Software Development II

Coursework Report 2021/2022

Name : K. A. Rashmi Sewmini Karunarathne

Student ID : 20210425

UoW ID : 18671441

Contents

Task 01 – Source Code

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.PrintStream;
import java.util.Scanner;

public class Task1 {
    private static boolean S_menu = true;


    public static void main(String[] args) throws IOException {
        Scanner input = new Scanner(System.in);
        String cabinName;
        int cabinNum = 0;
        String[] ship = new String[12];
        int[] guestNo = new int[12];
        String[] fName = new String[12];
        String[] SName = new String[12];
        long[] expense = new long[12];

        initialise(ship);


// menu

        while (S_menu){
            System.out.println("        ........WELCOME TO THE CRUISE
SHIP........        ");
            System.out.println("                ...CRUISE SHIP MENU...
");
            System.out.println("Enter 'A' to add a customer to a cabin
:");
            System.out.println("Enter 'V' to view all cabin
:");
            System.out.println("Enter 'E' to Display Empty cabin
:");
            System.out.println("Enter 'D' to Delete customer from cabin
:");
            System.out.println("Enter 'F' to Find cabin from customer name
:");
            System.out.println("Enter 'S' to Store program data into file
:");
            System.out.println("Enter 'L' to Load program data from file
:");
            System.out.println("Enter 'O' to View guests Ordered
alphabetically by name :");

            String selection = input.next();

            switch (selection){
                case "A":
                    add_customer(ship, guestNo, fName, SName, expense);
                    break;
```

```java
                case "V":
                    view_all_cabin(ship, guestNo, fName, SName, expense);
                    break;

                case "E":
                    empty_cabins(ship);
                    break;

                case "D":
                    del_cus_cabin(ship);
                    break;

                case "F":
                    find_cabin(ship);
                    break;

                case "S":
                    info_file(ship, guestNo, fName, SName, expense);
                    break;

                case "L":
                    load_prog_data(ship, guestNo, fName, SName, expense);
                    break;

                case "O":
                    order_name(ship);
                    break;
            }
            boolean choice = true;

            while (choice){
                System.out.println("Enter 1 to Continue or 2 to Exit: ");
                int com = input.nextInt();

                if (com == 1){
                    S_menu = true;
                    choice = false;
                }else if(com == 2){
                    S_menu = false;
                    choice = false;
                }else{
                    System.out.println("Invalid Input...");
                }

            }

        }

    }


    // initialising

    public static void initialise( String shipRef[] ) {
        for (int x = 0; x < 12; x++ ) shipRef[x] = "e";
        System.out.println( "initilise ");
    }

    // adding customers to cabins

    public static void add_customer(String[] ship, int[] guestNo, String[]
fName, String[] SName, long[] expense){
        Scanner input = new Scanner(System.in);
        System.out.println("Enter cabin Number 0 to 11: ");
        int cabin_num = input.nextInt();

        System.out.println("Enter Name for cabin Number " + cabin_num + ":
");
        String cabinName = input.next();
        ship[cabin_num] = cabinName;

        System.out.println("Enter the Number of Passengers in a cabin: ");
        int guest = input.nextInt();
        guestNo[cabin_num] = guest;

        System.out.println("Enter the First Name: ");
        String fNamePay = input.next();
        fName[cabin_num] = fNamePay;

        System.out.println("Enter the Surename: ");
        String sNamepay = input.next();
        SName[cabin_num] = sNamepay;

        System.out.println("Enter Expense : ");
        long cardNopay = input.nextLong();
        expense[cabin_num] = cardNopay;


    }

    // view all cabins

    public static void view_all_cabin (String[] ship , int[] guestNo,
String[] fName, String[] SName, long[] expense){
        for (int x=0; x<ship.length; x++){
            if(ship[x].equals("e")){
                System.out.println("cabin Number " + x + " is Empty");
            }else{
                System.out.println("--------------------------------------");
                System.out.println("cabin " + x +" Occupied by " + ship[x]);
                System.out.println("Number of Passengers " + guestNo[x]);
                System.out.println("First Name " + fName[x]);
                System.out.println("SurnameName " + SName[x]);
                System.out.println("Expenses " + expense[x]);
                System.out.println("--------------------------------------");
            }

        }
    }

    //view empty cabins

    private static void empty_cabins(String[] ship){
```

```java
        for (int x=0; x<ship.length; x++){
            if (ship[x].equals("e")){
                System.out.println("cabin " + x + " is empty");
            }
        }

    }

    //deleting a customer from a cabin

    public static void del_cus_cabin(String[] ship){
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the cabin number to remove customer: ");
        int cabinNumber=input.nextInt();
        ship[cabinNumber] = "e";
        System.out.println("Removed Successfully...");

    }

    // finding the cabin by a customer name

    public static void find_cabin(String[] ship){
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the Cabin Name: ");
        String cus_name = input.next();

        for(int i=0; i<ship.length; i++){
            if(ship[i].equals(cus_name)){
                System.out.println(cus_name + " is in cabin Number " + i);
            }
        }

    }

    //writing input data into a text file

    public static void info_file(String[] ship, int[] guestNo, String[]
fName, String[] SName, long[] expense)throws FileNotFoundException {
        File info_file = new File("Customer_Info.txt");
        PrintStream writer = new PrintStream(info_file);

        for (String temp : ship){
            writer.println(temp);
        }
        for (int temp : guestNo){
            writer.println(temp);
        }
        for (String temp : fName){
            writer.println(temp);
        }
        for (String temp : SName){
            writer.println(temp);
        }
        for (long temp : expense){
            writer.println(temp);
        }
        System.out.println("...Successfully Saved...");
        writer.close();

    }

    //load the input data which has written into the text file

    public static void load_prog_data(String[] ship, int[] guestNo, String[]
fName, String[] SName, long[] expense) throws IOException {

        try {
            File inputFile = new File("Customer_Info.txt");
            Scanner rf = new Scanner(inputFile);
            String fileLine;

            while (rf.hasNext()) {
                fileLine = rf.nextLine();
                System.out.println(fileLine);
            }
            rf.close();

        }

        catch (IOException e) {
            System.out.println("An error occurred");
        }

        System.out.println("...Successfully Loaded...");
    }



    //viewing the cabin names according to alphabetical order

    public static void order_name(String[] ship){
        String []shipArray = new String[ship.length];
        for(int i = 0; i < ship.length; i++){
            shipArray[i] = ship[i];
        }
        int n = shipArray.length;
        for (int x = 0; x < n - 1; x++)
        {
            for(int y = 0; y <= n - 2; y++)
            {
                if (shipArray[y].compareToIgnoreCase(shipArray[y + 1]) > 0)
                {
                    String temp = shipArray[y];
                    shipArray[y] = shipArray[y + 1];
                    shipArray[y + 1] = temp;
                }
            }
        }

        for(String temp:shipArray){
            if(!temp.equals("e")) {
                System.out.println(temp);
            }
        }
    }
}
```

Task 02 – Source Code

Ship.java

```
import java.io.File;                                     // Import the File class
import java.io.FileNotFoundException;                    // Import this class to
handle errors
import java.io.IOException;
import java.io.PrintStream;
import java.util.Scanner;                                  // Import the Scanner
class to read text files

public class Ship {
    private static boolean S_menu = true;


    public static void main(String[] args) throws IOException  {

        //initialising the array

        Ship shipObj = new Ship();
        Cabin[] cabinObj = new Cabin[12];
        Passenger[] perObj = new Passenger[12];

        Queue queObj = new Queue();

        Scanner input = new Scanner(System.in);


        for (int i = 0; i < cabinObj.length; i++){
            cabinObj[i] = new Cabin();

        }
        for(int i = 0; i < perObj.length; i++){
            perObj[i] = new Passenger();
        }

        shipObj.initialise(cabinObj);


        // Cruise ship menu

        while (S_menu){
            System.out.println("        ........WELCOME TO THE CRUISE
SHIP........       ");
            System.out.println("                ...CRUISE SHIP MENU...
");
            System.out.println("Enter 'A' to add a customer to a cabin
:");
            System.out.println("Enter 'V' to view all cabin
:");
            System.out.println("Enter 'E' to Display Empty cabin
:");
            System.out.println("Enter 'D' to Delete customer from cabin
:");
            System.out.println("Enter 'F' to Find cabin from customer name
:");
            System.out.println("Enter 'S' to Store program data into file
:");
            System.out.println("Enter 'L' to Load program data from file
:");
            System.out.println("Enter 'O' to View guests Ordered
alphabetically by name :");
            System.out.println("Enter 'T' to view customer expenses
:");

            String selection = input.next();

            switch (selection){

                case "A":
                    shipObj.add_customer(cabinObj, perObj , queObj);
                    break;

                case "V":
                    shipObj.view_all_cabins(cabinObj, perObj);
                    break;

                case "E":
                    shipObj.empty_cabins(cabinObj);
                    break;

                case "D":
                    shipObj.del_cus_cabin(cabinObj);
                    break;

                case "F":
                    shipObj.find_cabin(cabinObj , new Passenger[][]{perObj});
                    break;

                case "S":
                    shipObj.info_file(cabinObj , perObj);
                    break;

                case "L":
                    load_prog_data(cabinObj , perObj , queObj);
                    break;

                case "O":
                    shipObj.order_name(cabinObj);
                    break;

                case "T":
                    display_expenses(cabinObj , perObj);
                    break;

            }


            //exit or continue  the menu selection


            boolean choice = true;

            while (choice) {
                System.out.println("Enter 1 to Continue or 2 to Exit: ");
                int com = input.nextInt();

                if (com == 1) {
                    S_menu = true;
                    choice = false;
                }

                else if(com == 2){
                    S_menu = false;
                    choice = false;
                }

                else{
                    System.out.println("...Invalid Input...");
                }

            }

        }

    }


    public Ship(){}
    public void initialise(Cabin[] cabinObj) {
        for (int x = 0; x < 12; x++ ) cabinObj[x].setCusName("e");
        System.out.println("initialise");
    }

    //method for add a customer to a cabin

    public void add_customer(Cabin[] cabinObj, Passenger[] perObj , Queue
queueObj){

        boolean emCabin = false;

        for(int i = 0; i < cabinObj.length; i++){
            if(cabinObj[i].getCusName().equals("e")){
                emCabin = true;
            }
        }
        if(emCabin){
            Scanner input = new Scanner(System.in);
            System.out.println("Enter cabin Number 0 to 11: ");
            int cabin_num = input.nextInt();

            System.out.println("Enter Name for Cabin Number " + cabin_num +
": ");
            String cabinName = input.next();
            cabinObj[cabin_num].setCusName(cabinName);

            System.out.println("Enter the Number of Guests in a Cabin: ");
            int guest = input.nextInt();
            perObj[cabin_num].setnoCabin(guest);

            System.out.println("Enter the First Name: ");
            String fNamePay = input.next();
            perObj[cabin_num].setcusFname(fNamePay);

            System.out.println("Enter the Surname: ");
            String sNamepay = input.next();
            perObj[cabin_num].setcusSname(sNamepay);

            System.out.println("Enter your Expenses ($): ");
            long expensespay = input.nextLong();
            perObj[cabin_num].setExpenses(expensespay);

        }

        else{

            queObj.addQue();

        }


    }

    void setcusSname(String sNamepay) {
    }


    //method  for view all cabins

    public void view_all_cabins(Cabin[] cabinObj , Passenger[] perObj){
        for (int x=0; x< cabinObj.length; x++){
            if(cabinObj[x].getCusName().equals("e")){
                System.out.println("Cabin Number " + x + " is Empty");
            }else{
                System.out.println("---------------------------------------");
                System.out.println("Cabin " + x +" Occupied by " +
cabinObj[x].getCusName());
                System.out.println("Number of Guest " +
perObj[x].getnoCabin());
                System.out.println("First Name " + perObj[x].getcusFname());
                System.out.println("SurName " + perObj[x].getcusSname());
                System.out.println("Expenses " + perObj[x].getExpenses());
                System.out.println("---------------------------------------");
            }

        }
    }

    //method  for Display Empty cabins

    public void empty_cabins(Cabin[] cabinObj){
        for (int x=0; x<cabinObj.length; x++){
            if (cabinObj[x].getCusName().equals("e")){
                System.out.println("cabin " + x + " is empty");
            }
        }

    }

    //method  for Delete customer from cabin

    public void del_cus_cabin(Cabin[] cabinObj){
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the cabin number to remove customer: ");
        int cabinNumber=input.nextInt();
        cabinObj[cabinNumber].setCusName("e");
        System.out.println("Removed Successfully...");

    }

    //method Find cabin from customer name

    public static void find_cabin (Cabin[] cabinObj, Passenger[][] perObj) {
        Scanner User = new Scanner(System.in);
        System.out.println("Enter cabin name: ");
        String getcusFName = User.next();
        for (int i = 1; i < perObj.length; i++) {
            for (int j = 1; j < perObj.length; j++) {
                String findName = perObj[i][j].getcusFName();
                if (findName.equalsIgnoreCase(getcusFName)) {
                    System.out.println("Cabin is " + i + " Passenger location
is " + j + "\n");
                }
            }
        }

    }

    String getcusFName() {
        String cusFname = null;
        return cusFname;}


    //method Store program data into file

    public void info_file(Cabin[] cabinObj , Passenger[] perObj)throws
FileNotFoundException {
        File info_file = new File("Customer_Info.txt");
        PrintStream writer = new PrintStream(info_file);
        for (int i = 0; i < cabinObj.length; i++){
            writer.println(cabinObj[i].getCusName());
        }
        for(int i = 0; i < perObj.length; i++){
            writer.println(perObj[i].getnoCabin());
            writer.println(perObj[i].getcusFname());
            writer.println(perObj[i].getcusSname());
            writer.println(perObj[i].getExpenses());
        }
        System.out.println("Successfully Saved...");

    }

    //method Load program data from file

    public static void load_prog_data(Cabin[] cabinObj, Passenger[] perObj ,
Queue queObj) {

        try {
            File inputFile = new File("Customer_Info.txt");
            Scanner rf = new Scanner(inputFile);
            String fileLine;

            while (rf.hasNext()) {
                fileLine = rf.nextLine();
                System.out.println(fileLine);
            }
            rf.close();

        }

        catch (IOException e) {
            System.out.println("An error occured");
        }

        System.out.println("...Successfully Loaded...");
    }


    //method to view  expenses


    public static void display_expenses(Cabin[] cabinObj, Passenger[] perObj)
{
        for (int x = 0; x < cabinObj.length; x++) {
            if (cabinObj[x].getCusName().equals("e")) {
                System.out.println("Expenses " + perObj[x].getExpenses());
            } else {
                System.out.println("Expenses " + perObj[x].getExpenses());
            }

        }
    }
    //public void totExpenses(Person[] perObj){
    //long expensespay = input.nextLong();
    // perObj[cabin_num].setcrdNo(expensespay);
    //}


    //method to View customers  alphabetically ordered by name

    public void order_name(Cabin[] cabinObj){
        String []shipArray = new String[cabinObj.length];
        for(int i = 0; i < cabinObj.length; i++){
            shipArray[i] = cabinObj[i].getCusName();
        }
        int n = shipArray.length;
        for (int x = 0; x < n - 1; x++)
        {
            for(int y = 0; y <= n - 2; y++)
            {
                if (shipArray[y].compareToIgnoreCase(shipArray[y + 1]) > 0)
                {
                    String temp = shipArray[y];
                    shipArray[y] = shipArray[y + 1];
                    shipArray[y + 1] = temp;
                }
            }
        }

        for(String temp:shipArray){
            if(!temp.equals("e")) {
                System.out.println(temp);
            }
        }

    }
}
```

Passenger.java :

```
public class Passenger extends Ship{
    //initialise instanse veriable

    private int noCabin;
    private String cusFname;
    private String cusSname;
    private long expense;

    //initialise set & get methods

    public Passenger(){}

    public void setnoCabin(int noCabin){
        this.noCabin = noCabin;
    }
    public int getnoCabin(){
        return noCabin;
    }

    public void setcusFname(String cusFname){
        this.cusFname = cusFname;
    }
    public String getcusFname(){
        return cusFname;
    }

    public void setcusSname(String cusSname){this.cusSname = cusSname;}
    public String getcusSname() {return cusSname;}

    public void setExpenses(long expense){this.expense = expense;}
    public long getExpenses(){return expense;}

}
```

cabin.java :

```
public class Cabin extends Ship{
    private String cusName;
    public Cabin (){}

    public void setCusName(String cusName) {
        this.cusName = cusName;
    }

    public String getCusName() {
        return cusName;
    }
}
```

Task 03 – Source Code

Queue.java :

```
import java.util.Scanner;

public class Queue extends Ship {

    private int queSize = 36;
    private int front, rear;
    private String items[] = new String[queSize];
```

```java
    public Queue() {
        front = -1;
        rear = -1;
    }

    public void setFront(int front) {
        this.front = front;
    }

    public void setRear(int rear) {
        this.rear = rear;
    }

    public int getRear() {
        return rear;
    }

    public void setItems(String[] items) {
        this.items = items;
    }

    // Checking if the queue is full

    public boolean isFull() {
        if (front == 0 && rear == queSize - 1) {
            return true;
        }
        else if (front == rear + 1) {
            return true;
        }
        return false;
    }


    // Adding an element to queue

    public void addQue() {
        Scanner in = new Scanner(System.in);
        if (isFull()) {
            System.out.println("The Queue is Full...");
        } else {
            System.out.println("No Cabins Available Now. Please Enter Your
Name to Added to Waiting List...");
            String element = in.next();
            if (front == -1)
                front = 0;
            rear = (rear + 1) % queSize;
            items[rear] = element;
            System.out.println("Inserted " + element);
        }
    }

    // Removing an element in queue

    public String deQueue () {
        String element;
        element = items[front];
        if (front == rear) {
            front = -1;
            rear = -1;
        }

        /* Q has only one element, so we reset the queue after deleting it.
*/

        else {
            front = (front + 1) % queSize;
        }
        return (element);

    }
}
```

Task 04

Testing

Test Case	Expected Result	Actual Result	Pass/Fail
(Cabins Initialised correctly) After program starts, Press ‘V’	Displays ‘the cabin (cabin no.) is empty’ for all cabins	Displays ‘the cabin (cabin no.) is empty’ for all cabins	Pass

(Add customer “Bob” to cabin 5) Select A, enter “Bob”	Press ‘v’ Displays “Bob” for cabin 5	Displays “Bob” for cabin 5	Pass
Select E, Display Empty cabins	Press ‘E’ Display ‘the cabin (cabin no.) is empty’ for empty cabins	Display ‘the cabin (cabin no.) is empty’ for empty cabins	Pass
Select D, Delete customer from cabin	Press ‘D’ Ask the cabin number of the customer to remove	Removes the customer from the cabin and displays (“Successfully removed”)	pass
Select F, Find cabin from the customer’s name	Press ‘F’ Ask to enter the customer’s name Find the relevant cabin and display	Ask to enter the customer’s name , find and display the relevant cabin	Pass
Select S, Store program data into text file	Press ‘S’ Stores input data into the text file called ‘Customer_Info.txt’ And displays “successfully saved”	Stores input data into the text file called ‘Customer_Info.txt’ And displays “successfully saved”	Pass
Select L, Load program data from file	Press ‘L’ Load the input data which has written into the text file And displays “successfully loaded”	Load the input data which has written into the text file And displays “successfully loaded”	Pass
Select O,	Press ‘O’		

View passengers Ordered alphabetically by cabin name.	Displays the names of the cabins in alphabetical order	Displays the names of the cabins in alphabetical order	Pass
Select T, View the expenses per passenger and the total expenses of all passengers	Press ‘T’ Displays the expenses per passenger and display total expenses of all passengers	Displays the expenses per passenger but unable to display the total expenses of all passengers	Pass

Testing - Discussion

The functionality of the menu tested by using the codes; by pressing ‘V’ , all the cabins are viewed with empty or occupied. The menu works for only upper-case and if a lower-case character is entered, the functionality of that menu automatically iterates and asks “Enter 1 to Continue or 2 to Exit: ” (whether to continue or exit). As well as the option view all Cabins, view all empty Cabins, view passengers sorted in alphabetic order, store program data to a file, load program data from a file, and exit the program were tested similarly in all these tasks.

Task 1

Add method has tested by pressing ‘A’ and entering a cabin number, entering a name for cabin , number of customers in the relevant cabin , customer first name and customer last name and the expenses. The remove passenger option was tested by pressing ‘D’ and entering a Cabin number. Other options ‘E’,’F’,’S’,’L’,’O’ also done as the same as above.

Task 2

The newly created and added classes (passenger and cabin) and modifications to the methods too got tested. The addition of new data(expense) to the program was tested. The newly added viewing expenses menu option was also got tested.

Task 3

When the Cruise ship is full, adding of passengers to the waiting list was tested. Removing passengers from waiting list and adding them to a cabin that became vacant was also tested but failed.

Array Vs. Classes solution

For this boarding system of the cruise ship can be implemented with 2 methods: with an array solution as well as with a class solution. The class solution is the best technique to use when comparing with both techniques. In an array solution , all the parts of the application, methods, and input data fields etc… all are in a single java file, so it is so difficult to read the code and to find errors because we must scan the whole code. when the code is getting more complex, relatively the time consuming to read and find a bug in the error is also increasing though the parts of the codes are in various locations of the code. In a class solution , the program is well structured and organized into classes. This is so easy to read and navigate into the relevant code which w needs to. In array method as the application expands in functionality size , the array solution too gets complex and class method remains simple and it is easy to understand the code. when doing some modifications and extensions in the code , it is difficult to modify the array solutions but it is easy to modify the class solutions. So when considering the overall functionality of arrays and classes solutions , class solution acts as a pro array version . so , the class solution is the most suitable solution to use.

Self-Evaluation form

Criteria	Component marks	Expected Mark
Task 1 Three marks for each option (A, V, E, D, F, S, L, O) Menu works correctly	**24** **6**	21 6
Student comment: Fully Implemented and Partially Working (Ask to enter the customer's name but unable to find and display the relevant cabin)		
Task 2 Cabin class correctly implemented. Passenger class correctly implemented. Expenses correctly reported.	**14** **10** **6**	7 10 3
Student comment: Partially Implemented and Partially Working (one cabin do not limit up to 3 customers and total expenses is not displayed but the expenses of passengers are displayed separately)		
Task 3 Waiting list queue implementation "A : Add" works correctly "D: Delete" works correctly Circular queue implementation	**10** **3** **3** **4**	10 3 1 0
Student comment: Partially Implemented and Partially Working (add and deleting works correctly but adding customers automatically from the queue when a customer is deleted from a cabin is not working properly)		
Task 4 Test case coverage and reasons Writeup on which version is better and why	**6** **4**	5 3
Student comment:		

Coding Style (Comments, indentation, style) Complete the self-evaluation form indicating what you have accomplished to ensure appropriate feedback.	**7** **3**	7 3
Student comment: code is properly commented, indented, and styled		
Totals		79

Reference:

W3Schools: https://www.w3schools.com/java/java_arrays.asp
Stack overflow: https://stackoverflow.com