INFORMATICS INSTITUTE OF TECHNOLOGY

Software Development II

Coursework Report 2021/2022

Pasindu Geevinda

w1871471

20212016

Contents

1.Task 01 – Source Code

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.*;

public class Main {
    public static void main(String[] args) throws IOException {
        String[] name = new String[4];
//        to add the all cabin details.15/03/2022
        String[][] shipCabins;
        Scanner scanner = new Scanner(System.in);
        String[][] shipCabin_x = new String[12][3];
        System.out.print("""

                |-----------------------------------------------|
                |         WELCOME TO CRUISE SHIP BOARDING       |
                |-----------------------------------------------|
                | would you like to continue the program with the  |
```

```
        | previous file record? [Y/N]                  |
        |----------------------------------------------|
        | ANS :                                |""");
if (scanner.next().equalsIgnoreCase("N")) {
   shipCabins= initialise(shipCabin_x,"N");
}else {
   shipCabins= initialise(shipCabin_x,"file.txt");
}

while (true) {
   System.out.print("""
        |----------------------------------------------|
        | CHOOSE OPTION AND ENTER                      |
        | A: Add a customer to a cabin                 |
        | V: View all cabins                           |
        | E: Display Empty cabins                      |
        | D: Delete customer from cabin                |
        | F: Find cabin from customer name             |
        | S: Store program data into file              |
        | L: Load program data from file               |
        | O: View passengersOrdered alphabetically by name |
        | T: To STOP the program                       |
        |----------------------------------------------|
        | ANS :                                |""");
```

```
String answer = scanner.next();
System.out.println("|-----------------------------------------------|");
if (answer.equalsIgnoreCase("A")){
   addCustomer(shipCabins,scanner,name);
}
else if (answer.equalsIgnoreCase("V")) {
   viewAllCabins(shipCabins);
}
else if (answer.equalsIgnoreCase("E")) {
   emptyCabins(shipCabins);
}
else if (answer.equalsIgnoreCase("D")) {
   deleteCustomer(shipCabins,scanner,name);
}
else if (answer.equalsIgnoreCase("F")) {
   findCabinByName(shipCabins,scanner);
}
else if (answer.equalsIgnoreCase("S")) {
   intoFile(shipCabins, scanner);
}
else if (answer.equalsIgnoreCase("L")) {
   loadFile(scanner,shipCabin_x);
}
else if (answer.equalsIgnoreCase("O")) {
```

```
            passengersOrder(shipCabins);
        }
        else if (answer.equalsIgnoreCase("T")) {
            print(  "|                FINISHED                    |");
            break;
        }
        else {print("| INPUT IS NOT CORRECT. TRY AGAIN              |");
        }
    }
}

  private static void loadFile(Scanner scanner, String[][] shipCabin_x) {
    /*
     * You have two options as below. If your input is one it will load data from previous file that you saved data."file.txt"
     * if your option is 2 it will give you chance to load data from file, but it must have an order as printed msg when last
     printed msg on txt file when we store the data. If it is the program will load the data from that file. You just
     only have to give the path to txt file and put the .txt extension
     28/03/2022
      */

     System.out.print("""
             |-----------------------------------------------|
```

```
        | Choose the option what you want               |
        | 1: To load the previous stored data           |
        | 2: To load the New File data                  |
        | ANS                                :|""");
try {int ans1 = Integer.parseInt(scanner.next());
   if (ans1==2){
      System.out.println("|-----------------------------------------------|");
      System.out.print("""
         -------------------------------------------------
         | Enter the path to .txt file                   |
         | Eg: D:\\CW\\project\\file.txt                 |
         | PATH                               :|""");
      String path = scanner.next();
      initialise(shipCabin_x,path);
      System.out.println("|-----------------------------------------------|");
   }
   else if (ans1==1){
      initialise(shipCabin_x,"file.txt");
      System.out.println("|-----------------------------------------------|");}
   else print("| INPUT IS NOT CORRECT. TRY AGAIN                |");
} catch (InputMismatchException | FileNotFoundException e) {
   System.out.print("""
         |              Wrong input                |
           """);
```

```
        System.out.println("|-----------------------------------------------|");
    }
}


  private static String[][] initialise(String[][] cabins, String path) throws FileNotFoundException {
    /*Asking for would you like to continue the program with the previous file record?
     * If he said yes the previous record will be assigned to the cabin objects and passenger objects
     * IF he said No all cabin objects will create with empty cabin details
     15/03/2022
     */

    if (path.equalsIgnoreCase("N")){
//        adding details as ["Cabin num", "customer name", cabin status] and it added to cabin 2D array.
        for (int x = 0; x < 12; x++ ) {
            String[] sub = new String[3];
            sub[0]="Cabin "+(x+1);
            sub[1]=" - ";
            sub[2]= String.valueOf(0);
            cabins[x] = sub;
            System.out.println( "initialise ".concat(sub[0]));}
    }
    else {
```

```
        File file = new File(path);
        Scanner scanner = new Scanner(file);

        int i=1;
        for (String[] cabin : cabins){
//              read the line by line, and it will split using ":" and add to r1 array
           String[] r1 = scanner.nextLine().replace(" ","").split(":");
           cabin[2] = r1[2].replace(" ","");
           cabin[0] = "Cabin "+i;
           if (Integer.parseInt(cabin[2]) ==0) {
              cabin[1] = " - ";
           }
           else {
              cabin[1] = r1[1];
           }System.out.println("|   Initialise: "+ cabin[0]+"   |");
           i++;
        }
     }
   return cabins;}



   private static void intoFile(String[][] shipCabins,Scanner scanner) {
```

```
/*
* By this method will give you two option
1: To store data into the previous file
2: To store data into New File
* if you entered 1, the previous details recorded file "file.txt" use as the storing file if isn't you can give
path and enter your file name as you like with .txt extension.At last of print in file, it will show you what is
the pattern of data view on your txt file
28/03/2022
*/

String path = "file.txt";
System.out.println("|          Store program data into file         |");
System.out.print("""
        |-------------------------------------------------|
        | Choose the option what you want                 |
        | 1: To store data into the previous file         |
        | 2: To store data into New File                  |
        | ANS                                  :|""");

try {
   int ans = Integer.parseInt(scanner.next());
//        if answer is 2 you can let him/her to decide .txt file path to store data and that path will be assigned to path
```

```
//          variable
        if (ans==2) {
           System.out.print("""
              ---------------------------------------------------
              | Enter the path to .txt file that you want to store |
              | Eg: D:\\CW\\project\\file.txt                      |
              | PATH                                  :|""");
           path=scanner.next();
        }
        FileWriter file = new FileWriter(path);
        String a =""""



              |-------------------------------------------------|
              |                   PATTERN                       |
              |-------------------------------------------------|
              |     Cabin Num : Passengers Name : Cabin status     |
              |-------------------------------------------------|
              """;
        System.out.println(a);
        for (String[] cabin : shipCabins) {

           String writeDown = cabin[0] + " : " + cabin[1] + " : " + cabin[2];
           file.write(writeDown + "\n");
```

```
            System.out.println(writeDown);
        }
        file.write("""



                |-------------------------------------------------|
                |                   PATTERN                       |
                |-------------------------------------------------|
                |Room condition(R.C.): If 1, there is already a guest|
                |         E: Expenses;        C: Customer         |
                |   R.C  :  [1st C]  :  [2nd C]  :  [3rd C]  :  [E]  |
                |-------------------------------------------------|""");
        file.close();
    } catch (NumberFormatException | IOException e) {
        System.out.println("|-------------------------------------------------|");
        System.out.println("|           Wrong Input type                    :|");
        System.out.println("|-------------------------------------------------|");
    }

}



private static void findCabinByName(String[][] shipCabins, Scanner scanner) {
```

```
    /*
    * Go through entire cabins by loop and checking passengers surname or firstname is equal to the user input name and
    if it's found it will print out
    20/03/2022
    * */

    System.out.println("|          Find cabin from customer name         |");
    System.out.println("|------------------------------------------------|");
//      ask the name to find
    System.out.print("|          Enter the Customer name            :|");
    String customer = scanner.next().replace(" ","");int i=0;
    System.out.println("|------------------------------------------------|");
    for (String[] cabin :shipCabins){
//          going through the array by loop and if customer == your input it will go this way and print out
        if (cabin[1].equalsIgnoreCase(customer)){
            System.out.println("|              "+cabin[0] + " owned by " + cabin[1]+"        |");
            i++;
        }
    }// if customer couldn't be able to find out this will print
    if (i==0) System.out.println("|       No customer by this name was found       |");
    System.out.println("|------------------------------------------------|");
  }
  private static void deleteCustomer(String[][] shipCabins, Scanner scanner, String[] name) {
```

```
        /*
        * Go through entire cabins by loop and checking passengers surname or firstname is equal to the user input name and
        if it's found the equal passenger pass set the mainName value to cabinName value
        * If that name customer couldn't be able to find out in cabin it checked by line 183 and print msg
        * After that if you find out the correct passenger from cabin reset the values in whole cabin,using mainName.
        * Eventually is any customers in waiting List they will be added to the deleted cabin.
        28/03/2022
        */

        System.out.println("|            Delete customer from cabin          |");
        int l =0;
        while (l==0){
            String mainName =" - ";
//      Ask the customers name or cabin num for delete from cabin
            System.out.println("|------------------------------------------------|");
            System.out.print(  "|       Enter the Customer name or cabin num      :|");
            String firstname = scanner.next().replace(" ","");
            String g = firstname.substring(0,1).toUpperCase()+firstname.substring(1);
            System.out.println("|------------------------------------------------|");
            int a=0;
            for (String[] cabin :shipCabins){
//          going through cabin loop and check the cabin names and cabin numbers for is it equal to input or not
```

```
            if (g.equalsIgnoreCase(cabin[1])
                    ||  cabin[0].split("n")[1].replace(" ","").equals(g)) {
                mainName = cabin[1];
                a++;
                break;
            }
        }
        try {
            int k = Integer.parseInt(g);
            if (0>= k || k>= 13){
                print("|          We have only 12 cabins              |");
                continue;}
        }catch (NumberFormatException ignored){
        }
        if (a==0) {
//          if customer doesn't find this will print
            print("|       No customer by this name was found       |");
        }
        else {
            System.out.print("| Are you sure you want to delete this passenger (Y/N) :|");
            if (!scanner.next().equalsIgnoreCase("Y")) {
                l++;
                print("|          End process without deleting          |");
                continue;
```

```
            }
            for (String[] cabin:shipCabins) {
               if (cabin[1].equals(mainName)) {
                  cabin[1]=" - ";
                  cabin[2]= String.valueOf(0);
                  print("|           Deleted from this cabin              |");l++;
                  break;
               }
            }
         }
      }if (waitingListSize(name)>0) init2(shipCabins,name);
   }



   private static int emptyCabins(String[][] shipCabins) {
      /*
       * Go through the loop and check the cabin status. If cabin status is 0 it is empty, and it will
print
       03/04/2022*/

      System.out.println("|             Display Empty cabins              |");
      System.out.println("|-----------------------------------------------|");
      int i=12;
      int j=0;
      for (String[] cabin:shipCabins) {
```

```
//          going through array of cabins and if the cabin is empty go in this path
          if (cabin[2].equals(String.valueOf(0))) {
             System.out.println("|            " + cabin[0] + " :   empty                |");
             i--;
          }
       }if (i==12) {
//          if all cabins are fulled this will print
          print("|-------------All Cabins are fulled----------------|");
          j=1;
       }
       return j;
    }

    private static void viewAllCabins(String[][] shipCabins) {
       /*
        * If cabinStatus == 0 it will show cabin is empty if isn't it will show cabin owners name
        01/04/2022
        */

       System.out.println("|               View all cabins               |");
       System.out.println("|---------------------------------------------|");
       for (String[] cabin:shipCabins){
//          going through the array and if cabin is empty this will print and if isn't else option will
print out
          if (cabin[2].equals(String.valueOf(0))){
```

```
            System.out.println("|            "+cabin[0]+" :   empty                |");
        }else {
            System.out.println("|            "+cabin[0]+" :   " + cabin[1] +"              |");
        }
    }
}


    private static void addCustomer(String[][] shipCabins, Scanner scanner, String[] name) {
        /*04/04/2022*/
        int roomNum;
        System.out.println("|            Add customers to a cabin            |");
        int q;
        do {
            q = 0;
            try {
//              check all the cabins or full or not. If isn't path is here and ask for cabin number they require
                if (emptyCabins(shipCabins)==0) {
                    System.out.println("|------------------------------------------------|");
                    System.out.print("|          Enter the Cabin number              :|");
                    roomNum = Integer.parseInt(scanner.next());
//                  we have only 12 cabins
                    if (roomNum < 13 && 0 < roomNum) {
                        for (String[] cabin : shipCabins) {
```

```
                if (Integer.parseInt(cabin[0].split("n ")[1]) == roomNum) {
//                      If input cabin is full by customers this will print and if isn't customer add by
else option
                   if (cabin[2].equals(String.valueOf(1))) {
                      print("|----------------The cabin is full-----------------|");
                      print("|               Waiting List                 |");
                      print("""
                      | you can only add 4 customers to the waiting list |
                      |   When one customer deleted, You will be added   |""");
                      System.out.print("|          Enter the Customer's name          :|");
                      String firstname = scanner.next().replace(" ", "");
                      String a = firstname.substring(0, 1).toUpperCase() + firstname.substring(1);
                      int b = check(a,shipCabins);
                      if (b==1) {
                         q++;
                         break;
                      }int j=0;
                      for (int i=0;i<4;i++){
//                  if this is doesn't exist same name will add to the whole waiting list array.
                         if (j>=1){continue;}
//                  check the index position is free and adding to the waiting list and adding 1 to j
                         if (name[i]==null) {
                            name[i] = a;
                            j++;
                         }
```

```
                    }
                    q=1;
                    break;
                }

                else {
                    System.out.println("|-----------------------------------------------|");
                    System.out.print("|          Enter the Customer's name           :|");
                    String firstName = scanner.next().replace(" ", "");
                    String a = firstName.substring(0, 1).toUpperCase() + firstName.substring(1);
//                      by going to check method it will find that customer already exists or not. If him/her exists
//                       program will end.
                    int d = check(a,shipCabins);
                    if (d==1) {
                        q++;
                        break;
                    }
                    cabin[2]=String.valueOf(1);
                    cabin[1]=a;
                    System.out.println("|          "+cabin[1]+" added to "+cabin[0]+"             :|");
                    q++;
                }
            }
        }
```

```
            } else print("|          We have only 12 cabins               |");

            }// If all cabins are full the customer will add to waiting list

        } catch (NumberFormatException e) {
            print("|              Wrong Input type             :|");
        }

    }
    while (q == 0);
    if (waitingListSize(name)>0) init2(shipCabins,name);
  }

  private static int check(String a, String[][] shipCabins) {
    for (String[] cabins:shipCabins){
//          find the customer already exists or not
//          05/04/2022
        if (a.equalsIgnoreCase(cabins[1])) {
            System.out.println("|------------------------------------------------|");
            System.out.println("|------------This customer already exist-----------|");
            return 1;
        }
    }
    return 0;
```

```
}


public static void passengersOrder(String[][] shipCabins) {
    /*
    * When user enter the one option it will Go through the loop, and it will collect your surname of firstname to array
    as user entered input.
    * And going through the loop, and it will make as an order the list using selection
    Algorithm */
    String[] firstNames = new String[36];
    System.out.println("| View passengers Ordered alphabetically by name   |");
    int q =0;
    for (String[] cabin:shipCabins){
       firstNames[q]= cabin[1];
       q++;
    }
    for (int i=0;i<q;i++){
       for (int j=0;j<q;j++) {
          if (firstNames[j].compareTo(firstNames[i]) > 0) {
             String highNum = firstNames[i];
             String smallNum = firstNames[j];
             firstNames[i] = smallNum;
             firstNames[j] = highNum;
```

```
            }
        }
    }
    int d =1;
    System.out.print("|  ");
    for (int j=0;j<q;j++){
        if (firstNames[j].replace(" ","").equals("-")) {
            continue;
        }
        System.out.print("  "+firstNames[j]+"  ");
        d++;
        if (d%5==0) {
            System.out.println(" ");
        }
    }System.out.println("  |");
}

public static void init2(String[][] shipCabins, String[] name){
    for (String[] cabin:shipCabins) {
        int i=0;
        if (waitingListSize(name) > 0) {
            for (int j = 0; j < 4; j++) {
                if (name[j]==null || i>=1) {continue;}
                if (cabin[2].replace(" ", "").equals(String.valueOf(0))) {
```

```
                cabin[1] = name[j];
                cabin[2] = String.valueOf(1);
                name[j]=null;
                print("|          "+cabin[1]+" added to "+cabin[0]+"              |");
                i++;
            }
        }
    }

  }
}
private static void print(String expression){
  System.out.println("|------------------------------------------------|");
  System.out.println(expression);
  System.out.println("|------------------------------------------------|");
}

private static int waitingListSize(String[] name) {
  int j=0;
  for (int i=0; i<4;i++){
    if (!(name[i]==null)) j++;
  }
  return j;
}
```

```
}
```

2.Task 02 – Source Code

2.1. Main Class:

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.InputMismatchException;
import java.util.Objects;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws IOException {
//        This array is for store the cabins details and all cabins has own 3 passenger
objects.
        Cabin[] shipCabins = new Cabin[12];
        CircularQueue circularQueue;
        Scanner scanner = new Scanner(System.in);
//        Creating the 12 cabin objects using loop and by creating one cabin automatically
will create 3 passenger objects
        for (int i=0; i<12; i++) shipCabins[i] = new Cabin(i);
          /*Asking for would you like to continue the program with the previous file
record?
          * If he said yes the previous record will be assigned to the cabin objects and
passenger objects
          * IF he said No all cabin objects will create with empty cabin details
        15/03/2022
          */
        System.out.print("""




                    |-------------------------------------------------|
                    |          WELCOME TO CRUISE SHIP BOARDING        |
                    |-------------------------------------------------|
                    | would you like to continue the program with the |
                    | previous file record? [Y/N]                     |
                    |-------------------------------------------------|
                    | ANS :                                           |""");
        if (scanner.next().equalsIgnoreCase("N")) {
            initialise(shipCabins, "N");
        }else {
            initialise(shipCabins, "file.txt");
        }
        do {
          /*This will create CircularQueue object and also will be created the waiting list
array of type of
            cabin using the size we entered
        15/03/2022
            */
            System.out.print("""
```

```
                |-------------------------------------------------|
                |          WELCOME TO CRUISE SHIP BOARDING        |
                |-------------------------------------------------|
                | When the cabin is full, enter the size of the   |
                | waiting list to include the information about the|
                | cabin you want. (SIZE min = 2)                  |
                |-------------------------------------------------|
                | ANS :                                           |""");
            try {
                circularQueue = new CircularQueue(Integer.parseInt(scanner.next()));
                break;
            }catch (NumberFormatException e) {
                wrongInput();
            }
        }while (true);
        int loop=0;
        while (loop==0) {
//              Give the options to choose.
            System.out.print("""
                |-------------------------------------------------|
                | CHOOSE OPTION AND ENTER                         |
                | A: Add a customer to a cabin                    |
                | V: View all cabins                              |
                | E: Display Empty cabins                         |
                | D: Delete customer from cabin                   |
                | F: Find cabin from customer name                |
                | S: Store program data into file                 |
                | L: Load program data from file                  |
                | O: View passengersOrdered alphabetically by name |
                | T: Expenses/total Expenses                      |
                | Q: To STOP the program                          |

                |-------------------------------------------------|
                | ANS :                                           |""");
            String answer = scanner.next();
            switch (answer.toUpperCase()) {
                case ("A") -> addCustomer(shipCabins, circularQueue, scanner);
                case ("V") -> viewAllCabins(shipCabins);
                case ("E") -> emptyCabins(shipCabins);
                case ("D") -> deleteCustomer(shipCabins, circularQueue, scanner);
                case ("F") -> findCabinByName(shipCabins, scanner);
                case ("S") -> intoFile(shipCabins, scanner);
                case ("L") -> loadFile(shipCabins, scanner);
                case ("O") -> passengersOrder(shipCabins, scanner);
                case ("T") -> totalExpenses(shipCabins, scanner);
                case ("Q") -> {stop();loop++;}

                default -> print("|         INPUT IS NOT CORRECT. TRY AGAIN          |");
            }
        }
    }
```

```
    //-------------------------------------------------------------emptyCabins-----------------
-----------------------------|
    public static void emptyCabins(Cabin[] shipCabin){
        /*
         * Go through the loop and check the cabin status. If cabin status is 0 it is
empty, and it will print
         * 28/03/2022
         * */
        int a =0;
        print("|               Display Empty cabins               |");
        for (Cabin cabin:shipCabin) {
            if (cabin.status() == 0) {
                System.out.println("|              Cabin " + cabin.getCabinNum() + " is
empty                  |");
                a++;}
        }
//        If all cabins had fulled it will check using variable, and it is it will print
        if (a==0) print("|               All Cabins  are full               |");
    }




    //-------------------------------------------------------------viewAllCabins-----------------
-----------------------------|
    public static void viewAllCabins(Cabin[] shipCabin){
        /*
         * show the details of room. All the Passengers' names and Expenses
         * If cabinStatus == 0 it will show cabin is empty
         * If entered number is out of bond it will print and error msg by wrongInput
method
         * 01/04/2022
         * */
        print("|                  View all cabins                 |");
        int q=0;
        do {
            for (Cabin cabin:shipCabin){
                print("|                       "+cabin.getMainName()+"
|");
                if (cabin.status() == 1) {
                    for (Passenger passenger: cabin.getPassengers()){
                        if (passenger.getFirstName().equals(" - ")) continue;
                        System.out.println   ("|    Passenger
"+passenger.getPassengerNum()+"     : "+passenger.getFirstName()+"
"+passenger.getSurName()+"        |");
                        System.out.println   ("|    Expenses        :
"+passenger.getExpenses()+"                   |");
                    }q++;
                }else {q++;
                    print("|                  Room is empty                   |");
                }
            }

        } while (q<1);
        System.out.println("|---------------------------------------------------|");
    }
```

```
    //-------------------------------------------------------addCustomer-------------------
-----------------------------|
    public static void addCustomer(Cabin[] shipCabin, CircularQueue circularQueue, Scanner
scanner) {
        /*
        * Ask for cabin number and check it must be between 13 and 0. There haven't any
roomNum as 13 and 0
        * After that ask for cabin num they like to add the customer and check the cabin
status
        * If cabin states == 0 passengers added to the cabin and if isn't passengers will
add to the waiting list
        * Use the Initialize method and if there is any cabinStatus == 0 cabin customers
adding to that from waiting list
        as circular queue.
        * 20/03/2022
        * */
        int roomNum = 0;
        print("|             Add customers to a cabin             |");
        do {
            try {
                System.out.println("""
                                   |  instructions :                                    |
                                   | * You can only get 3 places in cabin.              |
                                   | * When you finished adding enter "N" for break     |
                                   | program from adding another customer
|""");
                System.out.println("|----------------------------------------------------|");
                System.out.print( "|          Do you want add passengers (Y/N)       :|");
                if (scanner.next().equalsIgnoreCase("N")) break;
                emptyCabins(shipCabin);
                System.out.println("|----------------------------------------------------|");
                System.out.print("|          Enter the Cabin number                 :|");
                roomNum = Integer.parseInt(scanner.next());
                if (roomNum<13 && 0<roomNum){
                    System.out.println("|----------------------------------------------------
|");
                    break;
                }
                print("|           We have only 12 cabins                 |");
            } catch (NumberFormatException e) {
                wrongInput();
            }
        }
        while (true);
        for (Cabin cabin :shipCabin){
            if (cabin.status() ==0){
                if (cabin.getCabinNum()==roomNum){
                    String[] tempCabin = addCustomerInput(scanner);
                    for (Passenger passenger : cabin.getPassengers()) {
                        System.out.println();
                        passenger.setFirstName(tempCabin[passenger.getPassengerNum()-
1].split(":")[0]);
                        passenger.setSurName(tempCabin[passenger.getPassengerNum()-
```

```
1].split(":")[1]);

passenger.setExpenses(Double.parseDouble(tempCabin[passenger.getPassengerNum()+2]));
                        cabin.setStatus(1);
                        System.out.println("|               Customers added
:| " + cabin.getCabinNum());
                    }
                }
            }else {
                if (cabin.getCabinNum()==roomNum) {
                    if (circularQueue.isFull()) break;
                    print("|               Adding to waiting list            |");

                    circularQueue.enqueue(addCustomerInput(scanner));
                }
            }
        }// Initialize method from Initialize class
        waitingListInitialize2(shipCabin,circularQueue);

    }




    //----------------------------addCustomerInput - belongs to add customer method-------
---------------------------|
    private static String[] addCustomerInput(Scanner scanner){
        // This method used for collect to the data from customers.
        //02/04/2022
        String[] names = new String[6];
        for (int i =0;i<3;i++){
            if (i>=1) {
                System.out.print("|  Do you want add another customer (Y/N)         :|");
                if (scanner.next().equalsIgnoreCase("N")) {
                    for (; i < 3; i++) {
                        names[i] = " - " + ":" + " - ";
                        names[i + 3] = String.valueOf(0.0);
                    }
                    break;
                }
            }
            print("|                  "+(i+1)+" Customer                      :|");
            System.out.print("|            Enter the Customer's first name      :|");
            String _1stName = scanner.next().replace(" ","");
            System.out.println("|--------------------------------------------------|");
            System.out.print("|            Enter the Customer's surname         :|");
            String _2ndName = scanner.next().replace(" ","");
            System.out.println("|--------------------------------------------------|");
            names[i] = _1stName.substring(0,1).toUpperCase()+_1stName.substring(1)+":"+
                    _2ndName.substring(0,1).toUpperCase()+_2ndName.substring(1);
            do {
                try {
                    System.out.println("|-------------------------------------------------
|");
                    System.out.print("|            Enter the Customer's Expenses
:|");
                    names[i+3]= String.valueOf((Double.valueOf(scanner.next())));
```

```
                    break;
                } catch (NumberFormatException kl) {
                    print("|         INPUT IS NOT CORRECT. TRY AGAIN          |");
                }
            }while (true);
            print("|                Customer added                   :|");
            System.out.println("|-------------------------------------------------|");
        }
        return names;
    }




    //-----------------------------------------------------delete Customer-------------------
-----------------------------|
    public static void deleteCustomer(Cabin[] _arrayCabins, CircularQueue circularQueue,
Scanner scanner) {
        /*
        * Go through entire cabins by loop and checking passengers surname or firstname is
equal to the user input name and
        if it's found the equal passenger pass set the mainName value to cabinName value
        * If that name customer couldn't be able to find out in cabin it checked by line
183 and print msg
        * After that if you find out the correct passenger from cabin reset the values in
whole cabin,using mainName.
        * Eventually is any customers in waiting List they will be added to the deleted
cabin using --------
        Initialize.waitingListInitialize2(_arrayCabins,circularQueue) method.
        * 03/04/2022
        */
        String mainName =" - ";
        int cabinNum = 0;
        print(           "|             Delete customer from cabin           |");
        System.out.println("|-------------------------------------------------|");
        System.out.print("|              Enter the Customer name            :|");
        String g = scanner.next().replace(" ","");
        String customer = g.substring(0,1).toUpperCase()+g.substring(1);
        System.out.println("|-------------------------------------------------|");
        for (Cabin cabin :_arrayCabins){
            for (Passenger passenger:cabin.getPassengers()){
                if (customer.equalsIgnoreCase(passenger.getSurName()) ||
customer.equalsIgnoreCase(passenger.getFirstName())) {
                    mainName = cabin.getMainName();
                    cabinNum = cabin.getCabinNum();
                    break;
                }
            }
        }
        if (mainName.equals(" - ")||cabinNum==0) {
            print("|        No customer by this name was found        |");
        }
        else {
            System.out.println("|--            Found the cabin number            --|");
            System.out.println("|-------------------------------------------------|");
            System.out.println("|--                      "+_arrayCabins[cabinNum-
1].getMainName()+"                    --|");
```

```
            for (Passenger passenger : _arrayCabins[cabinNum-1].getPassengers()) {
                System.out.println("|-----------------------------------------------------|");
                System.out.println("|                No "+passenger.getPassengerNum()+"
customer is deleted          |");
                System.out.println("|-----------------------------------------------------|");
                passenger.setDefFirstName();
                passenger.setDefSurName();
                passenger.setDefExpenses();
                _arrayCabins[cabinNum-1].setStatus(0);
            }
            waitingListInitialize2(_arrayCabins,circularQueue);
        }
    }




    //-----------------------------------------------------find By Cabins-----------------
----------------------------|
    public static void findCabinByName(Cabin[] shipCabin, Scanner scanner) {
        /*
        * Go through entire cabins by loop and checking passengers surname or firstname is
equal to the user input name and
        if it's found it will print out
        * 17/03/2022
        * */
        print("|           Find cabin from customer name          |");
        System.out.print("|            Enter the Customer name              :|");
        String customer = scanner.next().replace(" ","");int i=0;
        System.out.println("|-----------------------------------------------------|");
        for (Cabin cabin :shipCabin){
            for (Passenger passenger: cabin.getPassengers()){
                if
(customer.equalsIgnoreCase(passenger.getFirstName())||customer.equalsIgnoreCase(passenger.g
etSurName())){
                    System.out.println("|                "+cabin.getMainName() + " owned by
" + customer.substring(0,1).toUpperCase().concat(customer.substring(1))+"         |");
                    i++;
                }
            }
        }if (i==0) print("|        No customer by this name was found        |");
    }




    //-----------------------------------------------------into File-----------------
----------------------------|
    public static void intoFile(Cabin[] shipCabin, Scanner scanner) throws IOException {
        /*
        * By this method will give you two option
        1: To store data into the previous file
        2: To store data into New File
        * if you entered 1, the previous details recorded file "file.txt" use as the
storing file if isn't you can give
```

```
        path and enter your file name as you like with .txt extension.At last of print in
file, it will show you what is
        the pattern of data view on your txt file
        18/03/2022 */
        String path = "file.txt";
        print("|            Store program data into file          |");
        System.out.print("""
                    |-----------------------------------------------------|
                    | Choose the option what you want                     |
                    | 1: To store data into the previous file             |
                    | 2: To store data into New File                      |
                    | ANS                                               :|""");

        try {
            int ans = Integer.parseInt(scanner.next());
            if (ans==2) {
                System.out.print("""
                    -----------------------------------------------------
                    | Enter the path to .txt file that you want to store |
                    | Eg: D:\\CW\\project\\file.txt                        |
                    | PATH                                              :|""");
                path= scanner.next();
            }
            FileWriter file = new FileWriter(path);
            String a ="""


                    |-------------------------------------------------------------------------
--------------|
                    |                                          PATTERN
|
                    |-------------------------------------------------------------------------
--------------|
                    |                Room condition(R.C.): If 1, there is already a guest
|
                    |                           E: Expenses;          C: Customer
|
                    |   R.C  :  [1st C]  :  [2nd C]  :  [3rd C]  :  [1st C E] :  [2st C E]
:  [3st C E]  |
                    |-------------------------------------------------------------------------
--------------|
                    """;
            System.out.println(a);
            for (Cabin cabin : shipCabin) {
                String[] passengerNames = new String[3];
                String[] passengerSurNames = new String[3];
                String[] passengerExpenses = new String[3];
                int i = 0;
                for (Passenger passenger : cabin.getPassengers()) {
                    passengerNames[i]=passenger.getFirstName();
                    passengerSurNames[i]=passenger.getSurName();
                    passengerExpenses[i] = String.valueOf(passenger.getExpenses());
                    i++;
                }
                String writeDown = cabin.status() + " : [" + passengerNames[0] + " : " +
passengerSurNames[0] +
                        "] : [" + passengerNames[1] + " : " + passengerSurNames[1] + "] :
```

```
[" + passengerNames[2] + " : " + passengerSurNames[2] + "] : " +
                        passengerExpenses[0] + " : " + passengerExpenses[1] + " : " +
passengerExpenses[2] + " : " ;
                file.write(writeDown + "\n");
                System.out.println(writeDown);
            }
            file.write(a);
            file.close();
        } catch (NumberFormatException e) {
            wrongInput();
        }
    }




    //-----------------------------------------------------passenger Order-----------------
-----------------------------|
    public static void passengersOrder(Cabin[] shipCabin, Scanner scanner) {
        /*
        * When user enter the one option it will Go through the loop, and it will collect
your surname of firstname to array
        as user entered input.
        * And going through the loop and using selectionAlgorithm() method it will make as
an order the list using selection
        Algorithm
        29/03/2022
        */
        int option;
        String[] firstNames = new String[36];
        String[] sureName = new String[36];
        print("| View passengers Ordered alphabetically by name   |");
        do {
            try {
                System.out.print("""
            |   1:Ordered by FIRST NAME                          |
            |   2:Ordered by SUR NAME                            |
            |   ANS                                             :|""");
                option= Integer.parseInt(scanner.next());
                if (!(option==1||option==2)) {
                    wrongInput();
                    continue;
                }
                break;
            }catch (NumberFormatException e) {wrongInput();}
        }while (true);
        int i =0;
        for (Cabin cabin:shipCabin){
            for (Passenger passenger:cabin.getPassengers()) {
                if (!Objects.equals(passenger.getSurName(), " - ") &&
!Objects.equals(passenger.getFirstName(), " - ")) {
                    if (option == 1) {
                        firstNames[i] = passenger.getFirstName();
                    } else {
                        sureName[i] = passenger.getSurName();
                    }
```

```
                    i++;
                }
            }
        }
        System.out.println("|--------------------------------------------------|");

        if (option==1) {
            selectionAlgorithm(firstNames,i);
        }
        else {
            selectionAlgorithm(sureName,i);
        }
    }




    //--------------------------- selection Algorithm - belongs to passenger order --------
----------------------------|
    private static void selectionAlgorithm(String[] stringList,int intl) {
        /*
        * If stringList[j] string value is valuable more than stringList[i] it will swap
and make and order by going
        through the loop
        03/04/2022
        * */
        for (int i=0;i<intl;i++){
            for (int j=0;j<intl;j++) {
                if (stringList[j].compareTo(stringList[i]) > 0) {
                    String highNum = stringList[i];
                    String smallNum = stringList[j];
                    stringList[i] = smallNum;
                    stringList[j] = highNum;
                }
            }
        }System.out.print("|  ");
        for (int j=1;j<intl+1;j++){
            System.out.print("  "+stringList[j-1]+"  ");
            if (j%5==0) System.out.println(" ");
        }System.out.println("  |");
    }




    //-----------------------------------------------------total Expenses-----------------
----------------------------|
    public static void totalExpenses(Cabin[] shipCabins,Scanner scanner) {
        /*
        * Ask for user input as given below and if user input is 1 it will ask for the
cabin num
        * Then going through the loop, and it will find out sum of the expenses of
passengers of inputted cabin
        * If user input is 2 it will be going through cabins and will print sum of all
cabin expenses
```

```
        04/04/2022
         */
        System.out.print("""
                        ----------------------------------------------------
                        | Choose the option for seeking the expenses       |
                        | 1   : To see the passengers in the selected cabin|
                        | ANY : To see all the passengers                  :|""");
        try {
            if (Integer.parseInt(scanner.next()) == 1) {
                System.out.println("|----------------------------------------------------|");
                System.out.print("| Enter the cabin Number                             :|");
                int roomNum = Integer.parseInt(scanner.next());
                System.out.println("|----------------------------------------------------|");
                for (Cabin cabin : shipCabins) {
                    if (roomNum == cabin.getCabinNum()) {
                        double totalExpenses = 0;
                        System.out.println("|                        "+cabin.getMainName()+"
|");
                        for (Passenger passenger : cabin.getPassengers()) {
                            print("|           Passenger " + passenger.getPassengerNum() +
" expenses                :| "+
                                    passenger.getExpenses());
                            totalExpenses = totalExpenses+passenger.getExpenses();
                        }System.out.println("|               Total expenses
"+totalExpenses+"                |");} }
            }else {
                allOptionFind(shipCabins);
            }
        }catch (NumberFormatException ignored){
            allOptionFind(shipCabins);
        }
    }




    //------------------------------ all Option Find - belongs to total expenses ---------
-----------------------------|
    private static void allOptionFind(Cabin[] shipCabins) {
        //This is for find out sum of expenses of all passengers' in ship
        //28/03/2022
        double totalExpenses = 0;
        for (Cabin cabin : shipCabins) {
            print("|                        "+cabin.getMainName()+"                        |");
            for (Passenger passenger : cabin.getPassengers()) {
                System.out.print("|           Passenger " + passenger.getPassengerNum() + "
expenses                    :|");
                System.out.println(passenger.getExpenses());
                totalExpenses = totalExpenses+passenger.getExpenses();
            }
        }print("|               Total expenses "+totalExpenses+"                |");
    }
```

```
    //-----------------------------------------Load data from file option--------------
-----------------------------|
    public static void loadFile(Cabin[] shipCabins, Scanner scanner) {
        /*
        * You have two options as below. If your input is one it will load data from
previous file that you saved data."file.txt"
        * if your option is 2 it will give you chance to load data from file, but it must
have an order as printed msg when last
        printed msg on txt file when we store the data. If it is the program will load the
data from that file. You just
        only have to give the path to txt file and put the .txt extension
        02/04/2022
        * */
        int q = 0;
        do {
            try {
                System.out.print("""
                                        |-----------------------------------------------------|
                                        | Choose the option what you want                     |
                                        | 1: To load the previous stored data                 |
                                        | 2: To load the New File data                        |
                                        | ANY number: To go back                              |
                                        |-----------------------------------------------------|
                                        | ANS
:|""");
                int ans1 = Integer.parseInt(scanner.next());
                if (ans1 == 2) {
                    System.out.println("|-------------------------------------------------
|");
                    System.out.print("""
                                             -------------------------------------------------
-
                                             | Enter the path to .txt file
|
                                             | Eg: D:\\CW\\project\\file.txt
|
                                             | PATH
:|""");
                    String path = scanner.next();
                    initialise(shipCabins, path);
                    System.out.println("|-------------------------------------------------
|");
                    q++;
                } else if (ans1 == 1) {
                    initialise(shipCabins, "file.txt");
                    System.out.println("|-------------------------------------------------
|");
                } else q++;
            } catch (NumberFormatException | IOException | InputMismatchException e) {
                wrongInput();q++;
            }
        } while (q == 0);
    }
```

```
    //-----------------Initialise for starting and when doing the load file method --------
-----------------------------
    public static void initialise(Cabin[] shipRef, String path) throws IOException {
        /*Asking for would you like to continue the program with the previous file record?
in main class
         * If he said yes the previous record will be assigned to the cabin objects and
passenger objects
         * IF he said No all cabin objects will create with empty cabin details

        15/03/2022*/
        if (path.equalsIgnoreCase("N")){
            for (Cabin cabin:shipRef) {
                cabin.setStatus(0);
                System.out.println( "Initialise ".concat(cabin.getMainName()));}
        }
        else {
            File file = new File(path);
            Scanner scanner = new Scanner(file);

            for (Cabin cabin : shipRef){
                String[] r1 = scanner.nextLine().replace(" ","").split(":");
                cabin.setStatus(Integer.parseInt(r1[0].replace(" ","")));
                int i=1,j=2;
                int k = 7;
                for (Passenger passenger: cabin.getPassengers()){
                    if (cabin.status() ==0) {
                        passenger.setDefFirstName();
                        passenger.setDefSurName();
                        passenger.setDefExpenses();
                        continue;
                    }
                    String first = r1[i].replace("["," ").replace(" ","");
                    String second = r1[j].replace("]","").replace(" ","");

passenger.setFirstName(first.substring(0,1).toUpperCase()+first.substring(1));

passenger.setSurName(second.substring(0,1).toUpperCase()+second.substring(1));
                    passenger.setExpenses(Double.parseDouble(r1[k].replace(" ","")));
                    i+=2;j+=2;k++;
                }
                System.out.println("|-----------   Initialise: "+ cabin.getMainName()+"   -
----------|");
            }
        }
    }



    //-----------------Adding passengers to cabins from waiting list to cabins-------------
-----------------------------
    public static void waitingListInitialize2(Cabin[] shipRef, CircularQueue circularQueue)
{
        /*
        * when we have tried to add the passengers to cabin if cabin is fulled, we added
the details of passengers to waiting
        list, when we called this method last to the addCustomer and deleteCustomer methods
it will enter the passengers' data
```

```
        of in waiting list to the cabin.
        * Using circular queue this will be adding passengers to the cabins when he asked
cabin is fulled

        27/03/2022
        * */
        if (!circularQueue.isEmpty()) {
            for (Cabin cabin : shipRef) {
                if (cabin.status() == 0) {
                    if (!circularQueue.isEmpty()) {
                        System.out.println("|    Customers added from the waiting list to
:| " + cabin.getCabinNum());
                        String[] tempCabin =  circularQueue.dequeue();
                        for (Passenger passenger : cabin.getPassengers()) {
                            passenger.setFirstName(tempCabin[passenger.getPassengerNum()-
1].split(":")[0]);
                            passenger.setSurName(tempCabin[passenger.getPassengerNum()-
1].split(":")[1]);

passenger.setExpenses(Double.parseDouble(tempCabin[passenger.getPassengerNum()+2]));
                            cabin.setStatus(1);
                        }
                    }
                }
            }
        }
    }




    //-----------------------------------------------------print Options----------------------
-----------------------------|
    public static void stop() { print("|                 STOP the program
|"); }
    public static void wrongInput(){ print("|                Wrong Input type
:|"); }
    public static void print(String expression) {
        System.out.println("|-------------------------------------------------------|");
        System.out.println(expression);
        System.out.println("|-------------------------------------------------------|");
    }
}
```

2.2. Passenger Class:

```
public class Passenger {
    private String firstName;
    private final int passengerNum;
    private String surName;
    private double expenses;

    public Passenger(int cabinNameNum, int passengerNum){
        /*
        * This constructor will give and make own values of passenger when the creating the
objects
        15/03/2022
        * */
        this.passengerNum = passengerNum+1;
        System.out.println("|   made a passenger ID ("+ this.passengerNum +") for Room
number   |"+ cabinNameNum +"    |");
        System.out.println("|-------------------------------------------------|");
    }
    public int getPassengerNum() {return passengerNum;}
    public void setFirstName(String firstName1) {firstName=firstName1;}
    public String getFirstName() {return firstName;}
    public void setDefFirstName() {firstName=" - ";}
    public void setSurName(String surName1){surName=surName1;}
    public String getSurName() {return surName;}
    public void setDefSurName() {surName=" - ";}
    public void setExpenses(double Expenses1){expenses =Expenses1;}
    public double getExpenses(){return expenses;}
    public void setDefExpenses(){expenses=0.0;}

}
```

2.3. Cabin Class:

```
public class Cabin {
    private final String mainName;
    private final int cabinNum;
    private int guestsInCabin;
    private final Passenger[] passengers = new Passenger[3];

    public Cabin(int l) {
        /*This constructor will give and make own values of cabin when the creating the
objects 15/03/2022 */
        cabinNum = l+1;
        mainName = "Cabin "+(l+1);
        for (int i=0;i<3;i++) passengers[i]=new Passenger(cabinNum,i);
        System.out.println("|             made a Cabin ("+mainName+")              |");
    }
    public int getCabinNum() {return cabinNum;}
    public String getMainName() {return mainName;}
    public int status() {return guestsInCabin;}
    public void setStatus(int aStatus){ guestsInCabin = aStatus; }
    public Passenger[] getPassengers(){return passengers;}
}
```

3.Task 03 – Source Code

3.1. CircularQueue Class:

```
public class CircularQueue {
    private int  size = 2;
    private int front = -1;
    private int rear = -1;
    String[][] waitingList;
    public CircularQueue(int size){
        if (size>2) this.size = size;
        System.out.println("|       waiting list created with size of "+size+"        |");
        this.waitingList = new String[size][6];
    }

    public void enqueue(String[] cabin){
        /*
        * This method is used to change the value of rear and front when the passenger is
adding to the waiting lis.
        If waitingList is fulled it also will be show up
        15/04/2022 */
        if (isFull()) {
            System.out.println("|-----        Waiting List is full          -----|");
            return;
        }
        else if (isEmpty()) rear=front=0;
        else {
            if (rear==size-1) rear=0;
            else rear++;
        }
        waitingList[rear]=cabin;
        Main.print("|           Front = "+front+"           Rear = "+rear+"           |");
    }

    public String[] dequeue() {
        /*
        * This method is used to change the value of rear and front variables and delete
the details of passengers from the
        waiting list when their detail is added to the list
        15/04/2022
        */

        String[] x = new String[0];
        if (isEmpty()) {
            Main.print("|           Front = "+front+"           Rear = "+rear+"
|");
            return x;
        }
        else if (front==rear){
            x = waitingList[front];
            waitingList[front]=null;
            front = rear = -1;
        }
        else {
            x = waitingList[front];
            waitingList[front]=null;
```

```
            if (front==size-1) front = 0;
            else front++;
        }
        Main.print("|           Front = "+front+"           Rear = "+rear+"           |");
        return x;
    }



    public boolean isFull(){
        return size-1==Math.abs(rear-front);
    }

    public boolean isEmpty() {
        return (front==-1 && rear==-1);
    }
}
```

4.Task 04 – Testing

4.1. TEST PLAN for Task 1

Student Name: Pasindu Geevinda		Student ID: w1871471		
TEST PLAN for Task 1				
Test No.	Test Input	Expected Result	Actual Result (or state 'not attempted')	Pass /Fail ('Actual Result' matches 'Expected Result = Pass')
Initialize method				
1	Option 1 ANS = "Y" or any String~~"N"~~	All the Cabins are created with ~~empty cabins or created with~~ previous recorded information in "File.txt".	Displays "Initialize: Cabin X" (X = 1 -> 12)	Pass
2	Option 2 ANS = "N" or "n"	All the Cabins are created with empty cabins	Displays "Initialize: Cabin X" (X = 1 -> 12)	Pass
Menu Items				
1	'A' or 'a'	Empty cabins are displayed and ask for details to add passenger to a cabin	Displays	Pass
2	'V' or 'v'	All the cabins will be listed with details	Displays	Pass
3	'E' or 'e'	All the Empty cabins are displayed	Displays	Pass
4	'D' or 'd'	Delete passenger from cabin method displays	Displays	Pass
5	'F' or 'f'	Find Cabin from Customer Name method Displays	Displays	Pass
6	'S' or 's'	Store Program data into a File Method Displays	Displays	Pass
7	'L' or 'l'	Load program data from file method Displays	Displays	Pass
8	'O' or 'o'	All Passengers are Displayed in the Alphabetical Order	Displays	Pass

9	'T' or 't'	Stop the program	"FINISHED" Displays	Pass
10	4	Continue loop and print error message	"INPUT IS NOT CORRECT. TRY AGAIN" Displays	Pass
Add Passenger Method				
1	cabinNum = 1 firstName = "Pasindu" menuItem = V	Adds the passenger Pasindu to a Cabin Successfully showing Pasindu in cabin 1. (View All the Cabins)	Adds Passenger Pasindu to Cabin 1 Successfully and Displays "Pasindu added to Cabin 1" Updates data in the view all the cabins option.	Pass Pass
2	cabinNum = 1 firstName = "Kevin"	Failed to add passenger to Cabin 1 since it's already occupied. So customer will add to empty cabin using waiting list.	Displays "Waiting list" and adding passenger to a empty cabin or if all cabins are full waiting until a passenger is deleting.	Pass
3	cabinNum = "Hello"	"Wrong Input type" displayed	Displays "Wrong Input type" and going to the beginning of add passenger method	Pass
Delete Passenger from Cabin Method				
1	mainName = "Pasindu"	Customer deleted from cabin and displays "Deleted from this cabin"	Displays	Pass
2	mainName = 1	"Are you sure you want to delete this passenger?" is displayed	Displays	Pass
3	mainName = 20	"We have only 12 cabins" is displayed	Displays	Pass
4	mainName = "gfefr"	"No customer by this name was found" is displayed	Displays	Pass

5	Option 1 Ans = “Y” or “y” menuItem = V Option 2 Ans = Any string menuItem = V	“Deleted from this cabin” is displayed Doesn't show deleted customer on the cabin in view all option. “End process without deleting” is displayed Does not delete the passenger from the view all cabins option.	Displays Successfully deleted from Displays Has not been deleted from V.	Pass
Find Cabin from Passenger Name Method				
1	firstName = “Pasindu”	“Cabin 1 owned by Pasindu” is displayed	Displays	Pass
2	firstName = “Kevin” or 1	“No customer by this name was found” is displayed	Displays	Pass
Display Empty Cabins Method				
1		Displays all empty cabins like “Cabin x : empty” X = 1 -> 12	Displays	Pass
Store program data into file Method				
1	Option 1 Ans = 1	Updating the file (File.txt) with new updates.	All writing data in a text file is displayed on the console.	Pass
2	Option 2 Ans = 2 PATH = “lol.txt”	Ask for the file path to write data into inputted path file Creating file on the inputted path and writing all data into the program	Displays instruction All writing data in a text file is displayed on the console.	Pass Pass
3	Option 3 Ans = Any string	Displays “Wrong Input type”	Displays	Pass
Load program data from file Method				
1	Option 1 Ans = 1	All cabins are initializing and updating with previous file (File.txt) records.	Displays “Initialize: Cabin X” (X = 1 -> 12)	Pass

2	Option 2 Ans = 2 PATH = “lol.txt” PATH = “Amal”	Ask for file path to load data from inputted path file All cabins are initializing and updating with inputted file (File.txt) records. Displays “Wrong Input”	Displays instruction Displays “Initialize: Cabin X” (X = 1 -> 12) Displays	
3	Option 3 Ans = Any string	Displays “Wrong Input type”	Displays	Pass
View passengers Ordered alphabetically by name Method				
1		View passengers name ordered alphabetically by selection algorithm.	Displays name	Pass
To STOP the program Method				
3		Displays “Wrong Input type”	Displays	Pass

4.2. TEST PLAN for Task 2 and Task 3

Student Name: Pasindu Geevinda		Student ID: w1871471		
TEST PLAN for Task 2 and Task 3				
Test No.	Test Input	Expected Result	Actual Result (or state 'not attempted')	Pass /Fail ('Actual Result' matches 'Expected Result = Pass')
Initialize method				
1	Option 1 ANS = "Y" or any String~~"N"~~	All the Cabins are created with ~~empty cabins or created with~~ previous recorded information in "File.txt".	Displays "Initialize: Cabin X" (X = 1 -> 12)	Pass
2	Option 2 ANS = "N" or "n"	All the Cabins are created with empty cabins	Displays "Initialize: Cabin X" (X = 1 -> 12)	Pass
Waiting List				
1	4	Creating waiting list using given size and displays on console "waiting list created with size of 4"	Displays	Pass
Menu Items				
1	'A' or 'a'	Displays instruction to add for customers to a cabin	Displays	Pass
2	'V' or 'v'	All the cabins will be listed with details on all passengers	Displays	Pass
3	'E' or 'e'	All the Empty cabins are displayed	Displays	Pass
4	'D' or 'd'	Delete passenger from cabin method displays	Displays	Pass
5	'F' or 'f'	Find Cabin from Customer Name method Displays	Displays	Pass
6	'S' or 's'	Store Program data into a File Method Displays	Displays	Pass
7	'L' or 'l'	Load program data from file method Displays	Displays	Pass
8	'O' or 'o'	All Passengers are Displayed in the Alphabetical Order using selection Algorithm	Displays	Pass

9	'T' or 't'	Expenses/total Expenses Method Displays	Displays	Pass
10	'Q' or 'q'	Stop the program	"FINISHED" Displays	Pass
11	Any other input	Continue loop and print error message	"INPUT IS NOT CORRECT. TRY AGAIN" Displays	Pass
Add Passenger Method				
1		Display msg asking exit without adding passengers or not	Displays	Pass
2	Option 1 "Y" or any input Option 2 "N" or "n"	Continue the program to add customer and ask for cabin number Exit the program and go to the main menu	Displays Displays main menu	Pass
3	cabinNum = 1	Adds the passenger Pasindu to a Cabin Successfully	Adds Passenger Pasindu to Cabin 1 Successfully and Displays "Pasindu added to Cabin 1"	Pass
4	cabinNum = "ekj"	Displays "Wrong Input type" and again start from beginning of the Method by giving instructions	Displays	Pass
5	firstName = "Pasindu" lastName = "Geevinda" Expenses = 100	Adds the passenger Pasindu to a Cabin Successfully and displays "Customers added"	Displays	Pass
6	Expenses = "Pasindu"	Displays the error msg saying "INPUT IS NOT CORRECT. TRY AGAIN"	Displays	Pass
7	Option 1.1 Any input (Without "N" or "n") Option 1.2 "N" or "n"	Continue the loop and add passengers like did before until i variable got the value of 3. After adding the maximum number of 3 passengers, the loop is end and the cabin will be occupied. will be breaking the loop and will show the main menu	Displays "Customers added" Displays "Customers added"	Pass

8	cabinNum = 1	Displays “adding to waiting list” and passenger will add to the waiting list like test case 6 and 7. If there any empty cabin. They will add to that cabin	Displays	Pass
Delete Passenger from Cabin Method				
1	mainName = “Pasindu”	“Are you sure you want to delete this passenger?” is displayed	Displays	Pass
3	Option 1 Ans = “Y” or “y” menuItem = V	If inputted passenger’s name is equal to any first or sure name in a cabin, Passengers will delete from the cabin and display (X = 1->3) “Found the cabin number” “No X customer is deleted” Doesn't show deleted customer on the cabin in view all option.	Displays Successfully deleted from V.	Pass Pass
4	Option 2 Ans = Any string (Without “Y” or “y”) menuItem = V	“End process without deleting” is displayed Does not delete the passenger from the view all cabins option.	Displays Has not been deleted from V.	Pass Pass
5	mainName = 20	“We have only 12 cabins” is displayed	Displays	Pass
6	mainName = “mug”	“No customer by this name was found” is displayed	Displays	Pass
Find Cabin from Passenger Name Method				
1	mainName = “Pasindu”	“Cabin 1 owned by Pasindu” is displayed	Displays	Pass
2	mainName = “Kevin” or 1	“No customer by this name was found” is displayed	Displays	Pass
Store program data into file Method				
1	Option 1 Ans = 1	Updating the file (File.txt) with new updates.	All writing data in a text file is displayed on the console.	Pass

2	Option 2 Ans = 2 PATH = “lol.txt”	Ask for the file path to write data into inputted path file Creating file on the inputted path and writing all data into the program	Displays instruction All writing data in a text file is displayed on the console.	Pass Pass
3	Option 3 Ans = Any string	Displays “Wrong Input type”	Displays	Pass
Load program data from fileMethod				
1	Option 1 Ans = 1	All cabins are initializing and updating with previous file (File.txt) records.	Displays “Initialize: Cabin X” (X = 1 -> 12)	Pass
2 2.1 2.2	Option 2 Ans = 2 PATH = “lol.txt” PATH = “Amal”	Ask for file path to load data from inputted path file All cabins are initializing and updating with inputted file (File.txt) records. Displays “Wrong Input”	Displays instruction Displays “Initialize: Cabin X” (X = 1 -> 12) Displays	Pass Pass Pass
3	Option 3 Ans = Any string	Displays “Wrong Input type”	Displays	Pass

5.Task 04 – Testing – Discussion

5.1. how I chose my test cases to ensure that my tests cover all aspects of my program

We are knowing what we want input for results and what are the scope and rules for the program. But the user doesn't know. So we can give instructions to users to do tasks and follow the rules. But we don't know how are they going to work with this program. Sometimes they would do tasks out of instructions. We can't blame them because users thinking range would be able different from ours. So we have to manage our program, thinking about all users' think range. So we want to create test cases thinking about that. Also, I thought about all the possibilities it gives an error or the wrong program work wrongly. As an example, where there should be an Integer variable, but accidentally, sometimes users can input string data. Cause of that, the program will crash giving an error. So I had to think like users. Also, I had to ensure the program steps are working correctly and if it gives an error, it must be shown to the user without crashing the program and he must have again chance to do those things correctly. So I have followed these steps and written test cases as thinking as users and had been giving correct instructions to users. I checked all the test cases and correct the errors or wrong logic until the test cases are passed and ensured my program steps were working correctly and user-friendly.

5.2. Which version is better? Class solution or Array solution?

Object-oriented programming, or OOP, refers to a unique approach to solving all computational problems using objects. Objects are the basic units of object-oriented programming. In the case of a built-in program, we will take the example of a cabin that can accommodate up to 3 passengers, with information on one passenger. If using an array, it will have to enter the information related to 12 cabins and re-enter the information of 3 passengers in it as well. There you will have to face a lot of trouble in entering, filing, and retrieving data. Similarly, if something goes wrong, it is difficult to find.

The code of object-oriented programming is organized around objects. They can interact with each other to do something by using a variable that is unique to that class, filing relevant values and retrieving them at the desired time, and by using methods and actions related to that object. As I mentioned earlier, the class solution is easy to use with the object, but when it comes to the Array solution, 3 passengers have to be inserted into the cabin, making the array difficult and confusing to handle. Also can't easily read and understand the data included in the array. But in the case of the class solution, we can easily understand and read the data using objects and accessing values and methods. So for the class solution, I created 12 cabin objects using cabin class and 3 objects using separate passenger class and assign separate values to them, and accessed that value using the object and using it to easily and quickly handle data.

Also after assigning some object values I do not need to change them again. Example for that I didn't want to have to edit again the value of the cabin that I named variable mainName. But in an array solution, it is possible

to edit it. I added mainName to the first index position of Array. But if accessed accidentally it and edited it my program would have crashed. We can avoid that risk conditions in class solutions. Because we can modify their access using encapsulation in OOP. As an example for that when initialized the program had created cabins and assigned the value to the String variable mainName. And I don't wanted to edit that again. So I put access modifier privet to the mainName and used only the getter method. So I couldn't be able to edit it. Considering all of that, I could be able to have combined all datas together and easily handled it in class solution. So finally I consider the class solution to be easier than the Array solution.

6.Self-Evaluation form

Criteria	Component marks	Expected Mark
Task 1: One mark for each option (A, V, E, D, F, S, L, O) Menu works correctly	24 6	24 6
Student comment: fully Implemented and Working *[When you start the program will ask to load previous data from previously data stored file. If you answer "Y" it will help to check the functionalities.]*		
Task 2: Cabin class correctly implemented. Passenger class correctly implemented. Expenses correctly reported.	14 10 6	14 10 6
Student comment: fully Implemented and Working *[When you start the program will ask to load previous data from previously data stored file. If you answer "Y" it will help to check the functionalities]*		
Task 3: Waiting list queue implementation **6.1.** "A: Add" works correctly 6.2. "D: Delete" works correctly 6.3. Circular queue implementation	10 3 3 4	10 3 3 4
Student comment: fully Implemented and Working *[You have to give size for Waiting List]*		
Task 4: Test case coverage and reasons Writeup on which version is better and why	6 4	6 4
Student comment: fully Implemented		
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: fully Implemented		
Totals		**(100)**
Demo: At the discretion of your tutor, you may be called on to give a demo of your work to demonstrate understanding of your solutions. If you cannot explain your code and are unable to point to a reference within your code of where this code was found (i.e., in a textbook or on the internet) then significant marks will be lost for that marking component. If you do not attend a requested demo your mark will be capped at 50%.		

7. References

-