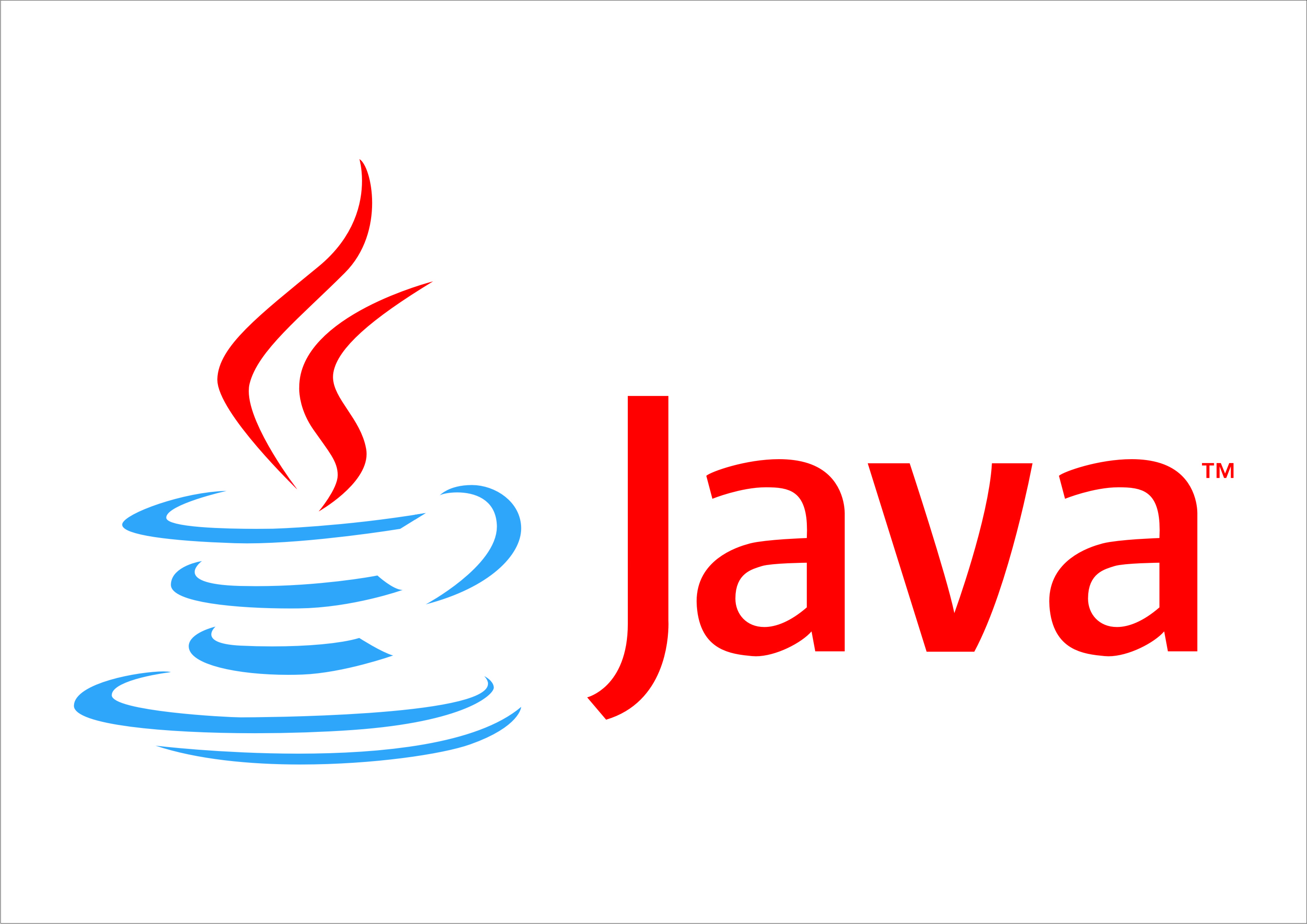


CIS016-1 Assignment 3

Hotel Booking System  
Group Report



**Group members:**

### Authors:

# Introduction

This report will discuss the solution to the CIS016-1 Group Project. The brief outlined that a hotel booking system was to be created**.** Any assumptions about what would be required within the system were to be identified too. The system was also required to be modelled using some UML techniques. A proof of concept program was also to be implemented, of which would act as a prototype version of the system itself. The brief also outlined the types of users who are expected to interact with the system, of which includes registered Customers, the Administrators of the system itself. They handle any of the bookings that are made or altered in some way.

Java was to be used as the programming language for this group project. Some of the basic requirements that were outlined for this system included the following:

* The system must store the following information about its users:
  + Name
  + Contact Details (This includes their phone number and email address).
  + Card Details
  + Whether the user is a customer or administrator.

# System Design

Below a series of UML diagrams have been included, which help to demonstrate the functionality of the program for this project. Explanations have been included for each of the diagrams.

## UML Use Case Diagram

Below the use case diagram indicates how each of the actors are able to interact with the hotel booking system, which was created in this group project. The actors include unregistered customers and administrators, as well as registered customers and administrators.

Notice how the users that are unregistered can only interact with the system in one way, and that is to create an account. It is the only action that is possible for them to do in the hotel booking system.

On the other hand, users which are registered have multiple actions that are possible for them to do. It is here that some assumptions were first made. It was assumed that the hotel booking system would allow a user to make more than one booking at one time, providing there are suitable rooms that are available to use on the dates that are requested by the customer. It should also be noted that a room may be available whilst Customer A is in the process of creating their booking, however if Customer B were to complete their booking for the same type of room, and on the same dates, then CustomerB would be issued the room.

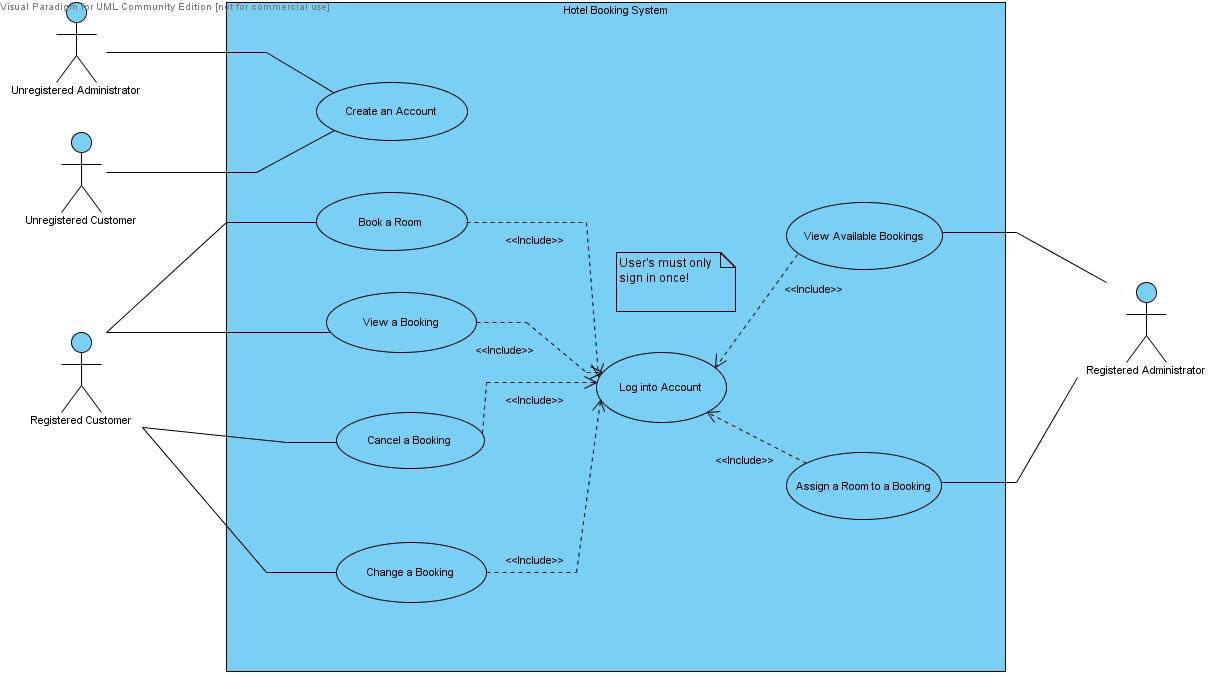


Figure 1 - Use Case Diagram, Hotel Booking System

## BLUEJ CLASS DIAGRAM

Below a screenshot from the BlueJ IDE shows the relationships between the eight classes within this project.

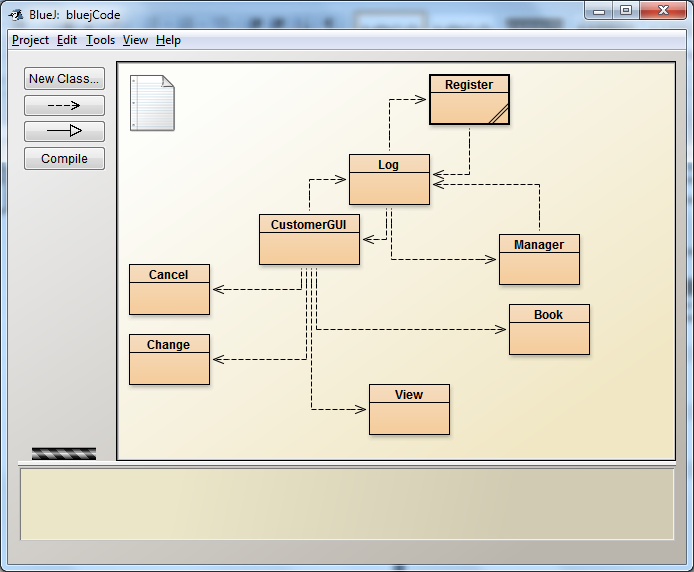


Figure 3 - Relationship between Classes

## UML Class Diagram

An UML Class Diagram showing the relationships between classes, their variables and methods:

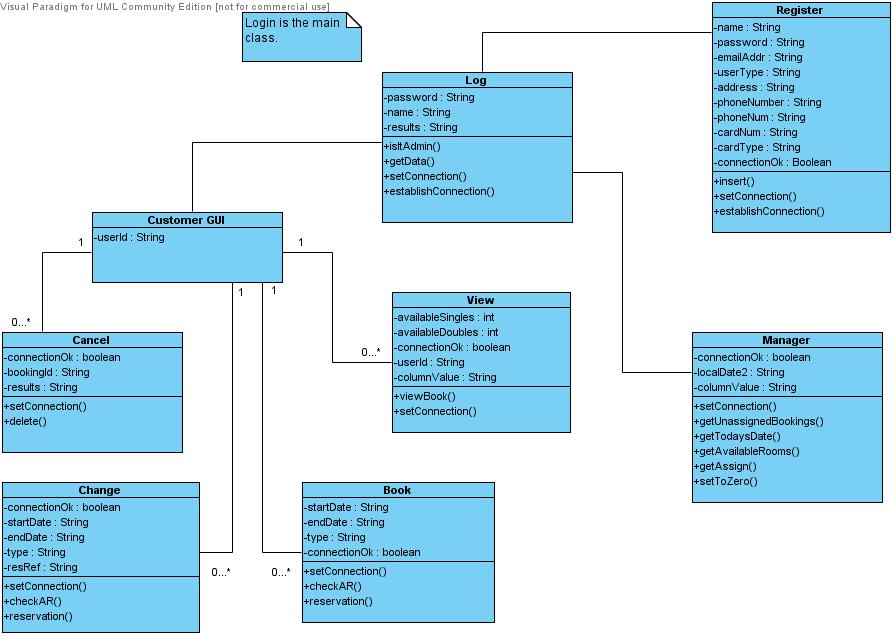


Figure 2 - Class Diagram, Hotel Booking System

## Datebase Design

The database ‘myhotel’ consists of four tables with their corresponding primary and foreign keys. Here we witness many-to-many relationship model where many rooms can be assigned to many bookings.

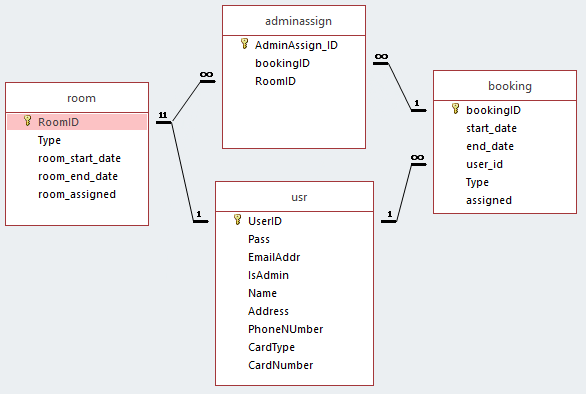


Figure 3 - Database Relationship between Tables

# Implementation

The group project was programmed and implemented on two different IDE’s (Integrated Development Environments), of which were BlueJ and Eclipse Neon. The project was built based on a two tier architecture using JDBC.

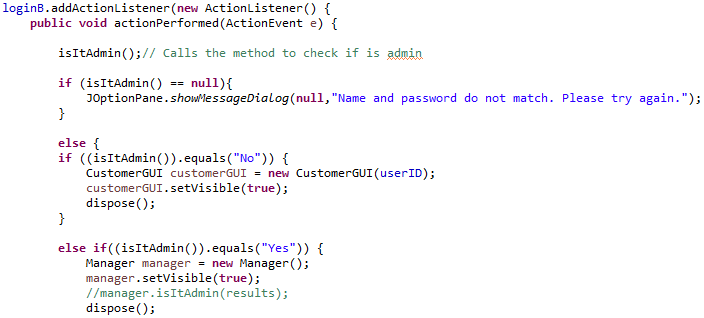
The program itself consists of eight separate graphical user interfaces, all of which interact with one another as well as the database. Apart from frame constructor and connection method, the classes have essential methods which assist the GUI’s to interact with the database in the background by manipulating the data within the tables. Some of the key methods and sections of the code are outlined in detail below. Alongside this, tests have been carried out on the individual graphical user interfaces too. Within each of these sections, the problems are also discussed.

## Login GUI

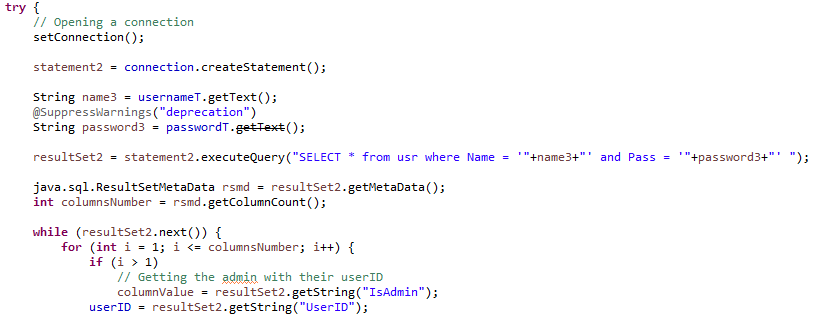
Some of the key methods that should be examined here include the following:

* Method One: isItAdmin()
* Method Two: getData()

Method one checksif the user is a customer or an administrator. This is done through executing a query which selects all of the values from the table ‘usr’. Here the columns entitled as ‘name’ and ‘password’ are used as the login credentials for the customer or administrator user. A while loop then iterates throughout the numbers of columns and checks column IsAdmin and column UserID for the user’s type and their unique ID. It is the button ‘Log In’ that fires an event. Notice how the action listener is used in the snippet of code:

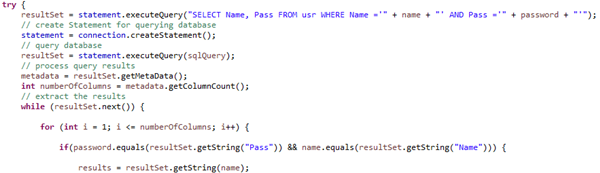


This same action listener uses the method *isItAdmin(),* alongside the use of several if-else statements. Within these statements, it accordingly opens either a manager’s or guest’sgraphical user interface, depending upon whether the user that has logged into the system is an admin or not:



If it is neither the admin nor customer, isItAdmin() == null, a show message dialog window pops up, alerting the user for the wrong credentials.

Method two, *getData(),* executes a query which selects the name and password from table ‘usr’. Here if all the data contains no errors, then the information is inserted. The name and password are checked in a while loop by string comparison.



In every Java program, there has to be a main method. Note that there are some keywords for doing this ‘public static void’, which essentially means this method is accessible because it is public and not private. Static means that this method can be called without creating an object. ‘Void’ is stating that the method here will not return anything! Note that within the main method its parameters ‘String [] args’ is used to say that this is a string of arrays, however ‘args’ can be called anything so long as it is not a Java keyword. There in this program it is the log in graphical user interface which has a main method, which in this system it means runs the entire system.

## Registration GUI

This graphical user interface required a lot of validation on the JTextFields, this was because the unregistered user must enter in specific data into the relevant fields. This validation will be illustrated through the use of screenshots and code snippets, of which both of these can be found within the appendices. It should also be noted that the entry fields for this registration window were all mandatory.

Refer to **Appendix A** to view the scenarios for this graphical user interface.

Two different layout managers were used within the Registration.java file. As the base for this graphical user interface the Border layout manager was used. This enabled the window to be divided into separate sections of which include the top, bottom, left, right and centre of the window. Most of the content was placed into the centre of the window, but another layout manager was required to overlap this area. This resulted in using the GridBag layout manager. It was selected because it enabled more control over the sizes of the individual cells within the table, unlike the Grid layout manager where the sizes of the cells cannot be altered. Using this approach was better practice when compared to absolute positioning.

The ‘insert**(**Component c**,** JPanel p**,**int gridX**,**int gridY**,**int gridW**,**int gridH**)’** method played a vital role with the positioning of the JComponents. It allowed an extensive amount of control as to where the multiple components could be placed on the registration page, as described above. By having this method, it also saved on repeating code for every single component, as the values could be passed into the relevant parameters of the method. This is shown in the snippet of code below:

For example the name label component would pass these values:

insert**(**nameLabel**,** centerPanel**,**0**,**0**,**1**,**1**);**

Into this method:

// Separate method, which handles the 'GridBag' positioning.

// Values are passed into this method from the components.

publicvoid insert**(**Component c**,** JPanel p**,**int gridX**,**int gridY**,**int gridW**,**int gridH**)**

**{**

GridBagConstraints constraint **=new** GridBagConstraints**();**

constraint**.**gridx **=** gridX**;**// X axis positioning

constraint**.**gridy **=** gridY**;**// Y axis positioning

constraint**.**gridwidth **=** gridW**;**// Width positioning

constraint**.**gridheight **=** gridH**;**// Height positioning

constraint**.**ipady **=**8**;**// Padding in y axis

constraint**.**weighty **=**1**;**// Spacing in the y axis

constraint**.**anchor **=** GridBagConstraints**.**LINE\_START**;**

// Alters the alignment (LINE\_START)

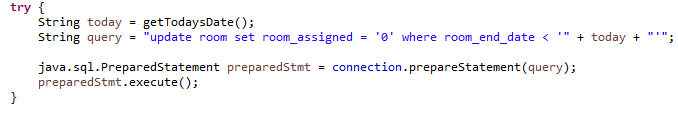
p**.**add**(**c**,** constraint**);**// Adding the panel

**}**

## MANAGER GUI

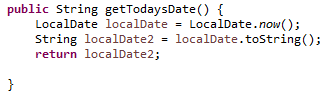
* Method one: *setToZero()*

This method turns assigned values in table ‘room’ back to zero for all rooms that have been occupied till today’s date. The method uses ‘PreparedStatement’ with update query to update the table:



* Method two: *getTodaysDate()*

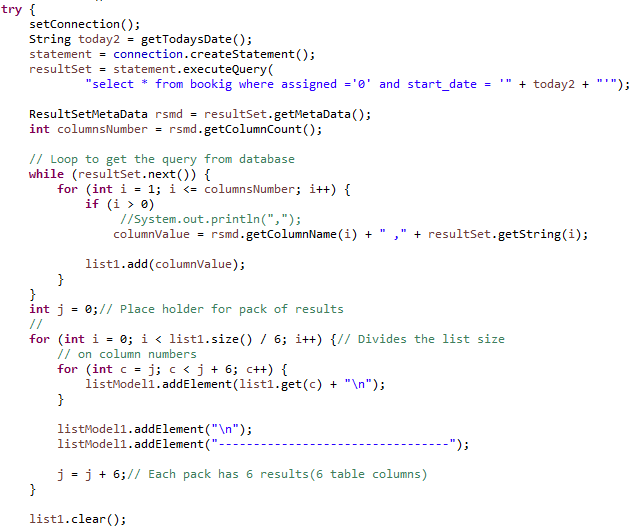
The method uses java.time.LocalDate, converts the result to a string and returns it:



* Method three: *getUnassignedBookings()*

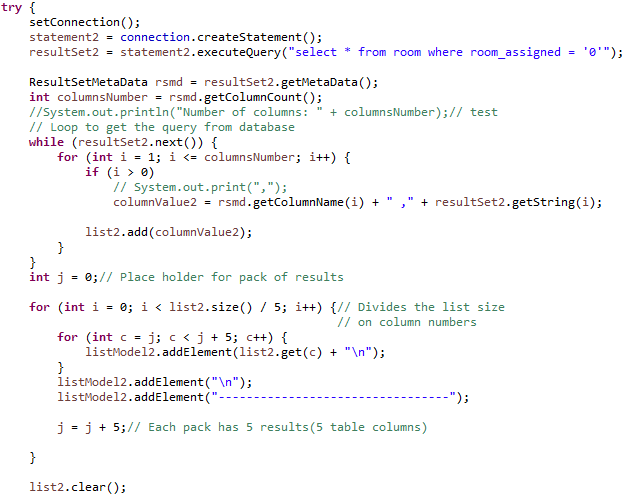
This method gets all unassigned bookings from today’s date and displays them in JList in the left side of the manager’s GUI. The method uses array list of strings and is also designed to display each pack of results separately for better view.

After an unassigned booking is selected from the list and assigned to a room, the objects are cleared from the list by manager pressing the button ‘Refresh’:



* Method four: *getAvailableRooms()*

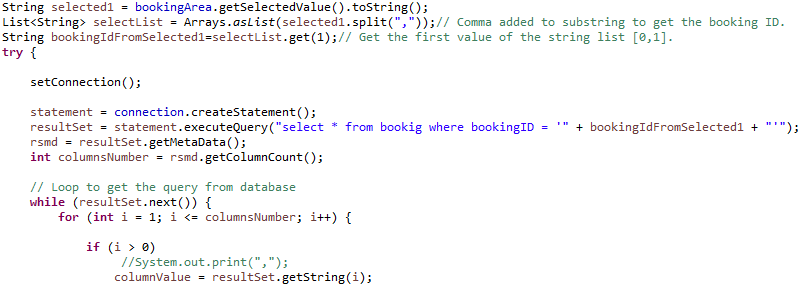
This method uses the same logic as the method above and works for the right JList of the manager’s GUI displaying all unassigned rooms.



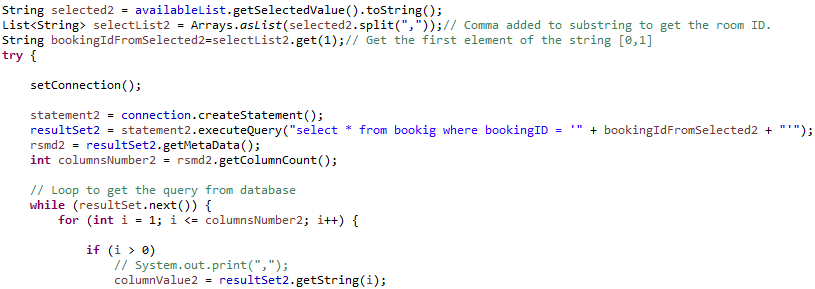
* Method five: *getAssign()*

This important method gets the chosen values from both JList fields and manipulates the data in the database by manager pressing the button ‘Assign’.

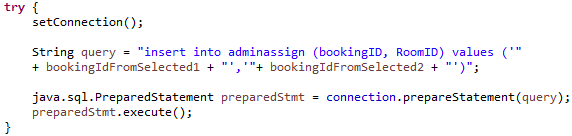
Selecting from the booking area:



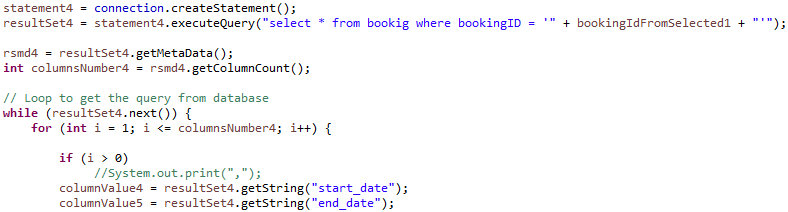
Selecting from the room area:



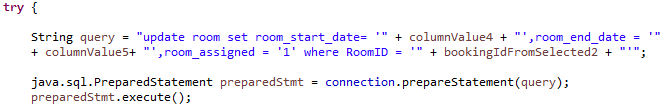
Populating table ‘adminassign’:



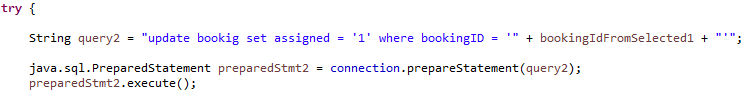
Extracting results from selected value in the booking (left) area of the GUI:



Updates table ‘room’ as assigned by using the chosen values from available bookings (left field) and room ID (right field):



Updates table ‘booking’ as assigned by using the selected booking ID from the left field of the GUI:



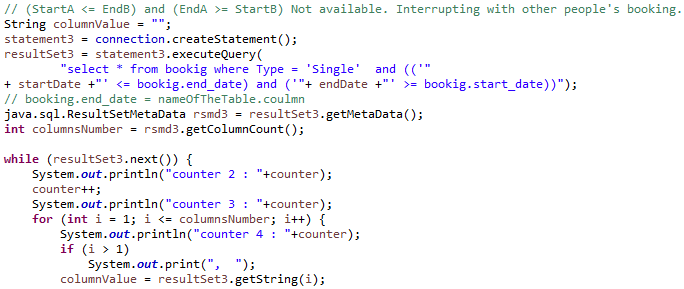
## CUSTOMER GUI

Since it does not use JDBC connection and does not manipulate data, this simple frame acts only as a navigator for customers’ choice. It contains five buttons with their performed action.

## BOOK A ROOM GUI

* Method one: *checkAR (String startDate, String endDate, String type)*

The first method checks for available single and double rooms by verifying bookings start and end dates are not overlapping with other booking dates.



The code which checks for double rooms uses the same logic.

* Method two:*reservation()*

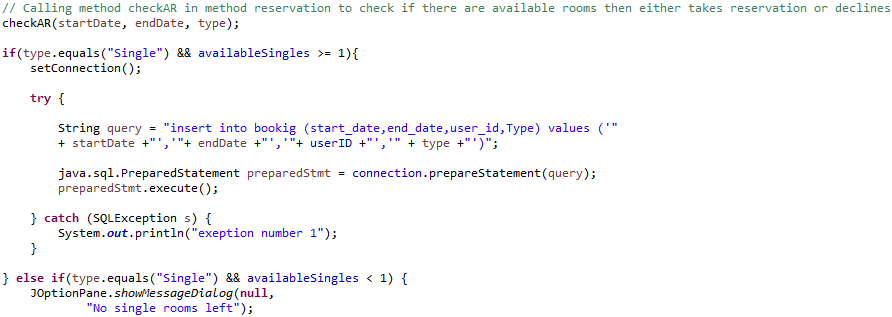
This method manipulates the data by inserting preferred booking values in to table ‘booking’, as well as checks if there are free single or double rooms left, based on which radio button is selected.

If(singleRB.isSelected()) {

type = "Single";

} else if(doubleRB.isSelected()) {

type = "Double";



The code for double rooms uses the same logic.

## CHANGE A BOOKING GUI

* Method one: *checkAR(String startDate, String endDate, String type)*
* Method two: *reservation()*

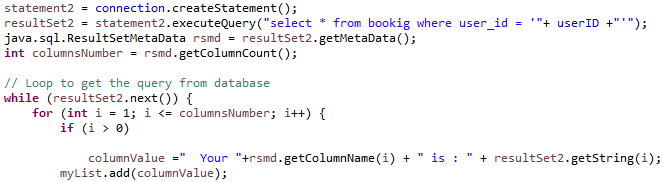
The class uses the same methods as *Book.java* but instead of inserting values in to table booking, it updates the table.



## View a Booking GUI

* Method one:*viewBook()*

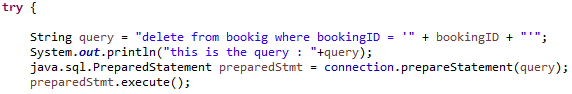
This method displays bookings in the GUI window by using array list of strings.



## Cancel a Booking GUI

* Method one: *delete()*

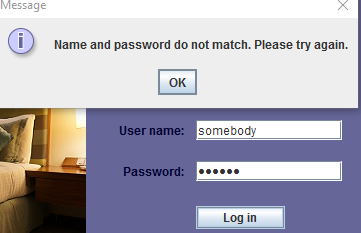
This method simply deletes data from table booking.



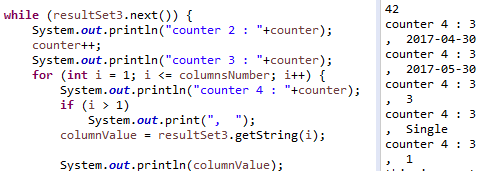
# TESTING

Some snippets of testing the system.

*Testing for wrong user details:*

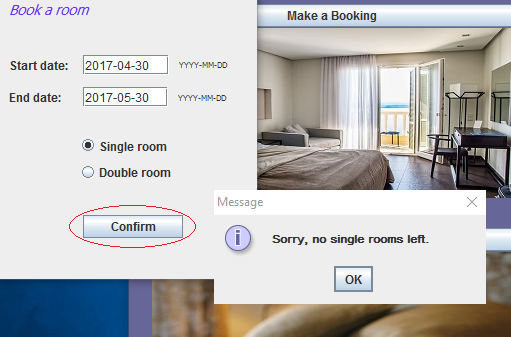


*Testing counter in the console during implementation*. The counter must go throughout all columns of table ‘booking’: bookingID, start\_date, end\_date, user\_id, Type, assigned. In this case, for three reservations. On the example, the third reservation:



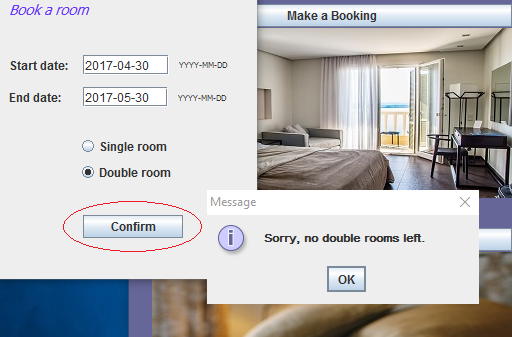
*Testing when all single rooms are booked:*

Since all 10 single rooms have been booked, the customer cannot book a room unless someone cancels a booking.

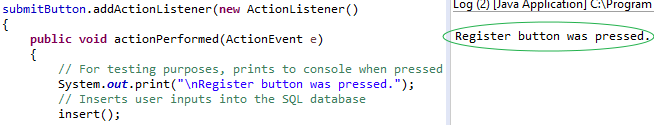


*Testing when all double rooms are booked:*

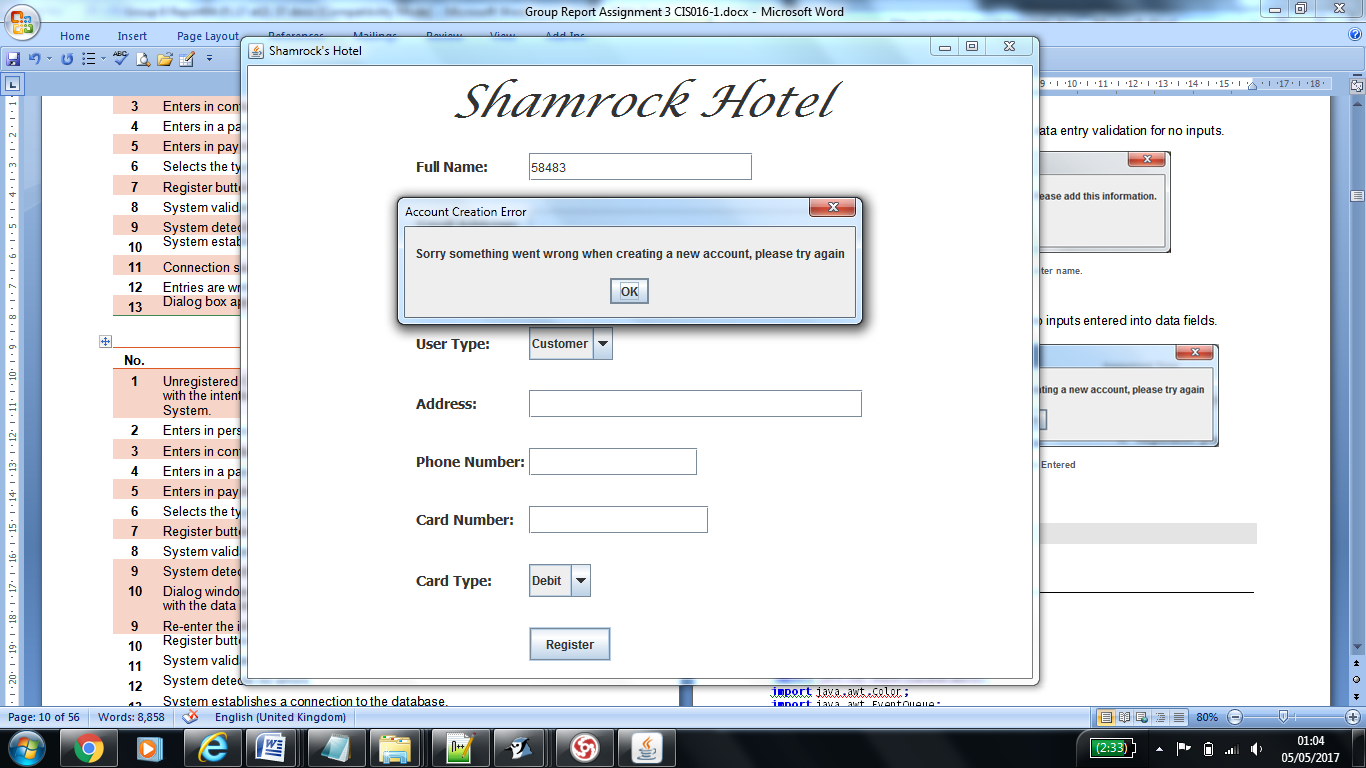
Since all 10 double rooms have been booked, the customer cannot book a room unless someone cancels a booking.



*Testing in the console if Register button was pressed:*



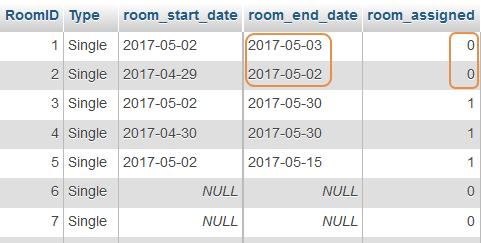
*Testing for Registration, graphical user interface, incorrect type of data entered into a field*:



*Testing the method setToZero():*

Below, table ‘room’.

Today’s date is **2017/05/05**. All rooms which have been assigned till later than today’s date have been automatically set to ‘0’ by the method.



# Reflections and Responsibilities

The group meant to be a group of five people, met obstacles at the beginning. This led to our group rearranged to a group of two at a quite late stage when the basics of the system were already laid, planned for five members. To save time, we decided to go for the more familiar JDBC connection, omitting the client-server architecture.

There were multiple problems concerning the structure of the group. The members within the group did change on multiple occasions. This wasted an excessive amount of time. Part of this was due to members not communicating or not showing up, clashes between characters, differences of opinions.

Group work presents multiple issues when a student’s mode of study is only part-time, as opposed to be being full-time. Again, this too was another problem which created further tensions within the original group at the start. It was not until much further on in the project that it was suggested a new group should be formed, with fewer members. Within this group, the members identified below had the following responsibilities:

|  |  |  |
| --- | --- | --- |
| Group Seven | | |
| Full Name | **Student ID** | **Responsibilities** |
|  |  | * Creating the database for the project * Creating the Database Entity Relationship diagram * Frontend and backend of: * *Book.java,* * *Cancel.java,* * *Change.java* * *CustomerGUI.java* * *Log.java* * *Manager.java* * *View.java.* * Debugging * Testing * Commenting aspects of the program * Report: Writing about her work and prove reading |
|  |  | * Creating the designs for the graphical user interfaces. * Conducting research into possible layout managers and experimenting with these findings. * Creating the UML diagrams for the project. * Writing the documentation for the group project. This includes the report, and UML diagrams. * Fully commenting the program and editing. * Altering aspects of the code and the database to suit the needs of the program. * Developing and testing the registration graphical user interface. * Carrying out testing on the program as a whole. |

# Conclusion

It was unfortunate that there have been multiple issues concerning the structure of the group in the first month of the project. Because of this multiple sections of the projects had to be scaled back, and compromises had to be made. For instance the original plan was to use socket programming, as both members of the current group felt this would be a fantastic opportunity to learn the subject is greater detail. It would have also meant that the end prototype system would have been more realistic in the way that it was written. This is because in the industry the larger companies such as Google use socket programming, and do not have SQL embedded within their code.

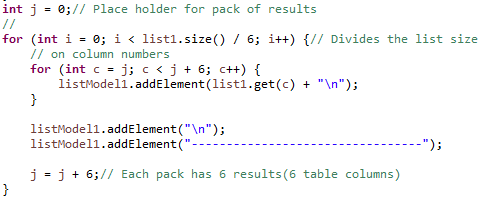
Each member of the group had their difficulties during the coding process.

**XXXXX’s:**

Main difficulties were in Book.java when checking for overlapping bookings. Mainly the idea behind it.



Also, in Manager.java displaying the results in the JList GUI.



A recent bug which caused rooms to appear all booked when they were not, was also dealt with. The case was a simple mistake in typing resultSet.getInt(i) instead of resultSet.getString(i).

**XXXXX’s:**

The initial problem with this project was due to being a first-year student, who had not yet taken the unit that covers SQL and databases. Because of this it meant that more time was taken here to grasp how data could be inputted into the database, through the means of a graphical user interface. Some SQL had to also be learnt and understood, in order to be able to embed the SQL statements into the Java code. For example, there was some difficulty with the SQL statement, which passes the values into the database. The code snippet is shown below:

st**.**executeUpdate**(**"INSERT INTO usr (Pass, EmailAddr, IsAdmin, Name, Address,PhoneNumber, CardType, CardNumber) VALUES ('"**+**Password**+**"', '"**+**EmailAddr**+**"', '"**+**IsAdmin**+**"', '"**+**Name**+**"', '"**+**Address**+**"', '"**+**PhoneNumber**+**"', '"**+**item**+**"', '"**+**CardNumber**+**"' )"**);**

The problem with the SQL statement was with the syntax. The principle of how to pass the values into the database was then, but the way in which it was written was not. This problem did not take too long to rectify.

Another problem that was encountered was with the combo boxes that were used on the form. It was not clear at first how the option could be obtained and then inserted into the database. This was due to it being a different component, from that of an entry textbox. The snippet of code below highlights how this was done:

// Obtains the selected option from combo box

Object item **=** cardType**.**getSelectedItem**();**

// Altering the value selected to suit the data within the 'usr' table

**if(**item **==**"Credit"**)**

**{**

item **=**"Credit Card"**;**// A credit card is being used

**}**

**else**

**{**

item **=**"Debit Card"**;**// A debit card is being used

**}**

The .getSelectedItem() had to be used as opposed to the .getText(). Hence the IF, ELSE statements were then used to determine which option had been selected by the customer.

Another problem that was faced throughout the project was the sensitivity of the tables within the database. By this it is meant that when an attempt was made to alter the database, or just one table within the database, it caused countless errors within the code. With hind sight, it is of the upmost importance that the database is finalised at the start of the project, prior to any connection being formed with the program. Hence this is why the user table remained as being named ‘usr’. Multiple attempts were made to rectify the problem towards the end of the project, but all failed. In fact, by making the alterations it almost meant that the entire project was broken. This is where it was quickly learnt that multiple backups were required for both the code and the database. By doing so,this error could be undone without too much trouble.

Nevertheless, much was learnt in the process of this project, which will prove valuable in future projects.

# Appendices

Included within the appendices is the entire program with comments to indicate how it functions. Code snippets are also included within the appendices, of which are referenced to throughout the report. These will help to illustrate any discussion covered within this report, concerning problems that were encountered within this project.

A full table of references is also included within the appendices, which acts as some of the evidence for research that was carried out during this project.

## Appendix A – Registration Scenarios

Below are some scenarios for the registration graphical user interface, to help demonstrate the processes that may occur when an unregistered customer attempts to create an account.

|  |  |
| --- | --- |
| **No.** | **Scenario One (Registration)** |
| **1** | Unregistered Customer clicks onto the ‘Register’ button from the log in page, with the intention to register for a new account with the Hotel Booking System. |
| **2** | Enters in personal information into the relevant entry fields. |
| **3** | Enters in contact information into the relevant entry fields. |
| **4** | Enters in a password. |
| **5** | Enters in payment card information into the relevant entry fields. |
| **6** | Selects the type of user this account is for. |
| **7** | Register button is clicked upon completion of the registration form. |
| **8** | System validates all of the mandatory entry fields. |
| **9** | System detects no errors. |
| **10** | System establishes a connection to the database. |
| **11** | Connection successful. |
| **12** | Entries are written to the database, in the appropriate columns and table. |
| **13** | Dialog box appears to confirm that an account has been made successfully. |

|  |  |
| --- | --- |
| **No.** | **Scenario Two (Registration)** |
| **1** | Unregistered Customer clicks onto the ‘Register’ button from the log in page, with the intention to register for a new account with the Hotel Booking System. |
| **2** | Enters in personal information into the relevant entry fields. |
| **3** | Enters in contact information into the relevant entry fields. |
| **4** | Enters in a password. |
| **5** | Enters in payment card information into the relevant entry fields. |
| **6** | Selects the type of user this account is for. |
| **7** | Register button is clicked upon completion of the registration form. |
| **8** | System validates all of the mandatory entry fields. |
| **9** | System detects errors. |
| **10** | Dialog window is displayed, informed the end user that an error has occurred with the data input. |
| **9** | Re-enter the information into the entry fields. |
| **10** | Register button is clicked upon completion of the registration form. |
| **11** | System validates all of the mandatory entry fields. |
| **12** | System detects no errors. |
| **13** | System establishes a connection to the database. |
| **14** | Connection successful. |
| **15** | Entries are written to the database, in the appropriate columns and table. |
| **16** | Dialog box appears to confirm that an account has been made successfully. |

SCREEN SHOTS:

1. Registration, graphical user interface, data entry validation for no inputs.

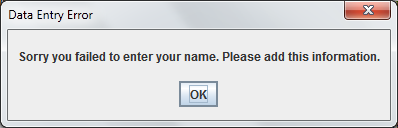


Figure 4 - Failed to enter name.

1. Registration, graphical user interface, no inputs entered into data fields.

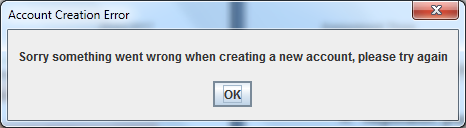


Figure 5 - No Inputs Entered

## Appendix b - The Program

The full program is outlined below:

Log In

**import** java**.**sql**.**Connection**;**

**import** java**.**sql**.**Statement**;**

**import** java**.**sql**.**DriverManager**;**

**import** java**.**sql**.**ResultSet**;**

**import** java**.**sql**.**SQLException**;**

**import** java**.**sql**.**ResultSetMetaData**;**

**import** java**.**awt**.**Color**;**

**import** java**.**awt**.**EventQueue**;**

**import** java**.**awt**.**Font**;**

**import** java**.**awt**.**Image**;**

**import** java**.**awt**.**event**.**ActionEvent**;**

**import** java**.**awt**.**event**.**ActionListener**;**

**import** javax**.**swing**.**JFrame**;**

**import** javax**.**swing**.**JPasswordField**;**

**import** javax**.**swing**.**JTextField**;**

**import** javax**.**swing**.**JOptionPane**;**

**import** javax**.**swing**.**JPanel**;**

**import** javax**.**swing**.**JLabel**;**

**import** javax**.**swing**.**ImageIcon**;**

**import** javax**.**swing**.**JButton**;**

publicclass Log **extends** JFrame **{**

privatestaticfinallong serialVersionUID **=**1L**;**// Serialised by JVM

private JLabel welcomeL **=new** JLabel**(**"Welcome to Shamrock hotel"**);**

private JLabel shamrockL **=new** JLabel**(**""**);**// Label placeholder for an image

private JLabel bedImgL **=new** JLabel**(**""**);**// Label placeholder for an image

private JLabel noaccountL **=new** JLabel**(**"No Account?"**);**

private JLabel usernameL **=new** JLabel**(**"User name:"**);**

private JLabel passwordL **=new** JLabel**(**"Password:"**);**

private JButton registerB **=new** JButton**(**"Register"**);**

private JButton loginB **=new** JButton**(**"Log in"**);**

private JTextField usernameT **=new** JTextField**(**20**);**

private JPasswordField passwordT **=new** JPasswordField**(**20**);**

private JPanel panel **=new** JPanel**();**

// the database connector details

privatestaticfinal String DRIVER **=**"com.mysql.jdbc.Driver"**;**// JDBC driver

privatestaticfinal String DATABASE **=**"myhotel"**;**// Database name

privatestaticfinal String HOST **=**"localhost"**;**// Database host IP

//static final String HOST = "10.1.63.200";// Uni

privatestaticfinal String DATABASE\_URL **=**"jdbc:mysql://"**+** HOST **+**"/"**+** DATABASE**;**

private Connection connection **=null;**

private Statement statement **=null;**

private String userID**;**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CONSTRUCTOR \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Initialising frame components

public Log**(){**

**super(**"Shamrock's hotel bookings"**);**

// Setting the frame

setSize**(**723**,**475**);**

setLocation**(**45**,**100**);**

setDefaultCloseOperation**(**JFrame**.**EXIT\_ON\_CLOSE**);**

panel**.**setBackground**(new** Color**(**102**,**102**,**153**));**

panel**.**setForeground**(**Color**.**BLACK**);**

panel**.**setLayout**(null);**

// Setting the components

welcomeL**.**setBounds**(**26**,**22**,**540**,**59**);**

welcomeL**.**setForeground**(new** Color**(**0**,**0**,**51**));**

welcomeL**.**setFont**(new** Font**(**"Lucida Calligraphy"**,** Font**.**ITALIC**,**28**));**

shamrockL**.**setBounds**(**588**,**22**,**89**,**71**);**// Source https://pixabay.com/en/shamrock-celtic-plant-clover-149821/

Image img2 **=new** ImageIcon**(this.**getClass**().**getResource**(**"/image2small.png"**)).**getImage**();**// Setting the image path

shamrockL**.**setIcon**(new** ImageIcon**(**img2**));**

bedImgL**.**setBounds**(**26**,**101**,**395**,**288**);**// Source https://pixabay.com/en/hotel-room-curtain-green-furniture-1979406/

Image img1 **=new** ImageIcon**(this.**getClass**().**getResource**(**"/hotel1.png"**)).**getImage**();**// Setting the image path

bedImgL**.**setIcon**(new** ImageIcon**(**img1**));**

noaccountL**.**setBounds**(**454**,**126**,**76**,**31**);**

noaccountL**.**setForeground**(new** Color**(**0**,**0**,**51**));**

noaccountL**.**setFont**(new** Font**(**"Tahoma"**,** Font**.**BOLD**,**11**));**

usernameL**.**setBounds**(**454**,**209**,**80**,**14**);**

usernameL**.**setForeground**(new** Color**(**0**,**0**,**51**));**

passwordL**.**setBounds**(**458**,**250**,**76**,**14**);**

passwordL**.**setForeground**(new** Color**(**0**,**0**,**51**));**

usernameT**.**setBounds**(**531**,**206**,**146**,**20**);**

passwordT**.**setBounds**(**531**,**247**,**146**,**20**);**

registerB**.**setBounds**(**531**,**130**,**89**,**23**);**

registerB**.**addActionListener**(new** ActionListener**(){**

publicvoid actionPerformed**(**ActionEvent e**){**

@SuppressWarnings**(**"unused"**)**

Register register **=new** Register**();**

**}});**

loginB**.**setBounds**(**531**,**292**,**89**,**23**);**

loginB**.**addActionListener**(new** ActionListener**(){**

publicvoid actionPerformed**(**ActionEvent e**){**

isItAdmin**();**// Calls the method to check if is admin

**if(**isItAdmin**()==null){**

JOptionPane**.**showMessageDialog**(null,**"Name and password do not match. Please try again."**);**

**}**

**else{**

**if((**isItAdmin**()).**equals**(**"No"**)){**

CustomerGUI customerGUI **=new** CustomerGUI**(**userID**);**

customerGUI**.**setVisible**(true);**

dispose**();**

**}**

**elseif((**isItAdmin**()).**equals**(**"Yes"**)){**

Manager manager **=new** Manager**();**

manager**.**setVisible**(true);**

//manager.isItAdmin(results);

dispose**();**

**}**

**}**

**}**

**});**

// Adding the components to the panel

panel**.**add**(**welcomeL**);**

panel**.**add**(**shamrockL**);**

panel**.**add**(**bedImgL**);**

panel**.**add**(**noaccountL**);**

panel**.**add**(**usernameL**);**

panel**.**add**(**passwordL**);**

panel**.**add**(**registerB**);**

panel**.**add**(**loginB**);**

panel**.**add**(**usernameT**);**

panel**.**add**(**passwordT**);**

getContentPane**().**add**(**panel**);**

**}**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CHECK IF ADMIN \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public String isItAdmin**(){**

ResultSet resultSet2 **=null;**

Statement statement2 **=null;**

String columnValue **=null;**

String answer**=**""**;**

// Querying the database

**try{**

// Opening a connection

setConnection**();**

statement2 **=** connection**.**createStatement**();**

String name3 **=** usernameT**.**getText**();**

@SuppressWarnings**(**"deprecation"**)**

String password3 **=** passwordT**.**getText**();**

resultSet2 **=** statement2**.**executeQuery**(**"SELECT \* from usr where Name = '"**+**name3**+**"' and Pass = '"**+**password3**+**"' "**);**

java**.**sql**.**ResultSetMetaData rsmd **=** resultSet2**.**getMetaData**();**

int columnsNumber **=** rsmd**.**getColumnCount**();**

**while(**resultSet2**.**next**()){**

**for(**int i **=**1**;** i **<=** columnsNumber**;** i**++){**

**if(**i **>**1**)**

// Getting the admin with their userID

columnValue **=** resultSet2**.**getString**(**"IsAdmin"**);**

userID **=** resultSet2**.**getString**(**"UserID"**);**

**}**

**}**

**}catch(**SQLException s**){**

**}**

**finally{**// Closing resultSet, statement and connection

**try{**

resultSet2**.**close**();**

statement2**.**close**();**

connection**.**close**();**

**}**

**catch(**Exception exception**){**

exception**.**printStackTrace**();**

**}**

**}**

answer **=** columnValue**;**

**return** answer**;**

**}**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* GET USER DATA \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public String getData**(){**

String password **=**""**;**

String name **=**"root"**;**

String results **=**"Yes"**;**

String sqlQuery **=null;**

ResultSetMetaData metadata **=null;**

ResultSet resultSet **=null;**

**try{**

resultSet **=** statement**.**executeQuery**(**"SELECT Name, Pass FROM usr WHERE Name ='"**+** name **+**"' AND Pass ='"**+** password **+**"'"**);**

// create Statement for querying database

statement **=** connection**.**createStatement**();**

// query database

resultSet **=** statement**.**executeQuery**(**sqlQuery**);**

// process query results

metadata **=** resultSet**.**getMetaData**();**

int numberOfColumns **=** metadata**.**getColumnCount**();**

// extract the results

**while(**resultSet**.**next**()){**

**for(**int i **=**1**;** i **<=** numberOfColumns**;** i**++){**

**if(**password**.**equals**(**resultSet**.**getString**(**"Pass"**))&&** name**.**equals**(**resultSet**.**getString**(**"Name"**))){**

results **=** resultSet**.**getString**(**name**);**

**}else{**

JOptionPane**.**showMessageDialog**(null,**"Login Failed. Please try again."**);**

**}**

**}**

**}**

**}**

**catch(**SQLException sqlException**){**

sqlException**.**printStackTrace**();**

// Print an error message instead of results

System**.**err**.**println**(**"ERROR! Something went wrong, please consult stack trace!"**);**

**}**

**finally{**

**try{**

resultSet**.**close**();**

statement**.**close**();**

connection**.**close**();**

**}**

**catch(**Exception exception**){**

exception**.**printStackTrace**();**

**}**

**}**

System**.**out**.**println**(**"name and pass are : "**+**results**);**// Debugging

**return** results**;**

**}**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CONNECTION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

publicboolean setConnection**(){**

boolean connectionOK **=false;**

// Connect to database

**try{**

// Load the driver class

Class**.**forName**(**DRIVER**);**

// Establish connection to database

connection **=** DriverManager**.**getConnection**(**DATABASE\_URL**,**"root"**,**""**);**

connectionOK **=true;**

**}**

**catch(**SQLException sqlException**){**

System**.**err**.**println**(**"Driver loaded, but something went wrong elsewhere!"**);**

sqlException**.**printStackTrace**();**

**}**

**catch(**ClassNotFoundException classNotFound**){**

System**.**err**.**println**(**"Couldn't find driver!"**);**

classNotFound**.**printStackTrace**();**

**}**

**return** connectionOK**;**

**}**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MAIN \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

publicstaticvoid main**(**String**[]** args**){**

EventQueue**.**invokeLater**(new** Runnable**(){**

publicvoid run**(){**

**try{**

Log log **=new** Log**();**

log**.**setVisible**(true);**

**}**

**catch(**Exception e**){**

e**.**printStackTrace**();**

**}**

**}**

**});**

**}**

**}**

Register

// Library Imports

// ---------------------------------->8

**import** java**.**awt**.\*;**

**import** java**.**awt**.**event**.\*;**

**import** javax**.**swing**.\*;**

**import** java**.**awt**.**event**.\*;**

**import** java**.**sql**.\*;**

// 'Register' Class (Creating an Account)

// ---------------------------------->8

publicclass Register **extends** JFrame

**{**

// Database connection details

// ------------------------------->8

staticfinal String DRIVER **=**"com.mysql.jdbc.Driver"**;**// JDBC driver

staticfinal String DATABASE **=**"myhotel"**;**// Database name

staticfinal String HOST **=**"localhost"**;**// Database host IP

// the full URL:

staticfinal String DATABASE\_URL **=**"jdbc:mysql://"**+** HOST **+**"/"**+** DATABASE**;**

// Window frame

private JFrame mainFrame**;**

// Window panels

private JPanel northPanel**,** centerPanel**;**

// Label components

private JLabel hotelName**,** pageTitleLabel**;**

private JLabel passwordLabel**,** userTypeLabel**;**

private JLabel nameLabel**;**

private JLabel emailLabel**,** phoneLabel**,** addressLabel**;**

private JLabel cardTypeLabel**,** cardNumLabel**;**

//private JLabel imageHolder;

// Password field component

private JPasswordField password**;**

// Textfield components

private JTextField name**,** address**;**

private JTextField email**,** phone**;**

private JTextField cardNum**;**

// Button components

private JButton submitButton**;**

// Combo box component

private JComboBox cardType**;**

private JComboBox userType**;**

boolean isAdminCondition **=false;**

// Error Labels

private JLabel passwordErr**;**

// Error checking

boolean error **=false;**

boolean status **=true;**

// Database connection initialisations

public Connection c **=null;**

boolean checkConnection **=false;**

//public MetaData metadata = null;

// Constructor for 'Register' class

// ---------------------------------->8

public Register**()**

**{**

setUpWindow**();**// Calls the 'setUpWindow' method

// establishConnection(); // Calls the 'establishConnection' method

**}**

privatevoid setUpWindow**()**

**{**

// Creating the main window

JFrame mainFrame **=new** JFrame**();**

mainFrame**.**setTitle**(**"Shamrock's Hotel"**);**

mainFrame**.**setLayout**(new** BorderLayout**());**

mainFrame**.**setSize**(**800**,**650**);**

// Creating the panels

// ---------------------------------->8

JPanel centerPanel **=new** JPanel**();**

centerPanel**.**setLayout**(new** GridBagLayout**());**

centerPanel**.**setBackground**(**Color**.**WHITE**);**

JPanel northPanel **=new** JPanel**();**

northPanel**.**setBackground**(**Color**.**WHITE**);**

// Adding Panels to the 'mainFrame'

// ---------------------------------->8

mainFrame**.**add**(**centerPanel**,** BorderLayout**.**CENTER**);**

mainFrame**.**add**(**northPanel**,** BorderLayout**.**NORTH**);**

// Creating the Components

// ---------------------------------->8

hotelName **=new** JLabel**(**" Shamrock Hotel "**);**

hotelName**.**setFont**(new** Font**(**"Lucida Calligraphy"**,** Font**.**ITALIC**,**44**));**

nameLabel **=new** JLabel**(**"Full Name: "**);**

nameLabel**.**setFont**(new** Font**(**"Tahoma"**,** Font**.**BOLD**,**14**));**

name **=new** JTextField**(**20**);**

addressLabel **=new** JLabel**(**"Address: "**);**

addressLabel**.**setFont**(new** Font**(**"Tahoma"**,** Font**.**BOLD**,**14**));**

address **=new** JTextField**(**30**);**

emailLabel **=new** JLabel**(**"Email Address: "**);**

emailLabel**.**setFont**(new** Font**(**"Tahoma"**,** Font**.**BOLD**,**14**));**

email **=new** JTextField**(**30**);**

phoneLabel **=new** JLabel**(**"Phone Number: "**);**

phoneLabel**.**setFont**(new** Font**(**"Tahoma"**,** Font**.**BOLD**,**14**));**

phone **=new** JTextField**(**15**);**

passwordLabel **=new** JLabel**(**"Password: "**);**

passwordLabel**.**setFont**(new** Font**(**"Tahoma"**,** Font**.**BOLD**,**14**));**

password **=new** JPasswordField**(**20**);**

userTypeLabel **=new** JLabel**(**"User Type: "**);**

userTypeLabel**.**setFont**(new** Font**(**"Tahoma"**,** Font**.**BOLD**,**14**));**

String**[]** userOptions **={**"Customer"**,**"Admin"**};**

userType **=new** JComboBox**<**String**>(**userOptions**);**

cardTypeLabel **=new** JLabel**(**"Card Type: "**);**

cardTypeLabel**.**setFont**(new** Font**(**"Tahoma"**,** Font**.**BOLD**,**14**));**

String**[]** options **={**"Debit"**,**"Credit"**};**

cardType **=new** JComboBox**<**String**>(**options**);**

cardNumLabel **=new** JLabel**(**"Card Number: "**);**

cardNumLabel**.**setFont**(new** Font**(**"Tahoma"**,** Font**.**BOLD**,**14**));**

cardNum **=new** JTextField**(**16**);**

submitButton **=new** JButton**(**"Register"**);**

submitButton**.**addActionListener**(new** ActionListener**()**

**{**

publicvoid actionPerformed**(**ActionEvent e**)**

**{**

// For testing purposes, prints to console when pressed

System**.**out**.**print**(**"\nRegister button was pressed."**);**

// Inserts user inputs into the SQL database

insert**();**

//Log tryin = new Log();

**}**

**});**

// Adding GUI Components to 'northPanel'

// ---------------------------------->8

northPanel**.**add**(**hotelName**);**

// Adding GUI Components to 'centerPanel'

// ---------------------------------->8

centerPanel**.**add**(**passwordLabel**);**

centerPanel**.**add**(**password**);**

centerPanel**.**add**(**userTypeLabel**);**

centerPanel**.**add**(**userType**);**

centerPanel**.**add**(**nameLabel**);**

centerPanel**.**add**(**name**);**

centerPanel**.**add**(**addressLabel**);**

centerPanel**.**add**(**address**);**

centerPanel**.**add**(**emailLabel**);**

centerPanel**.**add**(**email**);**

centerPanel**.**add**(**phoneLabel**);**

centerPanel**.**add**(**phone**);**

centerPanel**.**add**(**cardTypeLabel**);**

centerPanel**.**add**(**cardType**);**

centerPanel**.**add**(**cardNumLabel**);**

centerPanel**.**add**(**cardNum**);**

centerPanel**.**add**(**submitButton**);**

// Setting the Position of the Components

// ---------------------------------->8

insert**(**nameLabel**,** centerPanel**,**0**,**0**,**1**,**1**);**

insert**(**emailLabel**,** centerPanel**,**0**,**1**,**1**,**1**);**

insert**(**passwordLabel**,** centerPanel**,**0**,**2**,**1**,**1**);**

insert**(**userTypeLabel**,** centerPanel**,**0**,**3**,**1**,**1**);**

insert**(**addressLabel**,** centerPanel**,**0**,**4**,**1**,**1**);**

insert**(**phoneLabel**,** centerPanel**,**0**,**5**,**1**,**1**);**

insert**(**cardNumLabel**,** centerPanel**,**0**,**6**,**1**,**1**);**

insert**(**cardTypeLabel**,** centerPanel**,**0**,**7**,**1**,**1**);**

insert**(**name**,** centerPanel**,**1**,**0**,**1**,**1**);**

insert**(**email**,** centerPanel**,**1**,**1**,**1**,**1**);**

insert**(**password**,** centerPanel**,**1**,**2**,**1**,**1**);**

insert**(**userType**,** centerPanel**,**1**,**3**,**2**,**1**);**

insert**(**address**,** centerPanel**,**1**,**4**,**1**,**1**);**

insert**(**phone**,** centerPanel**,**1**,**5**,**1**,**1**);**

insert**(**cardNum**,** centerPanel**,**1**,**6**,**1**,**1**);**

insert**(**cardType**,** centerPanel**,**1**,**7**,**1**,**1**);**

insert**(**submitButton**,** centerPanel**,**1**,**10**,**1**,**1**);**

mainFrame**.**setVisible**(true);**

**}**

// Separate method, which handles the 'GridBag' positioning.

// Values are passed into this method from the components.

publicvoid insert**(**Component c**,** JPanel p**,**int gridX**,**int gridY**,**int gridW**,**int gridH**)**

**{**

GridBagConstraints constraint **=new** GridBagConstraints**();**

constraint**.**gridx **=** gridX**;**// X axis positioning

constraint**.**gridy **=** gridY**;**// Y axis positioning

constraint**.**gridwidth **=** gridW**;**// Width positioning

constraint**.**gridheight **=** gridH**;**// Height positioning

constraint**.**ipady **=**8**;**// Padding in y axis

constraint**.**weighty **=**1**;**// Spacing in the y axis

constraint**.**anchor **=** GridBagConstraints**.**LINE\_START**;**// Alters the alignment (LINE\_START)

p**.**add**(**c**,** constraint**);**// Adding the panel

**}**

// Inserting the form data into the database 'usr'

publicvoid insert**()**

**{**

**try**

**{**

// Acts as a flag to determine whether there were any errors here

error **=false;**

// The connection string

Class**.**forName**(**DRIVER**);**

// Establish connection to the database

c **=** DriverManager**.**getConnection**(**DATABASE\_URL**,**"root"**,**""**);**

Statement st **=** c**.**createStatement**();**

// Converting the data from the GUI component into a useful format

String Name **=** name**.**getText**();**

// Entering that some data is entered

**if(**Name**.**isEmpty**())**

**{**

JOptionPane**.**showConfirmDialog**(null,**"Sorry you failed to enter your name. Please add this information."**,**

"Data Entry Error"**,** JOptionPane**.**DEFAULT\_OPTION**,**

JOptionPane**.**PLAIN\_MESSAGE**);**

error **=true;**

**}**

@SuppressWarnings**(**"deprecation"**)**// Gets rid of the passsword error message

// Obtains the password the user has entered

String Password **=** password**.**getText**();**

// Entering that some data is entered

**if(**Password**.**isEmpty**())**

**{**

JOptionPane**.**showConfirmDialog**(null,**"Sorry you failed to enter a password. Please add this information."**,**

"Data Entry Error"**,** JOptionPane**.**DEFAULT\_OPTION**,**

JOptionPane**.**PLAIN\_MESSAGE**);**

error **=true;**

**}**

// Obtains the email address of the user

String EmailAddr **=** email**.**getText**();**

// Entering that some data is entered

**if(**EmailAddr**.**isEmpty**())**

**{**

JOptionPane**.**showConfirmDialog**(null,**"Sorry you failed to enter an email address. Please add this information."**,**

"Data Entry Error"**,** JOptionPane**.**DEFAULT\_OPTION**,**

JOptionPane**.**PLAIN\_MESSAGE**);**

error **=true;**

**}**

// Determining the type of user acount that is being created.

Object IsAdmin **=** userType**.**getSelectedItem**();**

// Altering the value selected to suit the data within the 'usr' table

**if(**IsAdmin **==**"Customer"**)**

**{**

IsAdmin **=**"No"**;**// Not an admin user

**}**

**else**

**{**

IsAdmin **=**"Yes"**;**// Is an admin user

**}**

// Obtains the address of the user

String Address **=** address**.**getText**();**

// Entering that some data is entered

**if(**Address**.**isEmpty**())**

**{**

JOptionPane**.**showConfirmDialog**(null,**"Sorry you failed to enter an address. Please add this information."**,**

"Data Entry Error"**,** JOptionPane**.**DEFAULT\_OPTION**,**

JOptionPane**.**PLAIN\_MESSAGE**);**

error **=true;**

**}**

// Obtains the phone number of the user

String PhoneNumber **=** phone**.**getText**();**

// Entering that some data is entered

**if(**PhoneNumber**.**isEmpty**())**

**{**

JOptionPane**.**showConfirmDialog**(null,**"Sorry you failed to enter contact number. Please add this information."**,**

"Data Entry Error"**,** JOptionPane**.**DEFAULT\_OPTION**,**

JOptionPane**.**PLAIN\_MESSAGE**);**

error **=true;**

**}**

// Ensures that only numberic data is entered here

**try**

**{**

int phoNum **=** Integer**.**parseInt**(**PhoneNumber**);**

**}**

**catch(**NumberFormatException ee**)**

**{**

JOptionPane**.**showConfirmDialog**(null,**"Only numeric values are allowed in the phone number field. Please enter this information again."**,**

"Data Entry Error"**,** JOptionPane**.**DEFAULT\_OPTION**,**

JOptionPane**.**PLAIN\_MESSAGE**);**

error **=true;**

**}**

// Obtains the card number of the user

String CardNumber **=** cardNum**.**getText**();**

// Entering that some data is entered

**if(**CardNumber**.**isEmpty**())**

**{**

JOptionPane**.**showConfirmDialog**(null,**"Sorry you failed to enter a card number. Please add this information."**,**

"Data Entry Error"**,** JOptionPane**.**DEFAULT\_OPTION**,**

JOptionPane**.**PLAIN\_MESSAGE**);**

error **=true;**

**}**

// Ensures that only numberic data is entered here

**try**

**{**

int cardNum **=** Integer**.**parseInt**(**CardNumber**);**

**}**

**catch(**NumberFormatException ee**)**

**{**

JOptionPane**.**showConfirmDialog**(null,**"Only numeric values are allowed in the card number field. Please enter this information again."**,**

"Data Entry Error"**,** JOptionPane**.**DEFAULT\_OPTION**,**

JOptionPane**.**PLAIN\_MESSAGE**);**

error **=true;**

**}**

// Obtains the selected option from combo box

Object item **=** cardType**.**getSelectedItem**();**

// Altering the value selected to suit the data within the 'usr' table

**if(**item **==**"Credit"**)**

**{**

item **=**"Credit Card"**;**// A credit card is being used

**}**

**else**

**{**

item **=**"Debit Card"**;**// A debit card is being used

**}**

/\* Important this line of the program, inserts the data obtained from the GUI into the database.

\* Notice the use of 'INSERT INTO' and 'VALUES'.

\*/

**if(**error **==false)**

**{**

st**.**executeUpdate**(**"INSERT INTO usr (Pass, EmailAddr, IsAdmin, Name, Address,PhoneNumber, CardType, CardNumber) VALUES ('"**+**Password**+**"', '"**+**EmailAddr**+**"', '"**+**IsAdmin**+**"', '"**+**Name**+**"', '"**+**Address**+**"', '"**+**PhoneNumber**+**"', '"**+**item**+**"', '"**+**CardNumber**+**"' )"**);**

// Creates a dialog message box when a new account is registered successfully.

JOptionPane**.**showConfirmDialog**(null,**"Thank you for registering an account with The Shamrock Hotel."**,**

"Account Creation Confirmation"**,** JOptionPane**.**DEFAULT\_OPTION**,**

JOptionPane**.**PLAIN\_MESSAGE**);**

status **=false;**

**}**

**else**

**{**

**thrownew** Exception**();**

**}**

// Closing connections, required, and good practise.

st**.**close**();**// Closes the statements

c**.**close**();**// Closes the connection

**}**

**catch(**Exception e**)**

**{**

JOptionPane**.**showConfirmDialog**(null,**"Sorry something went wrong when creating a new account, please try again"**,**

"Account Creation Error"**,** JOptionPane**.**DEFAULT\_OPTION**,**

JOptionPane**.**PLAIN\_MESSAGE**);**

**}**

**}**

// Establishing a connection with the database on MySQL

publicboolean establishConnection**(){**

// Attempting to make a connection to the database

**try**

**{**

checkConnection **=true;**

// Loading the driver into the program

Class**.**forName**(**DRIVER**);**

// Establish a connection to database

c **=** DriverManager**.**getConnection**(**DATABASE\_URL**,**"root"**,**""**);**

// Prints out this message on the console

System**.**out**.**print**(**"The driver loaded successfully."**);**

**}**

// Error message for when the driver cannot be found

**catch(**ClassNotFoundException classNotFound**)**

**{**

checkConnection **=false;**

classNotFound**.**printStackTrace**();**

System**.**err**.**println**(**"ERROR: No driver found."**);**

**}**

// Error message when there is an SQL problem but the driver is found

**catch(**SQLException sqlException**)**

**{**

checkConnection **=false;**

System**.**err**.**println**(**"ERROR: An SQL problem is apparent. Driver has loaded successfully."**);**

**}**

**return** checkConnection**;**

**}**

// Main Program

publicstaticvoid main**(**String**[]** args**)**

**{**

// Creating a new Register object

Register newAccount **=new** Register**();**

**}**

**}**

View Options

**import** java**.**sql**.**Connection**;**

**import** java**.**sql**.**DriverManager**;**

**import** java**.**sql**.**ResultSet**;**

**import** java**.**sql**.**SQLException**;**

**import** java**.**sql**.**Statement**;**

**import** java**.**awt**.**BorderLayout**;**

**import** java**.**awt**.**Color**;**

**import** java**.**util**.**ArrayList**;**

**import** javax**.**swing**.**JFrame**;**

**import** javax**.**swing**.**JPanel**;**

**import** javax**.**swing**.**JScrollPane**;**

**import** javax**.**swing**.**JTextArea**;**

**import** java**.**awt**.**Button**;**

**import** java**.**awt**.**Font**;**

**import** java**.**awt**.**event**.**ActionEvent**;**

**import** java**.**awt**.**event**.**ActionListener**;**

publicclass View **extends** JFrame **{**

privatestaticfinallong serialVersionUID **=**1L**;**

privatestaticfinalint LINES **=**15**;**

privatestaticfinalint CHAR\_PER\_LINE **=**25**;**

private JTextArea viewYourBooking**;**

// Connection

privatestaticfinal String DRIVER **=**"com.mysql.jdbc.Driver"**;**// JDBC driver

privatestaticfinal String DATABASE **=**"myhotel"**;**// Database name

privatestaticfinal String HOST **=**"localhost"**;**// "10.1.63.200";// Uni server

// the full URL:

privatestaticfinal String DATABASE\_URL **=**"jdbc:mysql://"**+** HOST **+**"/"**+** DATABASE**;**

private Connection connection **=null;**

private String userID**;**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CONSTRUCTOR \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public View**(**String userID**){**

**super(**"Shamrock hotel"**);**

**this.**userID **=** userID**;**

setSize**(**315**,**304**);**

setLocation**(**266**,**306**);**

setDefaultCloseOperation**(**JFrame**.**DISPOSE\_ON\_CLOSE**);**

JPanel textPanel **=new** JPanel**();**

textPanel**.**setBackground**(new** Color**(**102**,**102**,**153**));**

textPanel**.**setLayout**(new** BorderLayout**(**0**,**0**));**

viewYourBooking **=new** JTextArea**(**LINES**,** CHAR\_PER\_LINE**);**

viewYourBooking**.**setBackground**(new** Color**(**204**,**255**,**255**));**

viewYourBooking**.**setEditable**(false);**

JScrollPane scrolledText **=new** JScrollPane**(**viewYourBooking**);**

scrolledText**.**setHorizontalScrollBarPolicy**(**

JScrollPane**.**HORIZONTAL\_SCROLLBAR\_AS\_NEEDED**);**

scrolledText**.**setVerticalScrollBarPolicy**(**

JScrollPane**.**VERTICAL\_SCROLLBAR\_ALWAYS**);**

textPanel**.**add**(**scrolledText**);**

getContentPane**().**add**(**textPanel**,** BorderLayout**.**NORTH**);**

Button ok **=new** Button**(**"Ok"**);**

ok**.**setFont**(new** Font**(**"Dialog"**,** Font**.**PLAIN**,**13**));**

getContentPane**().**add**(**ok**,** BorderLayout**.**SOUTH**);**

ok**.**addActionListener**(new** ActionListener**(){**

publicvoid actionPerformed**(**ActionEvent e**){**

dispose**();**

**}});**

**}**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CONNECTION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

publicboolean setConnection**(){**

boolean connectionOK **=false;**

// connect to database

**try{**

// load the driver class

Class**.**forName**(**DRIVER**);**

// establish connection to database

connection **=** DriverManager**.**getConnection**(**DATABASE\_URL**,**"root"**,**""**);**

connectionOK **=true;**

**}catch(**SQLException sqlException**){**

System**.**err**.**println**(**"Driver loaded, but something went wrong elsewhere!"**);**

sqlException**.**printStackTrace**();**

**}catch(**ClassNotFoundException classNotFound**){**

System**.**err**.**println**(**"Couldn't find driver!"**);**

classNotFound**.**printStackTrace**();**

**}**

**return** connectionOK**;**

**}**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* VIEW BOOKING \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public String viewBook**(){**

String columnValue **=null;**

ResultSet resultSet2 **=null;**

Statement statement2 **=null;**

ArrayList**<**String**>** myList **=new** ArrayList**<**String**>();**

setConnection**();**

**try{**

setConnection**();**

statement2 **=** connection**.**createStatement**();**

resultSet2 **=** statement2**.**executeQuery**(**"select \* from bookig where user\_id = '"**+** userID **+**"'"**);**

java**.**sql**.**ResultSetMetaData rsmd **=** resultSet2**.**getMetaData**();**

int columnsNumber **=** rsmd**.**getColumnCount**();**

// Loop to get the query from database

**while(**resultSet2**.**next**()){**

**for(**int i **=**1**;** i **<=** columnsNumber**;** i**++){**

**if(**i **>**0**)**

columnValue **=**" Your "**+**rsmd**.**getColumnName**(**i**)+**" is : "**+** resultSet2**.**getString**(**i**);**

myList**.**add**(**columnValue**);**

**}**

**}**

// Displaying the results one after another as the inner loop iterates each 6 times, but the outer loop iterates depending on size

// of the list by dividing by the column number we get the number of reservations so this is the amount of times we need to loop

// and with the inner loop we get the elements for each reservation.

int j **=**0**;**// Place holder for pack of results

//

**for(**int i **=**0**;** i **<** myList**.**size**()/**6**;** i**++){**// Divides the list size on column numbers

**for(**int c **=** j**;** c **<** j **+**6**;** c**++){**

viewYourBooking**.**append**(**myList**.**get**(**c**)+**"\n"**);**

**}**

viewYourBooking**.**append**(**"\n"**);**

viewYourBooking**.**append**(**"\n"**);**

viewYourBooking**.**append**(**"\n"**);**

j **=** j **+**6**;**// Each pack has 6 results(6 table columns)

**}**

// testing

System**.**out**.**println**(**myList**);**

**}catch(**SQLException s**){**

System**.**out**.**println**(**"exeptionist"**);**

**}**

**finally{**

**try{**

resultSet2**.**close**();**

statement2**.**close**();**

connection**.**close**();**

**}**

**catch(**Exception exception**){**

exception**.**printStackTrace**();**

**}**

**}**

**return** columnValue**;**

**}**

**}**

Book a Room

**import** java**.**sql**.**Connection**;**

**import** java**.**sql**.**Statement**;**

**import** java**.**sql**.**DriverManager**;**

**import** java**.**sql**.**ResultSet**;**

**import** java**.**sql**.**SQLException**;**

**import** java**.**awt**.**Color**;**

**import** java**.**awt**.**Font**;**

**import** java**.**awt**.**event**.**ActionEvent**;**

**import** java**.**awt**.**event**.**ActionListener**;**

**import** javax**.**swing**.**JFrame**;**

**import** javax**.**swing**.**JTextField**;**

**import** javax**.**swing**.**JOptionPane**;**

**import** javax**.**swing**.**JPanel**;**

**import** javax**.**swing**.**JRadioButton**;**

**import** javax**.**swing**.**JLabel**;**

**import** javax**.**swing**.**ButtonGroup**;**

**import** javax**.**swing**.**JButton**;**

publicclass Book **extends** JFrame **{**

privatestaticfinallong serialVersionUID **=**1L**;**// Serial GUI ID implied by Java VM

private JLabel bookAroomL **=new** JLabel**(**"Book a room"**);**

private JLabel startDateL **=new** JLabel**(**"Start date:"**);**

private JLabel endDateL **=new** JLabel**(**"End date: "**);**

private JTextField startDateT **=new** JTextField**(**8**);**

private JTextField endDateT **=new** JTextField**(**8**);**

private ButtonGroup groupOfButtons **=new** ButtonGroup**();**

private JRadioButton singleRB **=new** JRadioButton**(**"Single room"**);**

private JRadioButton doubleRB **=new** JRadioButton**(**"Double room"**);**

private JButton confirmB **=new** JButton**(**"Confirm"**);**

private JPanel panelBook **=new** JPanel**();**

// the database connector details

privatestaticfinal String DRIVER **=**"com.mysql.jdbc.Driver"**;**// JDBC driver

privatestaticfinal String DATABASE **=**"myhotel"**;**// Database name

privatestaticfinal String HOST **=**"localhost"**;**// Database host IP

//static final String HOST = "10.1.63.200";// Uni server

// the full URL:

privatestaticfinal String DATABASE\_URL **=**"jdbc:mysql://"**+** HOST **+**"/"**+** DATABASE**;**

private Connection connection **=null;**

//available rooms

privateint availableSingles **=**0**;**

privateint availableDoubles **=**0**;**

private String userID**;**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CONSTRUCTOR \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public Book**(**String userID**){**

**super(**"Shamrock's hotel booking"**);**

**this.**userID **=** userID**;**

setBackground**(new** Color**(**153**,**153**,**204**));**

setSize**(**274**,**327**);**

setLocation**(**313**,**14**);**

setDefaultCloseOperation**(**JFrame**.**DISPOSE\_ON\_CLOSE**);**

panelBook**.**setLayout**(null);**

bookAroomL**.**setBounds**(**10**,**11**,**147**,**14**);**

bookAroomL**.**setFont**(new** Font**(**"Tahoma"**,** Font**.**ITALIC**,**14**));**

bookAroomL**.**setForeground**(new** Color**(**102**,**51**,**255**));**

startDateL**.**setBounds**(**10**,**53**,**75**,**43**);**

endDateL**.**setBounds**(**10**,**85**,**64**,**43**);**

startDateT**.**setBounds**(**84**,**96**,**85**,**20**);**

endDateT**.**setBounds**(**84**,**64**,**86**,**20**);**

singleRB**.**setBounds**(**79**,**144**,**109**,**23**);**

//singleRB.setActionCommand("Single room");

groupOfButtons**.**add**(**singleRB**);**

doubleRB**.**setBounds**(**79**,**170**,**109**,**23**);**

//doubleRB.setActionCommand("Double room");

doubleRB**.**setSelected**(true);**

groupOfButtons**.**add**(**doubleRB**);**

confirmB**.**setBounds**(**84**,**224**,**100**,**23**);**

confirmB**.**addActionListener**(new** ActionListener**(){**

publicvoid actionPerformed**(**ActionEvent e**){**

reservation**();**

dispose**();**

**}});**

// Adding the components to the panel

panelBook**.**add**(**bookAroomL**);**

panelBook**.**add**(**startDateL**);**

panelBook**.**add**(**endDateL**);**

panelBook**.**add**(**startDateT**);**

panelBook**.**add**(**endDateT**);**

panelBook**.**add**(**singleRB**);**

panelBook**.**add**(**doubleRB**);**

panelBook**.**add**(**confirmB**);**

getContentPane**().**add**(**panelBook**);**

JLabel lblNewLabel **=new** JLabel**(**"YYYY-MM-DD"**);**

lblNewLabel**.**setFont**(new** Font**(**"Tahoma"**,** Font**.**PLAIN**,**8**));**

lblNewLabel**.**setBounds**(**180**,**67**,**68**,**14**);**

panelBook**.**add**(**lblNewLabel**);**

JLabel lblYyyymmdd **=new** JLabel**(**"YYYY-MM-DD"**);**

lblYyyymmdd**.**setFont**(new** Font**(**"Tahoma"**,** Font**.**PLAIN**,**8**));**

lblYyyymmdd**.**setBounds**(**179**,**100**,**69**,**14**);**

panelBook**.**add**(**lblYyyymmdd**);**

**}**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CONNECTION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

publicboolean setConnection**(){**

boolean connectionOK **=false;**

// connect to database

**try**

**{**

// load the driver class

Class**.**forName**(**DRIVER**);**

// establish connection to database

connection **=** DriverManager**.**getConnection**(**DATABASE\_URL**,**"root"**,**""**);**

connectionOK **=true;**

**}**

**catch(**SQLException sqlException**)**

**{**

System**.**err**.**println**(**"Driver loaded, but something went wrong elsewhere!"**);**

sqlException**.**printStackTrace**();**

**}**

**catch(**ClassNotFoundException classNotFound**)**

**{**

System**.**err**.**println**(**"Couldn't find driver!"**);**

classNotFound**.**printStackTrace**();**

**}**

**return** connectionOK**;**

**}**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CHECKING FOR OVERLAPING BOOKINGS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

publicvoid checkAR**(**String startDate**,** String endDate**,** String type**){**

System**.**out**.**println**(**"the type in checkAR : "**+**type**);**// test, debugging

// CHECKING FOR SINGLE ROOMS

ResultSet resultSet3 **=null;**

Statement statement3 **=null;**

System**.**out**.**println**(**"firstline of :"**);**// test, debugging

setConnection**();**

System**.**out**.**println**(**"running here"**);**// test, debugging

int counter **=**0**;**

System**.**out**.**println**(**"start date ="**+** startDate**);**

**try{**

// (StartA <= EndB) and (EndA >= StartB) Not available. Interrupting with other people's booking.

String columnValue **=**""**;**

statement3 **=** connection**.**createStatement**();**

resultSet3 **=** statement3**.**executeQuery**(**

"select \* from bookig where Type = 'Single' and (('"**+** startDate **+**"' <= bookig.end\_date) and ('"**+** endDate **+**"' >= bookig.start\_date))"**);**

// booking.end\_date = nameOfTheTable.coulmn

java**.**sql**.**ResultSetMetaData rsmd3 **=** resultSet3**.**getMetaData**();**

int columnsNumber **=** rsmd3**.**getColumnCount**();**

**while(**resultSet3**.**next**()){**

System**.**out**.**println**(**"counter 2 : "**+**counter**);**

counter**++;**

System**.**out**.**println**(**"counter 3 : "**+**counter**);**

**for(**int i **=**1**;** i **<=** columnsNumber**;** i**++){**

System**.**out**.**println**(**"counter 4 : "**+**counter**);**

**if(**i **>**1**)**

System**.**out**.**print**(**", "**);**

columnValue **=** resultSet3**.**getString**(**i**);**

System**.**out**.**println**(**columnValue**);**

**}**

**}**

System**.**out**.**println**(**"this is counter : "**+** counter**);**

//iteration went throughout all columns and extracted the column values for 10 single rooms and their booking, subtracting the overlapping booking.

availableSingles **=**10**-** counter**;**

**}**

**catch(**SQLException s**){**

System**.**out**.**println**(**"sql exception"**);**

**}**

// CHECKING FOR DOUBLE ROOMS

ResultSet resultSet4 **=null;**

Statement statement4 **=null;**

//System.out.println("first line of :");

//String answer4 = "";

setConnection**();**

int counter2 **=**0**;**

**try{**

String columnValue **=**""**;**

statement4 **=** connection**.**createStatement**();**

resultSet4 **=** statement4**.**executeQuery**(**

"select \* from bookig where Type = 'Double' and (('"**+**startDate**+**"' <= bookig.end\_date) and ('"**+**endDate**+**"' >= bookig.start\_date))"**);**

java**.**sql**.**ResultSetMetaData rsmd4 **=** resultSet4**.**getMetaData**();**

int columnsNumber **=** rsmd4**.**getColumnCount**();**

**while(**resultSet4**.**next**()){**

counter2**++;**

**for(**int i **=**1**;** i **<=** columnsNumber**;** i**++){**

**if(**i **>**1**)**

System**.**out**.**print**(**", "**);**

columnValue **=** resultSet4**.**getString**(**i**);**

System**.**out**.**println**(**columnValue**);**

**}**

**}**

System**.**out**.**println**(**"this is counter2 : "**+** counter2**);**

//iteration went throughout all columns and extracted the column values for 10 double rooms and their booking, subtracting the overlapping booking.

availableDoubles **=**10**-** counter2**;**

**}**

**catch(**SQLException s**){**

System**.**out**.**println**(**"sql exception lol"**);**

**}**

**finally{**

**try{**

resultSet4**.**close**();**

statement4**.**close**();**

connection**.**close**();**

**}**

**catch(**Exception exception**){**

exception**.**printStackTrace**();**

**}**

**}**

**}**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* RESERVATION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

publicvoid reservation**(){**

String type **=null;**

String startDate **=** endDateT**.**getText**();**

String endDate **=** startDateT**.**getText**();**

**if(**singleRB**.**isSelected**()){**

type **=**"Single"**;**

**}elseif(**doubleRB**.**isSelected**()){**

type **=**"Double"**;**

**}**

System**.**out**.**println**(**"type is : "**+**type **);**

// Calling method checkAR in method reservation to check if there are available rooms then either takes reservation or declines

checkAR**(**startDate**,** endDate**,** type**);**

**if(**type**.**equals**(**"Single"**)&&** availableSingles **>=**1**){**

setConnection**();**

**try{**

String query **=**"insert into bookig (start\_date,end\_date,user\_id,Type) values ('"**+** startDate **+**"','"**+** endDate **+**"','"**+** userID **+**"','"**+** type **+**"')"**;**

java**.**sql**.**PreparedStatement preparedStmt **=** connection**.**prepareStatement**(**query**);**

preparedStmt**.**execute**();**

**}catch(**SQLException s**){**

System**.**out**.**println**(**"exeption number 1"**);**

**}**

**}elseif(**type**.**equals**(**"Single"**)&&** availableSingles **<**1**){**

JOptionPane**.**showMessageDialog**(null,**

"No single rooms left"**);**

**}elseif(**type**.**equals**(**"Double"**)&&** availableDoubles **>=**1**){**

setConnection**();**

**try{**

String query **=**"insert into bookig (start\_date,end\_date,user\_id,Type) values ('"**+** startDate **+**"','"**+** endDate **+**"','"**+** userID **+**"','"**+** type **+**"')"**;**

java**.**sql**.**PreparedStatement preparedStmt **=** connection**.**prepareStatement**(**query**);**

preparedStmt**.**execute**();**

System**.**out**.**println**(**"What?"**);**

**}catch(**SQLException s**){**

System**.**out**.**println**(**"exeption number 2"**);**

**}**

**}elseif(**type**.**equals**(**"Double"**)&&** availableDoubles **<**1**){**

JOptionPane**.**showMessageDialog**(null,**

"No double rooms left"**);**

**}**

**}**

**}**

Cancel a Booking

**import** java**.**sql**.**Connection**;**

**import** java**.**sql**.**DriverManager**;**

**import** java**.**sql**.**ResultSet**;**

**import** java**.**sql**.**ResultSetMetaData**;**

**import** java**.**sql**.**SQLException**;**

**import** java**.**sql**.**Statement**;**

**import** java**.**awt**.**Color**;**

**import** java**.**awt**.**Font**;**

**import** java**.**awt**.**event**.**ActionEvent**;**

**import** java**.**awt**.**event**.**ActionListener**;**

**import** javax**.**swing**.**JButton**;**

**import** javax**.**swing**.**JFrame**;**

**import** javax**.**swing**.**JLabel**;**

**import** javax**.**swing**.**JPanel**;**

**import** javax**.**swing**.**JTextField**;**

publicclass Cancel **extends** JFrame **{**

privatestaticfinallong serialVersionUID **=**1L**;**// Serialised by JVM

private JLabel topLabelL **=new** JLabel**(**"Cancel booking"**);**

private JLabel bookRefL **=new** JLabel**(**"Booking reference:"**);**

private JTextField bookRefT **=new** JTextField**(**10**);**

private JButton cancelB **=new** JButton**(**"Cancel booking"**);**

private JPanel panelCancel **=new** JPanel**();**

// the database connector details

privatestaticfinal String DRIVER **=**"com.mysql.jdbc.Driver"**;**// JDBC driver

privatestaticfinal String DATABASE **=**"myhotel"**;**// Database name

privatestaticfinal String HOST **=**"localhost"**;**// Database host IP

// static final String HOST = "10.1.63.200";// Uni server

// the full URL:

privatestaticfinal String DATABASE\_URL **=**"jdbc:mysql://"**+** HOST **+**"/"**+** DATABASE**;**

// Responsible for connection

public Connection connection **=null;**

public Statement statement **=null;**

public ResultSet resultSet **=null;**

public ResultSetMetaData metadata **=null;**

public String results **=null;**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CONSTRUCTOR \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public Cancel**(){**

**super(**"Shamrock's hotel booking"**);**

setBackground**(new** Color**(**153**,**153**,**204**));**

setSize**(**238**,**190**);**

setLocation**(**42**,**337**);**

setDefaultCloseOperation**(**JFrame**.**DISPOSE\_ON\_CLOSE**);**

panelCancel**.**setLayout**(null);**

topLabelL**.**setBounds**(**10**,**11**,**153**,**20**);**

topLabelL**.**setFont**(new** Font**(**"Tahoma"**,** Font**.**ITALIC**,**14**));**

topLabelL**.**setForeground**(new** Color**(**102**,**51**,**255**));**

cancelB**.**setBounds**(**46**,**102**,**129**,**23**);**

cancelB**.**addActionListener**(new** ActionListener**(){**

publicvoid actionPerformed**(**ActionEvent e**){**

delete**();**

dispose**();**

**}});**

bookRefL**.**setBounds**(**10**,**42**,**122**,**31**);**

bookRefT**.**setBounds**(**121**,**47**,**54**,**20**);**

// Adding the components to the panel

panelCancel**.**add**(**topLabelL**);**

panelCancel**.**add**(**bookRefL**);**

panelCancel**.**add**(**bookRefT**);**

panelCancel**.**add**(**cancelB**);**

getContentPane**().**add**(**panelCancel**);**

**}**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CONNECTION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

publicboolean setConnection**(){**

boolean connectionOK **=false;**

// connect to database

**try{**

// load the driver class

Class**.**forName**(**DRIVER**);**

// establish connection to database

connection **=** DriverManager**.**getConnection**(**DATABASE\_URL**,**"root"**,**""**);**

connectionOK **=true;**

**}catch(**SQLException sqlException**){**

System**.**err**.**println**(**"Driver loaded, but something went wrong elsewhere!"**);**

sqlException**.**printStackTrace**();**

**}catch(**ClassNotFoundException classNotFound**){**

System**.**err**.**println**(**"Couldn't find driver!"**);**

classNotFound**.**printStackTrace**();**

**}**

**return** connectionOK**;**

**}**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CANCEL BOOKING \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

publicvoid delete**(){**

setConnection**();**

String bookingID **=** bookRefT**.**getText**();**

System**.**out**.**println**(**"bookref : "**+** bookingID**);**

**try{**

String query **=**"delete from bookig where bookingID = '"**+** bookingID **+**"'"**;**

System**.**out**.**println**(**"this is the query : "**+**query**);**

java**.**sql**.**PreparedStatement preparedStmt **=** connection**.**prepareStatement**(**query**);**

preparedStmt**.**execute**();**

**}catch(**SQLException s**){**

**}**

**finally{**

**try{**

connection**.**close**();**

**}**

**catch(**Exception exception**){**

exception**.**printStackTrace**();**

**}**

**}**

**}**

**}**

Change a Booking

**import** java**.**sql**.**Connection**;**

**import** java**.**sql**.**Statement**;**

**import** java**.**sql**.**DriverManager**;**

**import** java**.**sql**.**ResultSet**;**

**import** java**.**sql**.**SQLException**;**

**import** java**.**awt**.**Color**;**

**import** java**.**awt**.**Font**;**

**import** java**.**awt**.**event**.**ActionEvent**;**

**import** java**.**awt**.**event**.**ActionListener**;**

**import** javax**.**swing**.**ButtonGroup**;**

**import** javax**.**swing**.**JButton**;**

**import** javax**.**swing**.**JFrame**;**

**import** javax**.**swing**.**JLabel**;**

**import** javax**.**swing**.**JOptionPane**;**

**import** javax**.**swing**.**JPanel**;**

**import** javax**.**swing**.**JRadioButton**;**

**import** javax**.**swing**.**JTextField**;**

publicclass Change **extends** JFrame **{**

privatestaticfinallong serialVersionUID **=**1L**;**// Serial GUI ID implied by

// Java VM

private JLabel bookAroomL **=new** JLabel**(**"Change booking"**);**

private JLabel startDateL **=new** JLabel**(**"Change start date:"**);**

private JLabel endDateL **=new** JLabel**(**"Change end date: "**);**

private JTextField startDateT **=new** JTextField**(**8**);**

private JTextField endDateT **=new** JTextField**(**8**);**

private ButtonGroup groupOfButtons **=new** ButtonGroup**();**

private JRadioButton singleRB **=new** JRadioButton**(**"Single room"**);**

private JRadioButton doubleRB **=new** JRadioButton**(**"Double room"**);**

private JButton confirmB **=new** JButton**(**"Update"**);**

private JPanel panelBook **=new** JPanel**();**

// the database connector details

privatestaticfinal String DRIVER **=**"com.mysql.jdbc.Driver"**;**// JDBC driver

privatestaticfinal String DATABASE **=**"myhotel"**;**// Database name

privatestaticfinal String HOST **=**"localhost"**;**// Database host IP

// static final String HOST = "10.1.63.200";// Uni server

// the full URL:

privatestaticfinal String DATABASE\_URL **=**"jdbc:mysql://"**+** HOST **+**"/"**+** DATABASE**;**

private Connection connection **=null;**

// available rooms

privateint availableSingles **=**0**;**

privateint availableDoubles **=**0**;**

private JTextField bookingReferenseT**;**

privatefinal JLabel lblNewLabel **=new** JLabel**(**"YYYY-MM-DD"**);**

privatefinal JLabel lblYyyymmdd **=new** JLabel**(**"YYYY-MM-DD"**);**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CONSTRUCTOR \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public Change**(){**

**super(**"Shamrock's hotel booking"**);**

setSize**(**342**,**363**);**

setLocation**(**24**,**13**);**

setDefaultCloseOperation**(**JFrame**.**DISPOSE\_ON\_CLOSE**);**

panelBook**.**setLayout**(null);**

bookAroomL**.**setBounds**(**10**,**11**,**147**,**31**);**

bookAroomL**.**setFont**(new** Font**(**"Tahoma"**,** Font**.**ITALIC**,**14**));**

bookAroomL**.**setForeground**(new** Color**(**102**,**51**,**255**));**

startDateL**.**setBounds**(**10**,**97**,**106**,**43**);**

endDateL**.**setBounds**(**10**,**138**,**120**,**43**);**

startDateT**.**setBounds**(**146**,**149**,**85**,**20**);**

endDateT**.**setBounds**(**146**,**108**,**85**,**20**);**

singleRB**.**setBounds**(**113**,**188**,**109**,**23**);**

groupOfButtons**.**add**(**singleRB**);**

doubleRB**.**setBounds**(**113**,**214**,**109**,**23**);**

doubleRB**.**setSelected**(true);**

groupOfButtons**.**add**(**doubleRB**);**

confirmB**.**setBounds**(**113**,**262**,**100**,**23**);**

// Adding the components to the panel

panelBook**.**add**(**bookAroomL**);**

panelBook**.**add**(**startDateL**);**

panelBook**.**add**(**endDateL**);**

panelBook**.**add**(**startDateT**);**

panelBook**.**add**(**endDateT**);**

panelBook**.**add**(**singleRB**);**

panelBook**.**add**(**doubleRB**);**

panelBook**.**add**(**confirmB**);**

getContentPane**().**add**(**panelBook**);**

bookingReferenseT **=new** JTextField**();**

bookingReferenseT**.**setBounds**(**146**,**69**,**85**,**20**);**

panelBook**.**add**(**bookingReferenseT**);**

bookingReferenseT**.**setColumns**(**10**);**

JLabel lblReservationRefference **=new** JLabel**(**"Reservation Reference:"**);**

lblReservationRefference**.**setBounds**(**10**,**72**,**136**,**14**);**

panelBook**.**add**(**lblReservationRefference**);**

lblNewLabel**.**setFont**(new** Font**(**"Tahoma"**,** Font**.**PLAIN**,**8**));**

lblNewLabel**.**setBounds**(**254**,**112**,**62**,**14**);**

panelBook**.**add**(**lblNewLabel**);**

lblYyyymmdd**.**setFont**(new** Font**(**"Tahoma"**,** Font**.**PLAIN**,**8**));**

lblYyyymmdd**.**setBounds**(**253**,**153**,**63**,**14**);**

panelBook**.**add**(**lblYyyymmdd**);**

// button action listener

confirmB**.**addActionListener**(new** ActionListener**(){**

publicvoid actionPerformed**(**ActionEvent e**){**

reservation**();**

dispose**();**

**}**

**});**

**}**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CONNECTION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

publicboolean setConnection**(){**

boolean connectionOK **=false;**

// connect to database

**try{**

// load the driver class

Class**.**forName**(**DRIVER**);**

// establish connection to database

connection **=** DriverManager**.**getConnection**(**DATABASE\_URL**,**"root"**,**""**);**

connectionOK **=true;**

**}catch(**SQLException sqlException**){**

System**.**err**.**println**(**"Driver loaded, but something went wrong elsewhere!"**);**

sqlException**.**printStackTrace**();**

**}catch(**ClassNotFoundException classNotFound**){**

System**.**err**.**println**(**"Couldn't find driver!"**);**

classNotFound**.**printStackTrace**();**

**}**

**return** connectionOK**;**

**}**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CHECK FOR AVAILABLE ROOMS

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

publicvoid checkAR**(**String startDate**,** String endDate**,** String Type**){**

// creating method variables

ResultSet resultSet3 **=null;**

Statement statement3 **=null;**

System**.**out**.**println**(**"firstline of :"**);**

// String answer3 = "";

setConnection**();**

System**.**out**.**println**(**"running here"**);**

int counter **=**0**;**

System**.**out**.**println**(**"start date ="**+** startDate**);**

**try{**

String columnValue **=**""**;**

statement3 **=** connection**.**createStatement**();**

resultSet3 **=** statement3**.**executeQuery**(**"select \* from bookig where Type = 'Single' and (('"**+** startDate

**+**"' <= bookig.end\_date) and ('"**+** endDate **+**"' >= bookig.start\_date))"**);**

java**.**sql**.**ResultSetMetaData rsmd3 **=** resultSet3**.**getMetaData**();**

int columnsNumber **=** rsmd3**.**getColumnCount**();**

**while(**resultSet3**.**next**()){**

counter**++;**

**for(**int i **=**1**;** i **<=** columnsNumber**;** i**++){**

**if(**i **>**1**)**

System**.**out**.**print**(**", "**);**

// the mistake : columnValue = resultSet3.getInt(i);

columnValue **=** resultSet3**.**getString**(**i**);**

// System.out.print(columnValue + " " +);// test

// rsmd.getColumnName(i));

// myList.add(columnValue);

System**.**out**.**println**(**columnValue**);**

**}**

**}**

System**.**out**.**println**(**"this is counter : "**+** counter**);**

availableSingles **=**10**-** counter**;**

**}**

**catch(**SQLException s**){**

System**.**out**.**println**(**"sql exception"**);**

**}**

// CHECKING FOR DOUBLE ROOMS

ResultSet resultSet4 **=null;**

Statement statement4 **=null;**

System**.**out**.**println**(**"firstline of :"**);**

// String answer4 = "";

setConnection**();**

int counter2 **=**0**;**

**try{**

String columnValue **=**""**;**

statement4 **=** connection**.**createStatement**();**

System**.**out**.**println**(**"start date is : "**+** startDate**);**

System**.**out**.**println**(**"end date is : "**+** endDate**);**

// System.out.println("start date is : "+ startDate);

resultSet4 **=** statement4**.**executeQuery**(**"select \* from bookig where Type = 'Double' and (('"**+** startDate

**+**"' <= bookig.end\_date) and ('"**+** endDate **+**"' >= bookig.start\_date))"**);**

java**.**sql**.**ResultSetMetaData rsmd4 **=** resultSet4**.**getMetaData**();**

int columnsNumber **=** rsmd4**.**getColumnCount**();**

**while(**resultSet4**.**next**()){**

counter2**++;**

**for(**int i **=**1**;** i **<=** columnsNumber**;** i**++){**

**if(**i **>**1**)**

System**.**out**.**print**(**", "**);**

// this is what was here : columnValue =

// resultSet4.getInt(i);

columnValue **=** resultSet4**.**getString**(**i**);**

// System.out.print(columnValue + " " +);

System**.**out**.**println**(**columnValue**);**

**}**

**}**

System**.**out**.**println**(**"this is counter2 : "**+** counter2**);**

availableDoubles **=**10**-** counter2**;**

// type2amountbooked = columnValue;

**}**

**catch(**SQLException s**){**

System**.**out**.**println**(**"sql exception lol"**);**

**}finally{**

**try{**

resultSet4**.**close**();**

statement4**.**close**();**

connection**.**close**();**

**}catch(**Exception exception**){**

exception**.**printStackTrace**();**

**}**

**}**

**}**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* RESSERVATION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

publicvoid reservation**(){**

String type **=null;**

String endDate **=** startDateT**.**getText**();**

String startDate **=** endDateT**.**getText**();**

String resRef **=** bookingReferenseT**.**getText**();**

**if(**singleRB**.**isSelected**()){**

type **=**"Single"**;**

**}elseif(**doubleRB**.**isSelected**()){**

type **=**"Double"**;**

**}**

checkAR**(**startDate**,** endDate**,** type**);**

// Type="1";

System**.**out**.**println**(**"till here please"**);**

System**.**out**.**println**(**"available singles here : "**+** availableSingles**);**

System**.**out**.**println**(**"available doubles here : "**+** availableDoubles**);**

System**.**out**.**println**(**"start date2 is : "**+** startDate**);**

System**.**out**.**println**(**"end date2 is : "**+** endDate**);**

**if(**type**.**equals**(**"Single"**)&&** availableSingles **>=**1**){**

setConnection**();**

**try{**

// We added the condition assigned = '0'; because we want the

// query to update database only when reservations are not

// assigned.

String query **=**"update bookig set start\_date= '"**+** startDate **+**"',end\_date = '"**+** endDate **+**"',Type = '"

**+** type **+**"' where bookingID = '"**+** resRef **+**"' and assigned ='0'"**;**

java**.**sql**.**PreparedStatement preparedStmt **=** connection**.**prepareStatement**(**query**);**

preparedStmt**.**execute**();**

**}catch(**SQLException s**){**

System**.**out**.**println**(**"exeption number 1"**);**

**}**

**}elseif(**type**.**equals**(**"Single"**)&&** availableSingles **<**1**){**

JOptionPane**.**showMessageDialog**(null,**"No single rooms left"**);**

**}elseif(**type**.**equals**(**"Double"**)&&** availableDoubles **>=**1**){**

setConnection**();**

**try{**

String query **=**"update bookig set start\_date= '"**+** startDate **+**"',end\_date = '"**+** endDate **+**"',Type = '"

**+** type **+**"' where bookingID = '"**+** resRef **+**"'"**;**

java**.**sql**.**PreparedStatement preparedStmt **=** connection**.**prepareStatement**(**query**);**

preparedStmt**.**execute**();**

**}catch(**SQLException s**){**

System**.**out**.**println**(**"exeption number 1"**);**

**}**

**}elseif(**type**.**equals**(**"Double"**)&&** availableDoubles **<**1**){**

JOptionPane**.**showMessageDialog**(null,**"No double rooms left"**);**

**}**

**}**

**}**

Customer GUI

**import** java**.**awt**.**Color**;**

**import** java**.**awt**.**Font**;**

**import** java**.**awt**.**Image**;**

**import** java**.**awt**.**event**.**ActionEvent**;**

**import** java**.**awt**.**event**.**ActionListener**;**

**import** javax**.**swing**.**JFrame**;**

**import** javax**.**swing**.**JLabel**;**

**import** javax**.**swing**.**JPanel**;**

**import** javax**.**swing**.**ImageIcon**;**

**import** javax**.**swing**.**JButton**;**

publicclass CustomerGUI **extends** JFrame **{**

privatestaticfinallong serialVersionUID **=**1L**;**// default SUID for the JVM to verify all versions are the same

private JLabel welcomeL **=new** JLabel**(**"Welcome to Shamrock hotel"**);**

private JLabel shamrockL **=new** JLabel**(**""**);**// Label placeholder for an image

private JLabel makeL **=new** JLabel**(**""**);**// Label placeholder for an image

private JLabel cancelL **=new** JLabel**(**""**);**// Label placeholder for an image

private JLabel changeL **=new** JLabel**(**""**);**// Label placeholder for an image

private JLabel viewL **=new** JLabel**(**""**);**// Label placeholder for an image

private JButton logOutB **=new** JButton**(**"Log Out"**);**

private JButton makeB **=new** JButton**(**"Make a Booking"**);**

private JButton cancelB **=new** JButton**(**"Cancel Booking"**);**

private JButton viewB **=new** JButton**(**"View Bookings"**);**

private JButton changeB **=new** JButton**(**"Change Bookings"**);**

private JPanel panel2 **=new** JPanel**();**

public String userID**;**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CONSTRUCTOR \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CustomerGUI**(**String userID**){**

**super(**"Shamrock's hotel bookings"**);**

**this.**userID **=** userID**;**

setSize**(**800**,**630**);**// 800,630

setLocation**(**547**,**15**);**//500,50

setDefaultCloseOperation**(**JFrame**.**EXIT\_ON\_CLOSE**);**

panel2**.**setBackground**(new** Color**(**102**,**102**,**153**));**

panel2**.**setForeground**(**Color**.**BLACK**);**

panel2**.**setLayout**(null);**// Since no layout is used

// Setting the components

welcomeL**.**setBounds**(**26**,**22**,**540**,**59**);**

welcomeL**.**setForeground**(new** Color**(**0**,**0**,**51**));**

welcomeL**.**setFont**(new** Font**(**"Lucida Calligraphy"**,** Font**.**ITALIC**,**28**));**

shamrockL**.**setBounds**(**672**,**21**,**89**,**71**);**

Image img2 **=new** ImageIcon**(this.**getClass**().**getResource**(**"/image2small.png"**)).**getImage**();**// Setting the image path

shamrockL**.**setIcon**(new** ImageIcon**(**img2**));**// Source https://pixabay.com/en/shamrock-celtic-plant-clover-149821/

makeL**.**setBounds**(**23**,**152**,**360**,**170**);**

Image img3 **=new** ImageIcon**(this.**getClass**().**getResource**(**"/img3.jpg"**)).**getImage**();**// Setting image path

makeL**.**setIcon**(new** ImageIcon**(**img3**));**// Source https://pixabay.com/en/travel-hotel-rooms-hotel-room-bed-1677347/

cancelL**.**setBounds**(**401**,**152**,**360**,**170**);**

Image img4 **=new** ImageIcon**(this.**getClass**().**getResource**(**"/img4.jpg"**)).**getImage**();**// Setting image path

cancelL**.**setIcon**(new** ImageIcon**(**img4**));**// Source https://pixabay.com/en/hotel-guest-room-new-1330846/

changeL**.**setBounds**(**401**,**376**,**360**,**170**);**

Image img5 **=new** ImageIcon**(this.**getClass**().**getResource**(**"/img5.jpg"**)).**getImage**();**// Setting image path

changeL**.**setIcon**(new** ImageIcon**(**img5**));**// Source https://pixabay.com/en/bed-bedroom-blanket-books-cover-1846251/

viewL**.**setBounds**(**23**,**376**,**360**,**170**);**

Image img6 **=new** ImageIcon**(this.**getClass**().**getResource**(**"/img6.jpg"**)).**getImage**();**// Setting image path

viewL**.**setIcon**(new** ImageIcon**(**img6**));**// Source https://pixabay.com/en/hotel-kl%C3%A9ber-strasbourg-macaroon-787383/

logOutB**.**setBounds**(**534**,**40**,**89**,**23**);**

logOutB**.**setBackground**(new** Color**(**153**,**153**,**153**));**

logOutB**.**addActionListener**(new** ActionListener**(){**

publicvoid actionPerformed**(**ActionEvent e**){**

Log getBackToLog **=new** Log**();**

getBackToLog**.**setVisible**(true);**

dispose**();**

**}});**

makeB**.**setBounds**(**23**,**128**,**360**,**23**);**

makeB**.**addActionListener**(new** ActionListener**(){**

publicvoid actionPerformed**(**ActionEvent e**){**

Book gui4 **=new** Book**(**userID**);**

gui4**.**setVisible**(true);**

**}});**

cancelB**.**setBounds**(**401**,**128**,**360**,**23**);**

cancelB**.**addActionListener**(new** ActionListener**(){**

publicvoid actionPerformed**(**ActionEvent e**){**

Cancel cancel **=new** Cancel**();**

cancel**.**setVisible**(true);**

**}});**

viewB**.**setBounds**(**23**,**352**,**360**,**23**);**

viewB**.**addActionListener**(new** ActionListener**(){**

publicvoid actionPerformed**(**ActionEvent e**){**

View view **=new** View**(**userID**);**

view**.**setVisible**(true);**

view**.**viewBook**();**

**}});**

changeB**.**setBounds**(**401**,**352**,**360**,**23**);**

changeB**.**addActionListener**(new** ActionListener**(){**

publicvoid actionPerformed**(**ActionEvent e**){**

Change gui5 **=new** Change**();**

gui5**.**setVisible**(true);**

**}});**

// Adding the components to the panel

panel2**.**add**(**welcomeL**);**

panel2**.**add**(**shamrockL**);**

panel2**.**add**(**makeL**);**

panel2**.**add**(**cancelL**);**

panel2**.**add**(**changeL**);**

panel2**.**add**(**viewL**);**

panel2**.**add**(**logOutB**);**

panel2**.**add**(**makeB**);**

panel2**.**add**(**cancelB**);**

panel2**.**add**(**viewB**);**

panel2**.**add**(**changeB**);**

getContentPane**().**add**(**panel2**);**

**}**

**}**

Manager GUI

**import** java**.**awt**.**Container**;**

**import** java**.**awt**.**Dimension**;**

**import** java**.**awt**.**EventQueue**;**

**import** java**.**awt**.**FlowLayout**;**

**import** java**.**awt**.**GridLayout**;**

**import** java**.**awt**.**event**.**ActionEvent**;**

**import** java**.**awt**.**event**.**ActionListener**;**

**import** java**.**sql**.**Connection**;**

**import** java**.**sql**.**DriverManager**;**

**import** java**.**sql**.**ResultSet**;**

**import** java**.**sql**.**ResultSetMetaData**;**

**import** java**.**sql**.**SQLException**;**

**import** java**.**sql**.**Statement**;**

**import** java**.**time**.**LocalDate**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Arrays**;**

**import** java**.**util**.**List**;**

**import** javax**.**swing**.**DefaultListModel**;**

**import** javax**.**swing**.**JButton**;**

**import** javax**.**swing**.**JFrame**;**

**import** javax**.**swing**.**JLabel**;**

**import** javax**.**swing**.**JList**;**

**import** javax**.**swing**.**JPanel**;**

**import** javax**.**swing**.**JScrollPane**;**

publicclass Manager **extends** JFrame **{**

privatestaticfinallong serialVersionUID **=**1L**;**

privatestaticfinal String DRIVER **=**"com.mysql.jdbc.Driver"**;**// JDBC driver

privatestaticfinal String DATABASE **=**"myhotel"**;**// Database name

privatestaticfinal String HOST **=**"localhost"**;**// Database host IP

// static final String HOST = "10.1.63.200";// Uni

// the full URL:

privatestaticfinal String DATABASE\_URL **=**"jdbc:mysql://"**+** HOST **+**"/"**+** DATABASE**;**

private Connection connection **=null;**

private JLabel bookingL **=new** JLabel**(**"Booking"**);**// Requested booking

private JLabel availableL **=new** JLabel**(**"Rooms available"**);**

private JList**<**String**>** bookingArea **=new** JList**<**String**>();**

private JList**<**String**>** availableList **=new** JList**<**String**>();**

private JScrollPane bookingScrl **=new** JScrollPane**(**bookingArea**);**

private JScrollPane availableScrl **=new** JScrollPane**(**availableList**);**

private JButton assignB **=new** JButton**(**"Assign"**);**

Container con **=new** Container**();**

private JPanel leftPanel **=new** JPanel**();**

private JPanel rightPanel **=new** JPanel**();**

privatefinal JButton btnRefresh **=new** JButton**(**"Refresh"**);**

privatefinal JButton btnLogOut **=new** JButton**(**"Log out"**);**

privatefinal JLabel label **=new** JLabel**(**"<>"**);**

private ResultSet resultSet2**;**

public Manager**(){**

**super(**"Shamrock's administration"**);**

con **=** getContentPane**();**

getContentPane**().**setLayout**(new** GridLayout**(**0**,**2**));**

getContentPane**().**add**(**leftPanel**);**

getContentPane**().**add**(**rightPanel**);**

bookingScrl**.**setPreferredSize**(new** Dimension**(**200**,**300**));**

availableScrl**.**setPreferredSize**(new** Dimension**(**200**,**300**));**

// set models on to the list

leftPanel**.**setLayout**(new** FlowLayout**());**

rightPanel**.**setLayout**(new** FlowLayout**());**

leftPanel**.**add**(**bookingL**);**

leftPanel**.**add**(**bookingScrl**);**

//RELOADS RESERVATION LIST AND ROOMS LIST

btnRefresh**.**addActionListener**(new** ActionListener**(){**

publicvoid actionPerformed**(**ActionEvent e**){**

setToZero**();**

getUnassignedBookings**();**

getAvailableRooms**();**

**}**

**});**

btnLogOut**.**addActionListener**(new** ActionListener**(){**

publicvoid actionPerformed**(**ActionEvent e**){**

Log getBackToLog **=new** Log**();**

getBackToLog**.**setVisible**(true);**

dispose**();**

**}**

**});**

leftPanel**.**add**(**btnLogOut**);**

leftPanel**.**add**(**label**);**

leftPanel**.**add**(**btnRefresh**);**

rightPanel**.**add**(**availableL**);**

rightPanel**.**add**(**availableScrl**);**

rightPanel**.**add**(**assignB**);**

assignB**.**addActionListener**(new** ActionListener**(){**

publicvoid actionPerformed**(**ActionEvent e**){**

getAssign**();**

**}**

**});**

setSize**(**500**,**420**);**

setVisible**(true);**

setDefaultCloseOperation**(**JFrame**.**EXIT\_ON\_CLOSE**);**

setToZero**();**

getUnassignedBookings**();**

getAvailableRooms**();**

**}**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CONNECTION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

publicboolean setConnection**(){**

boolean connectionOK **=false;**

// connect to database

**try{**

// load the driver class

Class**.**forName**(**DRIVER**);**

// establish connection to database

connection **=** DriverManager**.**getConnection**(**DATABASE\_URL**,**"root"**,**""**);**

connectionOK **=true;**

**}catch(**SQLException sqlException**){**

System**.**err**.**println**(**"Driver loaded, but something went wrong elsewhere!"**);**

sqlException**.**printStackTrace**();**

**}catch(**ClassNotFoundException classNotFound**){**

System**.**err**.**println**(**"Couldn't find driver!"**);**

classNotFound**.**printStackTrace**();**

**}**

**return** connectionOK**;**

**}**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* UPDATE BOOKING FOR TODAY'S DATE

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Method that turns assigned value in 'room' table to zero when the end

// date is less than todays date

// If the end\_date < todays\_date -> update table table room where

// 'room\_assigned' = 1

publicvoid setToZero**(){**

setConnection**();**

**try{**

String today **=** getTodaysDate**();**

String query **=**"update room set room\_assigned = '0' where room\_end\_date < '"**+** today **+**"'"**;**

java**.**sql**.**PreparedStatement preparedStmt **=** connection**.**prepareStatement**(**query**);**

preparedStmt**.**execute**();**

**}**

**catch(**SQLException s**){**

System**.**out**.**println**(**"sql exception"**);**

**}finally{**

**try{**

connection**.**close**();**

**}catch(**Exception exception**){**

exception**.**printStackTrace**();**

**}**

**}**

**}**

public String getTodaysDate**(){**

LocalDate localDate **=** LocalDate**.**now**();**

String localDate2 **=** localDate**.**toString**();**

**return** localDate2**;**

**}**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* GET ALL UNASSIGNED BOOKINGS

public String getUnassignedBookings**(){**

DefaultListModel**<**String**>** listModel1 **=new** DefaultListModel**<**String**>();**

bookingArea**.**setModel**(**listModel1**);**

String columnValue **=null;**

ResultSet resultSet **=null;**

Statement statement **=null;**

ArrayList**<**String**>** list1 **=new** ArrayList**<**String**>();**

setConnection**();**

**try{**

setConnection**();**

String today2 **=** getTodaysDate**();**

statement **=** connection**.**createStatement**();**

resultSet **=** statement**.**executeQuery**(**

"select \* from bookig where assigned ='0' and start\_date = '"**+** today2 **+**"'"**);**

ResultSetMetaData rsmd **=** resultSet**.**getMetaData**();**

int columnsNumber **=** rsmd**.**getColumnCount**();**

// Loop to get the query from database

**while(**resultSet**.**next**()){**

**for(**int i **=**1**;** i **<=** columnsNumber**;** i**++){**

**if(**i **>**0**)**

//System.out.println(",");

columnValue **=** rsmd**.**getColumnName**(**i**)+**" ,"**+** resultSet**.**getString**(**i**);**

list1**.**add**(**columnValue**);**

**}**

**}**

int j **=**0**;**// Place holder for pack of results

//

**for(**int i **=**0**;** i **<** list1**.**size**()/**6**;** i**++){**// Divides the list size

// on column numbers

**for(**int c **=** j**;** c **<** j **+**6**;** c**++){**

listModel1**.**addElement**(**list1**.**get**(**c**)+**"\n"**);**

**}**

listModel1**.**addElement**(**"\n"**);**

listModel1**.**addElement**(**"---------------------------------"**);**

j **=** j **+**6**;**// Each pack has 6 results(6 table columns)

**}**

list1**.**clear**();**

**}catch(**SQLException s**){**

System**.**out**.**println**(**"sql exception"**);**

**}**

**finally{**

**try{**

resultSet**.**close**();**

statement**.**close**();**

connection**.**close**();**

**}catch(**Exception exception**){**

exception**.**printStackTrace**();**

**}**

**}**

**return** columnValue**;**

**}**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* GET ALL AVALABLE ROOMS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public String getAvailableRooms**(){**

DefaultListModel**<**String**>** listModel2 **=new** DefaultListModel**<**String**>();**

availableList**.**setModel**(**listModel2**);**

String columnValue2 **=null;**

ResultSet resultSet2 **=null;**

Statement statement2 **=null;**

ArrayList**<**String**>** list2 **=new** ArrayList**<**String**>();**

**try{**

setConnection**();**

statement2 **=** connection**.**createStatement**();**

resultSet2 **=** statement2**.**executeQuery**(**"select \* from room where room\_assigned = '0'"**);**

ResultSetMetaData rsmd **=** resultSet2**.**getMetaData**();**

int columnsNumber **=** rsmd**.**getColumnCount**();**

//System.out.println("Number of columns: " + columnsNumber);// test

// Loop to get the query from database

**while(**resultSet2**.**next**()){**

**for(**int i **=**1**;** i **<=** columnsNumber**;** i**++){**

**if(**i **>**0**)**

// System.out.print(",");

columnValue2 **=** rsmd**.**getColumnName**(**i**)+**" ,"**+** resultSet2**.**getString**(**i**);**

list2**.**add**(**columnValue2**);**

**}**

**}**

int j **=**0**;**// Place holder for pack of results

**for(**int i **=**0**;** i **<** list2**.**size**()/**5**;** i**++){**// Divides the list size

// on column numbers

**for(**int c **=** j**;** c **<** j **+**5**;** c**++){**

listModel2**.**addElement**(**list2**.**get**(**c**)+**"\n"**);**

**}**

listModel2**.**addElement**(**"\n"**);**

listModel2**.**addElement**(**"---------------------------------"**);**

j **=** j **+**5**;**// Each pack has 5 results(5 table columns)

**}**

list2**.**clear**();**

**}catch(**SQLException s**){**

System**.**out**.**println**(**"sql exception"**);**

**}finally{**

**try{**

resultSet2**.**close**();**

statement2**.**close**();**

connection**.**close**();**

**}catch(**Exception exception**){**

exception**.**printStackTrace**();**

**}**

**}**

**return** columnValue2**;**

**}**

@SuppressWarnings**(**"unused"**)**

publicvoid getAssign**(){**

// SELECTING FROM BOOKING

String columnValue **=null;**

ResultSet resultSet **=null;**

Statement statement **=null;**

ResultSetMetaData rsmd **=null;**

String selected1 **=** bookingArea**.**getSelectedValue**().**toString**();**

List**<**String**>** selectList **=** Arrays**.**asList**(**selected1**.**split**(**","**));**// Comma added to substring to get the booking ID.

String bookingIdFromSelected1**=**selectList**.**get**(**1**);**// Get the first value of the string list [0,1].

**try{**

setConnection**();**

statement **=** connection**.**createStatement**();**

resultSet **=** statement**.**executeQuery**(**"select \* from bookig where bookingID = '"**+** bookingIdFromSelected1 **+**"'"**);**

rsmd **=** resultSet**.**getMetaData**();**

int columnsNumber **=** rsmd**.**getColumnCount**();**

// Loop to get the query from database

**while(**resultSet**.**next**()){**

**for(**int i **=**1**;** i **<=** columnsNumber**;** i**++){**

**if(**i **>**0**)**

//System.out.print(",");

columnValue **=** resultSet**.**getString**(**i**);**

**}**

**}**

//System.out.println(" this is selected 1" + selected1);

//System.out.println(columnValue);// not working

**}**

**catch(**SQLException sqlException**){**

sqlException**.**printStackTrace**();**

// Print an error message instead of results

System**.**err**.**println**(**"ERROR! Something went wrong, please consult stack trace!"**);**

**}**

// SELECTING FROM ROOMS AVALIABLE

String columnValue2 **=null;**

resultSet2 **=null;**

Statement statement2 **=null;**

ResultSetMetaData rsmd2 **=null;**

String selected2 **=** availableList**.**getSelectedValue**().**toString**();**

List**<**String**>** selectList2 **=** Arrays**.**asList**(**selected2**.**split**(**","**));**// Comma added to substring to get the room ID.

String bookingIdFromSelected2**=**selectList2**.**get**(**1**);**// Get the first element of the string [0,1]

**try{**

setConnection**();**

statement2 **=** connection**.**createStatement**();**

resultSet2 **=** statement2**.**executeQuery**(**"select \* from bookig where bookingID = '"**+** bookingIdFromSelected2 **+**"'"**);**

rsmd2 **=** resultSet2**.**getMetaData**();**

int columnsNumber2 **=** rsmd2**.**getColumnCount**();**

// Loop to get the query from database

**while(**resultSet**.**next**()){**

**for(**int i **=**1**;** i **<=** columnsNumber2**;** i**++){**

**if(**i **>**0**)**

// System.out.print(",");

columnValue2 **=** resultSet2**.**getString**(**i**);**

**}**

**}**

//System.out.println(selected2);

//System.out.println(columnValue2);

**}**

**catch(**SQLException sqlException**){**

sqlException**.**printStackTrace**();**

// Print an error message instead of results

System**.**err**.**println**(**"ERROR! Something went wrong, please consult stack trace!"**);**

**}**

// POPULATING TABLE adminassign

**try{**

setConnection**();**

String query **=**"insert into adminassign (bookingID, RoomID) values ('"**+** bookingIdFromSelected1 **+**"','"**+** bookingIdFromSelected2 **+**"')"**;**

java**.**sql**.**PreparedStatement preparedStmt **=** connection**.**prepareStatement**(**query**);**

preparedStmt**.**execute**();**

**}**

**catch(**SQLException sqlException**){**

sqlException**.**printStackTrace**();**

// Print an error message instead of results

System**.**err**.**println**(**"ERROR! Something went wrong, please consult stack trace!"**);**

**}**

String columnValue4 **=null;**

String columnValue5 **=null;**

ResultSet resultSet4 **=null;**

Statement statement4 **=null;**

ResultSetMetaData rsmd4 **=null;**

**try{**// Getting start and end dates from the selected booking

setConnection**();**

statement4 **=** connection**.**createStatement**();**

resultSet4 **=** statement4**.**executeQuery**(**"select \* from bookig where bookingID = '"**+** bookingIdFromSelected1 **+**"'"**);**

rsmd4 **=** resultSet4**.**getMetaData**();**

int columnsNumber4 **=** rsmd4**.**getColumnCount**();**

// Loop to get the query from database

**while(**resultSet4**.**next**()){**

**for(**int i **=**1**;** i **<=** columnsNumber4**;** i**++){**

**if(**i **>**0**)**

//System.out.print(",");

columnValue4 **=** resultSet4**.**getString**(**"start\_date"**);**

columnValue5 **=** resultSet4**.**getString**(**"end\_date"**);**

**}**

**}**

//System.out.println(" this is start\_date : " + columnValue4);

//System.out.println(" this is end\_date : " + columnValue5);

**}**

**catch(**SQLException sqlException**){**

sqlException**.**printStackTrace**();**

// Print an error message instead of results

System**.**err**.**println**(**"ERROR! Something went wrong, please consult stack trace!"**);**

**}**

**try{**

String query **=**"update room set room\_start\_date= '"**+** columnValue4 **+**"',room\_end\_date = '"**+** columnValue5**+**"',room\_assigned = '1' where RoomID = '"**+** bookingIdFromSelected2 **+**"'"**;**

java**.**sql**.**PreparedStatement preparedStmt **=** connection**.**prepareStatement**(**query**);**

preparedStmt**.**execute**();**

**}catch(**SQLException s**){**

System**.**out**.**println**(**"exeption number EDW"**);**

**}**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**try{**

String query2 **=**"update bookig set assigned = '1' where bookingID = '"**+** bookingIdFromSelected1 **+**"'"**;**

java**.**sql**.**PreparedStatement preparedStmt2 **=** connection**.**prepareStatement**(**query2**);**

preparedStmt2**.**execute**();**

**}catch(**SQLException s**){**

System**.**out**.**println**(**"exeption from updating bookingID to 1"**);**

**}**

**finally{**

**try{**

resultSet**.**close**();**

statement**.**close**();**

connection**.**close**();**

**}catch(**Exception exception**){**

exception**.**printStackTrace**();**

**}**

**}**

**}**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MAIN \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

publicstaticvoid main**(**String**[]** args**){**

EventQueue**.**invokeLater**(new** Runnable**(){**

publicvoid run**(){**

**try{**

Manager manager2 **=new** Manager**();**

manager2**.**setVisible**(true);**

**}catch(**Exception e**){**

e**.**printStackTrace**();**

**}**

**}**

**});**

**}**

**}**

## Appendix C - References

The following table shows the resources that were used throughout the group project:

|  |  |
| --- | --- |
| No. | Resources |
| 1 | Page 517, Chapter 8, Late Binding Savith Book |
| 2 | <http://stackoverflow.com/questions/3483156/whats-the-right-way-to-represent-phone-numbers> |
| 3 | <https://docs.oracle.com/javase/7/docs/api/java/awt/GridBagLayout.html> |
| 4 | <http://stackoverflow.com/questions/13028917/how-to-position-components-with-gridbaglayout> |
| 5 | <http://bioportal.weizmann.ac.il/course/prog2/tutorial/uiswing/layout/gridbag.html> |
| 6 | <http://www.formdev.com/jformdesigner/doc/layouts/gridbaglayout/> |
| 7 | <http://javatechniques.com/blog/gridbaglayout-example-a-simple-form-layout/> |
| 8 | <http://www.tutorialspoint.com/swing/swing_gridlayout.htm> |
| 9 | <http://www.java2s.com/Tutorial/Java/0240__Swing/HowtoUseGridLayout.htm> |
| 10 | <https://docs.oracle.com/javase/tutorial/uiswing/layout/grid.html> |
| 11 | Lecture slides CIS016-1 |
| 12 | Lecture slides CIS017-1 |