**Name:** Donark Patel (DP663)

**Class:** IS114-451

**Professor:** Maura Deek

**Date:** 10/07/2019

**1. Formulating the Problem**

**1.1 Problem Description**

Create a program in Java that creates a GUI. In the GUI, user are allowed to create a reservation of a conference room.

**1.2 Verbalization**

*What is the goal?*

Create a GUI for users to reserve a conference room.

*What are the givens?*

User’s first name, last name, date of reservation, start time of reservation, end time of reservation, number of attendees, size of room, beverage/food information, paper product information.

*What are the unknowns?*

Cost of the event.

1.3 **Information Elicitation**

*Goal*: Create a GUI for users to reserve a conference room.

*Givens*: User’s first name, last name, date of reservation, start time of reservation, end time of reservation, number of attendees, size of room, beverage/food information, paper product information.

*Unknowns*: Cost of the event.

*Conditions*:

Conference room pricing:

* Small conference room - $50/hour and holds 25 people
* Medium conference room - $100/hour and holds 50 people
* Large conference room - $150/hour and holds 75 people

Beverages/Food/Paper Products/Utensils options pricing:

* Coffee/Tea - $1/person
* Soda - $1.50/person
* Cookies/Cakes - $2/person
* Sandwiches - $4/person
* Paper products/utensils -$2.50/person

**2. Planning the Solution**

**2.1 Solution Strategy**

Ask the user to input the values of User’s first name, last name, date of reservation, start time of reservation, end time of reservation, number of attendees, size of room, beverage/food information, paper product information. Using this value calculate the cost of the event.

**)2.2 Goal Decomposition**

*Sub-goal 1*: Create the GUI panel

*Sub-goal 2*: Add the component to the GUI panel.

*Sub-goal 3*: Create an ActionListener class.

*Sub-goal 4*: Insert the value of users input from the component to local variable.

*Sub-goal 5*: Calculate the cost of the event from the condition provided and display result.

**2.3 Resources**

*Relevant formulas*

Sum += Time of event \* Selected size room cost.

Sum += number of attendees \* Selected item cost.

**2.4 Data Organization and Description**

Input:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Origin | Used in Sub-goal # |
| frame | GUi Frame | User | 1 |
| textFirstName | First name | User | 2, 3, 4, 5 |
| texLastName | Last name | User | 2, 3, 4, 5 |
| textReservationDate | Reservation Date | User | 2, 3, 4, 5 |
| textNumberOfAttendees | Number of attendees | User | 2, 3, 4, 5 |
| comboBoxStartTime | Start time | User | 2, 3, 4, 5 |
| comboBoxEndTime1 | End time | User | 2, 3, 4, 5 |
| chckbxSmallRoom | Small size room | User | 2, 3, 4, 5 |
| chckbxMediumRoom | Medium size room | User | 2, 3, 4, 5 |
| chckbxLargeRoom | Large size room | User | 2, 3, 4, 5 |
| chckbxCoffee | Coffee | User | 2, 3, 4, 5 |
| chckbxSoda | Soda | User | 2, 3, 4, 5 |
| chckbxCookies | Cookies | User | 2, 3, 4, 5 |
| chckbxSandwiches | Sandwich | User | 2, 3, 4, 5 |
| chckbxPaperProduct | Paper Product | User | 2, 3, 4, 5 |

Output:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Origin | Used in Sub-goal # |
| TextArea | To display the result | Screen | 5 |
| scrollPane | Provide scroll bar to TextArea | Screen | 5 |

**3. Designing the Solution**

**)3.1 Structure Chart**

*First Level Decomposition*

*Goal Refinement*

**Sub-goal 1**

Create the GUI Panel

**Sub-Goal 1.1**

From main class initialize the GUI panel class

**Sub-goal 2**

Create object of each component

**Sub-goal 2.1**

.setBound of the component.

**Sub-goal 2.2**

Add component to panel.

**Sub-goal 3**

Create a ActionListener class.

**Sub-goal 4**

Insert the value of users input from the component to local variable.

**Sub-goal 4.1**

Validate each user inputs.

**Sub-goal 5**

Calculate the cost of the event.

**Sub-goal 5.1**

Using IF statement, identify which size room user wants.

**Sub-goal 5.2**

Using IF statement, identify which products user wants.

**Sub-goal 5.3**

Displays the results in textArea.

*Second Level Decomposition*

**()3.2 Module and Data Specifications**

**Name**: actionPerformed – handles the calculate button event

**Input**: User first name, last name, Date of Reservation, Time room needed, Number of attendees, Size of room, products needed.

**Output**: Print’s the cost of the event.

**Logic**: Print user info and cost.

**Name**: isNumber – Validate the user input. It determines if it is number or not.

**Input**: String

**Output**: Boolean

**Logic**: Using a try catch statement. Try to parse the string into an integer. It is pass then return true, otherwise return false.

**Name**: Main – Class Frame class.

**Input**: None

**Output**: GUI panel

**Data:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Structure |
| clientFirstName | String | Variable |
| clientLastName | String | Variable |
| input | String | Variable |
| tempDate | Int | Variable |
| numberOfAttendees | Int | Variable |
| selectedEndTime | Int | Variable |
| sum | Double | Variable |
| totalHours | int | Variable |
| SMALL\_ROOM\_PRICE | Final Int | Variable |
| MEDIUM\_ROOM\_PRICE | Final Int | Variable |
| LARGE\_ROOM\_PRICE | Final Int | Variable |
| COFFEE\_PRICE | Final Double | Variable |
| SODA\_PRICE | Final Double | Variable |
| COOKIES\_PRICE | Final Double | Variable |
| SANDWICHES\_PRICE | Final Double | Variable |
| PAPER\_PRODUCT\_PRICE | Final Double | Variable |

**3.3 Algorithm**

*Logic*

1.0: Create the GUI Panel

1.1: From main class initialize the GUI panel class.

2.0: Create object of component.

2.1: .setBound of each component

2.2: Add component to panel.

3.0: Create an ActionListner class.

4.0: Insert the value of users input from the component to local variable.

4.1 Validate each user inputs.

5.0: Calculate the cost of the event.

5.1: Using IF statement, identify which size room user wants.

5.2: using IF statement, identify which products user wants.

5.3: Display the results in TextArea.

*Algorithm Description*

The program displays the GUI, where user can input the reservation information.

Once user inputs the data. Each input is validated using the methods. Ince the user inputs the data. Program finds the day of the week. From the day of the week, user can select start and ending time of the event. From the number of attendees that users had inputted, user can select the suitable size room.

From the input of size of the room, number of attendees, products, and time of the event, program calculates the total cost of the event. Then program prints out the information in TextArea.

**4. Translation**

**4.1** **Source Code**

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import java.awt.Font;

import javax.swing.JTextField;

import javax.swing.JComboBox;

import javax.swing.DefaultComboBoxModel;

import javax.swing.JCheckBox;

import javax.swing.JPanel;

import javax.swing.border.TitledBorder;

import javax.swing.border.EtchedBorder;

import java.awt.Color;

import javax.swing.JTextArea;

import javax.swing.JButton;

import java.awt.event.ActionListener;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.awt.event.ActionEvent;

import javax.swing.JScrollPane;

import java.util.Date;

//Java GUI Panel

public class Frame1 {

private JFrame frame;

private JTextField textFirstName; //Textbox for First Name

private JTextField textNumberOfAttendees; //Textbox for number of attendees

private JTextField texLastName;//Textbox for Last Name

private JTextField textReservationDate;// Textbox for reservation date

private JComboBox comboBoxStartTime; // Combobox for start time

private JComboBox comboBoxEndTime1; // Combobox for end time

private JPanel panel;

private JCheckBox chckbxSmallRoom,chckbxMediumRoom,chckbxLargeRoom, chckbxCoffee,chckbxSoda, chckbxCookies,chckbxSandwiches,chckbxPaperProduct; //Check box for size of rooms and products

private JTextArea textArea; // Textarea to show results.

private JScrollPane scrollPane; //Scrollpanel for textArea

private JLabel lblNote;

/\*\*

\* Launch the application.

\*/

//Main class

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

Frame1 window = new Frame1();

window.frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the application.

\*/

public Frame1() {

initialize();

}

/\*\*

\* Initialize the contents of the frame.

\*/

private void initialize()

{

frame = new JFrame();

frame.setBounds(100, 100, 788, 845);

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

frame.getContentPane().setLayout(null);

// Calculate button

JButton btnCalculate = new JButton("Calculate");

btnCalculate.addActionListener(new CalculateButtonListner() {

});

btnCalculate.setBounds(39, 524, 114, 45);

frame.getContentPane().add(btnCalculate);

// First Name Label

JLabel lblName = new JLabel("First Name:");

lblName.setFont(new Font("Tahoma", Font.PLAIN, 15));

lblName.setBounds(77, 91, 76, 19);

frame.getContentPane().add(lblName);

//Lasts Name Label

JLabel lblNumberOf = new JLabel("Number of attendees:");

lblNumberOf.setFont(new Font("Tahoma", Font.PLAIN, 15));

lblNumberOf.setBounds(77, 168, 145, 19);

frame.getContentPane().add(lblNumberOf);

// Panel

JPanel panel = new JPanel();

panel.setBorder(new TitledBorder(new EtchedBorder(EtchedBorder.LOWERED, new Color(255, 255, 255), new Color(160, 160, 160)), "", TitledBorder.LEADING, TitledBorder.TOP, null, new Color(0, 0, 0)));

panel.setBounds(393, 131, 231, 45);

frame.getContentPane().add(panel);

panel.setLayout(null);

// Start time lable.

JLabel lblStartTime = new JLabel("Start Time: ");

lblStartTime.setBounds(6, 16, 79, 19);

panel.add(lblStartTime);

lblStartTime.setFont(new Font("Tahoma", Font.PLAIN, 15));

// Start time Combobox

comboBoxStartTime = new JComboBox();

comboBoxStartTime.setBounds(91, 16, 70, 19);

comboBoxStartTime.setModel(new DefaultComboBoxModel(new String[] {"7 AM", "8 AM", "9 AM", "10 AM", "11 AM", "12 PM", "1 PM", "2 PM", "3 PM", "4 PM", "5 PM", "6 PM", "7 PM", "8 PM", "9 PM", "10 PM", "11 PM"}));

panel.add(comboBoxStartTime);

JPanel panel\_1 = new JPanel();

panel\_1.setBorder(new TitledBorder(new EtchedBorder(EtchedBorder.LOWERED, new Color(255, 255, 255), new Color(160, 160, 160)), "", TitledBorder.LEADING, TitledBorder.TOP, null, new Color(0, 0, 0)));

panel\_1.setBounds(393, 187, 231, 45);

frame.getContentPane().add(panel\_1);

panel\_1.setLayout(null);

// End time Label

JLabel lblEndTime = new JLabel("End Time:");

lblEndTime.setBounds(6, 16, 79, 19);

panel\_1.add(lblEndTime);

lblEndTime.setFont(new Font("Tahoma", Font.PLAIN, 15));

// End time Combobox

comboBoxEndTime1 = new JComboBox();

comboBoxEndTime1.setModel(new DefaultComboBoxModel(new String[] {"7 AM", "8 AM", "9 AM", "10 AM", "11 AM", "12 PM", "1 PM", "2 PM", "3 PM", "4 PM", "5 PM", "6 PM", "7 PM", "8 PM", "9 PM", "10 PM", "11 PM"}));

comboBoxEndTime1.setBounds(87, 11, 79, 22);

panel\_1.add(comboBoxEndTime1);

// First name text box.

textFirstName = new JTextField();

textFirstName.setBounds(163, 92, 135, 20);

frame.getContentPane().add(textFirstName);

textFirstName.setColumns(25);

// Number of Attendees Textbox

textNumberOfAttendees = new JTextField();

textNumberOfAttendees.setColumns(4);

textNumberOfAttendees.setBounds(229, 169, 38, 20);

frame.getContentPane().add(textNumberOfAttendees);

JPanel panel\_2 = new JPanel();

panel\_2.setBorder(new TitledBorder(new EtchedBorder(EtchedBorder.LOWERED, new Color(255, 255, 255), new Color(160, 160, 160)), "Conference room pricing :", TitledBorder.LEADING, TitledBorder.TOP, null, new Color(0, 0, 0)));

panel\_2.setBounds(25, 243, 373, 97);

frame.getContentPane().add(panel\_2);

panel\_2.setLayout(null);

//checkbox for small room

chckbxSmallRoom = new JCheckBox("Small conference room - $50/hour and holds 25 people");

chckbxSmallRoom.setBounds(6, 16, 346, 23);

panel\_2.add(chckbxSmallRoom);

//checkbox for medium room

chckbxMediumRoom = new JCheckBox("Medium conference room - $100/hour and holds 50 people");

chckbxMediumRoom.setBounds(6, 42, 361, 23);

panel\_2.add(chckbxMediumRoom);

//checkbox for small room

chckbxLargeRoom = new JCheckBox("Large conference room - $150/hour and holds 75 people");

chckbxLargeRoom.setBounds(6, 68, 346, 23);

panel\_2.add(chckbxLargeRoom);

JPanel panel\_3 = new JPanel();

panel\_3.setBorder(new TitledBorder(new EtchedBorder(EtchedBorder.LOWERED, new Color(255, 255, 255), new Color(160, 160, 160)), "Beverages/Food/Paper Products/Utensils options pricing :", TitledBorder.LEADING, TitledBorder.TOP, null, new Color(0, 0, 0)));

panel\_3.setBounds(25, 351, 373, 147);

frame.getContentPane().add(panel\_3);

panel\_3.setLayout(null);

//checkbox for coffee

chckbxCoffee = new JCheckBox("Coffee/Tea - $1/person");

chckbxCoffee.setBounds(6, 16, 139, 23);

panel\_3.add(chckbxCoffee);

//checkbox for soda

chckbxSoda = new JCheckBox("Soda - $1.50/person");

chckbxSoda.setBounds(6, 42, 139, 23);

panel\_3.add(chckbxSoda);

//checkbox for cookies

chckbxCookies = new JCheckBox("Cookies/Cakes - $2/person");

chckbxCookies.setBounds(6, 68, 155, 23);

panel\_3.add(chckbxCookies);

//checkbox for Sandwiches

chckbxSandwiches = new JCheckBox("Sandwiches - $4/person");

chckbxSandwiches.setBounds(6, 94, 141, 23);

panel\_3.add(chckbxSandwiches);

//checkbox for paper products

chckbxPaperProduct = new JCheckBox("Paper products/utensils -$2.50/person");

chckbxPaperProduct.setBounds(6, 118, 211, 23);

panel\_3.add(chckbxPaperProduct);

//Title label

JLabel lblNewLabel = new JLabel("Conference Room Registration");

lblNewLabel.setFont(new Font("Agency FB", Font.PLAIN, 36));

lblNewLabel.setBounds(166, 26, 326, 44);

frame.getContentPane().add(lblNewLabel);

// last Name label

JLabel lblLastName = new JLabel("Last Name:");

lblLastName.setFont(new Font("Tahoma", Font.PLAIN, 15));

lblLastName.setBounds(77, 131, 76, 19);

frame.getContentPane().add(lblLastName);

// Last Name textBox

texLastName = new JTextField();

texLastName.setColumns(25);

texLastName.setBounds(163, 132, 135, 20);

frame.getContentPane().add(texLastName);

//Reservation date label

JLabel lblNewLabel\_1 = new JLabel("Date of Reservation:");

lblNewLabel\_1.setFont(new Font("Tahoma", Font.PLAIN, 15));

lblNewLabel\_1.setBounds(393, 91, 135, 19);

frame.getContentPane().add(lblNewLabel\_1);

//Reservation date textbox

textReservationDate = new JTextField();

textReservationDate.setBounds(538, 92, 86, 20);

frame.getContentPane().add(textReservationDate);

textReservationDate.setColumns(10);

// Scroll Panel for TextArea

scrollPane = new JScrollPane();

scrollPane.setBounds(425, 253, 313, 395);

frame.getContentPane().add(scrollPane);

// TextArea

textArea = new JTextArea();

scrollPane.setViewportView(textArea);

textArea.setFont(new Font("Courier New", Font.PLAIN, 13));

//Clear Button and its Action Listener

JButton btnClear = new JButton("Clear");

btnClear.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

textFirstName.setText("");

texLastName.setText("");

textNumberOfAttendees.setText("");

textReservationDate.setText("");

if(chckbxSmallRoom.isSelected())

{chckbxSmallRoom.doClick();}

if(chckbxMediumRoom.isSelected())

{chckbxMediumRoom.doClick();}

if(chckbxLargeRoom.isSelected())

{chckbxLargeRoom.doClick();}

if(chckbxCoffee.isSelected())

{chckbxCoffee.doClick();}

if(chckbxSoda.isSelected())

{chckbxSoda.doClick();}

if(chckbxCookies.isSelected())

{chckbxCookies.doClick();}

if(chckbxSandwiches.isSelected())

{chckbxSandwiches.doClick();}

if(chckbxPaperProduct.isSelected())

{chckbxPaperProduct.doClick();}

textArea.setText("");

comboBoxStartTime.setSelectedIndex(0);

comboBoxEndTime1.setSelectedIndex(0);

}

});

btnClear.setBounds(163, 524, 104, 44);

frame.getContentPane().add(btnClear);

// Exit button and its Listener

JButton btnExit = new JButton("Exit");

btnExit.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

System.exit(0);

}

});

btnExit.setBounds(277, 524, 104, 44);

frame.getContentPane().add(btnExit);

JLabel lblmmddyyyy = new JLabel("(MM/DD/YYYY)");

lblmmddyyyy.setBounds(634, 95, 104, 14);

frame.getContentPane().add(lblmmddyyyy);

JLabel lblThisFacilityHolds = new JLabel("- The max number of people this facility holds is 75.");

lblThisFacilityHolds.setFont(new Font("Tahoma", Font.PLAIN, 15));

lblThisFacilityHolds.setBounds(51, 666, 339, 19);

frame.getContentPane().add(lblThisFacilityHolds);

JLabel lblYouMay = new JLabel("- You may choose any size room, as long as it does not exceed that room max limit. ");

lblYouMay.setFont(new Font("Tahoma", Font.PLAIN, 15));

lblYouMay.setBounds(51, 698, 563, 19);

frame.getContentPane().add(lblYouMay);

lblNote = new JLabel("Note:");

lblNote.setFont(new Font("Tahoma", Font.PLAIN, 15));

lblNote.setBounds(51, 638, 36, 19);

frame.getContentPane().add(lblNote);

}

// Calculate button listner

public class CalculateButtonListner implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

String clientFirstName; //First name

String clientLastName; // Last name

String input; //User input

int tempDate = 0;

int numberOfAttendees;

int selectedEndTime =0, selectedStartTime=0;

double sum=0;

int totalHours = 0;

final int SMALL\_ROOM\_PRICE = 50, MEDIUM\_ROOM\_PRICE = 100, LARGE\_ROOM\_PRICE = 150;

final double COFFEE\_PRICE = 1, SODA\_PRICE = 1.50, COOKIES\_PRICE = 2, SANDWICHES\_PRICE = 4, PAPER\_PRODUCT\_PRICE = 2.50;

textArea.setText("");

//validating first name input

input = textFirstName.getText();

if(isOnlyAlphabet(input))

{

clientFirstName = input;

}

else

{

JOptionPane.showMessageDialog(null,"Input valid first name.");

textFirstName.setText("");

return;

}

//validating last name input

input = texLastName.getText();

if(isOnlyAlphabet(input))

{

clientLastName = input;

}

else

{

JOptionPane.showMessageDialog(null,"Input valid last name.");

texLastName.setText("");

return;

}

//validating number of attendees input

input = textNumberOfAttendees.getText();

if(isNumber(input))

{

int tempnumberOfAttendees = Integer.parseInt(input);

if(tempnumberOfAttendees > 0 && tempnumberOfAttendees < 76)

{

numberOfAttendees = tempnumberOfAttendees;

}

else

{

JOptionPane.showMessageDialog(null,"Input valid number. \nThis facility holds up to 75 peoplee \nEnter number between 0 to 75");

textNumberOfAttendees.setText("");

return;

}

}

else

{

JOptionPane.showMessageDialog(null,"Input valid number of attendees.");

textNumberOfAttendees.setText("");

return;

}

//Validating Data input

String reservationDate = textReservationDate.getText();

try

{

Date date = (Date) new SimpleDateFormat("MM/dd/yyyy").parse(reservationDate);

//finding the data of the week

Calendar calendar = Calendar.getInstance();

calendar.setTime(date);

tempDate = calendar.get(Calendar.DAY\_OF\_WEEK);

;

}

catch (ParseException e1)

{

JOptionPane.showMessageDialog(null,"Please, enter date in corret format, MM/DD/YYYY");

textReservationDate.setText("");

return;

}

//If statement to set restriction on combobox.

if( tempDate == 1 || tempDate == 7 )

{

selectedStartTime = comboBoxStartTime.getSelectedIndex();

if(selectedStartTime == 0) {

JOptionPane.showMessageDialog(null,"Conference room is not available at chosen time.\n Available time:\n Monday to Friday: 7 AM to 9 PM\n Saturday and Sunday: 8 AM to 11 PM.");

return;}

}

if( tempDate > 1 && tempDate < 7 )

{

selectedEndTime = comboBoxEndTime1.getSelectedIndex();

if(selectedEndTime == 15 || selectedEndTime == 16 ) {

JOptionPane.showMessageDialog(null,"Conference room is not available at chosen time.\n Available time:\n Monday to Friday: 7 AM to 9 PM\n Saturday and Sunday: 8 AM to 11 PM.");

return;}

}

//Combobox input verification

if(comboBoxEndTime1.getSelectedIndex() < comboBoxStartTime.getSelectedIndex())

{

JOptionPane.showMessageDialog(null,"Please, check the time you have selected. End time that you have selected is early then the start time.");

return;

}

else if (comboBoxEndTime1.getSelectedIndex() == comboBoxStartTime.getSelectedIndex())

{

JOptionPane.showMessageDialog(null,"Please, check the time you have selected. Start and end time can not be equal.");

return;

}

//Printing out reservation info.

totalHours = comboBoxEndTime1.getSelectedIndex() - comboBoxStartTime.getSelectedIndex();

textArea.append("Reservation information:");

textArea.append("\nFirst Name: "+ clientFirstName + "\nLast Name: " + clientLastName + "\nDate: " + reservationDate + "\nNumber of attendees: " +numberOfAttendees);

textArea.append("\n-------------------------------");

textArea.append("\nServices Selected");

//Verification of number of attendees.

if(numberOfAttendees > 25) {

if(chckbxSmallRoom.isSelected())

{

JOptionPane.showMessageDialog(null,"Small conference room only holds upto 25 people ");

return;

}

}

else if (chckbxSmallRoom.isSelected())

{

sum += totalHours \* SMALL\_ROOM\_PRICE;

textArea.append("\nSmall Room: 50/hour");

}

if(numberOfAttendees > 50) {

if(chckbxMediumRoom.isSelected())

{

JOptionPane.showMessageDialog(null,"Midium conference room only holds upto 50 people ");

return;

}

}

else if (chckbxMediumRoom.isSelected())

{

sum += totalHours \* MEDIUM\_ROOM\_PRICE;

textArea.append("\nMedium Room: 100/hour");

}

if(numberOfAttendees > 75) {

if(chckbxLargeRoom.isSelected())

{

JOptionPane.showMessageDialog(null,"Large conference room only holds upto 75 people ");

return;

}

}

else if (chckbxLargeRoom.isSelected())

{

sum += totalHours \* LARGE\_ROOM\_PRICE;

textArea.append("\nLarge Room: 150/hour");

}

if(!chckbxSmallRoom.isSelected() && !chckbxMediumRoom.isSelected() && !chckbxLargeRoom.isSelected() )

{

JOptionPane.showMessageDialog(null,"You must select one size of room");

return;

}

totalHours = comboBoxEndTime1.getSelectedIndex() - comboBoxStartTime.getSelectedIndex();

//Verification of products input

if(chckbxCoffee.isSelected())

{

sum += numberOfAttendees \* COFFEE\_PRICE;

textArea.append("\nCoffee/Tea: $1/person");

}

if(chckbxSoda.isSelected())

{

sum += numberOfAttendees \* SODA\_PRICE;

textArea.append("\nSoda: $1.50/person");

}

if(chckbxCookies.isSelected())

{

sum += numberOfAttendees \* COOKIES\_PRICE;

textArea.append("\nCookies/Cakes: $2/person");

}

if(chckbxSandwiches.isSelected())

{

sum += numberOfAttendees \* SANDWICHES\_PRICE;

textArea.append("\nSandwiches: $4/person");

}

if(chckbxPaperProduct.isSelected())

{

sum += numberOfAttendees \* PAPER\_PRODUCT\_PRICE;

textArea.append("\nPaper products/utensils: $2.50/person");

}

textArea.append("\n-------------------------------");

textArea.append("\nTotal cost: $"+ sum);

}

public boolean isNumber(String in)

{

try

{

Integer.parseInt(in);

return true;

}

catch (Exception E)

{

return false;

}

}

public int parseInt(String selectedItem) {

return Integer.parseInt(selectedItem);

}

private boolean isOnlyAlphabet(String al)

{

return ((!al.equals("")) && (al.matches("^[a-zA-Z]\*$"))&& (al != null));

}

}

}

**4.2 Program and Module Description**

PrintStudentRecord

This function prints out student’s information.

getStudentID

This function returns the current student object’s student ID variable.

average

calculate and return the average of grades.

letterGrade

calculate and return a letter grade.

isNumber

Validate if the input is a number or not. Returns boolean.

isOnlyAlphabets

Validate if the input is only alphabets or not. Returns boolean.

isValidEmail

Validate if the input is in proper email format. Returns boolean.

Main

The main function asks under to input data of student’s information. Then the function creates a student object. For each object, it creates it also store in an array of object. Then the program asks the user to choose what they want to display. If a user wants to display an entire roster. A loop goes through an object array and calls the print method (each time it loops). In the print method, the program calls an average() and letterGrad() method. Which calculate the current student grads and print the information. If a user wants to display particular students information by using a Student ID. Loops go through an object array and get the StudentID (Using getStudentID method). Then the program matches the Student ID with the user input. If it matches, then it prints the student information of a current object.

**5. Solution Testing**

Test Case: 1 – Data validation.

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Entry | Status of input | Reason |
| First Name | ABC123 | Invalid | Name Cannot contain numbers |
| Last Name | ABC123 | Invalid | Name Cannot contain numbers |
| Number of Attendees | -20 | Invalid | Cannot accept negative number |
| Date of Reservation | Oct 25 2019 | Invalid | Not in correct format. |

Test Case: 2

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Entry | Status of input | Reason |
| First Name | Donark | Valid |  |
| Last Name | Patel | Valid |  |
| Number of Attendees | 20 | Valid |  |
| Date of Reservation | 10/05/2019 | Valid |  |
| Start time | 7:00 AM | Invalid | Weekend start time is after 8AM |
| Start time | 8:00 AM | Valid |  |
| End time | 11:00 PM | Valid |  |

Test Case: 3

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Entry | Status of input | Reason |
| First Name | Donark | Valid |  |
| Last Name | Patel | Valid |  |
| Number of Attendees | 20 | Valid |  |
| Date of Reservation | 10/03/2019 | Valid |  |
| Start time | 7:00 AM | Valid |  |
| End time | 10:00 PM | invlid | Weekend end time is after 9AM |

Test Case: 4

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Entry | Status of input | Reason |
| First Name | Donark | Valid |  |
| Last Name | Patel | Valid |  |
| Number of Attendees | 30 | Valid |  |
| Date of Reservation | 10/03/2019 | Valid |  |
| Start time | 7:00 AM | Valid |  |
| End time | 9:00 PM | Valid |  |
| Small Conference room. | Selected | Invlid | Small room can hold up to 25 people |

Test Case: 5

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Entry | Status of input | Reason |
| First Name | Donark | Valid |  |
| Last Name | Patel | Valid |  |
| Number of Attendees | 55 | Valid |  |
| Date of Reservation | 10/03/2019 | Valid |  |
| Start time | 7:00 AM | Valid |  |
| End time | 9:00 PM | Valid |  |
| Medium Conference room. | Selected | Invlid | Medium room can hold up to 50 people |

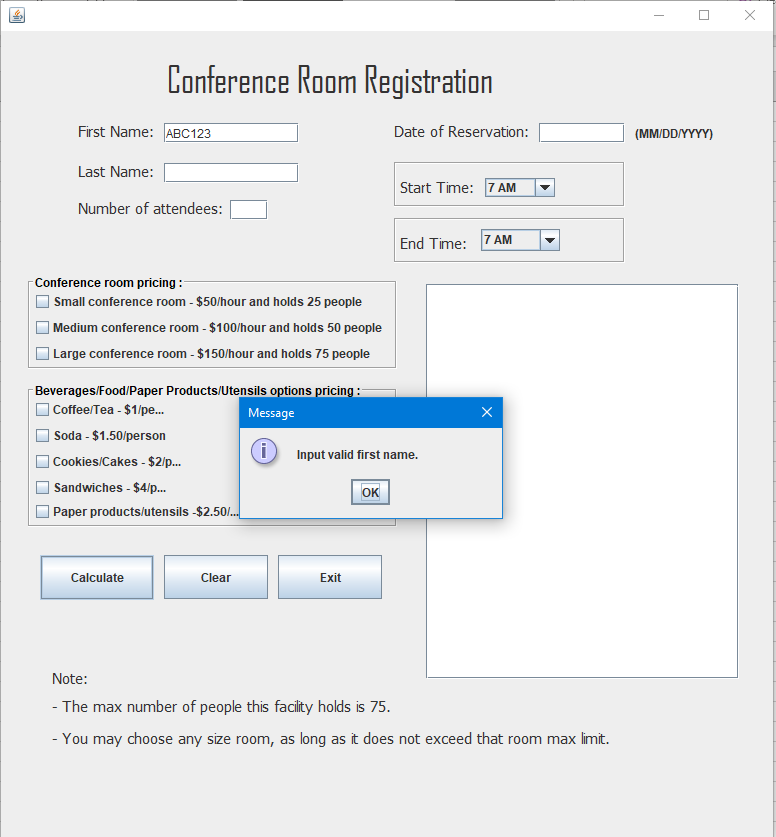
Test Case: 6

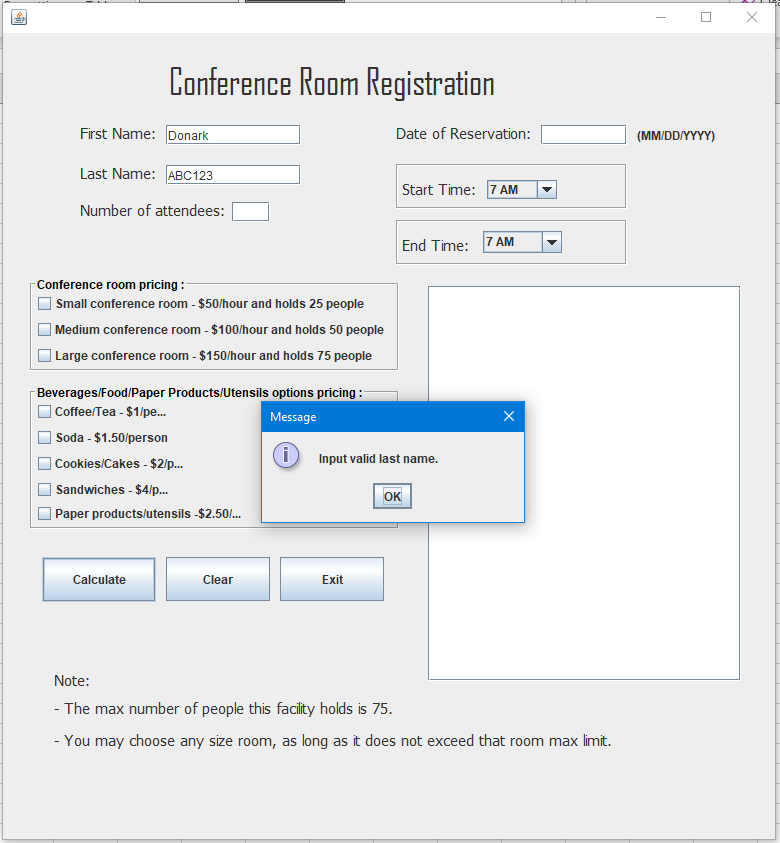
|  |  |  |  |
| --- | --- | --- | --- |
| Component | Entry | Status of input | Reason |
| First Name | Donark | Valid |  |
| Last Name | Patel | Valid |  |
| Number of Attendees | 85 | Valid |  |
| Date of Reservation | 10/03/2019 | Valid |  |
| Start time | 7:00 AM | Valid |  |
| End time | 9:00 PM | Valid |  |
| Large Conference room. | Selected | Invlid | large room can hold up to 75 people |

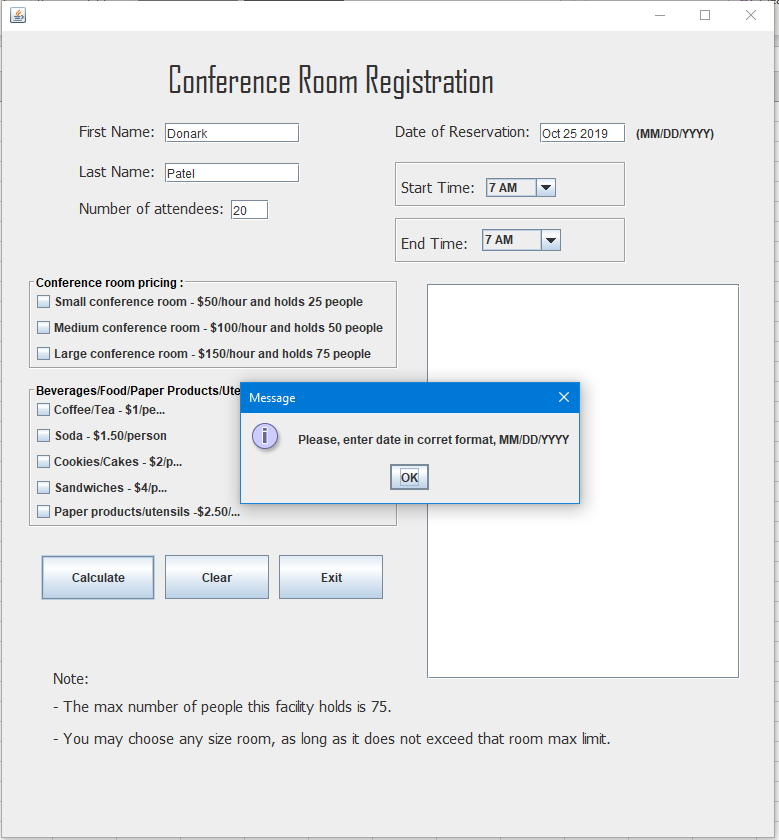
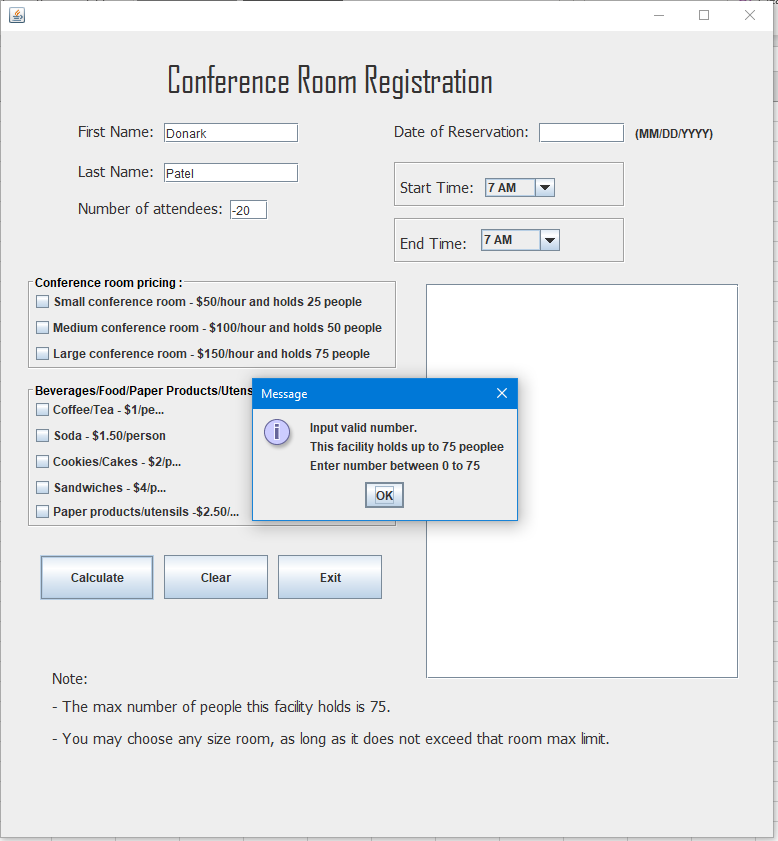
Test Case: 7

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Entry | Status of input | Reason |
| First Name | Donark | Valid |  |
| Last Name | Patel | Valid |  |
| Number of Attendees | 60 | Valid |  |
| Date of Reservation | 10/03/2019 | Valid |  |
| Start time | 7:00 AM | Valid |  |
| End time | 9:00 PM | Valid |  |
| Large Conference room. | Selected | valid |  |
| Coffee/Tea | Selected | valid |  |
| Soda | Selected | valid |  |
| Cookies/Cake | Selected | valid |  |
| Sandwiches | Selected | valid |  |
| Paper products/utensils | Selected | valid |  |

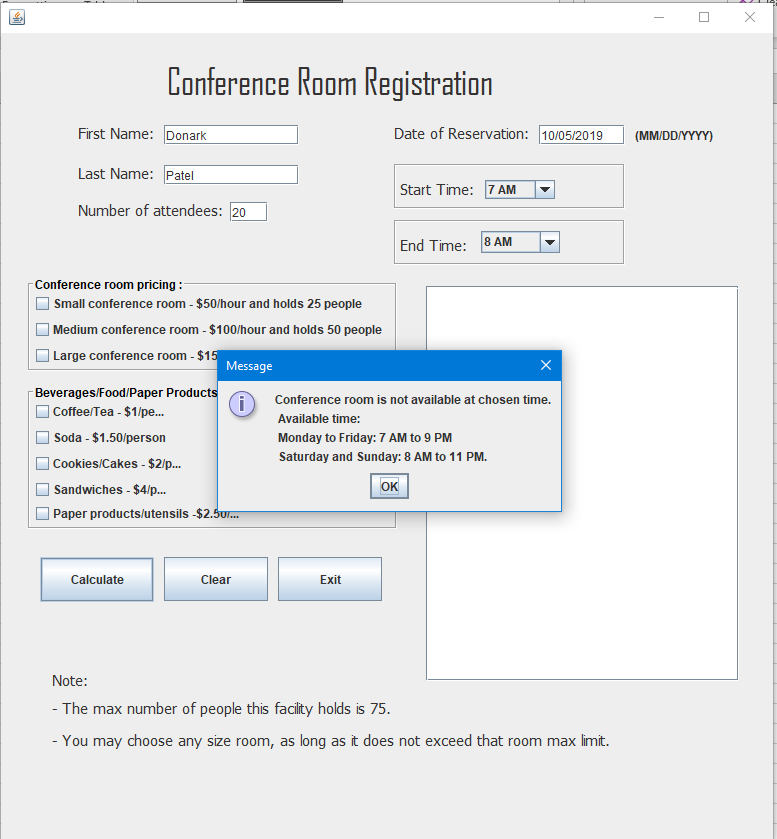
**(This is just a sample what you need to submit are actual screenshots of I/O or files)**

Test Case: 1 

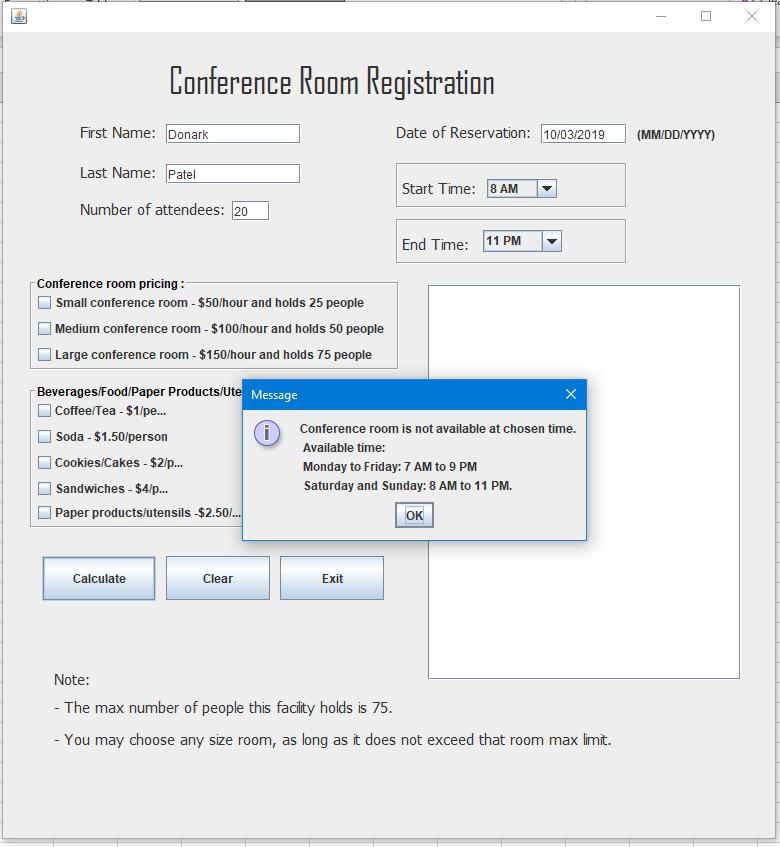




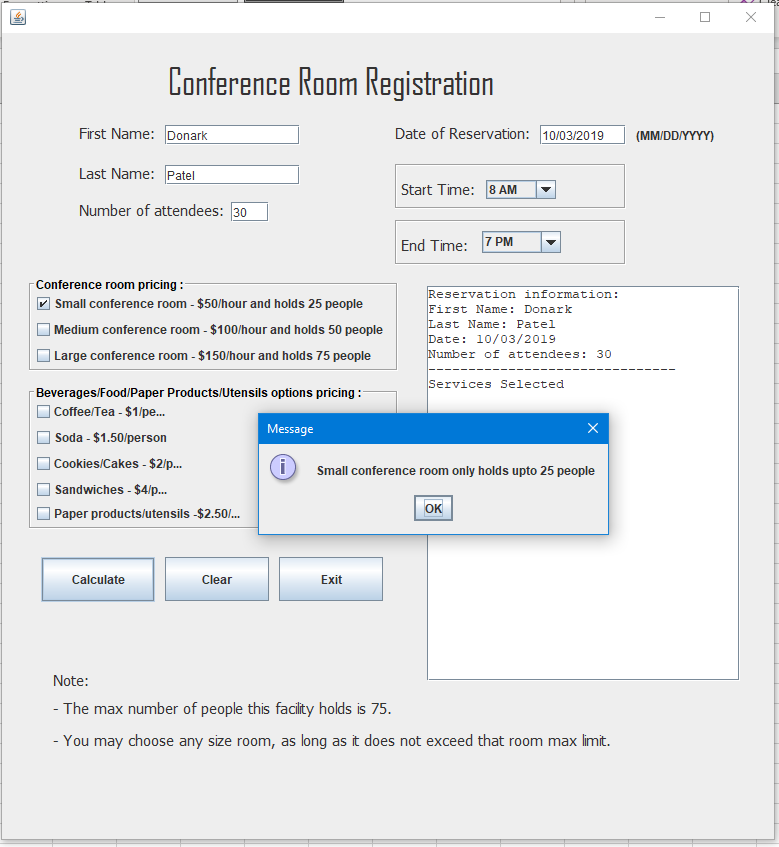
Test Case: 2

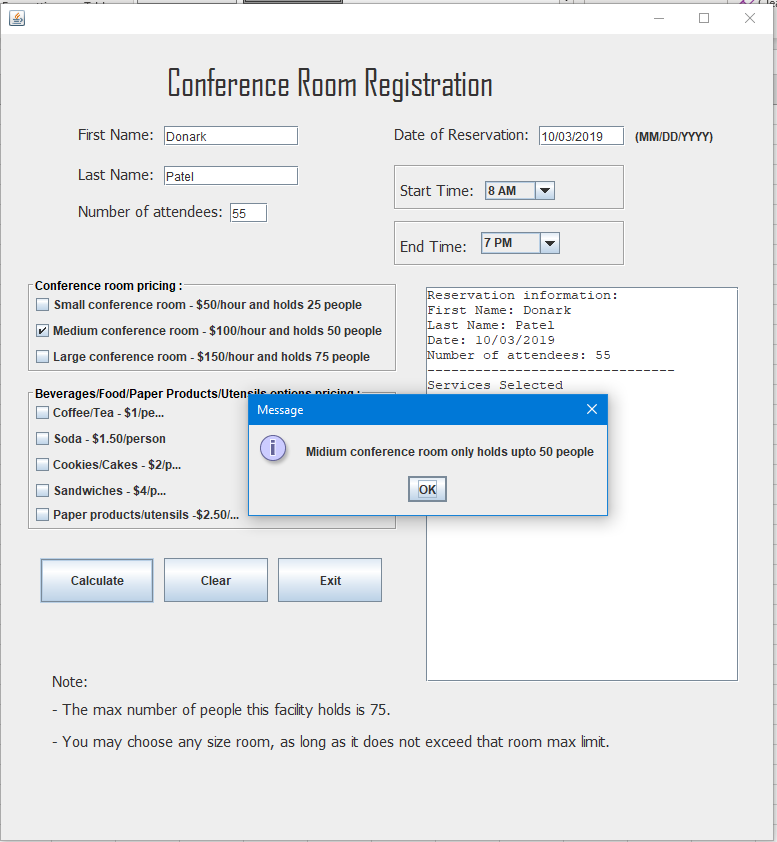


Test Case: 3

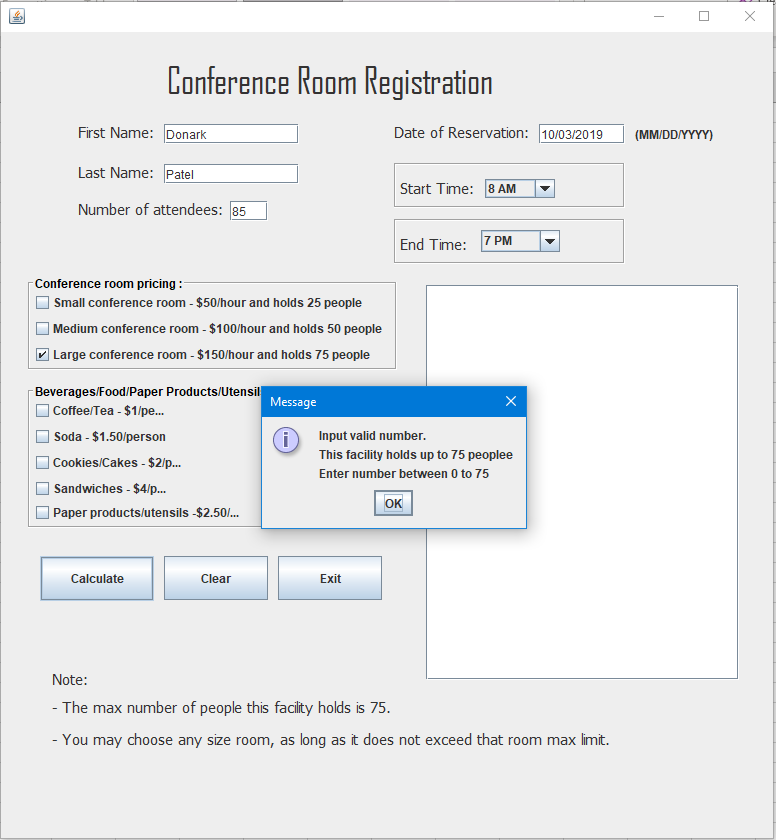


Test Case: 4



Test Case: 5 

Test Case: 6



Test Case: 8

