



INDIVIDUAL ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

AAPP004-4-2-JP

JAVA PROGRAMMING

STUDENT NAME: YIP ZI XIAN

TP NUMBER: TP059963

HAND OUT DATE: 10th January 2022

HAND IN DATE: 10th April 2022

WEIGHTAGE: 60%

INSTRUCTIONS TO CANDIDATES:

- 1. Students are advised to underpin their answers with the use of references (cited used the APA 7th generation System of Referencing.**
- 2. Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld.**
- 3. Cases of plagiarism will be penalised.**
- 4. The assignment should be bound in an appropriate style (comb bound or stapled).**
- 5. Where the assignment should be submitted in both hardcopy and softcopy, the softcopy of the written assignment and source code (where appropriate) should be on a CD in an envelope / CD cover and attached to the hardcopy.**

Table of Contents

1.0	Source Codes	3
1.1	Header Files / Import Files.....	3
1.2	Variables	5
1.3	Control Structures	7
1.4	Looping Structures	9
1.5	Object-Oriented Programming Concept	10
1.6	Try-Catch Structures.....	11
1.7	File Handling	12
2.0	Sample Output of the System.....	14
2.1	Login GUI.....	14
2.2	Main GUI.....	15
2.3	History GUI	16
2.4	Receipt GUI	17
3.0	Additional Features of Java	18
3.1	JTable.....	18
3.2	JCalendar.....	18
3.3	JDateChooser	19
4.0	Assumptions.....	20
5.0	References.....	21

1.0 Source Codes

1.1 Header Files / Import Files

```
import javax.swing.JOptionPane;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;

import java.awt.Color;
import java.text.SimpleDateFormat;
import java.util.*;
import java.util.concurrent.TimeUnit;
import java.io.*;
import java.text.ParseException;
import javax.swing.*;

import javax.swing.table.DefaultTableModel;
import java.io.*;
import java.io.*;

import com.toedter.calendar.JDateChooser;
import java.io.*;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.concurrent.TimeUnit;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;

import java.io.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
```

Figure 1 – 8: All import files of Login_GUI.java, History_GUI.java, Main_GUI.java, Receipt_GUI.java, OOP_Constructor_Main.java, OOP_GS_Credential.java, OOP_Normal_History.java, and OOP_Normal_Receipt.java respectively.

- **com.toedter.calendar.JDateChooser**

This import file is to acquire the permission or directory to use JDateChooser as a data type.

- **java.awt.Color**

This import file is to enable the colour class in order to apply different colour to components which allows colour changes like JPanel, using the value of RGB or HUE.

- **java.io.***

This import file is to allow the process of input and output like performing file handling.

- **java.text.ParseException**

This import file is to allow the usage of Parse exception error to handle error which occurs if a string with specific format failed to be parsed.

- **java.text.SimpleDateFormat**

This import file is to allow the parsing and formatting of dates with the user's format.

- **java.util.concurrent.TimeUnit**

This import file is to enable the time to be presented as a given unit of granularity. It can convert time with hours, minutes, and seconds to a single time unit like seconds, or milliseconds.

- **java.util.Date**

This import file is to allow the representation of date and time in Java.

- **javax.swing.***

This import file is to enable the interfaces which allows the development of input methods like JOptionPane to allow a popup message in JFrames or any other Java runtime environments.

- **javax.swing.table.DefaultTableModel**

This import file is to enable the initialization of table model which can pass the data in the table from table columns or rows.

1.2 Variables

- Date

```
// Convert String to Date
SimpleDateFormat dmyFormat = new SimpleDateFormat("dd-MM-yyyy");
Date tempDataDate;
```

Figure 9: Example of date variable

Date variables are used to declare an input as a date with the format which the user set using SimpleDateFormat. Date variables will have different functions like before() and after() to determine whether the date is before or after another date.

- Int

```
// Calculate new prices
int nBP = Integer.parseInt(newStay) * 350;
int nTT = Integer.parseInt(newStay) * 10;
int nST = (int) (nBP * 0.1);
int nTP = nBP + nTT + nST;
```

Figure 10: Examples of integer variable

Integer variables are used to declare numbers, either positive or negative. Integer variables can be used to perform arithmetic operations like addition, subtraction, multiplication, division etc.

- JButton[]

```
JButton[] buttonArr = new JButton[] {SR1BTN, SR2BTN, SR3BTN, SR4BTN, SR5BTN, SR6BTN,
```

Figure 11: Example of JButton[] array

JButton[] variables are used as a single-dimensional array to include only JButton data types.

- JPanel[]

```
JPanel[] panelArr = new JPanel[] {SR1, SR2, SR3, SR4, SR5, SR6,
```

Figure 12: Example of JPanel[] array

JPanel[] variables are used as a single-dimensional array to include only JPanel data types.

- JTextField[]

```
// Disable Textfields by default
JTextField[] fieldArr = new JTextField[] {inputCustomerName, inputIC, inputContact, inputEmail};
```

Figure 13: Example of JTextField[] array

JTextField variables are used as a single-dimensional array to include only JTextField data types.

- Long

```
long daysOfStay = checkout.getTime() - checkin.getTime();
long convertTime = TimeUnit.DAYS.convert(daysOfStay, TimeUnit.MILLISECONDS);
```

Figure 14: Examples of long variables

Long variables are used to store whole numbers with 8 bytes sizes.

- String

```
// Get Admin Input
String roomID = inputRoomID.getText();
String customerName = inputCustomerName.getText();
String IC = inputIC.getText();
String contact = inputContact.getText();
String email = inputEmail.getText();
```

Figure 15: Examples of string variables

String variables are used to declare an input as a text. For examples, equals() can be used to compare both string variables to find the exact same value or data.

- String[]

```
// Create Array in Array
String[] fileData;
```

Figure 16: Example of string[] array

String[] variables are used as a single-dimensional array to include only string data types.

- String[][]

```
String[][] tempData = new String[lineLength][12];
```

Figure 17: Example of string[][] multidimensional array

String[][] variables are used as a double-dimensional array to include only string data types. It can store multiple arrays within.

- Var

```
var dateOffset = 1;
var checkin = dmyFormat.parse(history[j][5]);
var checkout = dmyFormat.parse(history[j][6]);
```

Figure 18: Examples of var variables

Var variables are used to declare any text or integer as a local variable that can only be used within a function or method.

1.3 Control Structures

- If Statement

```
if((tempData[i][0].equals(room)) && (tempData[i][1].equals(name)) && (tempData[i][5].equals(checkin)))
{
    try
    {
        // Convert String to Date
        SimpleDateFormat dmyFormat = new SimpleDateFormat("dd-MM-yyyy");

        Date x = checkoutDate.getDate();
        String tempDataDate = dmyFormat.format(x);

        tempData[i][2] = IC.getText();
        tempData[i][3] = contact.getText();
        tempData[i][4] = email.getText();
        tempData[i][6] = tempDataDate;
        tempData[i][7] = stay.getText();
        tempData[i][8] = bP.getText();
        tempData[i][9] = tT.getText();
        tempData[i][10] = sT.getText();
        tempData[i][11] = tP.getText();
    }
    catch(Exception e)
    {
        System.out.println("Error 3-4");
    }
}
```

Figure 19: Example of if statement

If statements are used to determine whether the code should run or not based the argument given to it.

- If-Else Statement

```
if(j != 11)
{
    bw.write(tempData[i][j] + " | ");
}
else
{
    bw.write(tempData[i][j]);
    bw.write("\n");
}
```

Figure 20: Example of if-else statement

If-Else statements are used to determine whether the code should run or not based the argument given to it, and if the argument is not satisfied, it will run the code in Else statement.

- Nested-If Statement

```
if(newTempData[i][j] != null)
{
    if(j != 11)
    {
        bw.write(newTempData[i][j] + " | ");
    }
    else
    {
        bw.write(newTempData[i][j]);
        bw.write("\n");
    }
}
```

Figure 21: Example of nested-if statement

Nested-If statements are multiple If statement included together to determine a more complex logic or operations.

1.4 Looping Structures

- For Loop Statement

```
// For loop old and new data back into the file
for(int i = 0; i < newTempData.length; i++)
{
    for(int j = 0; j < 12; j++)
    {
        if(newTempData[i][j] != null)
        {
            if(j != 11)
            {
                bw.write(newTempData[i][j] + " | ");
            }
            else
            {
                bw.write(newTempData[i][j]);
                bw.write("\n");
            }
        }
    }
}
```

Figure 22: Example of for loop statement within a for loop statement

For loop statements are used to loop for a given amount of time instead of writing multiple lines of the same code.

- While Loop Statement

```
while((line = br.readLine()) != null)
{
    fileData = line.split("\\s\\s|\\s");

    for(int i = 0; i < fileData.length; i++)
    {
        tempData[pass][i] = fileData[i];
    }

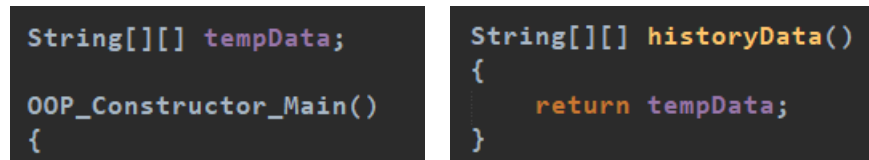
    pass++;
}
```

Figure 23: Example of while loop statement

While loop statements are used to loop as long as the argument is true, it won't break unless the argument returns a False value, or else it will be in a loop forever.

1.5 Object-Oriented Programming Concept

- Constructor Method

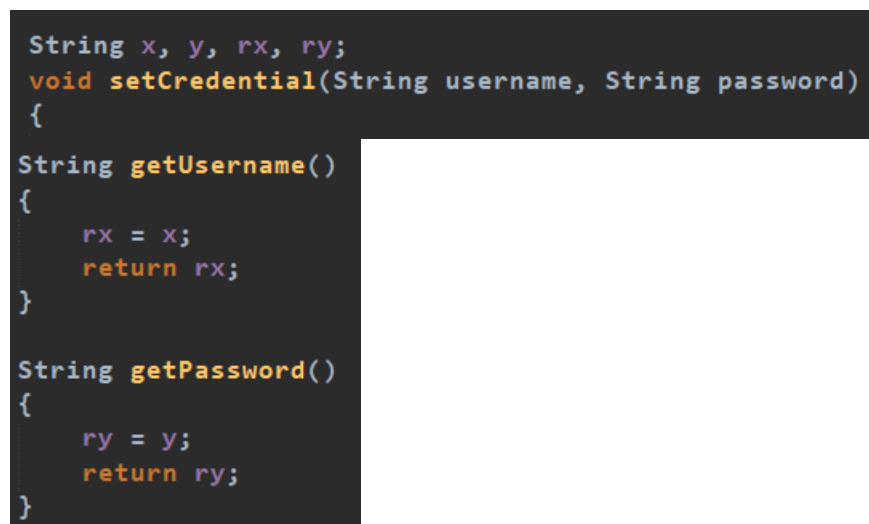


```
String[][] tempData;  
  
OOP_Constructor_Main()  
{  
  
String[][] historyData()  
{  
    return tempData;  
}
```

Figure 24: Example of constructor method and the return value

Constructor method is one of many object-oriented programming concepts in Java. It declares a method in a different class. There are some ground rules for constructor method. The constructor method name must be the same as the class name and it must not have a return type like void. Constructor is called when the object is created.

- Get-Set Method (Encapsulation)



```
String x, y, rx, ry;  
void setCredential(String username, String password)  
{  
  
String getUsername()  
{  
    rx = x;  
    return rx;  
}  
  
String getPassword()  
{  
    ry = y;  
    return ry;  
}
```

Figure 25: Example of Get-Set Method and the return values

Get-Set method is another object-oriented programming concepts in Java. It only works for class which has declared as private, and a public get and set method are used to retrieve and update the value of the private variable. It is totally inaccessible from an outside class. The public get method is used to return the variable value from the private class, and the public set method is used to set the value.

- Normal Method

```
// OOP Normal Method - Search and Display
void search(String searchName, JTable table)
{
// OOP Normal Method - Display full data from file
void viewSelected(String room, String name, String checkin
{
```

Figure 26: Examples of normal method with no return value

Normal method is the most common type of object-oriented programming concepts in Java. A class is created to run the code whenever the method of the class is being called.

1.6 Try-Catch Structures

```
try
{
// Create New File
File f = new File("Admin_Credential.txt");

// Insert Default Admin Username and Password
FileWriter fw = new FileWriter(f);
fw.write("Admin | admin");
fw.close();

// Read File
BufferedReader br = new BufferedReader(new FileReader("Admin_Credential.txt"));
String credential;

credential = br.readLine();

String[] credentialArr = credential.split("\\s\\s|\\s");

// Admin Username and Password
x = credentialArr[0];
y = credentialArr[1];

br.close();
}
catch (IOException e)
{
System.out.println("Error");
}
```

Figure 27: Example of try-catch statement

Try-Catch structures are used to return something the user written instead of crashing the system with errors. There are multiple exceptions to be caught like IOException for file handling, ParseException for parsing different data types and etc.

1.7 File Handling

- **BufferedReader**

```
// Read File  
BufferedReader br = new BufferedReader(new FileReader("Admin_Credential.txt"));
```

Figure 28: Example of BufferedReader

BufferedReader is used to read data from a given file or the file can be declared within by using FileReader.

- **BufferedWriter**

```
// Write File  
BufferedWriter bw = new BufferedWriter(new FileWriter(f));
```

Figure 29: Example of BufferedWriter

BufferedWriter is used to write data into a given file or the file can be declared within by using FileWriter.

- **Close**

```
// Close All Opened File  
br.close();  
bw.close();  
lnr.close();
```

Figure 30: Example of closing file

Close is used to close the access to an opened file to avoid any mistakes or errors.

- **File**

```
// Extract available data from file  
File f = new File("Resort_History.txt");
```

Figure 31: Example of declare a file

File is used to declare and create a new file with the given name, a Boolean can be included to avoid the existing data in the file to be overwritten by the new data.

- **Flush**

```
// Clear existing data in file  
bw.flush();
```

Figure 32: Example of flushing a file before writing

Flush is used to clear all the data in the file.

- **LineNumberReader**

```
LineNumberReader lnr = new LineNumberReader(new FileReader(f));
```

Figure 33: Example of LineNumberReader

LineNumberReader is used to count the number of lines in a file.

- **Readline**

```
while((line = br.readLine()) != null)  
{
```

Figure 34: Example of reading file line by line

Readline is used to read the file line by line instead of character by character.

- **Split**

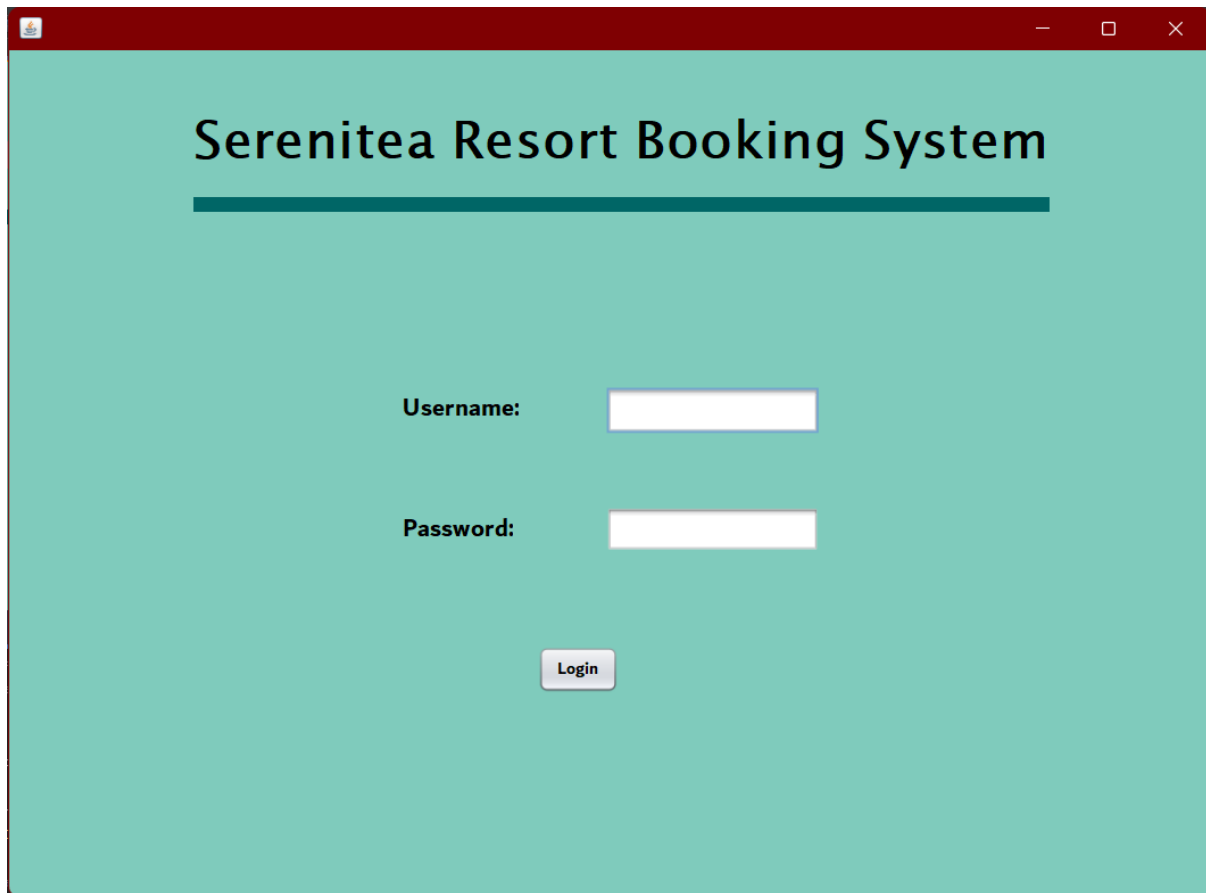
```
fileData = line.split("\\s\\|\\s");
```

Figure 35: Example of splitting each line

Split is used to separate a line with the given divider.

2.0 Sample Output of the System

2.1 Login GUI



Serenitea Resort Booking System

Username:

Password:

Login

Figure 36: Actual Login interface for Serenitea Resort Booking System

In Login GUI, admin will need type the correct username and password to access the system. In the case, the username is “Admin”, and the password is “admin”. The password input will be hidden to prevent any illegal behaviour by non-admin personnel.

2.2 Main GUI

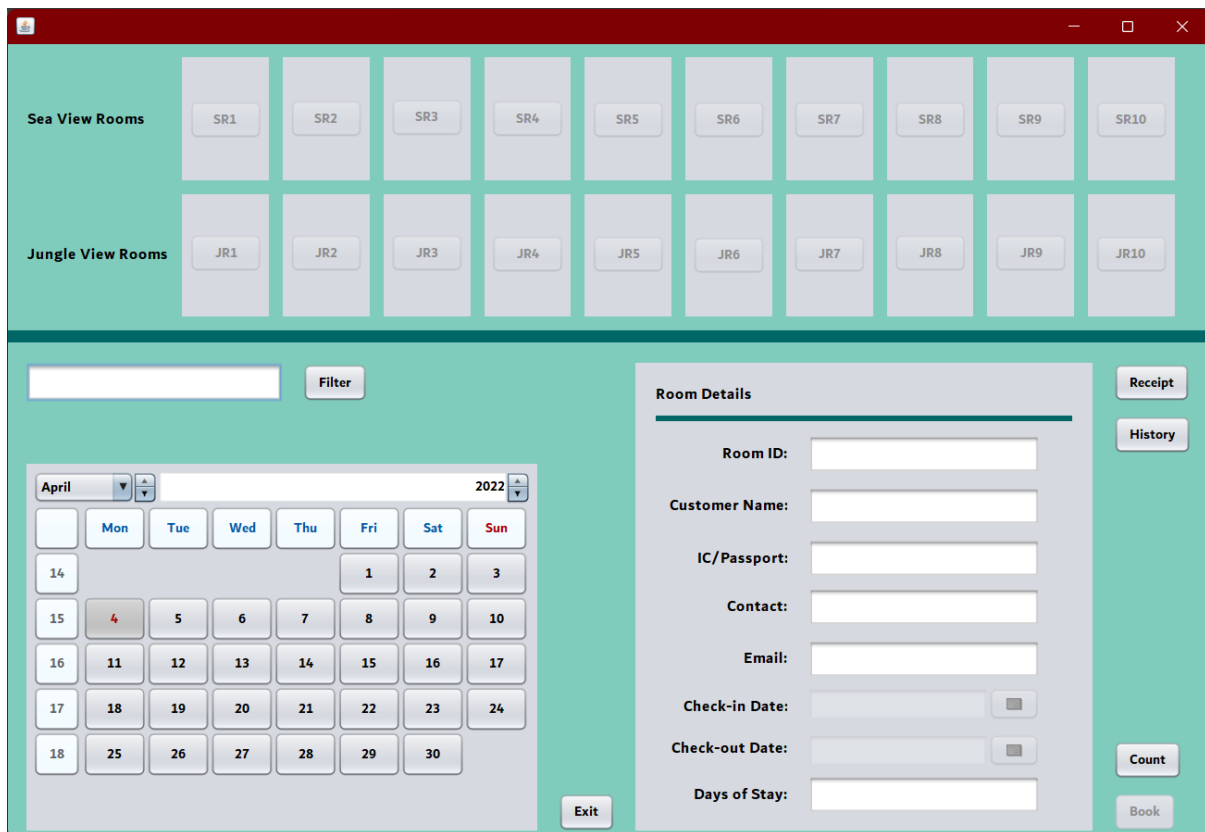


Figure 37: Actual Main interface for Serenitea Resort Booking System with filter calendar and booking form

In Main GUI, there are 20 rooms display, a calendar to check the room availabilities, and a form to include all customer booking details. The History and Receipt buttons are used to access another page which is to modify the booking details and view the invoice. Initially, when the admin hasn't filtered the date, all the room buttons are disabled to avoid any mis-click.

When the calendar date is clicked and filtered, the room which are booked will changed to Red and the button of the room can't be clicked. The Count button are used to calculate the days of stay and it will also run the receipt calculation in the back and save it into Customer_Receipt.txt. The Book button will be enabled when the admin has counted the days of stay.

When the admin clicked Book, a confirm message will prompt admin to accept or cancel the booking process. After a successful booking, the book button will be disabled again. Main GUI will not close when admin click History or Receipt GUI.

2.3 History GUI

Figure 38: Actual History interface for Serenitea Resort Booking System with table, managing buttons and preview form

In History GUI, there are a table and a preview form. There are edit, count, update, delete button. Admin can also search for the customer's exact name or just click Search to display all the data in the file. When the admin clicked one of the rows in the table, all data will be display in the preview form.

The Count and Update button will be available only when admin decided to edit. Count button is used to calculate the new days of stay and new prices. If admin wants to delete the data, a confirm message will prompt admin for accept or cancel. After editing or deleting the data, admin will need to refresh the table to view the updated data. The Count and Update button will be disabled after updating.

The Back button will exit the history GUI.

2.4 Receipt GUI

Search Customer Name:

Room ID	Customer Name	Check-in Date	Check-out Date
---------	---------------	---------------	----------------

Receipt Preview

Room ID:

Customer Name:

Admin Name:

Payment Date:

Check-in Date:

Check-out Date:

Days of Stay:

Base Price:

Tourism Tax (per night):

Service Tax (10%):

Total Charges:

Figure 38: Actual History interface for Serenitea Resort Booking System with table and preview form.

In Receipt GUI, there are also a table and a preview form. The Search button will display all the data in the file unless admin search for customer's exact name.

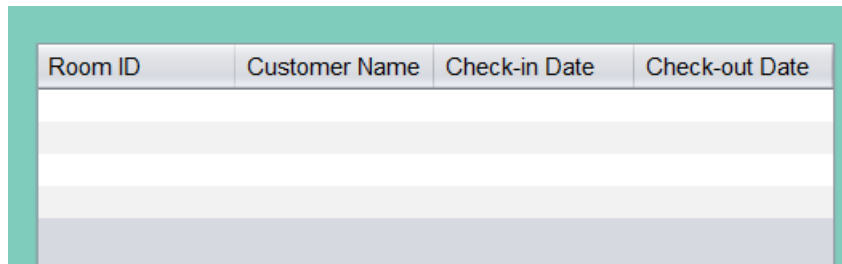
The Back button will exit the receipt GUI.

3.0 Additional Features of Java

3.1 JTable

```
// Get Selected Row in Table
int row = displayHistoryTable.getSelectedRow();

// Get Table Model
DefaultTableModel model = (DefaultTableModel) displayHistoryTable.getModel();
```



Room ID	Customer Name	Check-in Date	Check-out Date

Figure 39 - 40: Example of getting a row and model from JTable displayHistoryTable

JTable is a feature of JFrames to include selected data into the row and column of the table. Row and column can be set accordingly.

3.2 JCalendar

```
// Get and display filter date
SimpleDateFormat dmyFormat = new SimpleDateFormat("dd-MM-yyyy");
String sDate = dmyFormat.format(filterCalendar.getDate());
filterDate.setText(sDate);
```

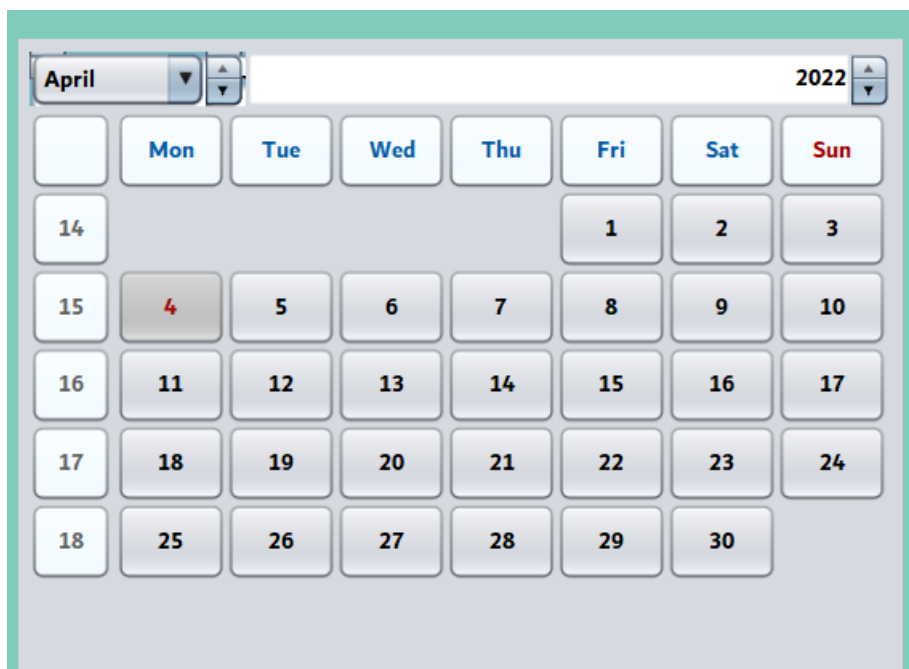


Figure 41 - 42: Example of getting date to filter from JCalendar filterCalendar

JCalendar is a library of external JAR file downloaded from web browser. It can be used to select a date with day, month, and year.

3.3 JDateChooser

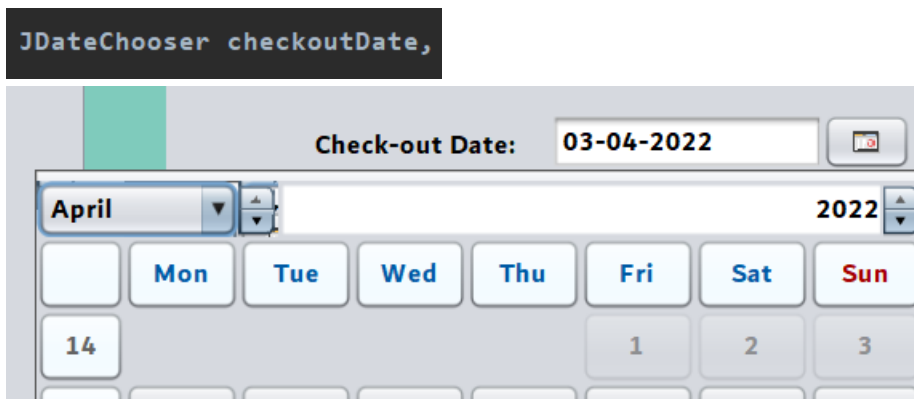


Figure 43 - 44: Example of JDateChooser variable in an argument

JDateChooser is from the same library of external JAR file as JCalendar. It displayed like a text field and when the box is clicked, a JCalendar will pop up. It can also be used as a data type.

4.0 Assumptions

1. No same room ID may exist in text file as the filter function will only work for distinct Room ID.
2. Admin will know the correct input for each text fields and will not input the wrong format.
3. Admin will check the room availability for the room if the room was booked. For example, if a customer wants to book SR1 room, but it has been booked for a certain period, admin will need to check for the available day to book the room in history GUI.
4. Only resort admin will know the username and password of the system and it is totally secure.
5. Room ID, customer name, and check-in date are used together as a unique key to identify each customer data.
6. Only one admin can access the system.

5.0 References

Oracle (n.d.) *Java Documentation*. Retrieved from:

<https://docs.oracle.com/en/java/index.html>

Refsnes Data (n.d.) *Java Tutorial*. Retrieved from: <https://www.w3schools.com/java/>

Parewa Labs Pvt. Ltd. (n.d.) *Java References*. Retrieved from:

<https://www.programiz.com/java-programming/library>

Stackoverflow (n.d.) *How can I determine if a date is between two dates in Java?* Retrieved from: <https://stackoverflow.com/questions/883060/how-can-i-determine-if-a-date-is-between-two-dates-in-java>