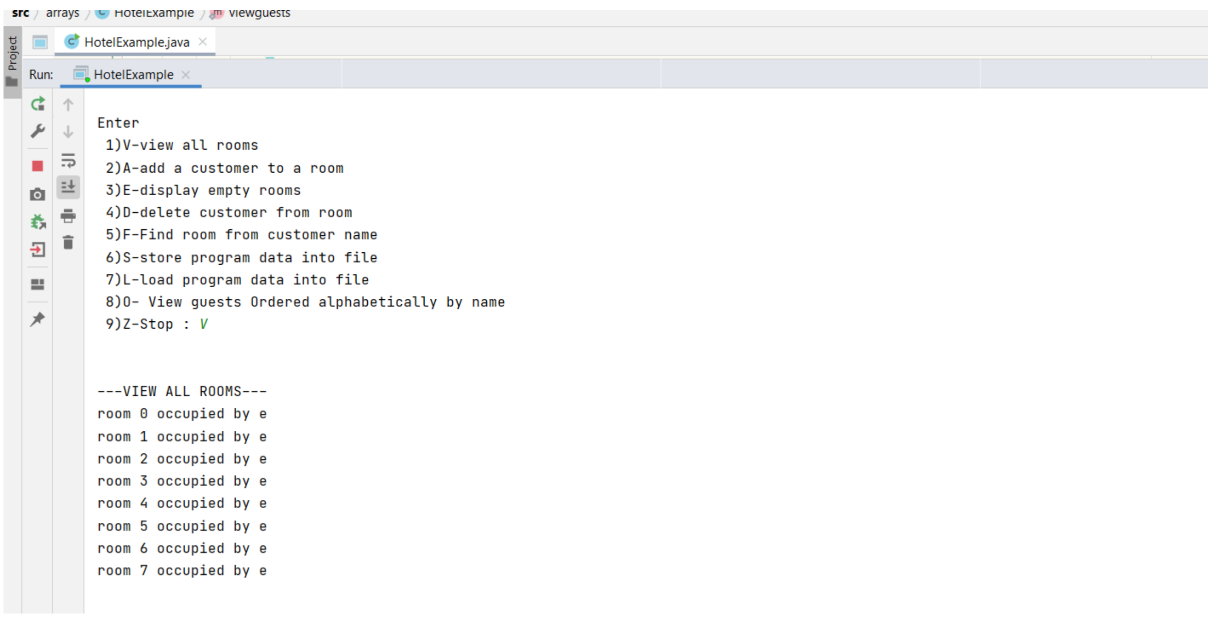
```
   static void loaddata() throws FileNotFoundException {                         //method for load data into
file.
      File f=new File("store.txt");
      Scanner sc=new Scanner(f);
      while ((sc.hasNext())){
         System.out.println(sc.nextLine());
      }
   }
   static void viewguests(){//method for view the guests ordered alphabetically by name.

       ArrayList<String> sortedRoomNames = new ArrayList<>(Arrays.asList(hotel));
       for(int i = 0; i< sortedRoomNames.size()-1; i++)
       {
          for (int j = i+1; j< sortedRoomNames.size()-1; j++)
          {
          //compares each elements of the array to all the remaining elements
             if(sortedRoomNames.get(i).compareTo(sortedRoomNames.get(j))>0)
             {   //swapping array elements
                String temp = sortedRoomNames.get(i);
                sortedRoomNames.set(i,sortedRoomNames.get(j));
                sortedRoomNames.set(j,temp);
             }
          }
       }

       for (int x = 0; x < 8; x++ ){
          if (!(sortedRoomNames.get(x)=="e")){
             System.out.println(sortedRoomNames.get(x));
          }

       }
   }
}
```

1.2 Screenshot of Code Snippet



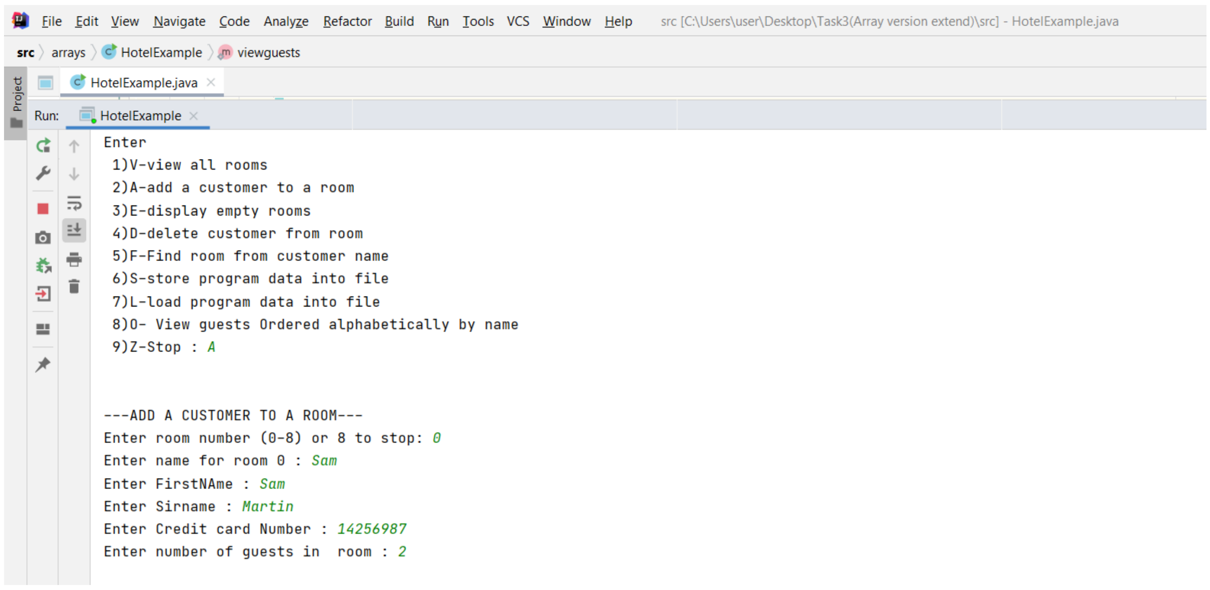
```
"C:\Program Files\Java\jdk-11.0.10\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2020.3.1\lib\idea_rt.jar=52206:C:\Program
---initilise---
Room 0 is empty
Room 1 is empty
Room 2 is empty
Room 3 is empty
Room 4 is empty
Room 5 is empty
Room 6 is empty
Room 7 is empty


Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : 
```

Figure 1 Initialize



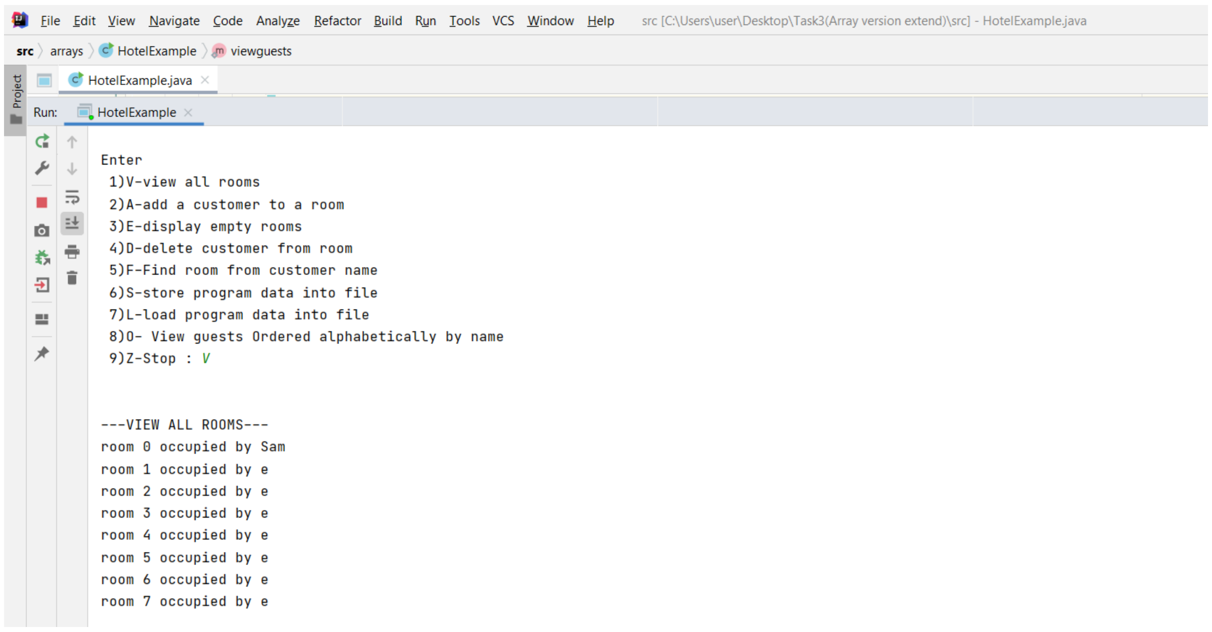
```

Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : V


---VIEW ALL ROOMS---
room 0 occupied by e
room 1 occupied by e
room 2 occupied by e
room 3 occupied by e
room 4 occupied by e
room 5 occupied by e
room 6 occupied by e
room 7 occupied by e
```

Figure 2 View All Rooms(before add the customer to the room)


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help src [C:\Users\user\Desktop\Task3(Array version extend)\src] - HotelExample.java
src arrays HotelExample viewguests
HotelExample.java
Run: HotelExample

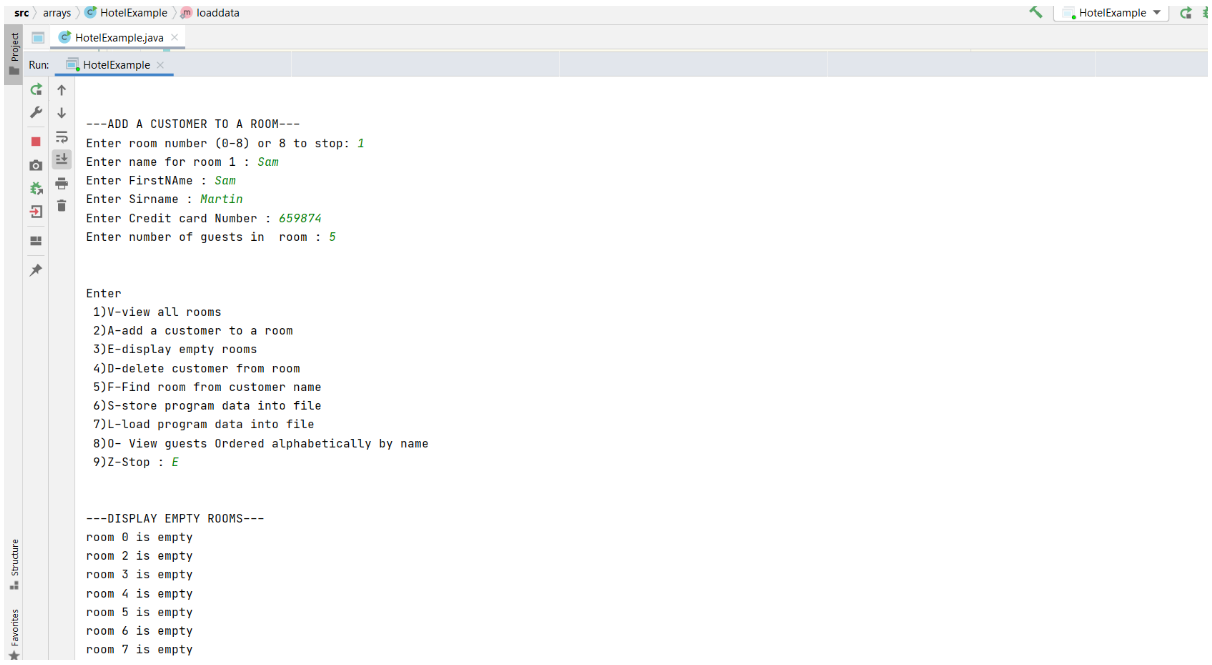
```
Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : A


---ADD A CUSTOMER TO A ROOM---
Enter room number (0-8) or 8 to stop: 0
Enter name for room 0 : Sam
Enter FirstNAme : Sam
Enter Sirname : Martin
Enter Credit card Number : 14256987
Enter number of guests in  room : 2
```

Figure 3 Add a customer to a room.


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help src [C:\Users\user\Desktop\Task3(Array version extend)\src] - HotelExample.java
src arrays HotelExample viewguests
HotelExample.java
Run: HotelExample

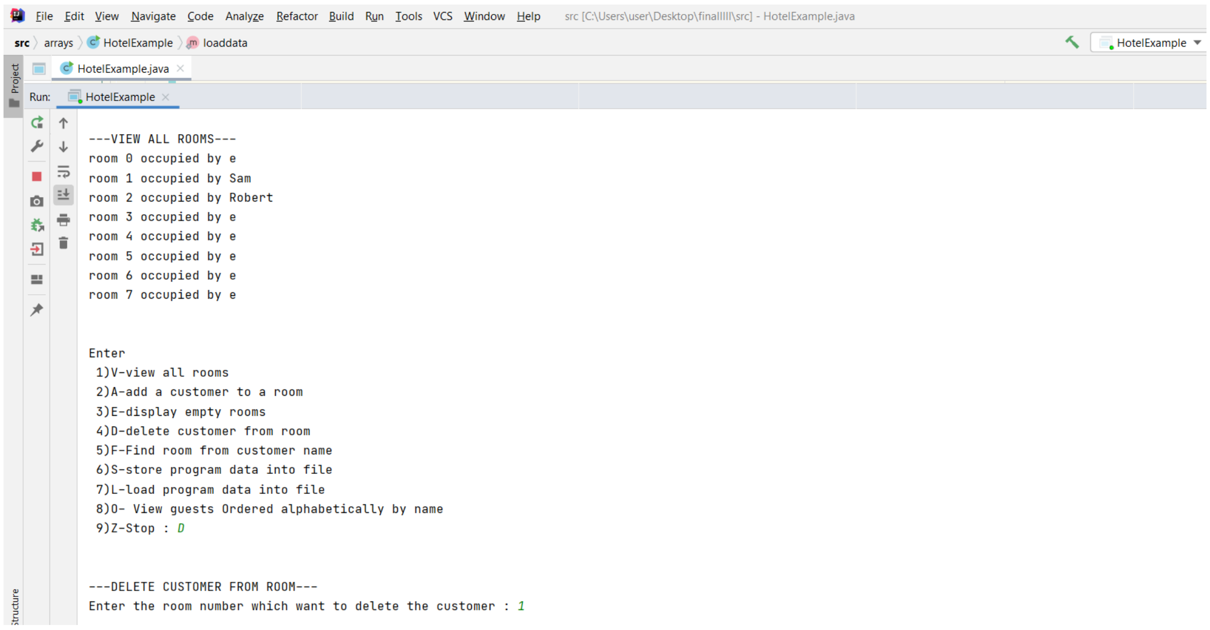
```
Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : V


---VIEW ALL ROOMS---
room 0 occupied by Sam
room 1 occupied by e
room 2 occupied by e
room 3 occupied by e
room 4 occupied by e
room 5 occupied by e
room 6 occupied by e
room 7 occupied by e
```

Figure 4 View All Rooms(after add the customer to the room)



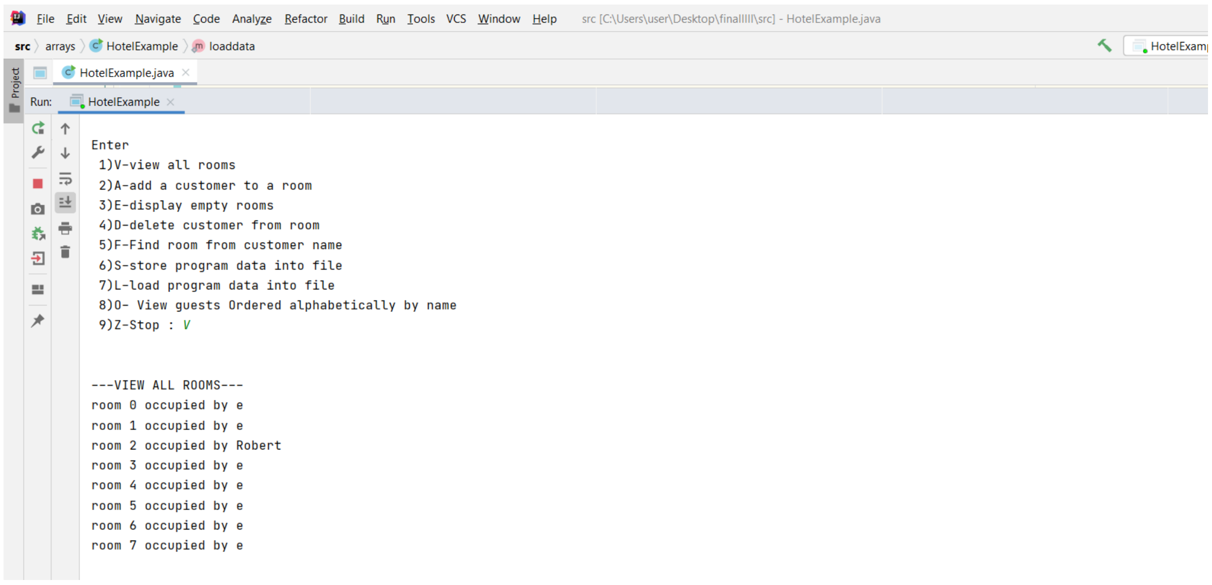
```
---ADD A CUSTOMER TO A ROOM---
Enter room number (0-8) or 8 to stop: 1
Enter name for room 1 : Sam
Enter FirstNAme : Sam
Enter Sirname : Martin
Enter Credit card Number : 659874
Enter number of guests in  room : 5


Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : E


---DISPLAY EMPTY ROOMS---
room 0 is empty
room 2 is empty
room 3 is empty
room 4 is empty
room 5 is empty
room 6 is empty
room 7 is empty
```

Figure 5 Display Empty Rooms(after add the "Sam"for room 1)


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
src [C:\Users\user\Desktop\finalllll\src] - HotelExample.java
src arrays HotelExample loaddata
HotelExample.java
Run: HotelExample
Project
Structure

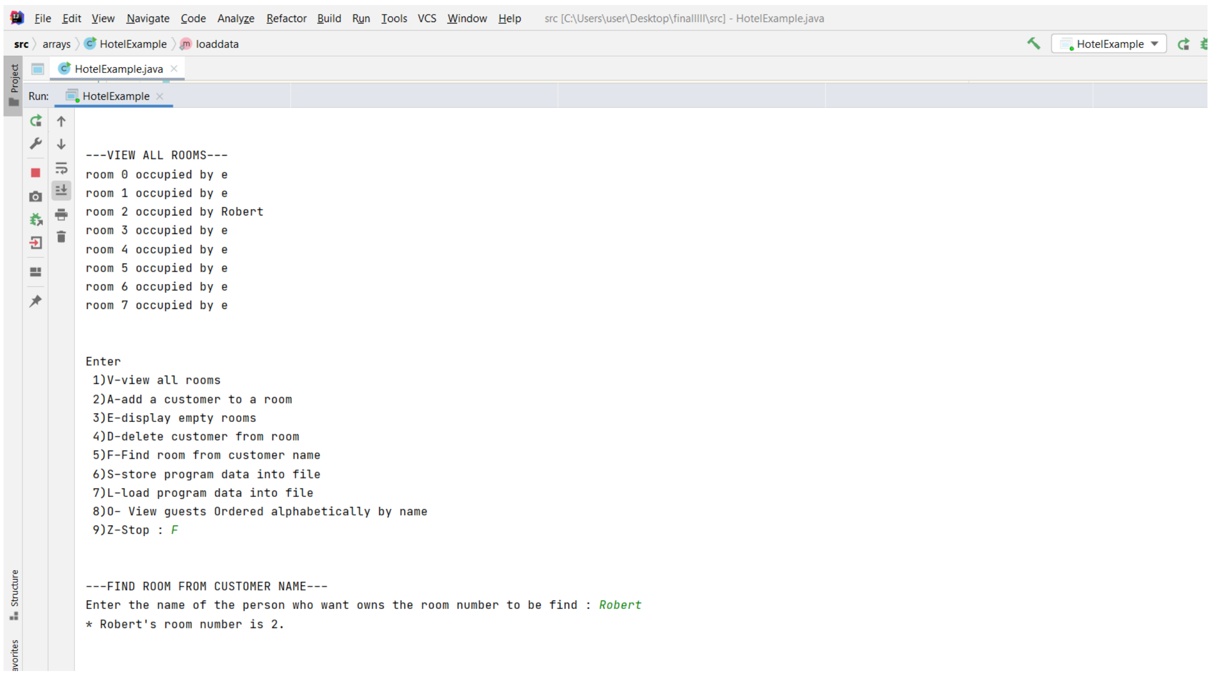
```
---VIEW ALL ROOMS---
room 0 occupied by e
room 1 occupied by Sam
room 2 occupied by Robert
room 3 occupied by e
room 4 occupied by e
room 5 occupied by e
room 6 occupied by e
room 7 occupied by e


Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : D


---DELETE CUSTOMER FROM ROOM---
Enter the room number which want to delete the customer : 1
```

Figure 6 View all rooms(before delete the "sam" from room 1)


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help src [C:\Users\user\Desktop\finalllll\src] - HotelExample.java
src arrays HotelExample loaddata
HotelExample.java
Run: HotelExample

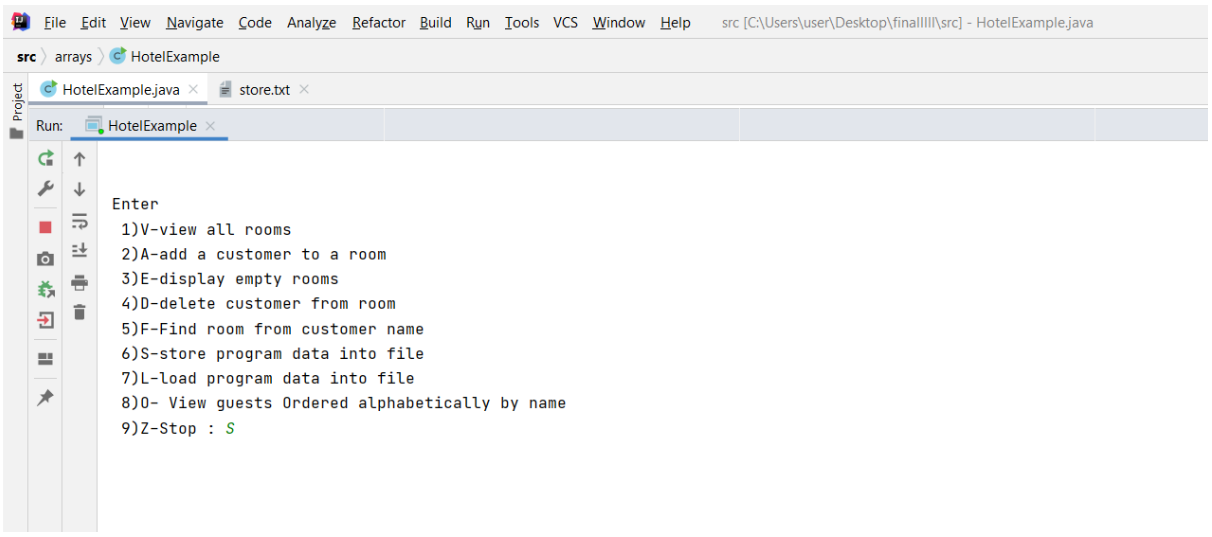
```
Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : V


---VIEW ALL ROOMS---
room 0 occupied by e
room 1 occupied by e
room 2 occupied by Robert
room 3 occupied by e
room 4 occupied by e
room 5 occupied by e
room 6 occupied by e
room 7 occupied by e
```

Figure 7 View all rooms (after delete the "Sam" from room 1)


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help src [C:\Users\user\Desktop\finalllll\src] - HotelExample.java
src arrays HotelExample loaddata
HotelExample.java
Run: HotelExample

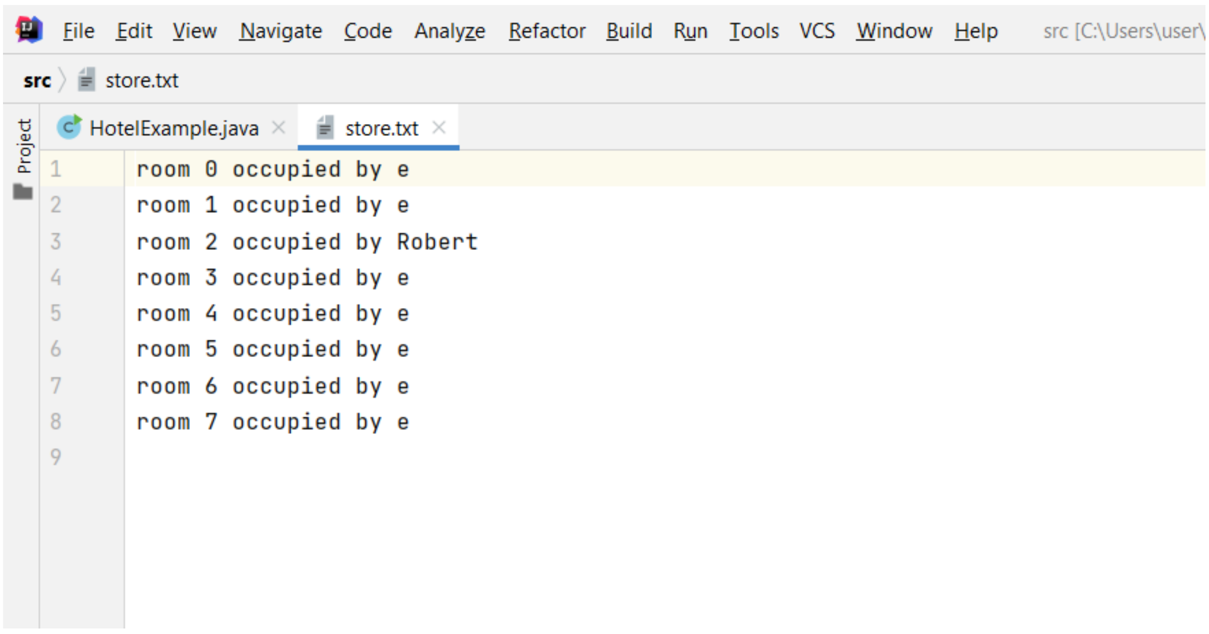
```
---VIEW ALL ROOMS---
room 0 occupied by e
room 1 occupied by e
room 2 occupied by Robert
room 3 occupied by e
room 4 occupied by e
room 5 occupied by e
room 6 occupied by e
room 7 occupied by e


Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : F


---FIND ROOM FROM CUSTOMER NAME---
Enter the name of the person who want owns the room number to be find : Robert
* Robert's room number is 2.
```

Figure 8 Find room from customer name


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
src [C:\Users\user\Desktop\finallllll\src] - HotelExample.java
src 〉 arrays 〉 HotelExample
HotelExample.java
store.txt
Run: HotelExample

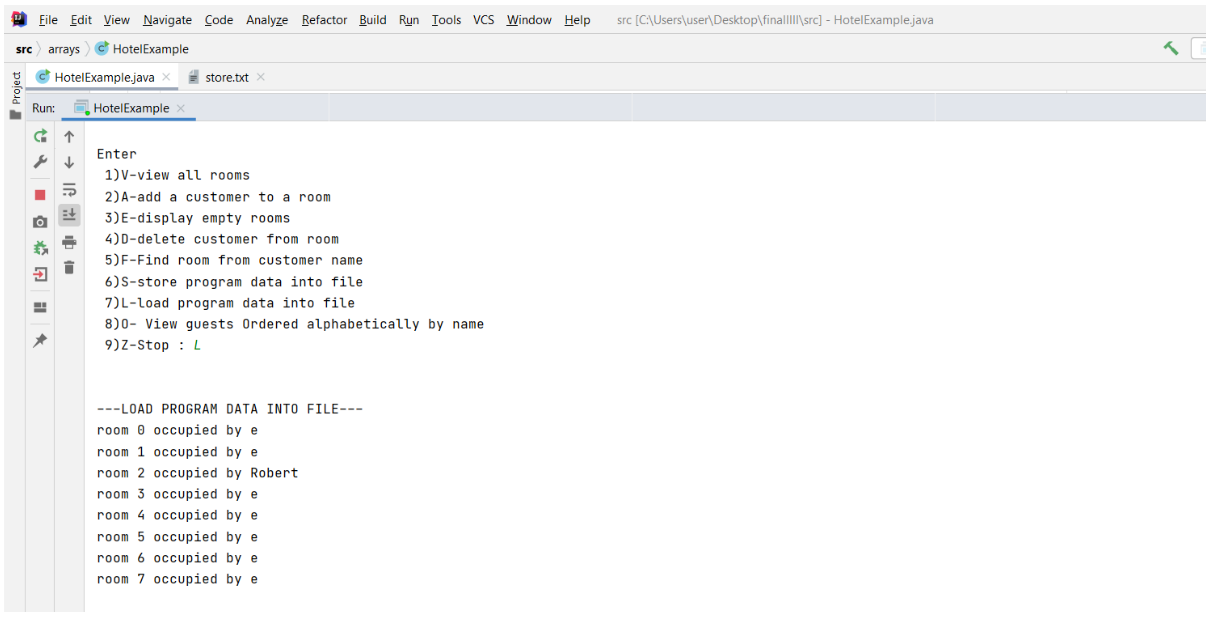
```
Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : S
```

Figure 9 Store program data into file


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
src [C:\Users\user\
src 〉 store.txt
HotelExample.java
store.txt

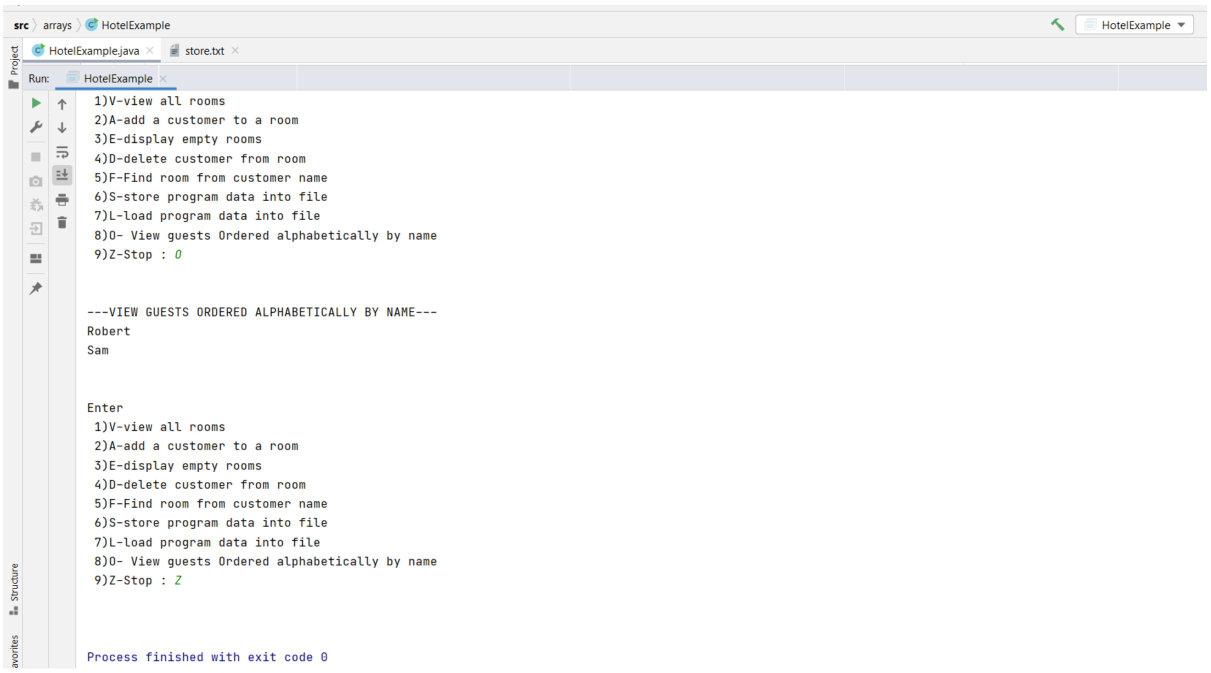
```
room 0 occupied by e
room 1 occupied by e
room 2 occupied by Robert
room 3 occupied by e
room 4 occupied by e
room 5 occupied by e
room 6 occupied by e
room 7 occupied by e
```

Figure 10 Store.txt (Text document)


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help src [C:\Users\user\Desktop\finalllll\src] - HotelExample.java
src arrays HotelExample
HotelExample.java store.txt
Run: HotelExample

```
Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : L


---LOAD PROGRAM DATA INTO FILE---
room 0 occupied by e
room 1 occupied by e
room 2 occupied by Robert
room 3 occupied by e
room 4 occupied by e
room 5 occupied by e
room 6 occupied by e
room 7 occupied by e
```

Figure 11 Load program data into file



```
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : O


---VIEW GUESTS ORDERED ALPHABETICALLY BY NAME---
Robert
Sam


Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : Z



Process finished with exit code 0
```

Figure 12 View guests ordered alphabeticaly by name and Stop

2.Task 4

Class Version

2.1Code Snippet

Main class(Task4)

```
import model.Hotel;
import model.Person;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Scanner;

public class Task4 {

   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      Hotel hotel =  new Hotel();
      hotel.initialise();

      while (true){                                   //execute the menu method.
         System.out.println("\n");
         System.out.print("Enter \n 1)V-view all rooms \n 2)A-add a customer to a room \n 3)E-display
empty rooms" +
               " \n 4)D-delete customer from room \n 5)F-Find room from customer name \n 6)S-store
program data into file \n " +
               "7)L-load program data into file \n 8)O- View guests Ordered alphabetically by name \n
9)Z-Stop : ");
         String choice= input.next();
         System.out.println("\n");
         if (choice.equals("Z")){
            break;
         }
         switch (choice) {
            case "V":
               System.out.println("---VIEW ALL ROOMS---");
               hotel.viewsallrooms();
               break;
            case "A":
               System.out.println("---ADD A CUSTOMER TO A ROOM---");
               System.out.print("Enter room number (0-8) or 8 to stop: " );
               int roomNum = input.nextInt();
               System.out.print("Enter name for room " + roomNum +" : " ) ;
               String roomName = input.next();
```

```
                System.out.print("Enter first name : " );
                String firstName = input.next();
                System.out.print("Enter surname : " );
                String surName = input.next();
                System.out.print("Enter credit card number : " );
                int creditCardNum = input.nextInt();
                System.out.print("Enter number of guests: " );
                int noOfGuests = input.nextInt();

                Person person  = new Person(roomName, firstName, surName, creditCardNum,
noOfGuests);

                hotel.addscustomertoroom(roomNum, person);
                break;
            case "E":
                System.out.println("---DISPLAY EMPTY ROOMS---");
                hotel.displayEmptyRooms();
                break;
            case "D":
                System.out.println("---DELETE CUSTOMER FROM ROOM---");
                System.out.print("Enter the room number which want to delete the customer : ");
                int roomdelete = input.nextInt();
                hotel.deleteCustomer(roomdelete);
                break;
            case "F":
                System.out.println("---FIND ROOM FROM CUSTOMER NAME---");
                System.out.print("Enter the name of the person who want owns the room number to be
find : ");
                String findperson=input.next();
                hotel.findcustomer(findperson);
                break;
            case "S":
                try {
                    hotel.storedata("store.txt");
                } catch (IOException e) {
                    System.out.println("File Not Found");
                }
                break;
            case "L":
                System.out.println("---LOAD PROGRAM DATA INTO FILE---");
                try {
                    hotel.loaddata("store.txt");
                } catch (FileNotFoundException e) {
                    System.out.println("File Not Found");
                }
                break;
            case "O":
                System.out.println("---VIEW GUESTS ORDERED ALPHABETICALLY BY NAME---");
//method for bview guests ordered alphabetically by name.
                hotel.viewguests();
                break;
            default:
                System.out.println("Invalid input");
                break;
          }
       }
    }
}
```

Hotel Class

```
package model;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

public class Hotel {
   private Room[] roomList;
   private CircularQueue<Person> waitingList;

   public Hotel(){
      roomList = new Room[8];
      waitingList = new CircularQueue<>(8);
   }

   public Hotel(int noOfRooms, int sizeOfWaitingList){
      roomList = new Room[noOfRooms];
      waitingList = new CircularQueue<>(sizeOfWaitingList);
   }
   public void initialise() {
      System.out.println( "---initialise---");
      for (int x = 0; x < 8; x++ ) {
         roomList[x]=(new Room("e"));
         System.out.println("Room " + x + " is empty");
      }
   }
```

```
  public void viewsallrooms(){                                    //method for view all rooms.
    for (int x = 0; x < 8; x++ ){
      System.out.println("room " + x + " occupied by " + roomList[x].getName());
    }
  }

  public void displayEmptyRooms() {                                    //method for display empty rooms.
    for (int x = 0; x < 8; x++ ){
      if (roomList[x].getName().equals("e")){
        System.out.println("room " + x + " is empty");
      }
    }
  }

  public void addscustomertoroom(int roomNum, Person
person){                                    //method for add a customer to a room.
    Room r = roomList[roomNum];
    if (r.getName().equals("e")){
      r.setName(person.getName());
      r.setPayingCustomer(person);
    } else {
      waitingList.enqueue(person);
    }

  }

  public void deleteCustomer(int roomdelete){                                    //method for delete
customer from room.
    if (!waitingList.isEmpty()){
      Person p = waitingList.dequeue();
      roomList[roomdelete].setName(p.getName());
      roomList[roomdelete].setPayingCustomer(p);
    } else {
      roomList[roomdelete].setName("e");
    }
  }

  public void findcustomer(String findperson){                                    //method for find room
from customer name.
    for(int x=0;x<8;x++){
      if(roomList[x].getName().equals(findperson)){
        System.out.println("* "+findperson+"'s room number is " +x+".");
      }
    }
  }

  public void storedata(String filename) throws IOException {                              //method for
store the data into file.
```

```
        FileWriter fw=new FileWriter (filename);
        for (int x = 0; x < 8; x++ ){
            fw.write("room " + x + " occupied by " + roomList[x].getName() + "\n");
        }
        fw.close();
    }

    public void loaddata(String fileName) throws FileNotFoundException {                     //method
for load data into file.
        File f=new File(fileName);
        Scanner sc=new Scanner(f);
        while ((sc.hasNext())){
            System.out.println(sc.nextLine());
        }
    }
    public void viewguests(){   //method for view the guests ordered alphabetically by name.

        ArrayList<Room> sortedRoomNames = new ArrayList<Room>(Arrays.asList(roomList));
        for(int i = 0; i< sortedRoomNames.size()-1; i++)
        {
            for (int j = i+1; j< sortedRoomNames.size()-1; j++)
            {
                //compares each elements of the array to all the remaining elements
                if(sortedRoomNames.get(i).getName().compareTo(sortedRoomNames.get(j).getName())>0)
                {   //swapping array elements
                    Room temp = sortedRoomNames.get(i);
                    sortedRoomNames.set(i,sortedRoomNames.get(j));
                    sortedRoomNames.set(j,temp);
                }
            }
        }

        for (int x = 0; x < 8; x++ ){
            if (!(sortedRoomNames.get(x).getName()=="e")){
                System.out.println(sortedRoomNames.get(x).getName());
            }

        }
    }

}
```

Person class

```
package model;
public class Person {
   private String name;
   private String firstName;
   private String  surName;
   private int creditCardNo;
   private int noOfGuests;

   public Person(String name, String firstName, String surName, int creditCardNo, int noOfGuests) {
      this.name = name;
      this.firstName = firstName;
      this.surName = surName;
      this.creditCardNo = creditCardNo;
      this.noOfGuests = noOfGuests;
   }

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   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 int getCreditCardNo() {
      return creditCardNo;
   }
```

```
    public void setCreditCardNo(int creditCardNo) {
        this.creditCardNo = creditCardNo;
    }

    public int getNoOfGuests() {
        return noOfGuests;
    }

    public void setNoOfGuests(int noOfGuests) {
        this.noOfGuests = noOfGuests;
    }
}
```

Circular Queue class

```
package model;
import exception.QueueEmptyException;
import exception.QueueFullException;
import java.util.Arrays;
public class CircularQueue<E> {

    private int currentSize; //Current Circular Queue Size
    private E[] circularQueueElements;
    private int maxSize; //Circular Queue maximum size

    private int rear;//rear position of Circular queue(new element enqueued at rear).
    private int front; //front position of Circular queue(element will be dequeued from front).

    public CircularQueue(int maxSize) {
        this.maxSize = maxSize;
        circularQueueElements = (E[]) new Object[this.maxSize];
        currentSize = 0;
        front = -1;
        rear = -1;
    }

    /**
     * Enqueue elements to rear.
     */
    public void enqueue(E item) throws QueueFullException {
        if (isFull()) {
            throw new QueueFullException("Circular Queue is full. Element cannot be added");
        }
        else {
            rear = (rear + 1) % circularQueueElements.length;
            circularQueueElements[rear] = item;
            currentSize++;

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

    /**
     * Dequeue element from Front.
     */
    public E dequeue() throws QueueEmptyException {
        E deQueuedElement;
        if (isEmpty()) {
            throw new QueueEmptyException("Circular Queue is empty. Element cannot be retrieved");
        }
        else {
            deQueuedElement = circularQueueElements[front];
            circularQueueElements[front] = null;
            front = (front + 1) % circularQueueElements.length;
            currentSize--;
        }
        return deQueuedElement;
    }
    /**
     * Check if queue is full.
     */
    public boolean isFull() {
        return (currentSize == circularQueueElements.length);
    }

    /**
     * Check if Queue is empty.
     */
    public boolean isEmpty() {
        return (currentSize == 0);
    }
    @Override
    public String toString() {
        return "model.CircularQueue [" + Arrays.toString(circularQueueElements) + "]";
    }
}
```

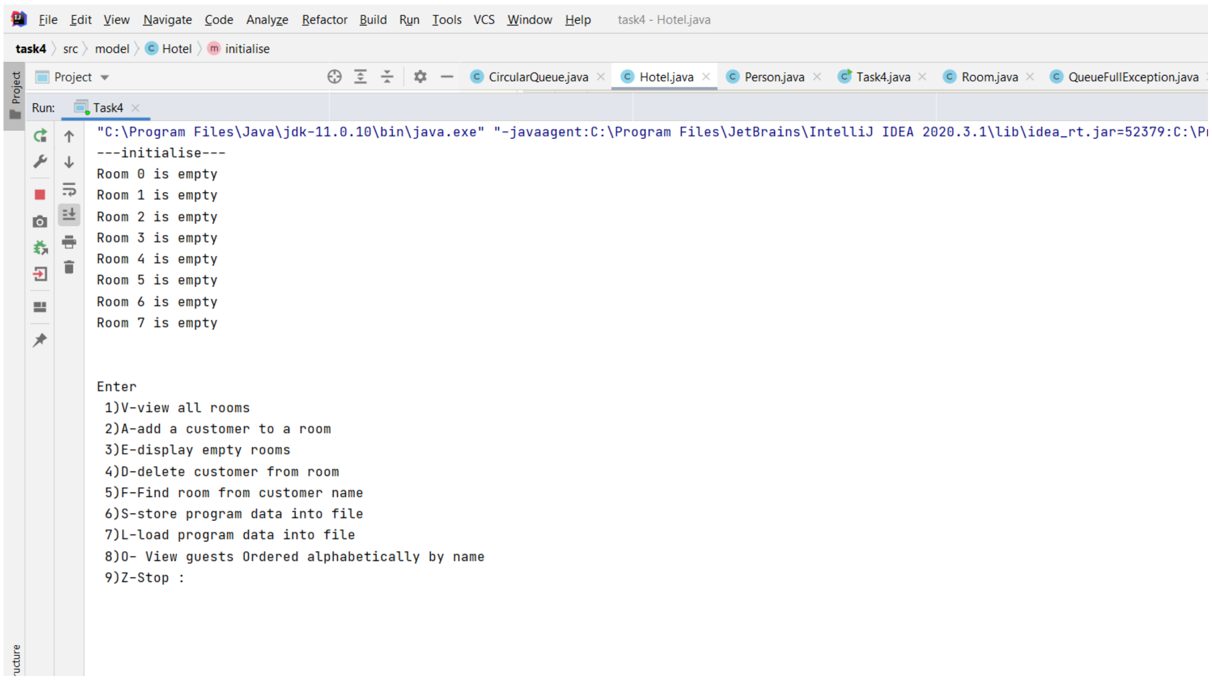
QueueFullException class

```
package exception;

public class QueueFullException extends RuntimeException {

    private static final long serialVersionUID = 1L;

    public QueueFullException() {
        super();
    }

    public QueueFullException(String message) {
        super(message);
    }

}
```

QueueEmptyException

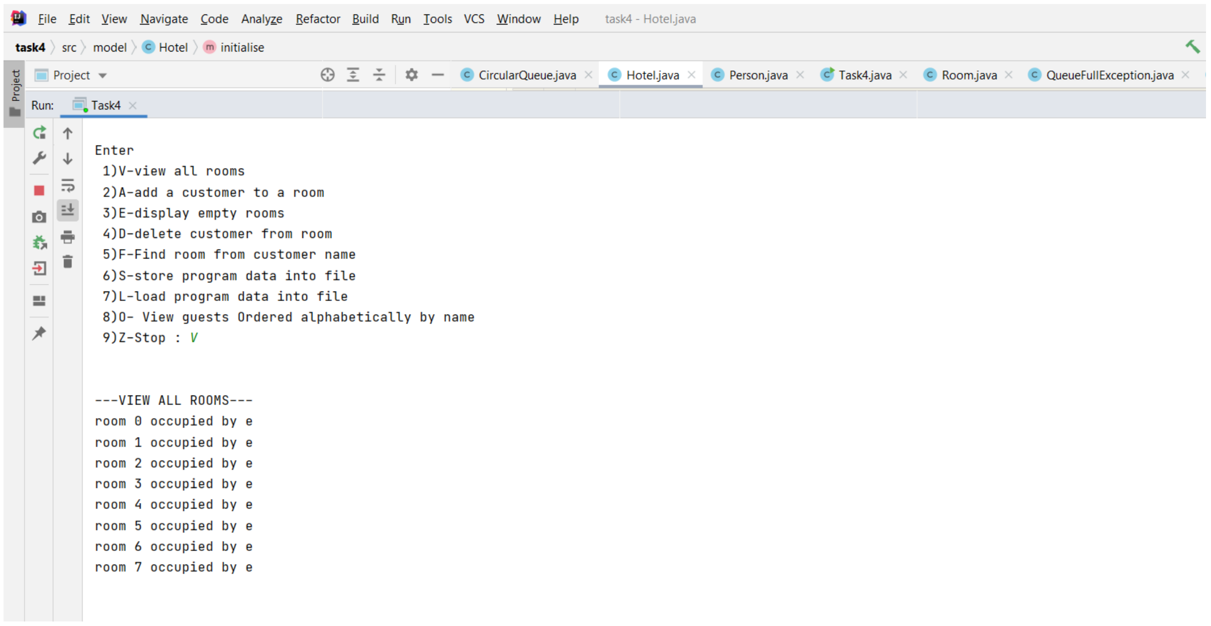
```
package exception;

public class QueueEmptyException extends RuntimeException {

    private static final long serialVersionUID = 1L;

    public QueueEmptyException() {
        super();
    }

    public QueueEmptyException(String message) {
        super(message);
    }

}
```

2.2 Screenshots Of Code Snippet


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help task4 - Hotel.java
task4 src model Hotel initialise
Project CircularQueue.java Hotel.java Person.java Task4.java Room.java QueueFullException.java
Run: Task4

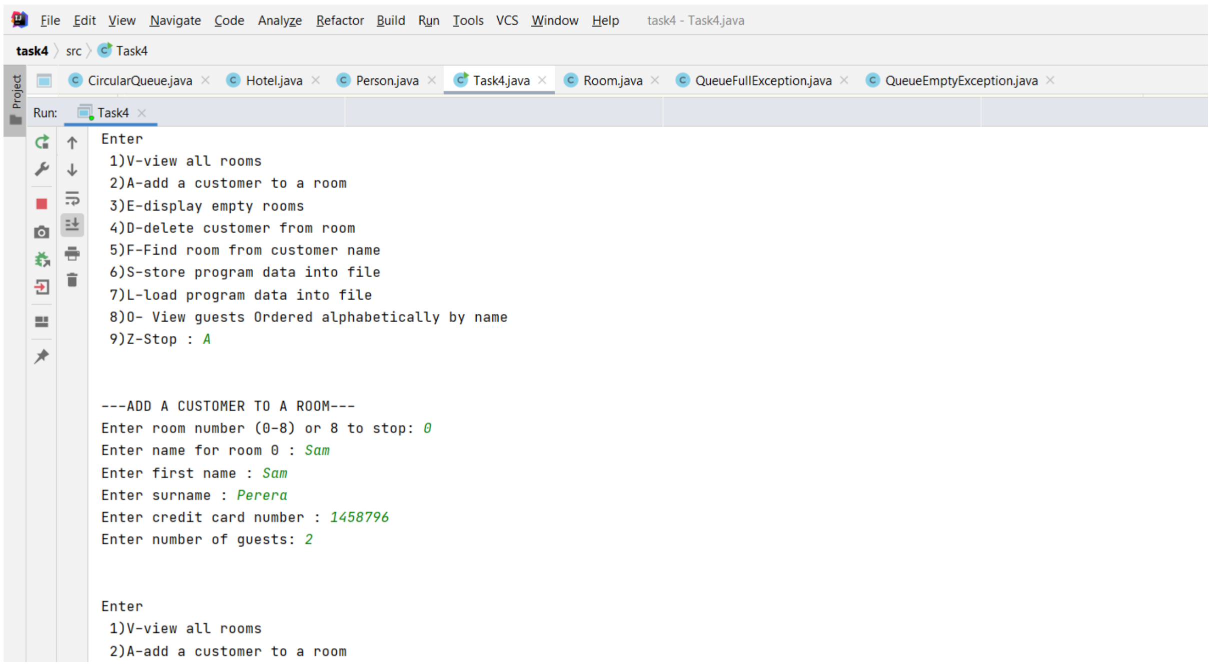
```
"C:\Program Files\Java\jdk-11.0.10\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2020.3.1\lib\idea_rt.jar=52379:C:\Pr
---initialise---
Room 0 is empty
Room 1 is empty
Room 2 is empty
Room 3 is empty
Room 4 is empty
Room 5 is empty
Room 6 is empty
Room 7 is empty


Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop :
```

Figure 13 Initialize the program


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help task4 - Hotel.java
task4 src model Hotel initialise
Project CircularQueue.java Hotel.java Person.java Task4.java Room.java QueueFullException.java
Run: Task4

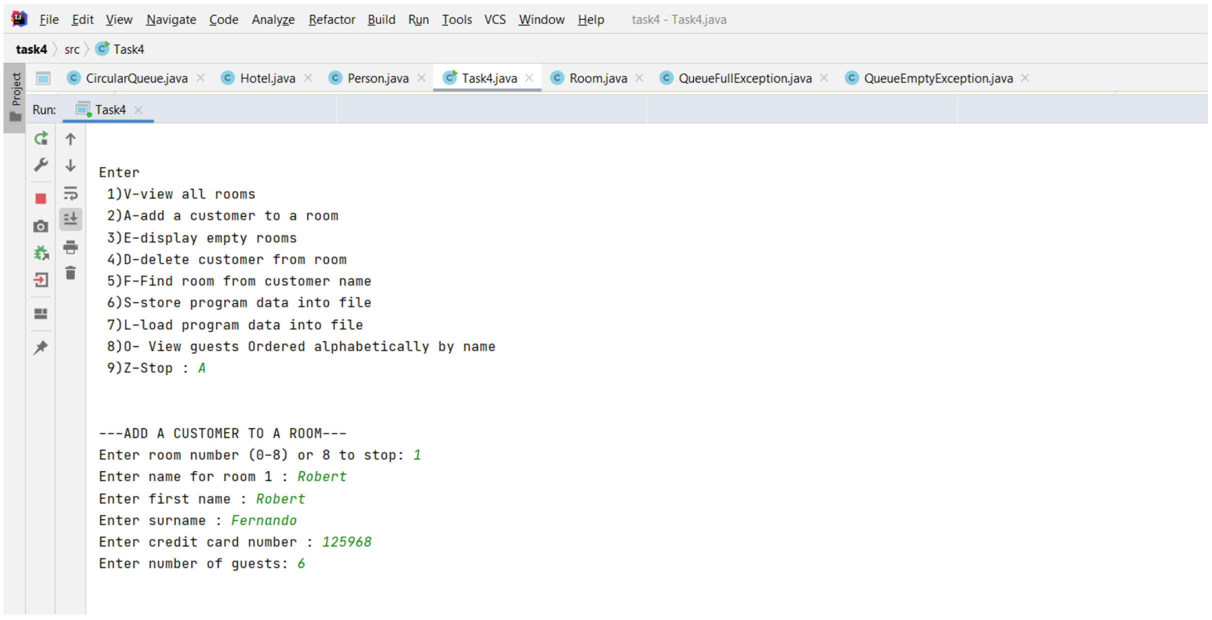
```
Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : V


---VIEW ALL ROOMS---
room 0 occupied by e
room 1 occupied by e
room 2 occupied by e
room 3 occupied by e
room 4 occupied by e
room 5 occupied by e
room 6 occupied by e
room 7 occupied by e
```

Figure 14 View all rooms(before add a customer to a room)


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help task4 - Task4.java
task4 src Task4
CircularQueue.java Hotel.java Person.java Task4.java Room.java QueueFullException.java QueueEmptyException.java
Run: Task4

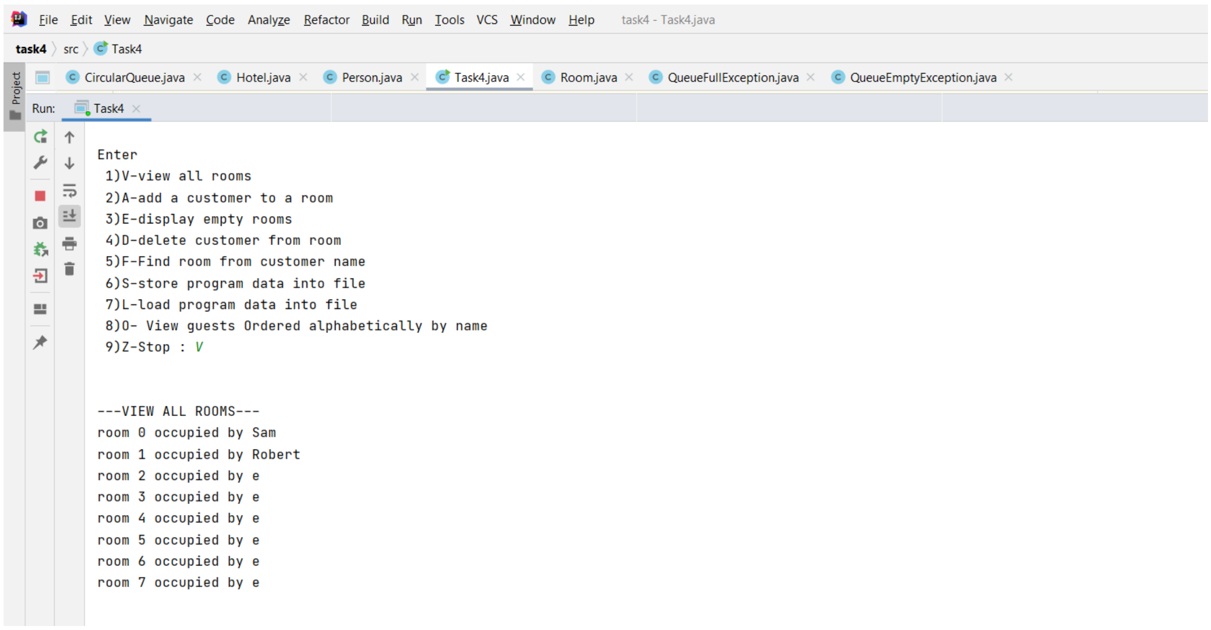
```
Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : A


---ADD A CUSTOMER TO A ROOM---
Enter room number (0-8) or 8 to stop: 0
Enter name for room 0 : Sam
Enter first name : Sam
Enter surname : Perera
Enter credit card number : 1458796
Enter number of guests: 2


Enter
 1)V-view all rooms
 2)A-add a customer to a room
```

Figure 15 Add a customer to a room


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help task4 - Task4.java
task4 src Task4
CircularQueue.java Hotel.java Person.java Task4.java Room.java QueueFullException.java QueueEmptyException.java
Run: Task4

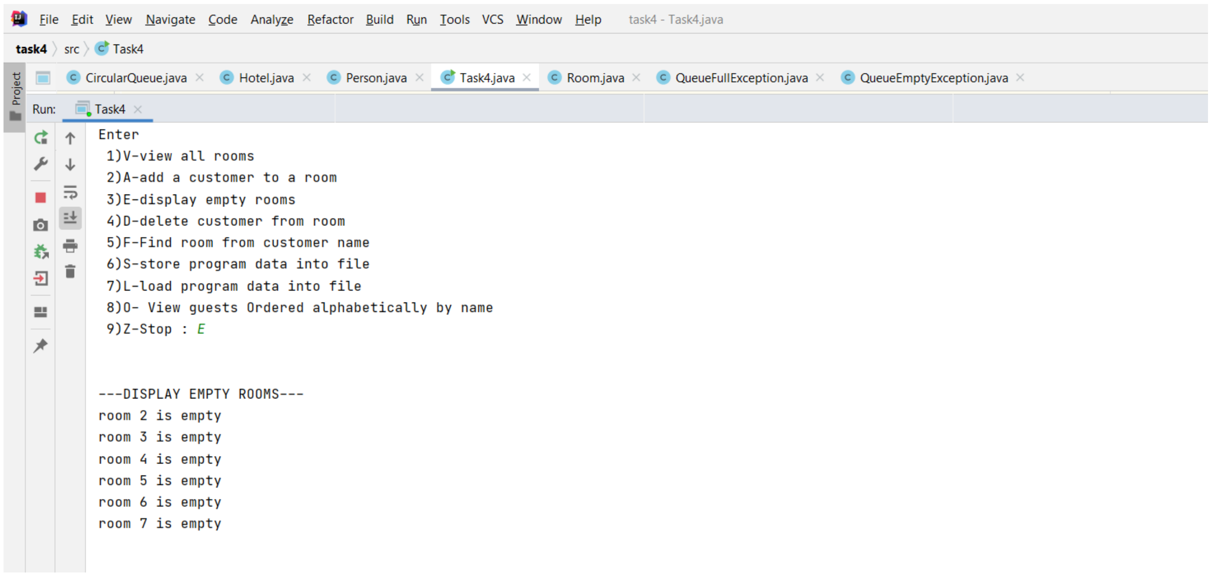
```
Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : A


---ADD A CUSTOMER TO A ROOM---
Enter room number (0-8) or 8 to stop: 1
Enter name for room 1 : Robert
Enter first name : Robert
Enter surname : Fernando
Enter credit card number : 125968
Enter number of guests: 6
```

Figure 16 Add a customer to a room(2)


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help task4 - Task4.java
task4 src Task4
CircularQueue.java Hotel.java Person.java Task4.java Room.java QueueFullException.java QueueEmptyException.java
Run: Task4

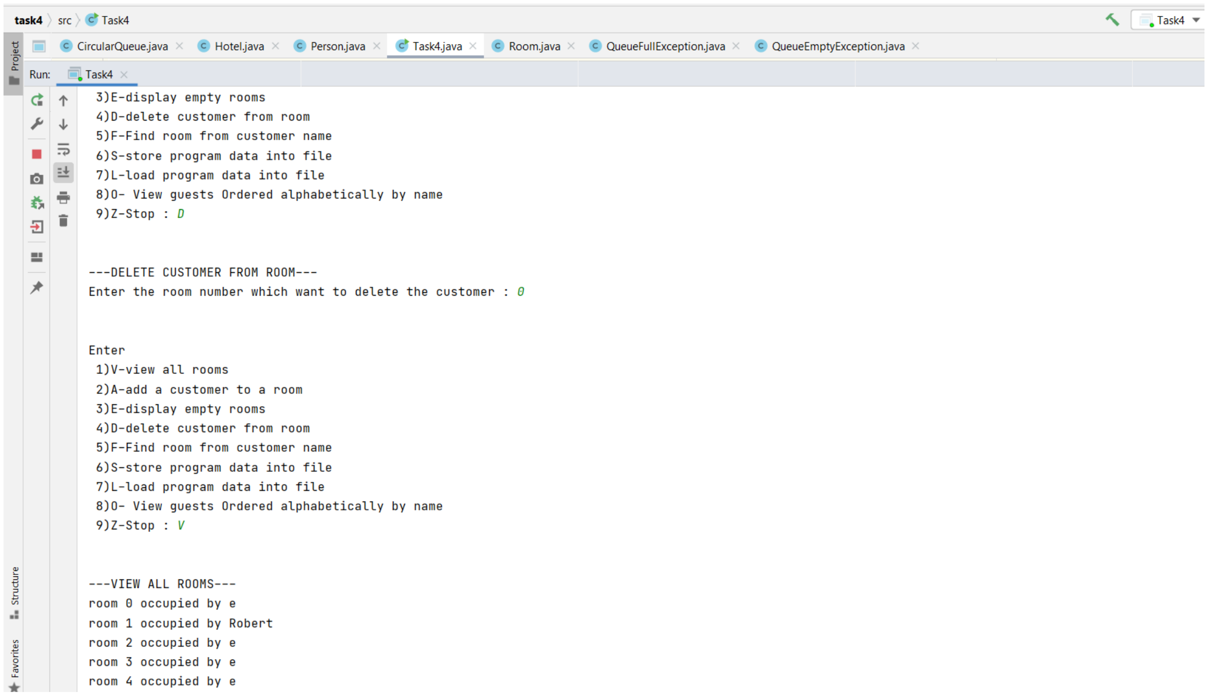
```
Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : V


---VIEW ALL ROOMS---
room 0 occupied by Sam
room 1 occupied by Robert
room 2 occupied by e
room 3 occupied by e
room 4 occupied by e
room 5 occupied by e
room 6 occupied by e
room 7 occupied by e
```

Figure 17 View all Rooms(after add the customers to the rooms)


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help task4 - Task4.java
task4 src Task4
CircularQueue.java Hotel.java Person.java Task4.java Room.java QueueFullException.java QueueEmptyException.java
Run: Task4

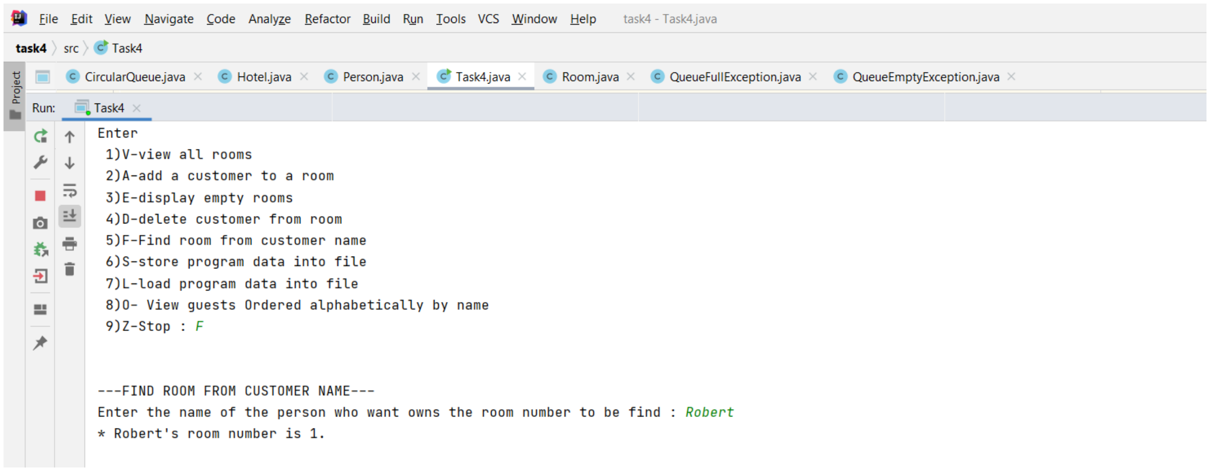
```
Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : E


---DISPLAY EMPTY ROOMS---
room 2 is empty
room 3 is empty
room 4 is empty
room 5 is empty
room 6 is empty
room 7 is empty
```

Figure 18 Display empty rooms



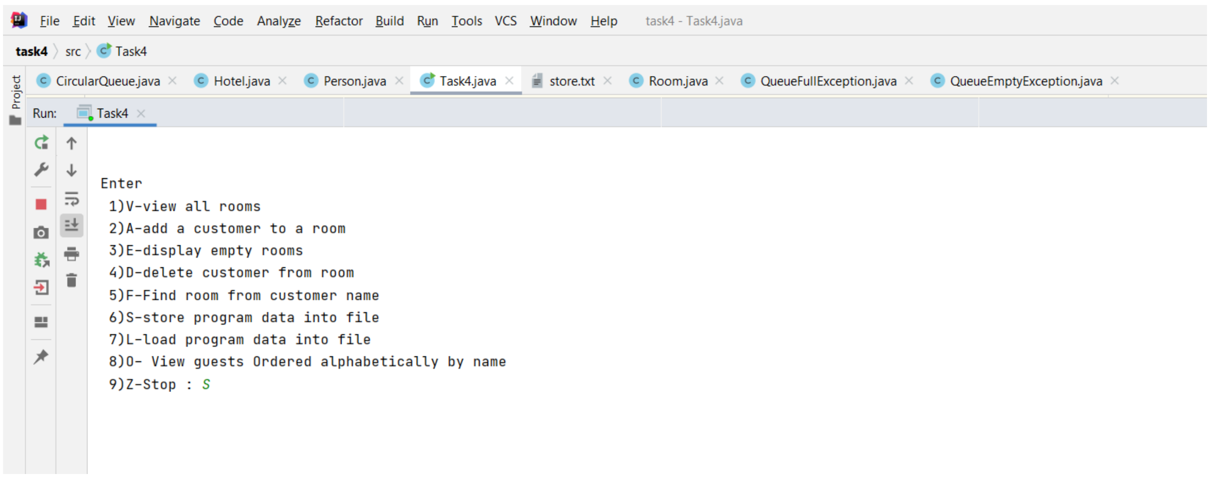
```
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : D


---DELETE CUSTOMER FROM ROOM---
Enter the room number which want to delete the customer : 0


Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : V


---VIEW ALL ROOMS---
room 0 occupied by e
room 1 occupied by Robert
room 2 occupied by e
room 3 occupied by e
room 4 occupied by e
```

Figure 19 Delete customer from room and view all rooms.



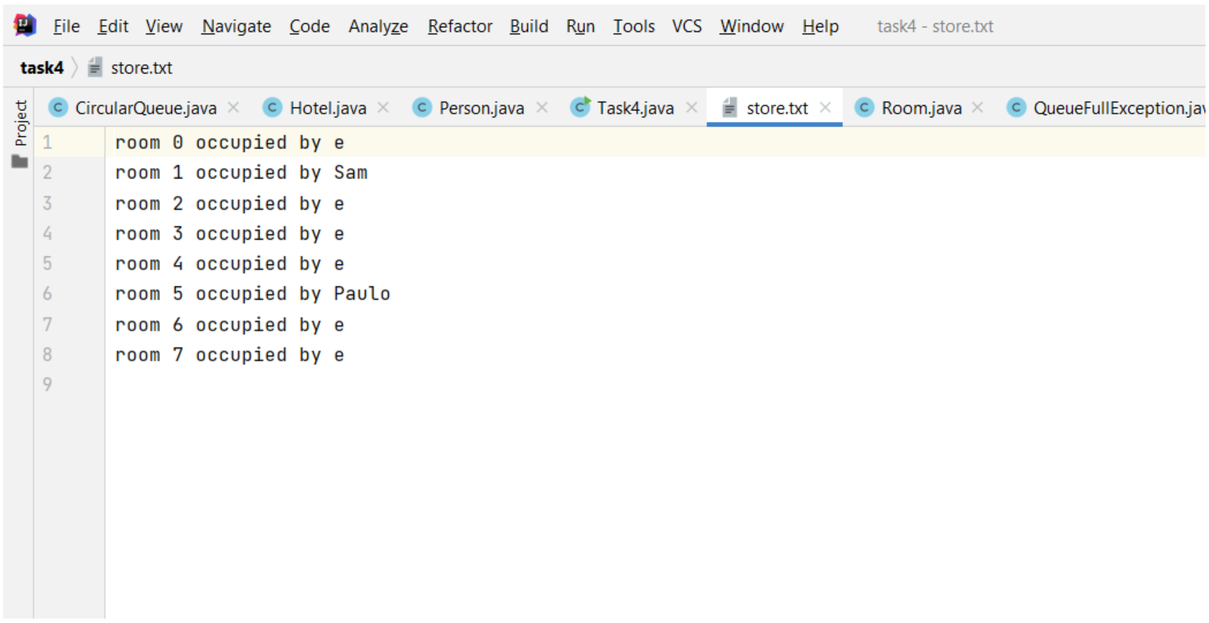
```
Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : F


---FIND ROOM FROM CUSTOMER NAME---
Enter the name of the person who want owns the room number to be find : Robert
* Robert's room number is 1.
```

Figure 20 Find room from customer name



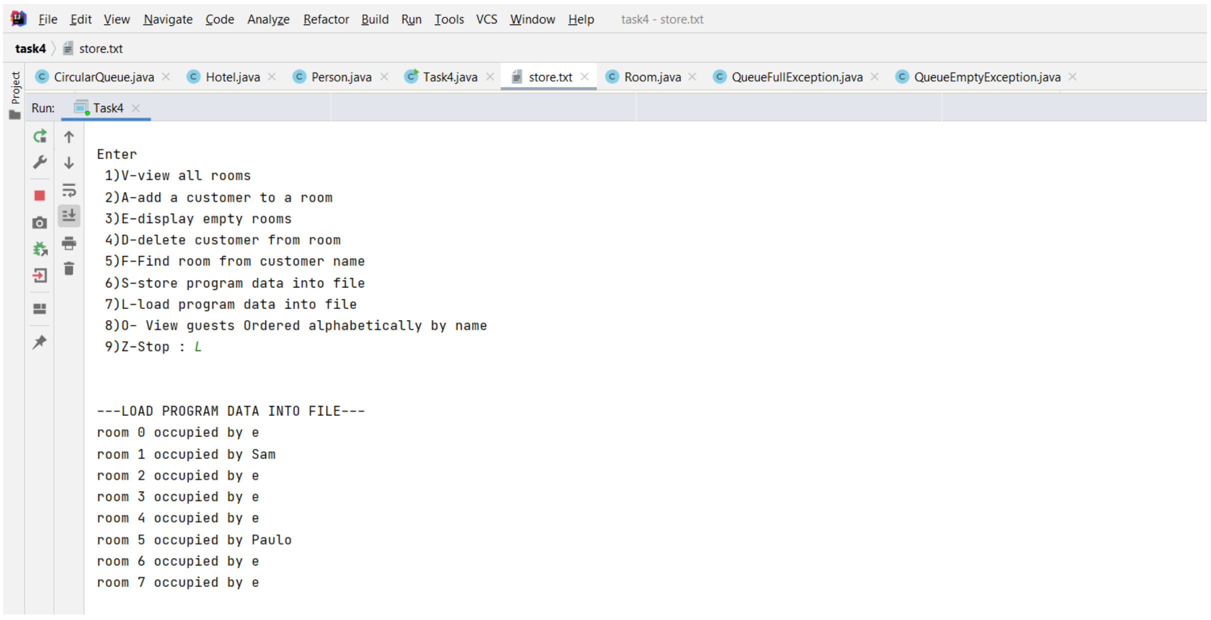
```
Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : S
```

Figure 21 store program data into file



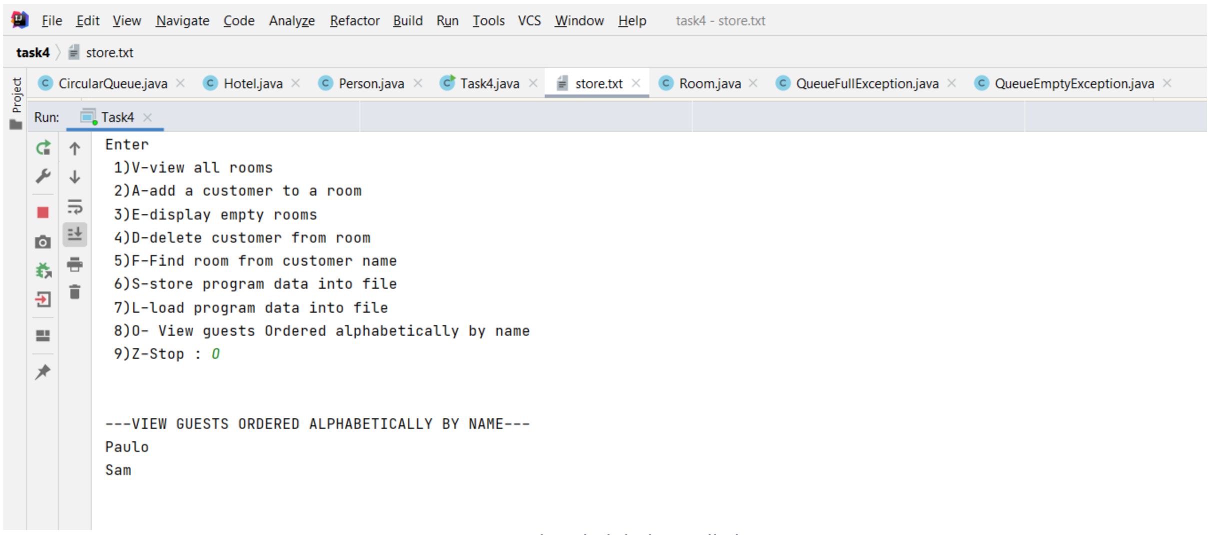
```
room 0 occupied by e
room 1 occupied by Sam
room 2 occupied by e
room 3 occupied by e
room 4 occupied by e
room 5 occupied by Paulo
room 6 occupied by e
room 7 occupied by e
```

Figure 22 Store program data into file(2)


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help task4 - store.txt
task4 store.txt
CircularQueue.java Hotel.java Person.java Task4.java store.txt Room.java QueueFullException.java QueueEmptyException.java
Run: Task4

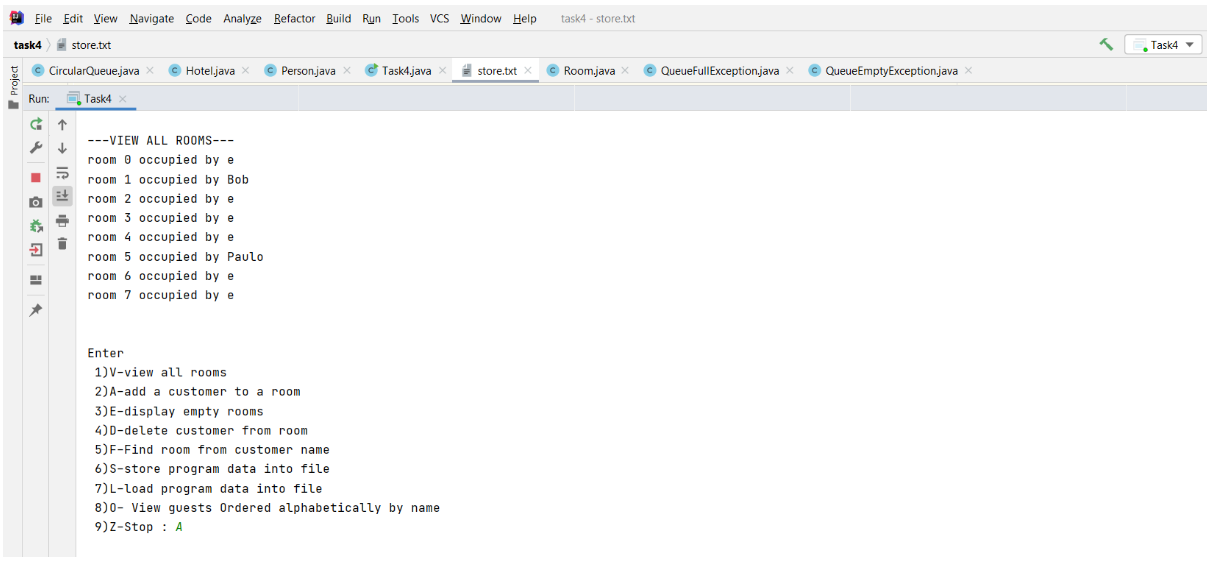
```
Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : L


---LOAD PROGRAM DATA INTO FILE---
room 0 occupied by e
room 1 occupied by Sam
room 2 occupied by e
room 3 occupied by e
room 4 occupied by e
room 5 occupied by Paulo
room 6 occupied by e
room 7 occupied by e
```

Figure 23 Load program data into file


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help task4 - store.txt
task4 store.txt
CircularQueue.java Hotel.java Person.java Task4.java store.txt Room.java QueueFullException.java QueueEmptyException.java
Run: Task4

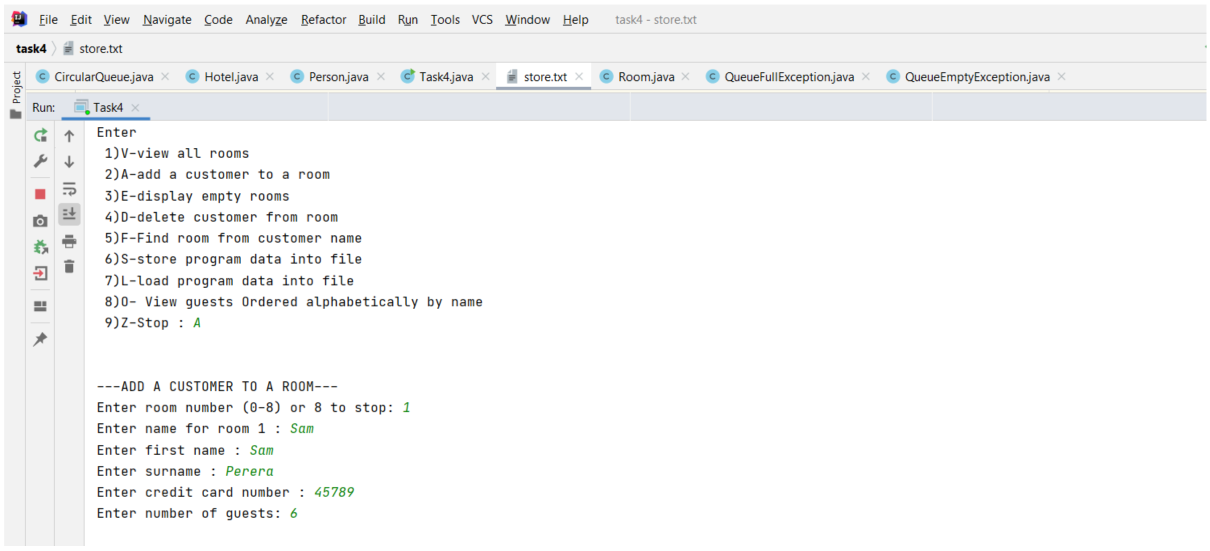
```
Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : O


---VIEW GUESTS ORDERED ALPHABETICALLY BY NAME---
Paulo
Sam
```

Figure 24 View guests ordered alphebetically by name


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help task4 - store.txt
task4 store.txt
CircularQueue.java Hotel.java Person.java Task4.java store.txt Room.java QueueFullException.java QueueEmptyException.java
Run: Task4

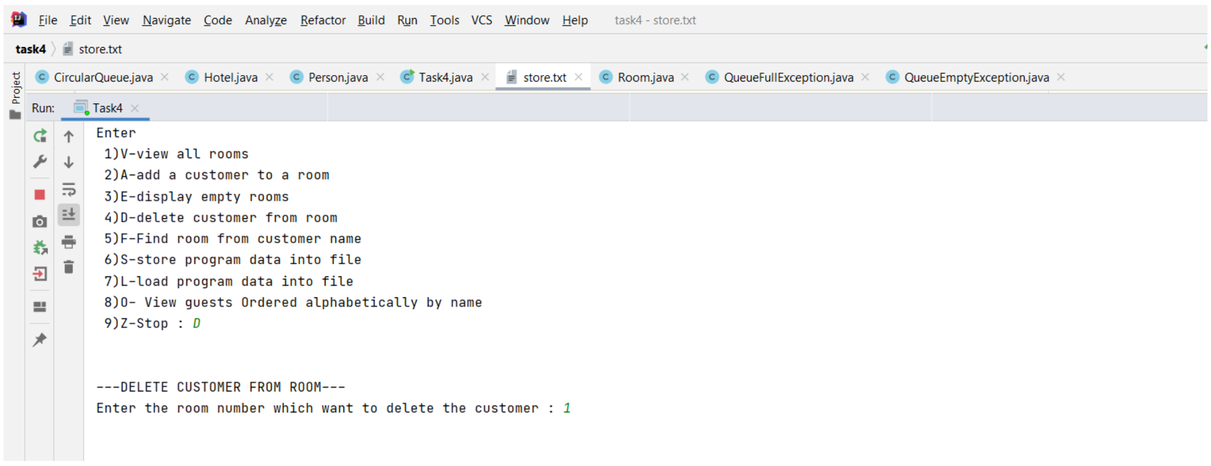
```
---VIEW ALL ROOMS---
room 0 occupied by e
room 1 occupied by Bob
room 2 occupied by e
room 3 occupied by e
room 4 occupied by e
room 5 occupied by Paulo
room 6 occupied by e
room 7 occupied by e


Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : A
```

Figure 25 View all rooms(room1-Bob & room5-Paulo)


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help task4 - store.txt
task4 store.txt
CircularQueue.java Hotel.java Person.java Task4.java store.txt Room.java QueueFullException.java QueueEmptyException.java
Run: Task4

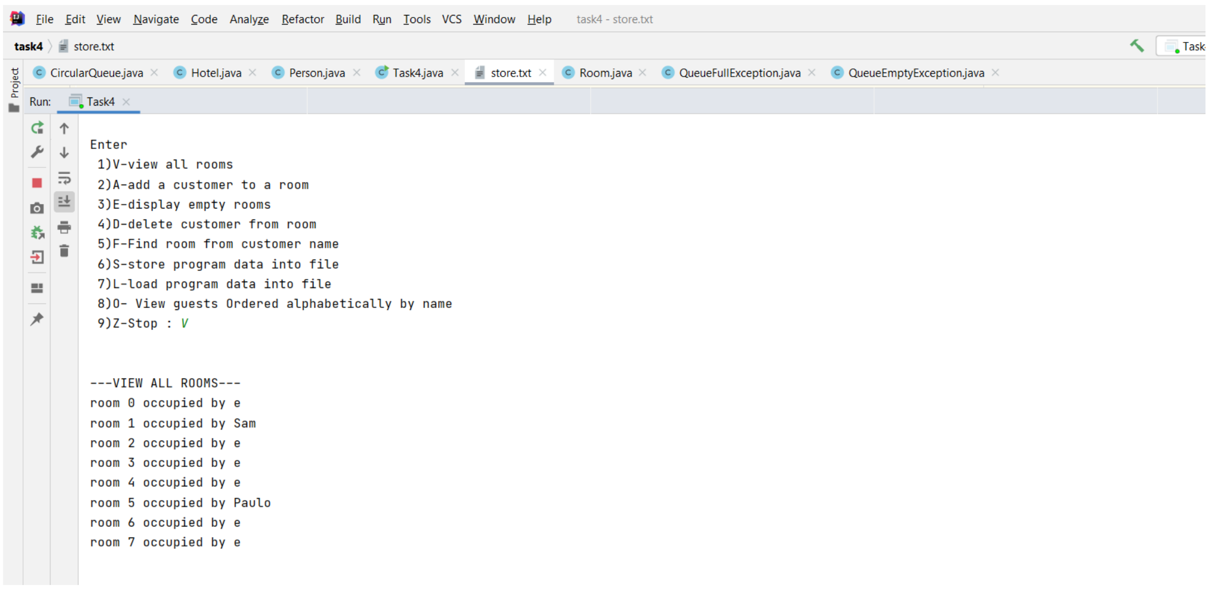
```
Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : A


---ADD A CUSTOMER TO A ROOM---
Enter room number (0-8) or 8 to stop: 1
Enter name for room 1 : Sam
Enter first name : Sam
Enter surname : Perera
Enter credit card number : 45789
Enter number of guests: 6
```

Figure 26 Add a customer to a room(again add Sam to room number 1)


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help task4 - store.txt
task4 store.txt
CircularQueue.java Hotel.java Person.java Task4.java store.txt Room.java QueueFullException.java QueueEmptyException.java
Run: Task4

```
Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : D


---DELETE CUSTOMER FROM ROOM---
Enter the room number which want to delete the customer : 1
```

Figure 27 Delete customer from room(delete Bob from room1)


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help task4 - store.txt
task4 store.txt
Task
CircularQueue.java Hotel.java Person.java Task4.java store.txt Room.java QueueFullException.java QueueEmptyException.java
Run: Task4

```
Enter
 1)V-view all rooms
 2)A-add a customer to a room
 3)E-display empty rooms
 4)D-delete customer from room
 5)F-Find room from customer name
 6)S-store program data into file
 7)L-load program data into file
 8)O- View guests Ordered alphabetically by name
 9)Z-Stop : V


---VIEW ALL ROOMS---
room 0 occupied by e
room 1 occupied by Sam
room 2 occupied by e
room 3 occupied by e
room 4 occupied by e
room 5 occupied by Paulo
room 6 occupied by e
room 7 occupied by e
```

Figure 28 View all rooms(since deleted the Bob from room1 Sam automatically add to the room 1-circularqueue)