**Name**

Joshua Turner

**Student Number**

S0258441

**Program**

Bachelor of Information Technology

**Course Code**

COIT11222

**Assignment**

Assessment 3 – JAVA Program and Specification Project

**Task**

Develop a Java program that uses Java constructs including input/output, Java primitive and built-in types, selection and looping statements and various other Java commands. Write a specification report outlining the program scenario, specifications and marking scheme.

**Word Count**

1983

**Lecturer**

Bruce McKenzie

**Due Date**

18 February 2020

Assignment 3 – Project

# Scenario

The Den Barbershop needs a program to store customer details and calculate the cost of the service they received. The shop’s service will be broken down into three categories: Head, Face, and Extras. Each category will contain multiple options including:

**Hair Face Extras**Standard **$25** Full Shave **$20** Monobrow **$5**  
Skin Fade **$35**  Beard Trim **$15** Full Eyebrows **$10**  
Style Cut **$30**

The program will allow for “no option” to be selected in each category to allow for customers who may only require one service. Extras will allow for one or no option to be selected as they are mutually exclusive.

# Customer Haircut Class

The project requires a class to be created for the customer details services received by The Den Barbershop. This will be called **CustomerHaircut.java**.

The CustomerHaircut class require five instance variables declared as private for the customer’s details:

* customerName – String
* customerHairOption – String
* customerFaceOption – String
* customerMonobrowOption – boolean
* customerFullEyebrowsOption – boolean

It will also require seven private static double constants for the prices of services:

* STANDARD\_CUT\_COST
* SKIN\_FADE\_COST
* STYLE\_CUT\_COST
* FULL\_SHAVE\_COST
* BEARD\_TRIM\_COST
* MONOBROW\_COST
* FULL\_EYEBROWS\_COST

As these are static name constants, they will not require mutator methods.

The class will contain the following public methods:

* A default constructor (no parameters)
* A parameterised constructor (five parameters – three strings, two booleans)
* Five set methods (mutators) (only required for the instance variables)
* 12 get methods (accessors) (required for and instance variables and named constants)
* A calculateServiceCharge method\*

\*A calculation method will be implemented within the CustomerHaircut.java class using the method header:

public double calculateServiceCharge()

It will calculate the cost by basic addition using if or switch statements. All prices will be included in the CustomerHaircut.java class.

# Project Class

The GUI program will be written in the Project.java class and will contain the main method in which the program will run. The Project class extends the JFrame class and implements the ActionListener class.

There code should not be repeated throughout the program. Where code is required multitple times, a method should be created to eliminate repeat code.

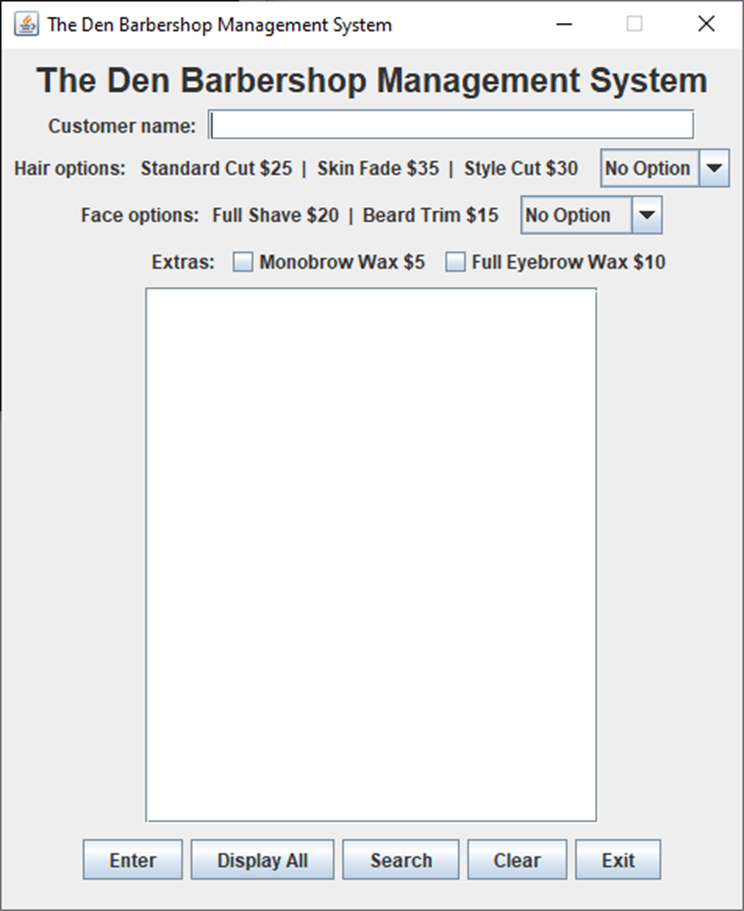


Figure 3.2.1 Project.java default screen.

* 1. GUI Attributes and Constructor

### Attributes

All items on the GUI frame will first be declared as class attributes of the Project class. The following components will be used:

**JLabel** will be used for all labels seen on the frame, including: the title, customer name, hair options etc. The hair and face labels will use the **String.format()** method, as follows:

String.format("Hair options: Standard Cut $%.0f | Skin Fade $%.0f | Style Cut $%.0f ",

**STANDARD\_PRICE**, **SKIN\_FADE\_PRICE**, **STYLE\_CUT\_PRICE**)

String.format("Face options: Full Shave $%.0f | Beard Trim $%.0f ",

**FULL\_SHAVE\_PRICE**, **BEARD\_TRIM\_PRICE**)

Accessor methods will be used to retrieve the above prices.

**JTextField** will be used for the customer name input field.

**JTextArea** and **JScrollPane** will be used for the output text area.

Drop down boxes will be created using **JComboBox<String>.** These will be used for the hair and face options.

Two **JCheckBox** will be placed inside a **ButtonGroup** for the Extras options. A third checked will also be declared; it will be named **invisibleBox**.

Five **JButtons** will be declared, as well as a **JPanel** for formatting purposes.

### Constructor

The constructor for this class does not require parameters. The objects declared as attributes will be added to using the **add()** method.

First, the layout will be set to flow layout using **this.setLayout(new FlowLayout());**

Second, the title label, name label and name field will be added to the frame.

Third, the hair and face labels and dropdown boxes will be added to the frame. The options will be added to each dropdown box using the **“objectName”.add()** method.

Forth, all three check boxes will be added to the button group using **“objectName”.add().** Then, extras label and check boxes will be added to the frame. The invisible checkbox will be set to invisible using **invisibleBox.setVisible(false).**

Fifth, the output text area font style will be set with **displayTextArea.setFont(new Font("Monospaced", Font.PLAIN, 12)).** The **displayTextArea.setEditable(false)** method will be used to disable editing. The scroll bar declared as a class attributed will now be constructed, **displayTextArea** will be passed to it as an argument. As horizontal scrolling will not be required, it will need to be disabled using **scrollPane.setHorizontalScrollBarPolicy()** method and passing the **JScrollPane.HORIZONTAL\_SCROLLBAR\_NEVER** argument.

Then, the JPanel will be added to the frame and all buttons will be added to the JPanel. This is to ensure all buttons appear together on the same row.

Finally, action listeners will be added to all buttons using **“button name”.addActionListener(this).** The close button in the top right-hand corner will be set to run the exit() method using the following code:

addWindowListener(

new WindowAdapter()

{

public void windowClosing(WindowEvent e)

{

exit();

}

}

);

An instance of this class now be created in the main method.

The class is instantiated using **Project p = new Project();**

To allow the exit() method to run the **setDefaultCloseOperation()** method used so that nothing is done when the close button is pressed using the **JFrame.DO\_NOTHING\_ON\_CLOSE** argument.

The size and position of the frame will set with **setBounds(1000, 500, 490, 590).**

The frame’s title will be set using the **setTitle()** method. The frame will be set to visible using the **setVisible(true)** method. To maintain the frame layout the **setResizable(false)** method will be used so the user cannot change the size of the frame.

## Data Structures

The program will use an array of CustomerHaircut objects to store each customer’s details. The array will be declared as an instance variable of the Project class; it will contain **N\*** elements. N will be declared as a constant for the maximum entries allowed.

\*N = largest number in the programmer’s student ID

private CustomerHaircut[] haircutArray = new CustomerHaircut[MAX\_HAIRCUTS];

An instance variable will also be required for tracking how many customers have been entered. This will serve as the index for the array as it is populated with customer details.

private int currentHaircut = 0;

## Enter Button

When the enter button is pressed it will store the customer name and the options they have selected into the haircutArray. It will also output the customer’s name, the services that have been received, the cost of each service and the total service cost.

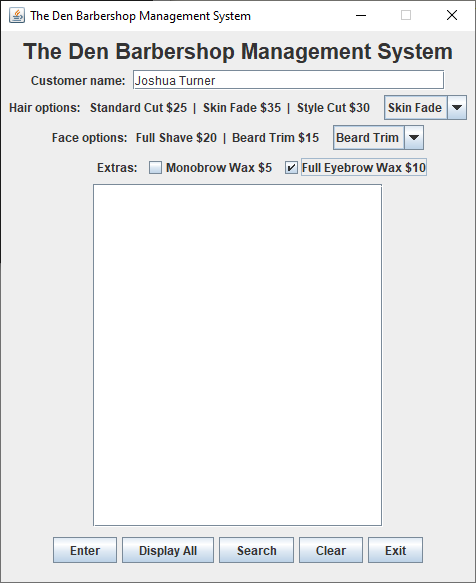


Figure 3.3.1 Default screen with customer details entered.

The following statements will be required to retrieve the user’s input:

String customerName = nameField.getText();

String customerHair = hairChoice.getSelectedItem().toString();

String customerFace = faceChoice.getSelectedItem().toString();

boolean customerMonobrow = monobrowBox.isSelected();

boolean customerEyebrows = fullEyebrowsBox.isSelected();

Those input will be temporarily saved to the declared variables. Those variables will then be used in the parametrised constructor:

haircutArray[currentHaircut] = new CustomerHaircut(customerName, customerHair,

customerFace, customerMonobrow, customerEyebrows);

Next, the program will the print out the customer details and total price. A series of decision-making statements will be used to determine what service options are to be printed in the output text area. These can be if/else statements or switch statements. Accessor methods will be used to retrieve details from the array, for example:

haircutArray[index].getCustomerName()

As service prices are not instance variables, but are class constants, the class name is used instead of an object name:

CustomerHaircut.getStandardCutPrice()

The format String method will be used to format the output correctly for the text area.

For the customer name and header use:

String.format("%s%s\n%-32s%s\n\n", "Customer name: ", **customerName**, "Services", "Price")

For each of the options selected by the customer use:

String.format("%-32s$%.2f\n", "Monobrow wax", **SERVICE\_PRICE**)

For the total use:

String.format("\n%-32s$%.2f\n", "Total", **serviceCharge**)

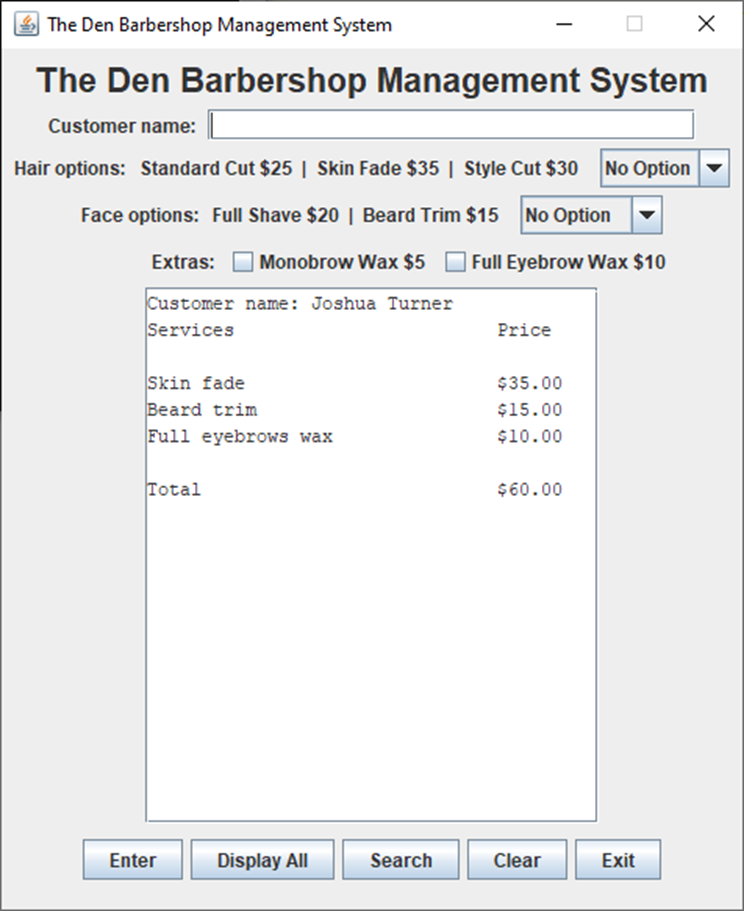


Figure 3.3.2 customer details output

At the end of the method the currentHaircut variable should be incremented and focus should be returned to the customer name JTextField using the following method:

nameField.requestFocus();

### Data Validation

The program should validate that appropriate data has been entered and prompt the user when data has been entered incorrectly.

When the maximum number of customers has been entered, the program should no longer attempt to create new array elements.

If statements should be used to determine error conditions. At the end of each error if statement block, the return statement should be used to stop the method from executing any further.

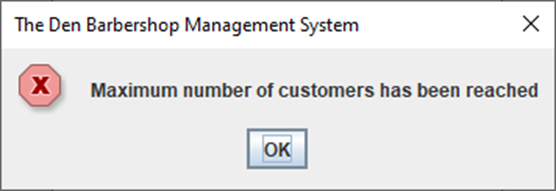


Figure 3.3.3 maximum customer has been reached error message.

The program should always require the user to enter a customer name. This is determined by using:

if(nameField.getText().compareTo("") == 0)

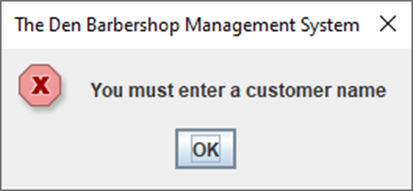


Figure 3.3.4 no customer name entered error.

**There is no need to verify character types.**

The program must also verify the user has selected services for that customer. To do this, it must check the hair option, face option, and extras options.

The following code will be used:

if(hairChoice.getSelectedItem().toString().compareTo("No Option") == 0 &&

faceChoice.getSelectedItem().toString().compareTo("No Option") == 0

&& invisibleBox.isSelected())

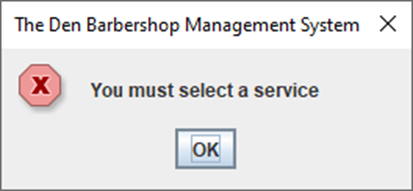


Figure 3.3.5 no service options selected error

## Display All Button

When the display all button is pressed the program will output all customer details that have been entered in the output text area; this will include their service details and total price. The program will also track the average cost of each service and the total sales amount, these will both outputted below all the entered orders.

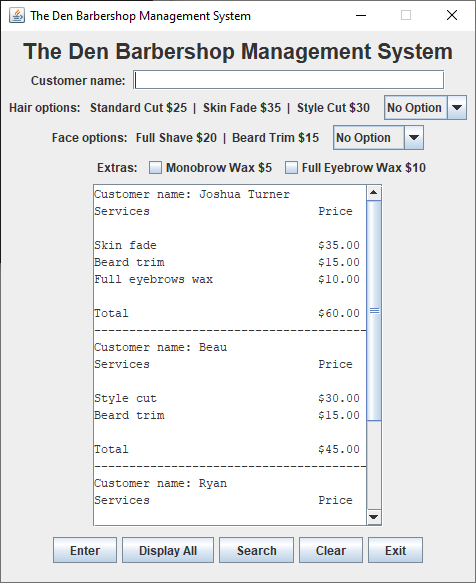


Figure 3.4.1 display all output - customer details

A loop structure should be used to the achieve this output. The currentHaircut variable should be used to terminate the loop; this is to prevent the whole array from being printed.

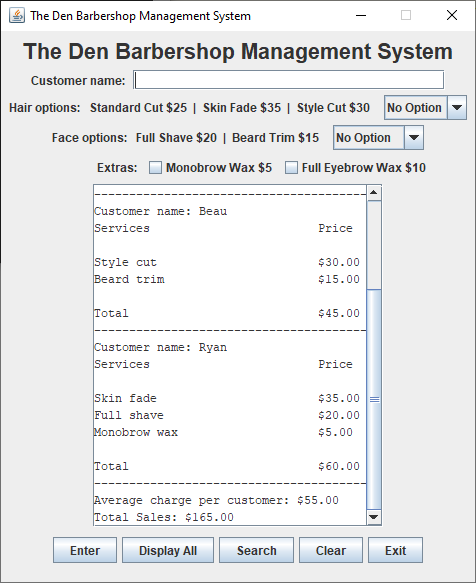


Figure 3.4.2 display all output - average charge and total sales.

Variable should be declared to track the totals sales; this will be added to with each iteration of the loop. The average of charge per customer can be calculated using the totalSales variable and the currentHaircut variable.

If no customer details have been entered, an error message should be displayed.

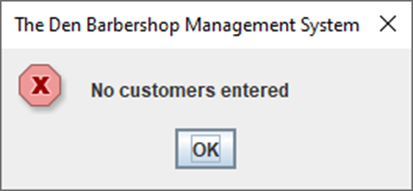


Figure 3.4.3 display all, no customer entered error.

## Search Button

The search button will allow the user to search for entered customers. A simple linear search can used for this; it must also be case insensitive.

The program will use the JOptionPane.showInputDialog() method to allow the user to input the customer name they want to search.

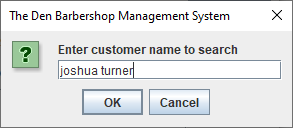


Figure 3.5.1 customer name search input dialog.

If the search is successful, the program with output the customer’s details in the output text area.

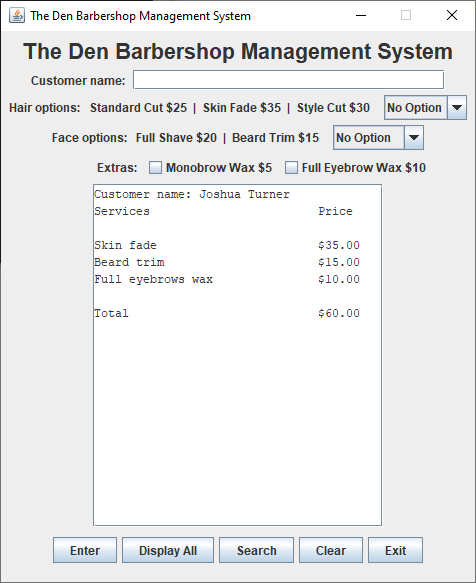


Figure 3.5.2 Successful search result output.

If the search is unsuccessful an error message will be displayed using a JOptionPane.showMessageDialog() method.

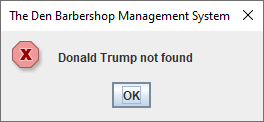


Figure 3.5.3 Unsuccessful search error message.

If the user attempts to search for a customer before any customer have been entered, an error message will be displayed. This will use the same logic as the error message in the displayAll() method.

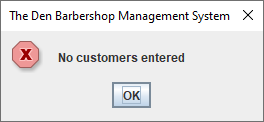


Figure 3.5.4 Customer search, no customers entered error message.

If the user attempts to search without entering customer name, an error message will be displayed. This will use similar logic to the error message in the enter() method.

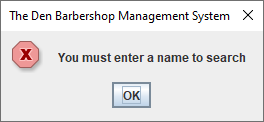


Figure 3.5.5 Customer search, no name entered error message.

## Clear Button

As the program requires several different inputs, a clear button will be implemented to allow the user to quickly clear all inputs.

**nameField.setText("");** will be used to clear the customer name text field.

**hairChoice.setSelectedIndex(0);** will reset the customer hair service choice to “No Option”.

**faceChoice.setSelectedIndex(0);** will reset the customer face service choice to “No Option”.

As the monobrow wax and full eyebrow wax JCheckBoxes have been declared in a ButtonGroup; once one of the check boxes are selected, they cannot be unselected. To circumvent this, a check box with it’s visibility set to false has also been declared in the in the ButtonGroup. To clear the extras option check boxes the **invisibleBox.setSelected(true);** method will be used.

The output text area will also be cleared using the **displayTextArea.setText("");** method. This allows the clear() method to be used in other areas of the program.

At the end of the clear() method, focus will be returned to the customer name text field.

## Exit Button

When the exit button, is pressed a thank you message will be displayed. The JOptionPane.showMessageDialog() method will be used to display this message.

The program will then be ended using **System.exit(0);**.

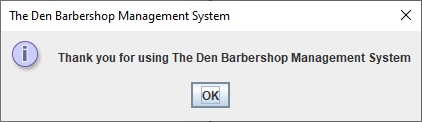


Figure 3.7.1 Exit thank you message.

# Marking Scheme



Figure 3.7.1 Marking Scheme created in MS Excel.

Appendix A – CustomerHaircut Class

// Programmer: Joshua Turner s0258441

// Course: Programming Fundamentals COIT 11222 T320

// File: CustomerHaircut.java

// Purpose: Assignment Three -- Project: The Den Barber Shop GUI program

// Date: 31 January 2021

// Class for services received from The Den Barbershop

public class CustomerHaircut

{

// Class attributes

private String customerName;

private String customerHairOption;

private String customerFaceOption;

private boolean customerMonobrowOption;

private boolean customerFullEyebrowsOption;

// constant attributes

private static double STANDARD\_CUT\_COST = 25.00; // cost for standard haircut

private static double SKIN\_FADE\_COST = 35.00; // cost for skin fade haircut

private static double STYLE\_CUT\_COST = 30.00; // cost for style cut haircut

private static double FULL\_SHAVE\_COST = 20.00; // cost for full face shave

private static double BEARD\_TRIM\_COST = 15.00; // cost for beard trim

private static double MONOBROW\_COST = 5.00; // cost for monobrow waxing

private static double FULL\_EYEBROWS\_COST = 10.00; // cost for full eyebrows waxing

// Full parametised constructor

public CustomerHaircut(String name, String hair, String face,

boolean monobrow, boolean eyebrows)

{

customerName = name;

customerHairOption = hair;

customerFaceOption = face;

customerMonobrowOption = monobrow;

customerFullEyebrowsOption = eyebrows;

}

// Empty constructor

public CustomerHaircut()

{

customerName = "null";

customerHairOption = "No Option";

customerFaceOption = "No Option";

customerMonobrowOption = false;

customerFullEyebrowsOption = false;

}

// Mutator methods

public void setCustomerName(String name)

{

customerName = name;

}

public void setCustomerHairOption(String hair)

{

customerHairOption = hair;

}

public void setCustomerFaceOption(String face)

{

customerFaceOption = face;

}

public void setCustomerMonobrowOption(boolean monobrow)

{

customerMonobrowOption = monobrow;

}

public void setcustomerFullEyebrowsOption(boolean eyebrows)

{

customerFullEyebrowsOption = eyebrows;

}

// Accessor methods

public String getCustomerName()

{

return customerName;

}

public String getCustomerHairOption()

{

return customerHairOption;

}

public String getCustomerFaceOption()

{

return customerFaceOption;

}

public boolean getCustomerMonobrowOption()

{

return customerMonobrowOption;

}

public boolean getCustomerFullEyebrowsOption()

{

return customerFullEyebrowsOption;

}

// price accessor methods

public static double getStandardCutPrice()

{

return STANDARD\_CUT\_COST;

}

public static double getStyleCutPrice()

{

return STYLE\_CUT\_COST;

}

public static double getSkinFadePrice()

{

return SKIN\_FADE\_COST;

}

public static double getFullShavePrice()

{

return FULL\_SHAVE\_COST;

}

public static double getBeardTrimPrice()

{

return BEARD\_TRIM\_COST;

}

public static double getMonobrowPrice()

{

return MONOBROW\_COST;

}

public static double getFullEyebrowsPrice()

{

return FULL\_EYEBROWS\_COST;

}

// Cost calculation method

public double calculateServiceCharge()

{

double serviceCharge = 0.0;

switch(customerHairOption) // add the cost of the customer hair option to the service charge

{

case "No Option":

break;

case "Standard":

serviceCharge += STANDARD\_CUT\_COST;

break;

case "Skin Fade":

serviceCharge += SKIN\_FADE\_COST;

break;

case "Style Cut":

serviceCharge += STYLE\_CUT\_COST;

break;

}

switch(customerFaceOption) // add the cost of the customer face option to the service charge

{

case "No Option":

break;

case "Full Shave":

serviceCharge += FULL\_SHAVE\_COST;

break;

case "Beard Trim":

serviceCharge += BEARD\_TRIM\_COST;

break;

}

if(customerMonobrowOption) // add cost of monobrow waxing if customer has chosen this option

{

serviceCharge += MONOBROW\_COST;

}

if(customerFullEyebrowsOption) // add cost of full eyebrows waxing if customer has chosen this option

{

serviceCharge += FULL\_EYEBROWS\_COST;

}

return serviceCharge;

}

}

Appendix B – Project Class

// Programmer: Joshua Turner s0258441

// Course: Programming Fundamentals COIT 11222 T320

// File: Project.java

// Purpose: Assignment Three -- Project: The Den Barber Shop GUI program

// Date: 6 February 2021

// GUI constructor and main method for program

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class Project extends JFrame implements ActionListener

{

final int MAX\_HAIRCUTS = 8; // Maximum number of haircuts that can be provided (N = highest number in student ID)

private int currentHaircut = 0; // varible used to track the current haircut

private CustomerHaircut[] haircutArray = new CustomerHaircut[MAX\_HAIRCUTS]; // object array declaration and initialisation

private JLabel titleLabel = new JLabel("The Den Barbershop Management System"); // program title

private JLabel nameLabel = new JLabel("Customer name: ");// label for customer name field

private JTextField nameField = new JTextField(28); // field for customer name input

private JTextArea displayTextArea = new JTextArea("", 20, 41); // declare text area

private JScrollPane scrollPane; // scroll pane for the text area

// declare JComboBoxes

private JLabel hairLabel = new JLabel(String.format("Hair options: Standard Cut $%.0f | Skin Fade $%.0f | Style Cut $%.0f ",

CustomerHaircut.getStandardCutPrice(), CustomerHaircut.getSkinFadePrice(),

CustomerHaircut.getStyleCutPrice())); // label for drop down box with prices

private JComboBox<String> hairChoice = new JComboBox<String>(); // Drop down box for selecting hair options

private JLabel faceLabel = new JLabel(String.format("Face options: Full Shave $%.0f | Beard Trim $%.0f ",

CustomerHaircut.getFullShavePrice(), CustomerHaircut.getBeardTrimPrice()));// label for drop down box with prices

private JComboBox<String> faceChoice = new JComboBox<String>(); // Drop down box for selecting face options

// declare button group

private JLabel extrasLabel = new JLabel(String.format("%25s", "Extras: ")); // Extras label for check box options

private ButtonGroup extrasChoice = new ButtonGroup(); // create button group extras option check boxes

private JCheckBox monobrowBox = new JCheckBox(String.format("Monobrow Wax $%.0f",

CustomerHaircut.getMonobrowPrice())); // monobow option check box

private JCheckBox fullEyebrowsBox = new JCheckBox(String.format("Full Eyebrow Wax $%.0f",

CustomerHaircut.getFullEyebrowsPrice())); // full eyebrow option check box

private JCheckBox invisibleBox = new JCheckBox(""); // for no selection is made

// declare all of the buttons

private JPanel buttonPanel = new JPanel(); // used to format buttons together

private JButton enterButton = new JButton("Enter");

private JButton displayButton = new JButton("Display All");

private JButton searchButton = new JButton("Search");

private JButton clearButton = new JButton("Clear");

private JButton exitButton = new JButton("Exit");

public Project() // Constructor for Project GUI

{

this.setLayout(new FlowLayout()); // set JFrame to FlowLayout

titleLabel.setFont(new Font("Ariel", Font.BOLD, 22)); // Set title format

add(titleLabel); // add title label to frame

add(nameLabel); // add name label to frame

add(nameField); // add name field to frame

add(hairLabel); // add hair option label to frame

add(hairChoice); // add hair choice combo box to frame

hairChoice.addItem("No Option"); // add option to hair choice combo box

hairChoice.addItem("Standard"); // add option to hair choice combo box

hairChoice.addItem("Skin Fade"); // add option to hair choice combo box

hairChoice.addItem("Style Cut"); // add option to hair choice combo box

add(faceLabel); // add face option label to frame

add(faceChoice); // add face choice combo box to frame

faceChoice.addItem("No Option"); // add option to face choice combo box

faceChoice.addItem("Full Shave"); // add option to face choice combo box

faceChoice.addItem("Beard Trim"); // add option to face choice combo box

add(extrasLabel); // add extras label to frame

extrasChoice.add(monobrowBox); // add check box to button group

extrasChoice.add(fullEyebrowsBox); // add check box to button group

extrasChoice.add(invisibleBox); // add invisible check box to button group

invisibleBox.setSelected(true); // sets the invisible checkbox to be selected, used for error message in enter()

add(monobrowBox); // add check box to frame

add(fullEyebrowsBox); // add check box to frame

add(invisibleBox); // add invisible check box to frame

invisibleBox.setVisible(false);

// set text area to a monospaced font so the columns can be aligned using a format string

displayTextArea.setFont(new Font("Monospaced", Font.PLAIN, 12));

displayTextArea.setEditable(false); // make text area read only

scrollPane = new JScrollPane(displayTextArea); // add text area to the scroll pane

scrollPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_NEVER); // just need vertical scrolling

add(scrollPane);

// add buttons to frame

add(buttonPanel);

buttonPanel.add(enterButton);

buttonPanel.add(displayButton);

buttonPanel.add(searchButton);

buttonPanel.add(clearButton);

buttonPanel.add(exitButton);

// add the action listener to the buttons

enterButton.addActionListener(this);

displayButton.addActionListener(this);

searchButton.addActionListener(this);

clearButton.addActionListener(this);

exitButton.addActionListener(this);

// action taken when user presses window close button (x in top right corner)

addWindowListener(

new WindowAdapter()

{

public void windowClosing(WindowEvent e)

{

exit(); // run exit() method below

}

}

);

}

public void actionPerformed(ActionEvent e)

{ // processes the user button clicks

String command = e.getActionCommand();

if (command.compareTo("Enter") == 0)

enter();

else if (command.compareTo("Display All") == 0)

displayAll();

else if (command.compareTo("Search") == 0)

search();

else if (command.compareTo("Clear") == 0)

clear();

else if (command.compareTo("Exit") == 0)

exit();

}

// Button Methods

private void enter()

{ // action for when enter button is pressed

// Error Conditions

if(currentHaircut >= MAX\_HAIRCUTS) // maximum customer error message

{

displayError("Maximum number of customers has been reached");

return;

}

if(nameField.getText().compareTo("") == 0) // no customer name entered error message

{

displayError("You must enter a customer name");

return;

}

if(hairChoice.getSelectedItem().toString().compareTo("No Option") == 0 &&

faceChoice.getSelectedItem().toString().compareTo("No Option") == 0 &&

invisibleBox.isSelected()) // if no services are selected entered error message

{

displayError("You must select a service");

return;

}

// Enter Button functions

String customerName = nameField.getText();

String customerHair = hairChoice.getSelectedItem().toString();

String customerFace = faceChoice.getSelectedItem().toString();

boolean customerMonobrow = monobrowBox.isSelected();

boolean customerEyebrows = fullEyebrowsBox.isSelected();

haircutArray[currentHaircut] = new CustomerHaircut(customerName, customerHair,

customerFace, customerMonobrow, customerEyebrows); // Constructs new array element

clear();

displayCustomerHaircut(currentHaircut); // displays customer details that were just inputted

nameField.requestFocus(); // returns focus to the customer name field

++currentHaircut; // increments haircut counter for next array element

}

private void displayAll()

{ // displays all haircutArray elements, average service cost and total sales

// Error message conditions

if(currentHaircut == 0) // no customer entered error

{

displayError("No customers entered");

return;

}

// display all button functions

clear();

double totalSales = 0.0; // used to track sales totals across all orders

for(int i = 0; i < currentHaircut; ++i) // loop to display all orders

{

displayCustomerHaircut(i);

appendLine();

totalSales += haircutArray[i].calculateServiceCharge(); // adds each haircutArray elements charge to the total sales tracker

}

double averageServiceCharge = totalSales / currentHaircut; // average cost of each customers service calculation

displayTextArea.append(String.format("Average charge per customer: $%.2f\nTotal Sales: $%.2f",

averageServiceCharge, totalSales)); // display average meals and total sales in text area

nameField.requestFocus(); // returns focus to the customer name field

}

private void search()

{ // allows user to search for a customer's service details by searching the customer name

// Error message conditions

if(currentHaircut == 0) // no customers entered error

{

displayError("No customers entered");

return;

}

// Search button functions

boolean nameFound = false; // used in search loop to determine if a name has been found

int searchCounter = 0; // used to increment through array elements in the search loop

String nameSearch = JOptionPane.showInputDialog(null, "Enter customer name to search",

"The Den Barbershop Management System", JOptionPane.QUESTION\_MESSAGE); // search input dialog box

if(nameSearch.compareTo("") == 0) // no name entered in search input dialog error message

{

displayError("You must enter a name to search");

return;

}

while(!nameFound && searchCounter < currentHaircut) // Name search loop

{

if(nameSearch.equalsIgnoreCase(haircutArray[searchCounter].getCustomerName()))

{ // action for if name has been found

nameFound = true;

}

else

{ // action for if name has not been found

++searchCounter; // increments counter for next pass through loop

}

}

if(nameFound)

{ // display successful search results or error message

clear();

displayCustomerHaircut(searchCounter);

}

else

{ // error message if search has been unsuccessful

displayError(nameSearch + " not found");

return;

}

}

private void clear()

{ // clears all input and output text area

nameField.setText(""); // clears customer name field

hairChoice.setSelectedIndex(0); // resets hair choice combo box

faceChoice.setSelectedIndex(0); // resets face choice combo box

invisibleBox.setSelected(true); // resets monobrow and eyebrow check box by selecting invisible box

displayTextArea.setText(""); // clears large text display area

nameField.requestFocus(); // returns focus to the customer name field

}

private void exit()

{ // standard exit operation

JOptionPane.showMessageDialog(null, "Thank you for using The Den Barbershop Management System",

"The Den Barbershop Management System", JOptionPane.INFORMATION\_MESSAGE); // thank you message

System.exit(0);

}

// Text area display methods

private void appendLine()

{ // Append line in output text area

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

}

private void displayCustomerHaircut(int index) // index = haircutArray element that will be display

{ // displays the haircutArray element details in the output text area

displayTextArea.append(String.format("%s%s\n%-32s%s\n\n", "Customer name: ",

haircutArray[index].getCustomerName(), "Services", "Price")); // displays customer name

switch(haircutArray[index].getCustomerHairOption())

{ // display hair option customer has chosen along with price

case "No Option":

break;

case "Standard":

displayTextArea.append(String.format("%-32s$%.2f\n", "Standard cut",

CustomerHaircut.getStandardCutPrice()));

break;

case "Skin Fade":

displayTextArea.append(String.format("%-32s$%.2f\n", "Skin fade",

CustomerHaircut.getSkinFadePrice()));

break;

case "Style Cut":

displayTextArea.append(String.format("%-32s$%.2f\n", "Style cut",

CustomerHaircut.getStyleCutPrice()));

break;

}

switch(haircutArray[index].getCustomerFaceOption())

{ // display face option customer has chosen along with price

case "No Option":

break;

case "Full Shave":

displayTextArea.append(String.format("%-32s$%.2f\n", "Full shave",

CustomerHaircut.getFullShavePrice()));

break;

case "Beard Trim":

displayTextArea.append(String.format("%-32s$%.2f\n", "Beard trim",

CustomerHaircut.getBeardTrimPrice()));

break;

}

if(haircutArray[index].getCustomerMonobrowOption())

{ // displays if customer has received monobrow wax service

displayTextArea.append(String.format("%-32s$%.2f\n", "Monobrow wax",

CustomerHaircut.getMonobrowPrice()));

}

if(haircutArray[index].getCustomerFullEyebrowsOption())

{ // displays if customer has received full eyebrows wax service

displayTextArea.append(String.format("%-32s$%.2f\n", "Full eyebrows wax",

CustomerHaircut.getFullEyebrowsPrice()));

}

displayTextArea.append(String.format("\n%-32s$%.2f\n", "Total",

haircutArray[index].calculateServiceCharge())); // displays customer total

}

// Error message method

private void displayError(String message) // message = the message that is displayed in dialog box

{

displayTextArea.setText("");

JOptionPane.showMessageDialog(rootPane, message,

"The Den Barbershop Management System",

JOptionPane.ERROR\_MESSAGE);

nameField.requestFocus();

}

// Main method the program runs from

public static void main(String[] args)

{ // Create the instance of the GUI class above

Project p = new Project(); // Creates instance GUI class above

p.setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE); // Allow the code to close the program

p.setBounds(1000, 500, 490, 590); // Define position and size of app

p.setTitle("The Den Barbershop Management System"); // Sets the title of the frame

p.setVisible(true); // Sets frame to be visible

p.setResizable(false); // Does not allow user to resize frame

}

}