Western Cape Education Department
Common Examination

SEPTEMBER EXAMINATION – 2015

INFORMATION TECHNOLOGY

PAPER 1 – PRACTICAL

TIME: 3 HOURS MARKS: 150

This paper consists of 16 pages, and there is one NetBeans project in your exam folder. Make sure that you have a complete set.

INSTRUCTIONS and INFORMATION:

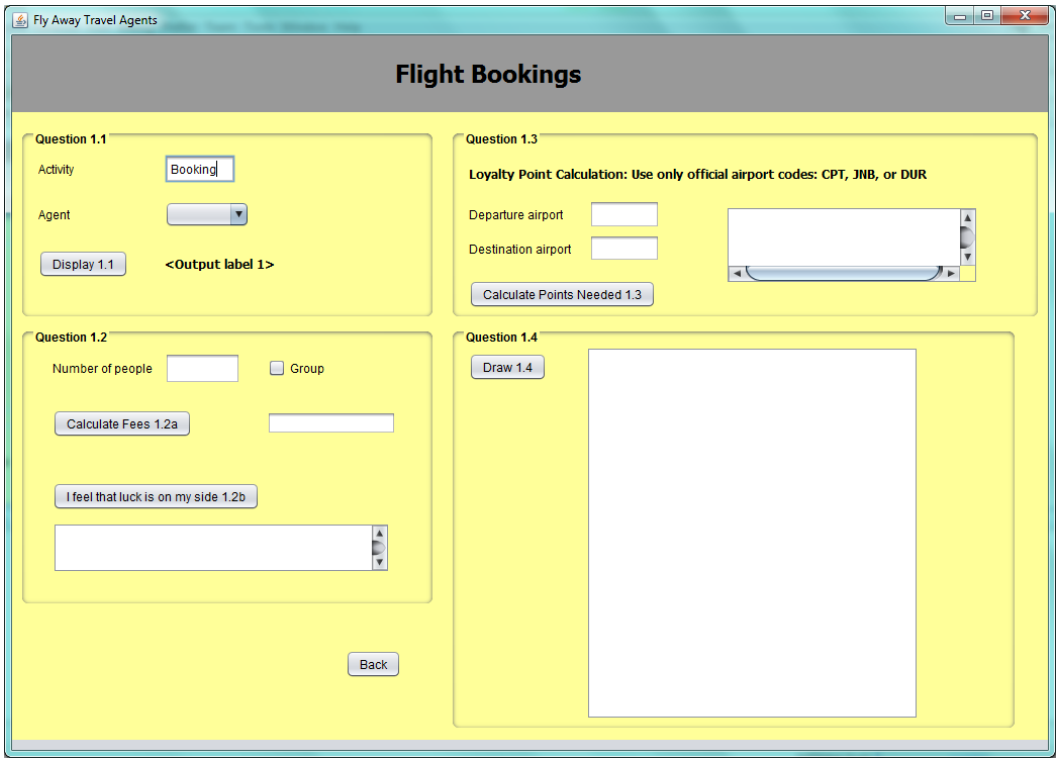
1. THIS IS A THREE-HOUR EXAMINATION. BECAUSE OF THE NATURE OF THIS EXAMINATION IT IS IMPORTANT TO NOTE THAT YOU WILL **NOT** BE ALLOWED TO LEAVE THE EXAMINATION ROOM BEFORE THE END OF THE EXAMINATION SESSION.
2. Log on and open your exam folder. You require the files listed below in order to answer the questions.
   Question 1: NetBeans project **FATAApp** – form **Question1UI**
   Question 2: NetBeans project **FATAApp** - form **Question2UI**, class **Booking**, and the **BookingData.txt** text file
   Question 3: NetBeans project **FATAApp** – form **Question3UI**
3. The mark allocation for the 3 questions is indicated below. Use the mark allocation to plan your time wisely.
   Q1 – (47) Q2 – (62) Q3 – (41)
4. Save ALL your work in your exam folder under the required names.
   NB: Only programs saved in your exam folder will be marked!
5. Save your work regularly as a precaution against power failures.
6. You may make use of the Java's API files. You may NOT consult any other resource material.
7. Type your name and the question number as a comment in the first line of each program, e.g. // Joe Soap, Q. 2.
8. **Attempt ALL questions**.
9. Read the requirements for each program thoroughly and understand the details before starting to program. Do ONLY what is required by each question.
10. In ALL programs marks will be allocated to functional parts, even if the whole program does not work (or hasn't been completed). So make sure that you complete each step before starting the next one.
11. Good luck!

SCENARIO:

Tourism is one of the cornerstones of the South African economy. It creates many jobs, both in the tourism industry (hotels, travel agencies, tour guides, transport companies, etc.) and in the programming industries as many apps have to be written to process the many tasks that make up the tourism industry.

Fly Away Travel Agents (FATA) have commissioned a few apps to be coded.

Question 1: Flight Bookings – General Programming

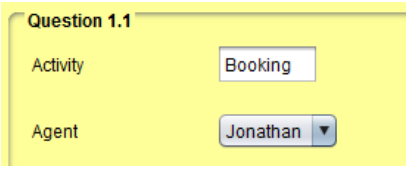


1.1 **Button [Display 1.1]**

When the user arrives at the FATA he/she has to choose the type of activity he/she has come for ("Booking" has been supplied as the default value) plus the agent that he/she would like to see. For that the user has to select an agent's name from the combo box.

Obtain the type of activity and the selected agent's name from the provided components. Create and display a line of text as output that indicates the type of activity in capitals and the selected agent's name, as shown in the example below.

Example of possible input:



Example of possible output:

(5)

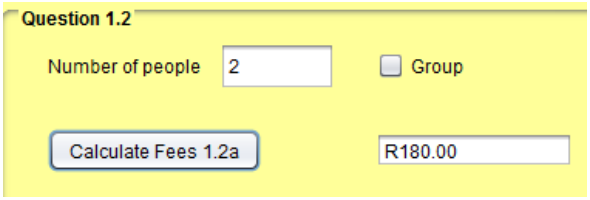
1.2 **Button [Calculate Fees 1.2a]**

(a) The agent's fees are calculated as follows:

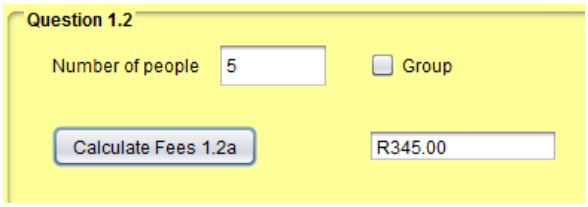
1 person	R100.00
2 people booking together	R100 + R80.00
3 people booking together	R100 + R80 + R65.00
Any other people in the same booking	+ R50.00 each

- Enter the number of people the booking is for.
- Make use of the global variable "fees" provided.
- Calculate the total fees charged.
- As the fees can get quite high, especially for groups, it has been decided to introduce a capping (limiting) of the fees to R500.00. When the check box "Group" gets ticked, the fees should be no more than R500.00.
- Format the amount as currency with an "R" in front and 2 decimal places.
- Display the calculated fees in the text box provided.

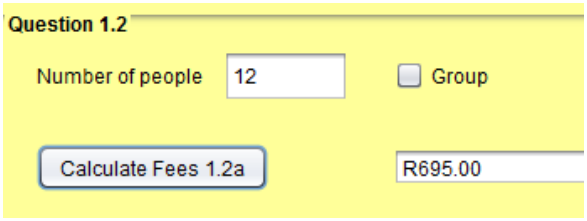
Example 1: Number of people is 2.



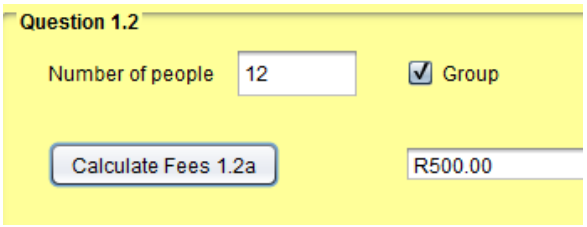
Example 2: Number of people is 5.



Example 3: Number of people is 12.



Example 4: Number of people is 12 and "Group" has been checked.



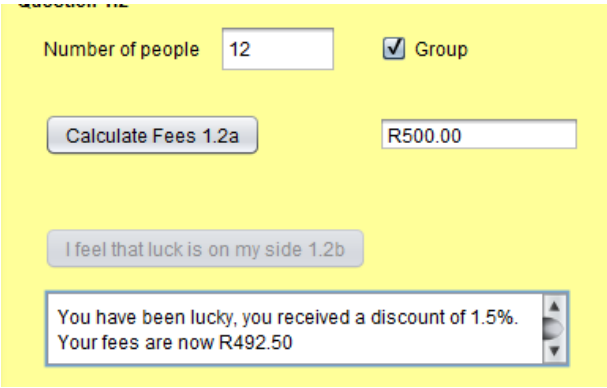
(12)

1.2 (b) **Button [I feel that luck is on my side 1.2b]**

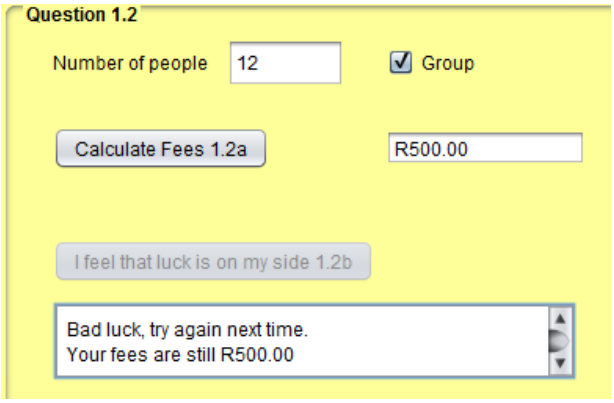
When the user presses the button "I feel that luck is on my side" the fees could get reduced. There must be 20% chance for a 3% discount and a 20% chance for a 1.5% discount. This means that there is a 60% chance that no discount will be granted and the fees will stay the same.

- Create and display two lines of text as output that indicate whether the user is lucky (including the discount granted and the final fees) or not as shown in the examples below.
- To ensure that the user can click this button only ONCE (i.e. not continue until he/she gets a maximum discount), the button must be deactivated after one click.

Example 1: The user is granted a discount.



Example 2: The user is not granted a discount.



(13)

1.3 **Button [Calculate Points Needed 1.3]**

Airlines have loyalty programmes that reward you with points if you are flying often. These points can then be used to "pay" for other flights. Depending on the distances you want to fly, you need a certain number of points.

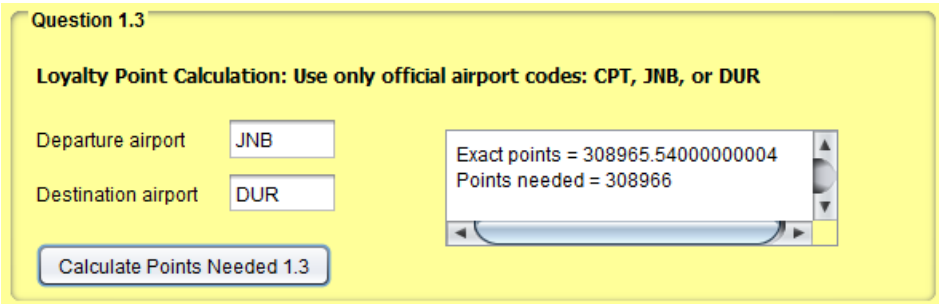
It requires exactly **321.17 points per kilometre** to be flown.

Using the 2-dimensional array "flightInfo" with data given (flight origin and destination, distance in kilometres), work out how many points the user will need for the flight.

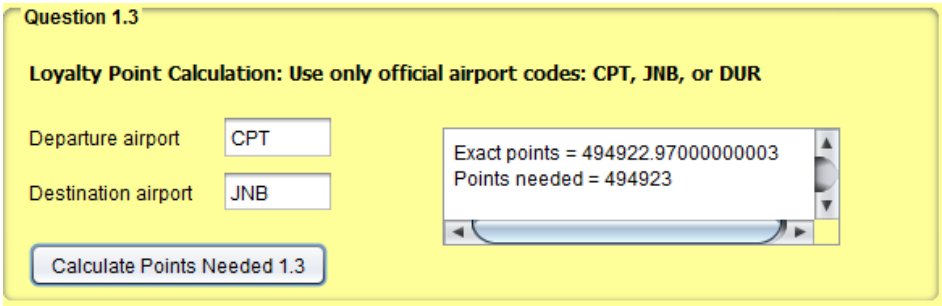
```
String[][] flightInfo = {   {"CPT - JNB", "1541"},
                            {"CPT - DUR", "1973"},
                            {"JNB - DUR", "962"}     };
```

- Read in the departure and destination airports (using only the airport codes given in the data structure) and determine how far the flight will be. It does not matter which way the flight is going, from Cape Town to Durban or from Durban to Cape Town; obviously the distance will be the same. So it should not matter in which order the two airports are entered.
- Display a warning message if the user accidentally enters the same airport twice.
- Calculate how many points the user will require (321.17 points per kilometre): once exactly (to any amount of decimals) and then rounded up to nearest whole number.
- Display the points required as in the examples.

Example 1: JNB and DUR entered



Example 2: CPT and JNB entered



(12)

1.4 **Button [Draw 1.4]**

In order to visualise the layout of the aeroplane, code has been provided to program a layout diagram and display it in the text area.

- This aeroplane has 33 rows. Each row must be numbered on the aisle, i.e. in the middle.
- Each row has 6 seats.
- The seats are numbered A to F from left to right as you walk into the plane at the front, i.e. NOT the way you sit facing forward.
- There is an aisle/passage down the middle of the plane.
- The wings are at rows 14, 15 and 16 and must be indicated on each side.

When the given code is executed an incorrect pattern is displayed due to errors and/or omissions in the code. **Correct the code so that the layout of the aeroplane will display as in the example below.**

NOTE: No marks will be allocated if you code a new solution!

Example of correct layout of the aeroplane:

```
ENTRANCE        COCKPIT
             F E D   1  C B A
             F E D   2  C B A
             F E D   3  C B A
             F E D   4  C B A
             F E D   5  C B A
             F E D   6  C B A
             F E D   7  C B A
             F E D   8  C B A
             F E D   9  C B A
             F E D  10  C B A
             F E D  11  C B A
             F E D  12  C B A
             F E D  13  C B A
    WING     F E D  14  C B A     WING
    WING     F E D  15  C B A     WING
    WING     F E D  16  C B A     WING
             F E D  17  C B A
             F E D  18  C B A
             F E D  19  C B A
             F E D  20  C B A
             F E D  21  C B A
             F E D  22  C B A
             F E D  23  C B A
             F E D  24  C B A
             F E D  25  C B A
             F E D  26  C B A
             F E D  27  C B A
             F E D  28  C B A
             F E D  29  C B A
             F E D  30  C B A
             F E D  31  C B A
             F E D  32  C B A
             F E D  33  C B A
                   TAIL
```

(5)

Total Question 1 = [47]

Question 2: Processing Hotel Bookings – Object-oriented Programming

The second app is designed to process hotel bookings.

2.1 An object class called **Booking** which represents a booking with some methods has been provided.

Complete the code in the given booking class (**Booking**) as described in Questions 2.1.1 to 2.1.6 that follow.

2.1.1 Declare the private attributes of the class using the information in the list given below:

firstName	The first name of the guest
surname	The surname of the guest
idNum	The ID number of the guest
gender	The character f or m to indicate the gender of the guest
paid	Whether the guest has paid or not, must be set to **false**
dailyRate	The daily rate charged by the hotel, must be set to R**895.00**.

(5)

2.1.2 Write code for a constructor that receives the guest's first and surnames, the ID number and gender as parameters.

Initialise the attributes of the class using the parameters. (1)

2.1.3 Remove the comment symbols from the assessor and mutator methods supplied. (1)

2.1.4 Write a method called "paymentMade" that returns either the word "Paid" if the guest has paid his/her account, or the words "Not paid" if the guest has not paid his/her account in full. (3)

2.1.5 Write a **toString** method to construct a string which contains the information on a booking in the format shown below: (5)

```
Name: <first name and surname of guest>
ID Num: <ID number of guest>
Gender: <gender of guest>
Payment: <Paid or Not paid>
```

When this method is called with the first dataset, it will display as follows:

```
Name: Ken Green
ID Num: 5609155123088
Gender: m
Payment: not paid
```

2.1.6 Write a method called "amountPayable" that will accept as a parameter the number of days the guest will stay/has stayed at the hotel and that will return the amount due. The amount will be calculated by multiplying the number of days by the daily rate. (3)

Subtotal 2.1 [18]

2.2 The text file **BookingsData.txt** contains data to process the bookings of guests at a hotel.

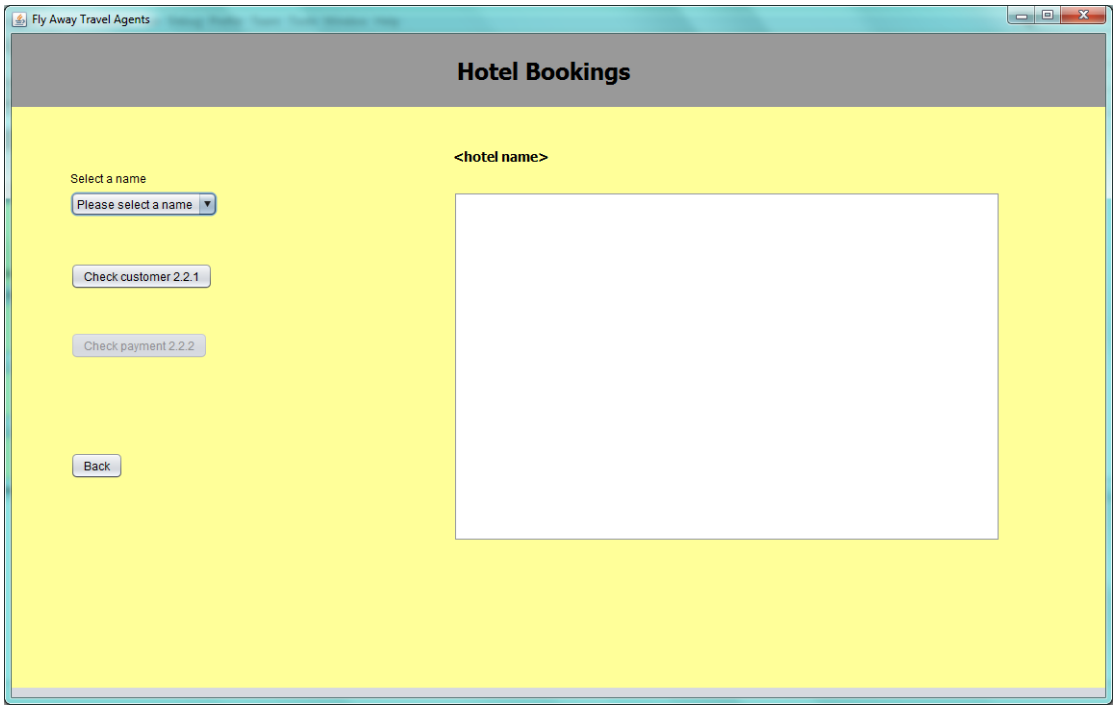
The name of the hotel is stored in the first line of the text file. Each line of text thereafter contains the following information:

<first name of guest>#<surname of guest>#<ID number of guest>,<gender of guest ('f' or 'm')>

Example of the first six lines of the data stored in the text file:

```
Beach Hotel, Mauritius
Ken#Green#5609155123088,m
Raymond#Sebe#8503305254086,m
Simona#Moonsamy#8702010296084,f
Richard#Pillay#4905120236081,m
Deon#Smith#5906065194082,m
:
```

An example of the GUI for Question 2:



Do the following to complete the code for the buttons in the GUI class as described below:

2.2.1 Obtain the surname of the guest who has stayed/is staying at the hotel from the combo box provided.

If the user did not select a name from the combo box, a suitable message must be displayed and the program must NOT proceed.

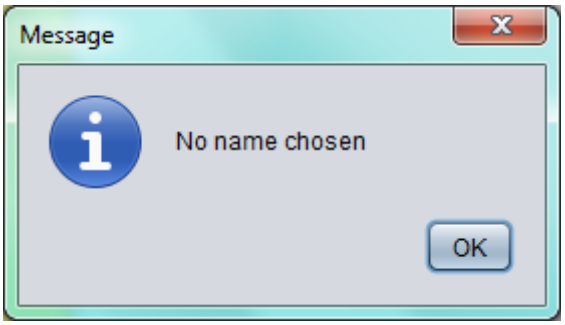
The program must check whether the text file **BookingsData.txt** does exist.

Display a suitable message if the text file does not exists. The program must NOT proceed further.

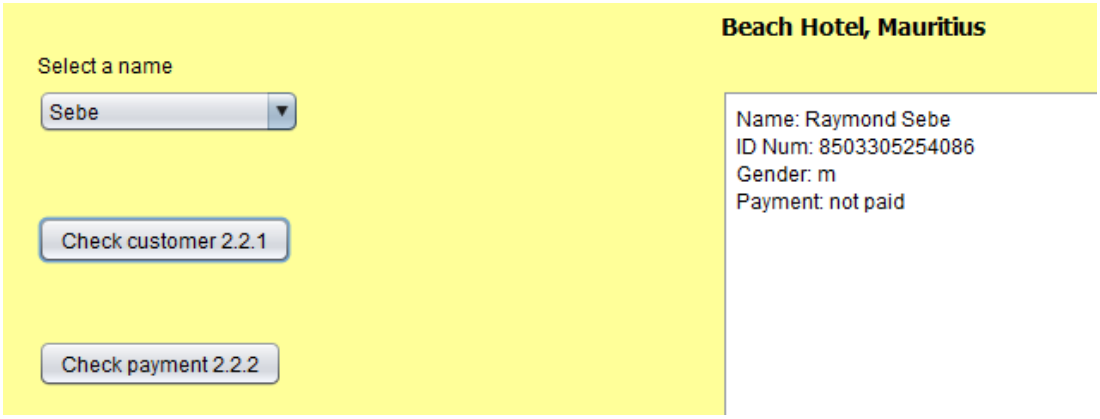
The program must do the following if the text file exists:

- Read the name of the hotel from the file and display the name in the label component provided.
- Use a conditional loop and search the text file for the name of the guest that was obtained from the combo box.
  - If the name is found:
    - Use the data from the line of text to instantiate a new Booking object. Use the **bookingItem** object that has been declared globally as part of the given code to store the object.
    - Display the details of the booking using the toString() method.
    - Enable the button for Question 2.2.2.
  - If the name is NOT found, display a suitable message and disable the button for Question 2.2.2. (28)

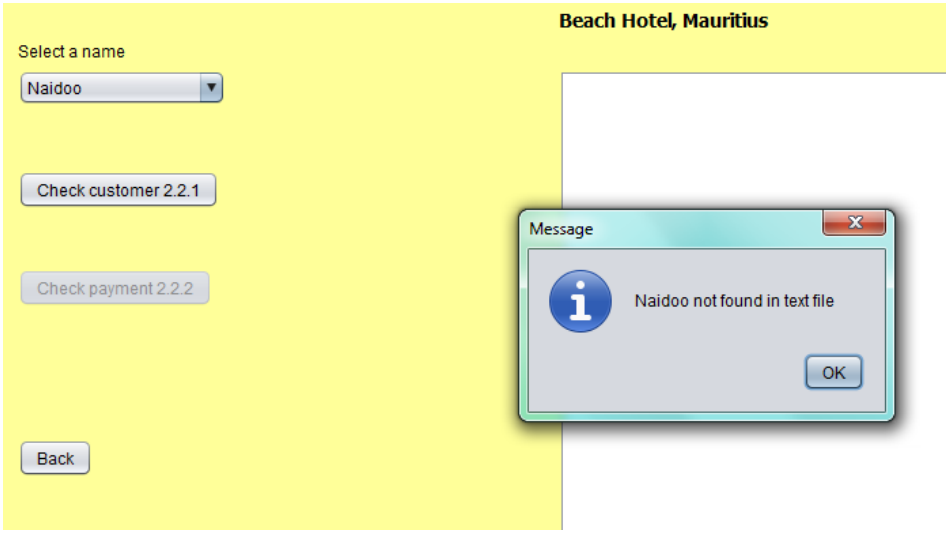
Example of output if no name was selected:



Example of output if the name is found in the text file:



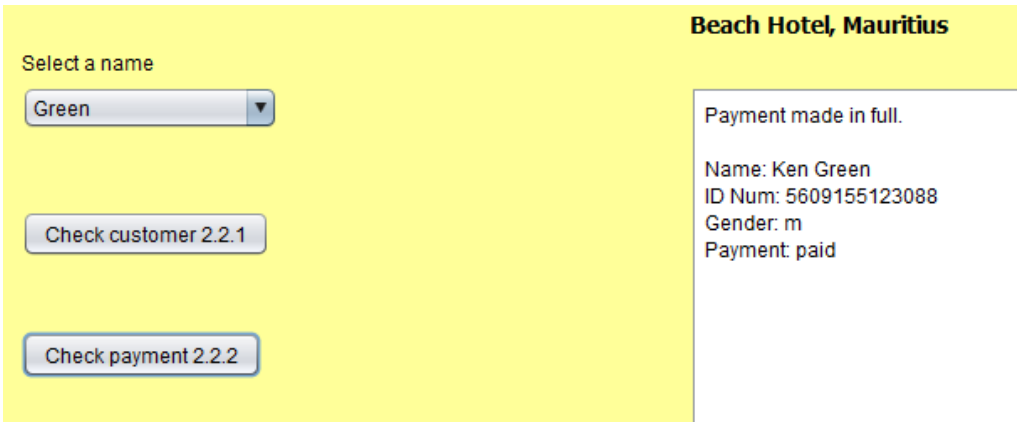
Example of output if the name is not found in the text file:



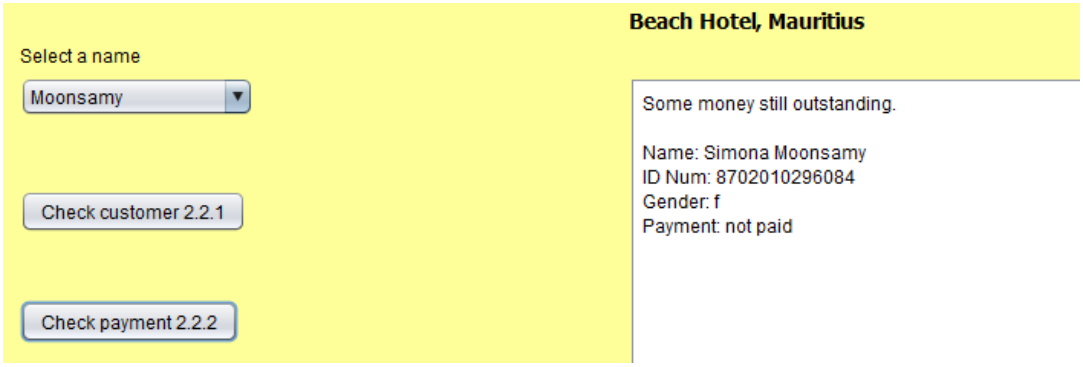
2.2.2 If a payment has not been made already (remember, by default none of the guests have paid), do the following:

- Using a dialog box (JOptionPane) ask the user to enter the number of days/nights the guest has stayed.
- Using the appropriate method of the bookingItem object, inform the user of how much the guest has to pay using a dialog box.
- With the same or another dialog box ask the user to indicate how much the guest is paying.
- If the amount paid is the full amount owing (or more), update the object's paid attribute.
- Clear the text area, write either "Payment made in full" or "Some money still outstanding" and display the status of the object with the toString() method. (16)

Example of output if paid in full:



Example of output if not paid in full:



Subtotal 2.2 [44]

Total Question 2 = [62]

Question 3: Processing Bookings – Problem Solving

The third app is a "booking engine", an app that allocates bookings and allows the user to add more bookings.

A number of bookings have been received. These bookings have to be allocated to the 10 rooms of the Beach Hotel.

NOTE: Read the following sections carefully before attempting to answer this question:

- GUI and DATA supplied
- Instructions
- Program requirements
- Mark allocation

Data for the bookings already received for September is supplied in an array called septemberBookings containing 20 text strings.

Each text string contains THREE parts, separated by commas: the number of nights the guest wants to stay, the date of the first night at the hotel, the surname of the booking.

Thus the first line of the array is

```
"3,2,Jones"
```

And means that Mr Jones has booked a room for 3 nights, starting on the 2nd of September.

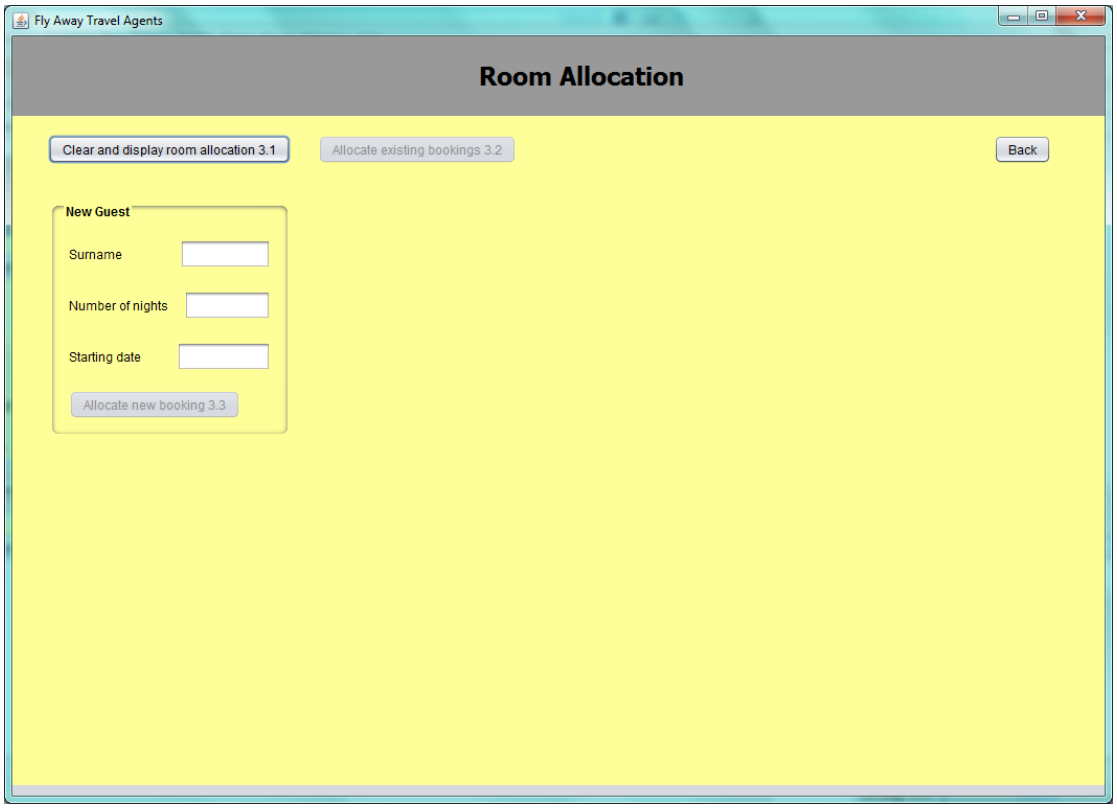
You are also provided with a two-dimensional array to be used for the allocation of bookings:

```
String[][] allocations = new String[15][10];
```

NOTE:

- You are NOT allowed to modify the content of the supplied array!
- You may add any further data structures and/or classes necessary for your solution.
- The use of good programming techniques and modular design must be applied in the design and coding of your solution.
- The GUI that is supplied contains components for user input only. You have to add suitable components as you see fit. Use suitable descriptive names. (6)

Example of GUI supplied:



Open the Question3UI.java file and write code to answer the questions according to the requirements stated in the sections that follow.

It is important that the three buttons only become available as the program is executed. Guide the user by enabling the buttons one-by-one.

3.1 Button [Clear and display room allocation 3.1]

- Add an output component of your choice to the GUI.
- Using the two-dimensional array supplied, program the display of an empty room list, similar to the example below: (14)

```
Date  Room 1   Room 2   Room 3   Room 4   Room 5   Room 6   Room 7   Room 8   Room 9   Room 10
1     -        -        -        -        -        -        -        -        -        -
2     -        -        -        -        -        -        -        -        -        -
3     -        -        -        -        -        -        -        -        -        -
4     -        -        -        -        -        -        -        -        -        -
5     -        -        -        -        -        -        -        -        -        -
6     -        -        -        -        -        -        -        -        -        -
7     -        -        -        -        -        -        -        -        -        -
8     -        -        -        -        -        -        -        -        -        -
9     -        -        -        -        -        -        -        -        -        -
10    -        -        -        -        -        -        -        -        -        -
11    -        -        -        -        -        -        -        -        -        -
12    -        -        -        -        -        -        -        -        -        -
13    -        -        -        -        -        -        -        -        -        -
14    -        -        -        -        -        -        -        -        -        -
15    -        -        -        -        -        -        -        -        -        -
```

3.2 **Button [Allocate existing bookings 3.2]**

This button may only be active once the data structures have been cleared, i.e. the previous button has been clicked.

Write code to read in the data from the array and allocate the bookings to the rooms so that there are no overlapping bookings. **ANY solution that allocates all bookings will be correct!**

NOTE:

- The data supplied will allow for a solution without errors as shown in the possible output below.
- Your output may look different depending on your solution. (14)

Example of output after the execution of the [Allocate existing bookings] button:

Day	Room 1	Room 2	Room 3	Room 4	Room 5	Room 6	Room 7	Room 8	Room 9	Room 10
1	Nick	-	Jack	Haigh	-	-	-	-	Rudd	-
2	Jones	Pieters	Jack	Haigh	-	-	-	-	Rudd	-
3	Jones	Pieters	Jack	Haigh	Leppers	-	Zuma	-	Rudd	Mabida
4	Jones	Pieters	Naidoo	Haigh	Leppers	-	Zuma	Thomas	Rudd	Mabida
5	-	Pieters	Naidoo	Haigh	Leppers	-	Zuma	Thomas	Rudd	Mabida
6	Ebrahim	Pieters	Naidoo	Ally	Leppers	Smith	Zuma	Thomas	Rudd	Mabida
7	Khumalo	-	Naidoo	Ally	Sebe	Smith	Zuma	Thomas	Rudd	Mabida
8	Khumalo	Morris	Naidoo	Ally	Sebe	Smith	Zuma	-	Rudd	Mabida
9	Khumalo	Morris	Naidoo	Ally	Sebe	Smith	Zuma	Helper	Rudd	Mabida
10	Khumalo	Morris	Omar	-	Sebe	Smith	Zuma	Helper	-	-
11	-	Morris	Omar	-	Sebe	-	Zuma	Helper	-	-
12	Obama	-	Omar	-	Sebe	-	Zuma	Helper	-	-
13	-	-	-	-	Sebe	-	-	Helper	-	-
14	-	-	-	-	-	-	-	Helper	-	-
15	-	-	-	-	-	-	-	-	-	-

3.3 **Button [Allocate new booking 3.3]**

Only when the allocation of existing bookings (from the array of data) has been made, may this button become active.

Write code to read in the data for a new booking supplied by the user.

You may assume that the user has entered everything correctly – and has in fact checked visually that there is in fact a room available for the period requested.

The program must then allocate a room to the user (probably by the same method as done before) and update the display to reflect this new booking.

Example of output after the following data has been entered: Yoko, 4, 10. The frame around the booking has been added merely so that you can easily see where the booking has been allocated. This should NOT form part of your solution! (7)

Example of output after the following data has been entered: Yoko, 4, 10.

Day	Room 1	Room 2	Room 3	Room 4	Room 5	Room 6	Room 7	Room 8	Room 9	Room 10
1	Nick	-	Jack	Haigh	-	-	-	-	Rudd	-
2	Jones	Pieters	Jack	Haigh	-	-	-	-	Rudd	-
3	Jones	Pieters	Jack	Haigh	Leppers	-	Zuma	-	Rudd	Mabida
4	Jones	Pieters	Naidoo	Haigh	Leppers	-	Zuma	Thomas	Rudd	Mabida
5	-	Pieters	Naidoo	Haigh	Leppers	-	Zuma	Thomas	Rudd	Mabida
6	Ebrahim	Pieters	Naidoo	Ally	Leppers	Smith	Zuma	Thomas	Rudd	Mabida
7	Khumalo	-	Naidoo	Ally	Sebe	Smith	Zuma	Thomas	Rudd	Mabida
8	Khumalo	Morris	Naidoo	Ally	Sebe	Smith	Zuma	-	Rudd	Mabida
9	Khumalo	Morris	Naidoo	Ally	Sebe	Smith	Zuma	Helper	Rudd	Mabida
10	Khumalo	Morris	Omar	Yoko	Sebe	Smith	Zuma	Helper	-	-
11	-	Morris	Omar	Yoko	Sebe	-	Zuma	Helper	-	-
12	Obama	-	Omar	Yoko	Sebe	-	Zuma	Helper	-	-
13	-	-	-	Yoko	Sebe	-	-	Helper	-	-
14	-	-	-	-	-	-	-	Helper	-	-
15	-	-	-	-	-	-	-	-	-	-

Example of output after the following data has been entered: Walters, 5, 1.

Day	Room 1	Room 2	Room 3	Room 4	Room 5	Room 6	Room 7	Room 8	Room 9	Room 10
1	Nick	-	Jack	Haigh	-	Walters	-	-	Rudd	-
2	Jones	Pieters	Jack	Haigh	-	Walters	-	-	Rudd	-
3	Jones	Pieters	Jack	Haigh	Leppers	Walters	Zuma	-	Rudd	Mabida
4	Jones	Pieters	Naidoo	Haigh	Leppers	Walters	Zuma	Thomas	Rudd	Mabida
5	-	Pieters	Naidoo	Haigh	Leppers	Walters	Zuma	Thomas	Rudd	Mabida
6	Ebrahim	Pieters	Naidoo	Ally	Leppers	Smith	Zuma	Thomas	Rudd	Mabida
7	Khumalo	-	Naidoo	Ally	Sebe	Smith	Zuma	Thomas	Rudd	Mabida
8	Khumalo	Morris	Naidoo	Ally	Sebe	Smith	Zuma	-	Rudd	Mabida
9	Khumalo	Morris	Naidoo	Ally	Sebe	Smith	Zuma	Helper	Rudd	Mabida
10	Khumalo	Morris	Omar	Yoko	Sebe	Smith	Zuma	Helper	-	-
11	-	Morris	Omar	Yoko	Sebe	-	Zuma	Helper	-	-
12	Obama	-	Omar	Yoko	Sebe	-	Zuma	Helper	-	-
13	-	-	-	Yoko	Sebe	-	-	Helper	-	-
14	-	-	-	-	-	-	-	Helper	-	-
15	-	-	-	-	-	-	-	-	-	-

MARK ALLOCATION for Question 3:

Requirements	**Marks**
Components and application of good programming techniques and modular programming	6
Question 3.1 – Clear and display room allocation	14
Question 3.2 – Allocate the existing bookings and display	14
Question 3.3 – Allocate new bookings and display	7
Total Question 3 =	**[41]**

GRAND TOTAL = 150