**GardenGUi.java**

package week\_8.q3\_garden;

import week\_8.q3\_garden.InvoiceWriter;

import javax.swing.\*;

import javax.swing.event.ChangeEvent;

import javax.swing.event.ChangeListener;

import javax.swing.text.DateFormatter;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.ItemEvent;

import java.awt.event.ItemListener;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

import java.util.Objects;

/\*\*

\* See the Lab 8 Questions.md file for the instructions.

\* Don't forget to add screenshots of your program's GUI running.

\* Save the screenshots in the screenshots directory of this project.

\*/

public class GardenGUI extends JFrame {

JPanel mainPanel;

JPanel dataEntryPanel;

JPanel invoicePreviewPanel;

JTextArea invoicePreviewTextArea;

JButton saveInvoiceButton;

JTextField customerNameTextField;

JTextField customerAddressTextField;

JButton generateInvoicePreviewButton;

JSpinner serviceDateSpinner;

JCheckBox mowingServiceCheckBox;

JLabel mowingServiceCost;

private JComboBox gardenSizeComboBox;

private JCheckBox leafRakingCheckBox;

private JLabel leafRakingCost;

private JLabel invoiceTotal;

double costMowing = 0.00;

double costLeafRaking = 0.00;

double costTotal = 0.00;

GardenGUI() {

setTitle("Garden Service Invoice");

setContentPane(mainPanel);

setPreferredSize(new Dimension(1000, 600));

pack();

setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

setVisible(true);

configureDateSpinner(); // Sets up the date spinner for you.

// TODO add event handlers here

// Remember to use an \*\*Item Changed\*\* listener (not an action listener) for the JCheckBox components.

// Don't write all of your code in this constructor. You should create methods for different tasks.

mowingServiceCheckBox.addChangeListener(new ChangeListener() {

@Override

public void stateChanged(ChangeEvent e) {

String size = Objects.requireNonNull(gardenSizeComboBox.getSelectedItem()).toString();

if (mowingServiceCheckBox.isSelected()){

if (Objects.equals(size, "SMALL")){

costMowing = 15.15;

}

if (Objects.equals(size, "MEDIUM")){

costMowing = 15.15 \* 2;

}

if (Objects.equals(size, "LARGE")){

costMowing = 15.15 \* 3;

}

} else {

costMowing = 0.0;

}

mowingServiceCost.setText("$ " + String.valueOf(costMowing));

costTotal = costMowing + costLeafRaking;

String output = String.valueOf(costTotal);

output = '$' + output;

invoiceTotal.setText(output);

}

});

gardenSizeComboBox.addItemListener(new ItemListener() {

@Override

public void itemStateChanged(ItemEvent e) {

String size = Objects.requireNonNull(gardenSizeComboBox.getSelectedItem()).toString();

if (leafRakingCheckBox.isSelected()){

if (Objects.equals(size, "SMALL")){

costLeafRaking = 12.25;

}

if (Objects.equals(size, "MEDIUM")){

costLeafRaking = 12.25 \* 2;

}

if (Objects.equals(size, "LARGE")){

costLeafRaking = 12.25 \* 3;

}

} else {

costLeafRaking = 0.0;

}

if (mowingServiceCheckBox.isSelected()){

if (Objects.equals(size, "SMALL")){

costMowing = 15.15;

}

if (Objects.equals(size, "MEDIUM")){

costMowing = 15.15 \* 2;

}

if (Objects.equals(size, "LARGE")){

costMowing = 15.15 \* 3;

}

} else {

costMowing = 0.0;

}

costTotal = costMowing + costLeafRaking;

String output = String.valueOf(costTotal);

output = '$' + output;

leafRakingCost.setText("$ " + String.valueOf(costLeafRaking));

mowingServiceCost.setText("$ " + String.valueOf(costMowing));

invoiceTotal.setText(output);

}

});

leafRakingCheckBox.addChangeListener(new ChangeListener() {

@Override

public void stateChanged(ChangeEvent e) {

String size = Objects.requireNonNull(gardenSizeComboBox.getSelectedItem()).toString();

if (leafRakingCheckBox.isSelected()){

if (Objects.equals(size, "SMALL")){

costLeafRaking = 12.25;

}

if (Objects.equals(size, "MEDIUM")){

costLeafRaking = 12.25 \* 2;

}

if (Objects.equals(size, "LARGE")){

costLeafRaking = 12.25 \* 3;

}

} else {

costLeafRaking = 0.0;

}

leafRakingCost.setText("$ " + String.valueOf(costLeafRaking));

costTotal = costMowing + costLeafRaking;

String output = String.valueOf(costTotal);

output = '$' + output;

invoiceTotal.setText(output);

}

});

generateInvoicePreviewButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String customerName = customerNameTextField.getText().trim();

String customerAddress = customerAddressTextField.getText().trim();

if (!leafRakingCheckBox.isSelected() && !mowingServiceCheckBox.isSelected()) {

showMessageDialog("You haven't selected any services.\n Please select a service to Continue", "NO SERVICE SELECTED!", 2);

invoicePreviewTextArea.setText("");

} else if (customerName.isEmpty() || customerAddress.isEmpty()) {

showMessageDialog("Please enter a CUSTOMER NAME and ADDRESS to CONTINUE!.", "CHECK NAME and ADDRESS!", 2);

invoicePreviewTextArea.setText("");

} else {

SimpleDateFormat formater = new SimpleDateFormat("MM/dd/yyyy");

String spinnerValue = formater.format(serviceDateSpinner.getValue());

// System.out.println(spinnerValue);

String invoiceTemplate =

"\*\*\*\*\*\*\*\*\*\*\*\* Garden Services Invoice \*\*\*\*\*\*\*\*\*\*\*\*" +

"\n" +

"\nRose the Gardener, 123 Main Street, Minneapolis. Telephone 612-123-4567" +

"\n" +

"\nCustomer Name: " + customerNameTextField.getText() +

"\nAddress of garden: " + customerAddressTextField.getText()+

"\n" +

"\nDate of service: " + spinnerValue+

"\nSize of garden: " + Objects.requireNonNull(gardenSizeComboBox.getSelectedItem()).toString()+

"\n" +

"\nLawn mowing service charge: $ " + costMowing+

"\nLeaf raking service charge: $ " + costLeafRaking+

"\n" +

"\nTotal: $ " + costTotal+

"\n" +

"\nPlease send payment to the address above." +

"\nThank you for your business.";

invoicePreviewTextArea.setText(invoiceTemplate);

}

}

});

saveInvoiceButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if (invoicePreviewTextArea.getText().isEmpty()) {

JOptionPane.showMessageDialog(null, "Please generate an invoice preview first.");

showMessageDialog("Please generate an invoice preview first inorder to CONTINUE!.", "Please Generate Invoice!", 2);

} else {

String customerName = customerNameTextField.getText().trim();

SimpleDateFormat formater = new SimpleDateFormat("MM/dd/yyyy");

String spinnerValue = formater.format(serviceDateSpinner.getValue());

String filename = InvoiceWriter.createFileName(customerName, spinnerValue);

InvoiceWriter.writeToFile(filename, invoicePreviewTextArea.getText());

}

}

});

}

// 1 : Information Message

// 2 : Warning Message

// TODO use this method to show an alert dialog

// type can be JOptionPane.ERROR\_MESSAGE, or JOptionPane.INFORMATION\_MESSAGE

void showMessageDialog(String message, String title, int type) {

JOptionPane.showMessageDialog(this, message, title, type);

}

// You don't need to modify this method.

private void configureDateSpinner() {

// Dates between Jan 1, 1970 and some time in 2920. I don't suppose this program will be around this long though...

SpinnerDateModel spinnerDateModel = new SpinnerDateModel(new Date(), new Date(0), new Date(30000000000000L), Calendar.DAY\_OF\_YEAR);

serviceDateSpinner.setModel(spinnerDateModel);

// Create a DateEditor to configure the way dates are displayed and edited

// Define format the dates will have - month, day, year

JSpinner.DateEditor editor = new JSpinner.DateEditor(serviceDateSpinner, "MM-dd-yyyy");

DateFormatter formatter = (DateFormatter) editor.getTextField().getFormatter();

// Attempt to prevent invalid input

formatter.setAllowsInvalid(false);

// Allow user to type as well as use up/down buttons

formatter.setOverwriteMode(true);

// And tell the serviceDataSpinner to use this Editor

serviceDateSpinner.setEditor(editor);

}

}

**InvoiceWriter.java**

package week\_8.q3\_garden;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.text.SimpleDateFormat;

import java.util.Date;

/\*\*

\* Handles writing invoices to disk. Invoices are saved in the

\* directory given by the INVOICE\_DIRECTORY constant.

\*

\* You should not need to modify this file, but you will need to call the methods in it.

\*/

public class InvoiceWriter {

static final String INVOICE\_DIRECTORY = "GardeningInvoices";

private static final String dateFormatString = "MMM\_dd\_yyyy"; // e.g. "sep\_09\_2017"

private static final SimpleDateFormat simpleDateFormat = new SimpleDateFormat(dateFormatString);

static {

File invoiceDir = new File(INVOICE\_DIRECTORY);

try {

invoiceDir.mkdir(); // We may ignore the warning about return value being ignored.

} catch (SecurityException e) {

if (!invoiceDir.exists()) {

System.out.println("ERROR - could not create Invoice Directory. " + INVOICE\_DIRECTORY);

}

// Otherwise, if it exists, presumably it has already been created, so no problem.

}

}

/\*

Create a valid filename from a date, and the customer's name.

Names may have characters that are not permitted in filenames, these must be removed or replaced.

This is a very basic solution: remove all characters from customer name that are not A-Z or a-z.

This could definitely be improved. There are many names that would be distorted by this method.

This would perform poorly on names with characters outside A-Z and a-z.

What if a customer has several characters removed from their name?

Many characters outside A-Z and a-z range are valid filename characters.

A more exhaustive solution to preserve names and create valid filenames would be more work, and more testing.

Looking into a 3rd party library to handle this would be recommended in a real program; it's a fairly common problem.

You are not required to improve this method; but you are welcome to contribute an improved version if you like :)

\*/

public static String createFileName(String customer, String date) {

String name = removeBannedCharacters(customer);

if (name.length() == 0) {

name = "Customer"; // Something, if there are no valid filename characters. Can you think of a better solution? Ask the user for a name?

}

// Format the date into a String

// String dateString = simpleDateFormat.format(date);

String filename = String.format("%s\_%s\_invoice", name, date);

filename = removeBannedCharacters(filename);

filename = filename + ".txt";

return filename;

}

protected static String removeBannedCharacters(String st) {

// Replace every character that's not in the set a-z or A-Z with an empty String

// In other words, remove every character that's not a-z or A-Z.

return st.replaceAll("[^a-zA-Z0-9]", "");

}

/\* Warning! This method overwrites an existing file. A real program

\* should warn the user that a file with the proposed name exists, and offer them

\* the choice to overwrite or give a new name.

\*

\* Returns true if the data is successfully written to the file given by the filename.

\* Prints an error message and returns false otherwise.

\* \*/

public static boolean writeToFile(String filename, String text) {

// TODO you don't need to modify this method, but you will need to call it.

try (BufferedWriter writer = new BufferedWriter(new FileWriter(new File(INVOICE\_DIRECTORY, filename)))) {

writer.write(text);

} catch (IOException e) {

System.out.println("Unable to write to file " + filename + ". Error message:\n" + e.getMessage());

return false;

}

return true;

}

}

**InvoiceGenerator.java**

package week\_8.q3\_garden;

import java.util.Map;

/\*\*

\* You should not need to modify this file,

\* but you will need to call the methods here.

\*/

public class InvoiceGenerator {

static final String GARDENER\_CONTACT = "GARDENER\_CONTACT";

static final String NAME = "NAME";

static final String ADDRESS = "ADDRESS";

static final String DATE = "DATE";

static final String GARDEN\_SIZE = "GARDEN\_SIZE";

static final String MOWING = "MOWING";

static final String LEAVES = "LEAVES";

static final String TOTAL = "TOTAL";

private static String invoiceTemplate =

"\*\*\*\*\*\*\*\*\*\*\*\* Garden Services Invoice \*\*\*\*\*\*\*\*\*\*\*\*" +

"\n" +

"\n&{GARDENER\_CONTACT}" +

"\n" +

"\nCustomer Name: &{NAME}" +

"\nAddress of garden: &{ADDRESS}" +

"\n" +

"\nDate of service: &{DATE}" +

"\nSize of garden: &{GARDEN\_SIZE}" +

"\n" +

"\nLawn mowing service charge: $ &{MOWING}" +

"\nLeaf raking service charge: $ &{LEAVES}" +

"\n" +

"\nTotal: $ &{TOTAL}" +

"\n" +

"\nPlease send payment to the address above." +

"\nThank you for your business.";

public static String generate(Map<String, String> invoiceData) {

// TODO you don't need to modify this method, but you will need to call it.

// In GardenGUI, create a HashMap with invoice data, using the keys and values described above,

// and call this method to generate the invoice String.

// Add in the gardener info String

invoiceData.put(GARDENER\_CONTACT, GardenServiceData.gardenerContactString);

// Create a String Substitutor with the HashMap;

// The HashMap keys will be used to look for placeholders in the invoice template

// When a placeholder, eg. ${LEAVES} is found in the template, it will be replaced

// with the value for the LEAVES key in the HashMap.

// StrSubstitutor sub = new StrSubstitutor(invoiceData);

//

// // Use our own template prefix

// sub.setVariablePrefix("&{");

//

// // Replace named placeholders with data from Map

// String invoice = sub.replace(invoiceTemplate);

// return invoice;

return null;

}

}

**GardenServiceData.java**

package week\_8.q3\_garden;

/\*\*

\* Constants to represent price of each service.

\* A future program would read these from a data store, and/or permit modification

\*/

public class GardenServiceData {

static String[] gardenSizes = {"Small", "Medium", "Large"};

// Prices of services

static final double MOWING = 15.15;

static final double LEAF\_RAKING = 12.25;

static final double MEDIUM\_PRICE\_MULTIPLY = 2;

static final double LARGE\_PRICE\_MULTIPLY = 3;

// Example gardener contact String, used when generating invoices

static final String gardenerContactString = "Rose the Gardener, 123 Main Street, Minneapolis. Telephone 612-123-4567";

}

**GardenProgram.java**

package week\_8.q3\_garden;

/\*

\* For this lab, you probably don't need to modify this class.

\* \*/

class GardenProgram {

public static void main(String[] args) {

GardenGUI gui = new GardenGUI();

}

}