package edu.neu.csye6200;

import java.awt.Color;

import java.awt.Component;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.ItemEvent;

import java.awt.event.ItemListener;

import java.awt.event.KeyEvent;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Comparator;

import java.util.Date;

import java.util.List;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.util.stream.Collectors;

import javax.swing.ButtonGroup;

import javax.swing.DefaultComboBoxModel;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JRadioButton;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.WindowConstants;

import javax.swing.table.DefaultTableModel;

public class SimpleSwingDemo3 implements Runnable {

public static String MAJOR\_REVISION;

public static String MINOR\_REVISION;

private JLabel m\_dateStatusLabel = new JLabel();

private JLabel m\_fruitStatusLabel = new JLabel();

private String m\_currentPattern = null;

private final Logger m\_logger = Logger.getLogger(this.getClass().getCanonicalName());

/\*\*

\* static initialization block

\* initialize all static data

\*/

static {

SimpleSwingDemo3.MAJOR\_REVISION = "2";

SimpleSwingDemo3.MINOR\_REVISION = "1";

}

/\*\*

\* allow each object created from this class to be a Runnable object

\*/

@Override

public void run() {

System.out.println("I'm a Swing'N '" + SimpleSwingDemo3.class.getName() + "' Object running in a new Thread");

}

private class MyPerson {

private Integer id = null;

private String firstName = null;

private String lastName = null;

private Integer age = null;

public MyPerson() {

super();

this.init(1, "John", "Doe", 1);

}

public MyPerson(Integer id, String firstName, String lastName, Integer age) {

super();

this.init(id, firstName, lastName, age);

}

private void init(Integer id, String firstName, String lastName, Integer age) {

this.id = id;

this.firstName = firstName;

this.lastName = lastName;

this.age = age;

}

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public Integer getAge() {

return age;

}

public void setAge(Integer age) {

this.age = age;

}

@Override

public String toString() {

StringBuilder info = new StringBuilder("Person: ");

info.append(this.getFirstName());

info.append(" ").append(this.getLastName());

info.append(", age: ").append(this.getAge());

info.append(", id: ").append(this.getId());

return info.toString();

}

public void show() {

System.out.println(this.toString());

}

}

private class MyItem {

private int id;

private double price;

private String name;

public MyItem() {

super();

// TODO Auto-generated constructor stub

}

public MyItem(int id, double price, String name) {

super();

this.id = id;

this.price = price;

this.name = name;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return "# " + id + " $" + price + ", " + name + "";

}

}

/\*\*

\* demonstrate how java auto-boxing works

\* for using primitive types in a collection

\*/

public void demoAutoBoxing() {

List<Integer> integers = new ArrayList<Integer>();

/\*

\* auto-boxing:

\* literal int type is placed inside Integer object

\* to add to sequential container

\* (sequential containers cannot work with primitive types)

\*/

integers.add(2); // autobox literal int 2

integers.add(3); // autobox literal int 3

integers.add(1); // autobox literal int 1

/\*\*

\* no auto-boxing required

\* the following is the equivalent to auto-boxing of literal 4 integer

\*/

integers.add(new Integer(4)); // add Integer holding literal int 4

}

/\*\*

\* show how to use a Thread object to execute a Runnable object

\*/

public void runThySelf() {

System.out.println("\n\t" + SimpleSwingDemo3.class.getName() + ".runThySelf()...");

Thread t1 = new Thread(this); // create Thread

t1.start(); // Start Thread running

//I'm a Swing'N Object running in a new Thread

try {

t1.join(); // Original Thread waits for new Thread to complete

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

Runnable r1 = () -> System.out.println("I'm a r1 Lambda running in a new Thread");

// I'm a r1 Lambda running in a new Thread

Thread t2 = new Thread(r1); // create Thread

t2.start(); // Start Thread running

//I'm a Swing'N Object running in a new Thread

try {

t2.join(); // Original Thread waits for new Thread to complete

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

Thread t3 = new Thread(() -> System.out.println("Hi")); // create Thread

// Hi

t3.start(); // Start Thread running

try {

t3.join(); // Original Thread waits for new Thread to complete

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

Thread t4 = new Thread(new Runnable() {

@Override

public void run() {

System.out.println("Hi, It's Anonymous ME!");

}

}); // create Thread

t4.start(); // Start Thread running

// Hi, It's Anonymous ME!

try {

t4.join(); // Original Thread waits for new Thread to complete

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

/\*\*

\* Use Thread pool and Runnable

\*/

ExecutorService executor = Executors.newFixedThreadPool(10);

// executor.execute(()-> System.out.println("Executing..."));

System.out.println("\n\t" + SimpleSwingDemo3.class.getName() + ".runThySelf()... done!");

}

/\*\*

\* create a GUI pop-up window containing supplied component

\*

\* @param c GUI component to

\*/

public void simplePopUpWindow(String windowTitle, Component c) {

/\*\*

\* create a pop-up window (JFrame) for containing table

\*/

JFrame frame = new JFrame(windowTitle);

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

/\*\*

\* create a JScrollPane for wrapping the table and

\* 1) add table to JScrollPane;

\* 2) add JScrollPane to the JFrame

\*/

frame.add(c);

/\*

\* size window (JFrame) to contained components for a simple window layout

\*/

frame.pack();

frame.setVisible(true); // make JFrame visible to user

}

/\*\*

\* create a GUI pop-up window containing supplied component

\*

\* @param c GUI component to

\*/

public void simplePopUpWindow(Component c) {

simplePopUpWindow("Simple Pop-up Window Example", c);

}

/\*\*

\* create a table (JTable) with the supplied data

\*

\* @param tableTitle String table name

\* @param colnames String array containing table column names

\* @param data Object array containing table data

\*/

public void simpleTable(String tableTitle, String[] colnames, Object[][] data) {

JTable table = new JTable(data, colnames);

table.setAutoCreateRowSorter(true); // sortable table

table.setAutoResizeMode(JTable.AUTO\_RESIZE\_ALL\_COLUMNS);

/\*

\* create scroll pane for wrapping the table

\*/

simplePopUpWindow(tableTitle, new JScrollPane(table));

}

/\*\*

\* create a table (JTable) with the supplied data

\*

\* @param myTM table model containing table data

\*/

public void simpleTable(DefaultTableModel myTM) {

JTable table = new JTable(); // create JTable

table.setModel(myTM); // set table model in JTable

/\*\*

\* make table automatically sort-able with click on table column

\* and automatically resize-able with a mouse dragging border or corner stretch

\*/

table.setAutoCreateRowSorter(true); // sortable table

table.setAutoResizeMode(JTable.AUTO\_RESIZE\_ALL\_COLUMNS);

/\*

\* create scroll pane for wrapping the table

\*/

simplePopUpWindow("MyPerson Table Example", new JScrollPane(table));

}

/\*\*

\* create a table model of MyPerson objects for simpleTable().

\*

\* Given:

\* 1) a String array of titles and

\* 2) a list of MyPerson objects, (id is President #)

\* create a Table Model and pass it to simpleTable() for GUI display.

\*

\* @param colTitles String array of table column titles

\* @param list List of MyPerson objects

\*/

public void personTable(String[] colTitles, List<MyPerson> list) {

/\*\*

\* Create a table model for containing table data

\*/

DefaultTableModel myTM = new DefaultTableModel();

/\*\*

\* Set the table columns and their titles

\*/

myTM.setColumnCount(colTitles.length);

myTM.setColumnIdentifiers(colTitles);

int ix = 0; // use ix as an index, i.e. id for object in table

for (MyPerson p : list) {

String fname = p.getFirstName();

String lname = p.getLastName();

int age = p.getAge();

myTM.addRow(new Object[]{++ix, p.getId(), fname, lname, age});

}

simpleTable(myTM);

}

public void myPersonTableDemo() {

String [] colTitles = {"ID", "Pres #", "First Name","Last Name", "Age"};

List<MyPerson> list = new ArrayList<>();

// List<Person> list = new ArrayList<>();

// Person p1 = new Person(1, "George", "Washington", 43);

// Person p2 = new Person(4, "James", "Madison", 49);

// Person p3 = new Person(2, "John", "Adams", 48);

// Person p4 = new Person(45, "Danald", "Trump", 71);

// Person p5 = new Person(46, "Daniel", "Peters", 17);

// list.add(p1);

// list.add(p2);

// list.add(p3);

// list.add(p4);

/\*\*

\* Object instance method can instantiate inner class

\* because BOTH are object instance members of the same class

\* i.e.

\* myPersonTableDemo object instance method has access to object

\* which ALSO contains class MyPerson

\*/

list.add(this.new MyPerson(46, "Daniel", "Peters", 17));

list.add(new MyPerson(1, "George", "Washington", 43));

list.add(new MyPerson(4, "James", "Madison", 49));

list.add(new MyPerson(2, "John", "Adams", 48));

list.add(new MyPerson(45, "Danald", "Trump", 71));

personTable(colTitles, list);

}

public void filteredMyPersonTableDemo() {

System.out.println("\n\t" + SimpleSwingDemo3.class.getName() + ".filteredPersonTableDemo()..." );

String [] colTitles = {"ID", "Pres #", "First Name", "Last Name", "Age"};

List<MyPerson> allPresidents = new ArrayList<>();

allPresidents.add(new MyPerson(45, "Donald", "Trump", 71));

allPresidents.add(new MyPerson(1, "George", "Washington", 43));

allPresidents.add(new MyPerson(4, "James", "Madison", 49));

allPresidents.add(new MyPerson(2, "John", "Adams", 48));

// list.add(Person(46, "Daniel", "Peters", 17));

allPresidents.add(new MyPerson(46, "Daniel", "Peters", 17));

/\*\*

\* Filter list showing ONLY early presidents (first 10)

\*/

List<MyPerson> earlyPresidents = // use with .collect terminal operation

allPresidents.stream()

.filter(p -> p.getId() < 10) // filter only EARLY presidents in US history

// sort stream using supplied Lambda method reference as keyExtractor (for Comparator)

.sorted(Comparator.comparing(MyPerson::getFirstName))

// .sorted(comparing(Person::getLastName))

.collect( Collectors.toList() );// terminate Stream in a new List

// .forEach(System.out::println); // terminate Stream with output to stdout

personTable(colTitles, earlyPresidents); // show list in a pop-up GUI table

personTable(colTitles, allPresidents); // show list in a pop-up GUI table

System.out.println("\n\t" + SimpleSwingDemo3.class.getName() + ".filteredPersonTableDemo()... done!" );

}

public void itemTableDemo() {

String[] colNames = {"ID", "PRICE", "NAME"};

List<SimpleSwingDemo3.MyItem> list = new ArrayList<SimpleSwingDemo3.MyItem>();

list.add(new MyItem(1,1.29,"Bread"));

list.add(new MyItem(2,3.49,"Oj"));

list.add(new MyItem(3,2.79,"Milk"));

list.add(new MyItem(4,.79,"Candy"));

Object[][] data = new Object[list.size()][3];

int i = 0;

for (MyItem myItem : list) {

data[i][0] = myItem.getId();

data[i][1] = myItem.getPrice();

data[i][2] = myItem.getName();

i++;

}

simpleTable("MyItem Table Example", colNames, data);

}

/\*\*

\* reformat the current date and time according to supplied pattern.

\* @param curPattern

\*/

private void reformat(String curPattern) {

Date today = new Date();

/\*\*

\* create a a class for formatting dates

\* i.e. date to text and text to date

\* according to the supplied format pattern

\*/

SimpleDateFormat formatter = new SimpleDateFormat(curPattern);

try{

/\*\*

\* format current date according to supplied pattern

\*/

String dateString = formatter.format(today);

m\_dateStatusLabel.setForeground(Color.BLACK);

m\_dateStatusLabel.setText(dateString);

System.out.println(dateString);

} catch (IllegalArgumentException e) {

m\_dateStatusLabel.setForeground(Color.red);

m\_dateStatusLabel.setText("ERROR: "+e.getMessage());

System.err.println("ERROR: IllegalArgumentException: "+ e.getMessage());

e.printStackTrace();

}

}

public void demoDateComboboxDemo() {

JPanel controlPanel = new JPanel();

JLabel headerLabel = new JLabel();

headerLabel.setText("Control in action: Date JComboBox ");

/\*\*

\* java.lang.SimpleDateFormat patterns

\*/

String[] patternExamples = {

"dd MMMMM yyyy",

"dd.MM.yy",

"MM/dd/yy",

"yyyy.MM.dd G 'at' hh:mm:ss z",

"EEE, MMM d, ''yy",

"h:mm a",

"H:mm:ss:SSS",

"K:mm a,z",

"yyyy.MMMMM.dd GGG hh:mm aaa"

};

final JComboBox<String> patternListCombo = new JComboBox<>(patternExamples);

patternListCombo.setEditable(true);

/\*\*

\* add an ActionListener on ComboBox

\* so our listener (handler) is invoked

\* i.e. executed on Event Dispatch Thread (EDT)

\* when the user make a selection from

\* the ComboBox drop down list

\*/

patternListCombo.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

JComboBox<String> cb = (JComboBox<String>) e.getSource();

String newSelection = (String) cb.getSelectedItem();

m\_currentPattern = newSelection;

reformat(m\_currentPattern);

m\_logger.log(Level.ALL, "patternList CB Selected'{0}'", newSelection);

}

});

m\_currentPattern = patternExamples[0];

patternListCombo.setSelectedIndex(0);

JScrollPane patternListScrollPane = new JScrollPane(patternListCombo);

JButton showButton = new JButton("Show");

/\*\*

\* add a listener to the button

\* so our listener (handler) is invoked

\* i.e. executed on Event Dispatch Thread (EDT)

\* When the user clicks on the button

\*/

showButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String data = "";

/\*\*

\* determine which ComboBox index has been selected by user

\* and then retrieve the ComboBox entry from list by index

\*/

if (patternListCombo.getSelectedIndex() != -1) {

data = "Pattern Selected: " + patternListCombo.getItemAt(patternListCombo.getSelectedIndex());

}

m\_dateStatusLabel.setText(data);

}

});

controlPanel.add(headerLabel);

controlPanel.add(patternListScrollPane);

controlPanel.add(showButton);

controlPanel.add(m\_dateStatusLabel);

/\*

\* create pop-up window (JFrame with a JScrillPanel) for this GUI

\*/

simplePopUpWindow("Date combo box Example", new JScrollPane(controlPanel));

}

public void demoFruitCombobox() {

JPanel controlPanel = new JPanel();

JLabel headerLabel = new JLabel();

final JLabel fruitStatusLabel = new JLabel();

fruitStatusLabel.setText(" LShow Selected Fruit Here");

headerLabel.setText("Control in action: Fruit JComboBox");

m\_fruitStatusLabel.setText(" aShow Selected Fruit Here");

final DefaultComboBoxModel<String> fruitNameComboModel = new DefaultComboBoxModel<>();

fruitNameComboModel.addElement("Apple");

fruitNameComboModel.addElement("Grapes");

fruitNameComboModel.addElement("Mango");

fruitNameComboModel.addElement("Pear");

fruitNameComboModel.addElement("BlueBerry");

fruitNameComboModel.addElement("Banana");

fruitNameComboModel.addElement("Cherry");

final JComboBox<String> fruitCombo = new JComboBox<>(fruitNameComboModel);

fruitCombo.setSelectedIndex(0); // initially selected a combo box element

JScrollPane fruitNameScrollPane = new JScrollPane(fruitCombo);

JButton showButton = new JButton("Show");

/\*\*

\* add a listener to the button

\* so our listener (handler) is invoked

\* i.e. executed on Event Dispatch Thread (EDT)

\* When the user clicks on the button

\*

\* Implement ActionListener as a in-line (anonymous) Lambda

\*/

showButton.addActionListener( (ActionEvent e) -> {

/\*\*

\* determine which ComboBox index has been selected by user

\* and then retrieve the ComboBox entry from list by index

\*/

if (fruitCombo.getSelectedIndex() != -1) {

fruitStatusLabel.setText( " LFruits Selected: " + fruitCombo.getItemAt(fruitCombo.getSelectedIndex()) );

}

}

);

/\*\*

\* add a listener to the button

\* so our listener (handler) is invoked

\* i.e. executed on Event Dispatch Thread (EDT)

\* When the user clicks on the button

\*

\* Implement ActionListener as a (named) Lambda

\*/

// ActionListener actionListenerLambda = (ActionEvent e) -> {

// /\*\*

// \* determine which ComboBox index has been selected by user

// \* and then retrieve the ComboBox entry from list by index

// \*/

// if (fruitCombo.getSelectedIndex() != -1) {

// fruitStatusLabel.setText( " LFruits Selected: " + fruitCombo.getItemAt(fruitCombo.getSelectedIndex()) );

// }

// };

// showButton.addActionListener( actionListenerLambda);

/\*\*

\* Implement (another) ActionListener as an anonymous inner class

\*/

showButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String data = "Invalid Selection!";

if (fruitCombo.getSelectedIndex() != -1) {

data = "aFruits Selected: " + fruitCombo.getItemAt(fruitCombo.getSelectedIndex());

}

m\_fruitStatusLabel.setText(data);

}

});

/\*\*

\* Add all GUI components to JPanel

\* i.e. our sub window in our JFrame window

\* to make use of our simplePopUpWindow method

\* which can only add a sinle component to the

\* JFrame window it creates and displays as our

\* pop-up window GUI.

\*/

controlPanel.add(headerLabel);

controlPanel.add(fruitNameScrollPane);

controlPanel.add(showButton);

controlPanel.add(fruitStatusLabel);

controlPanel.add(m\_fruitStatusLabel);

/\*

\* create pop-up window (JFrame with a JScrillPanel) for this GUI

\*/

simplePopUpWindow("Fruit name combo box Example", new JScrollPane(controlPanel));

}

public void demoRadioButtonDemo() {

JPanel controlPanel = new JPanel();

JLabel headerLabel = new JLabel();

final JLabel statusLabel = new JLabel();

statusLabel.setText("Show Selected RadioButton Here");

headerLabel.setText("Control in action: RadioButton");

final JRadioButton radApple = new JRadioButton("Apple", true);

final JRadioButton radMango = new JRadioButton("Mango");

final JRadioButton radPear = new JRadioButton("Pear");

radApple.setMnemonic(KeyEvent.VK\_A);

radMango.setMnemonic(KeyEvent.VK\_M);

radPear.setMnemonic(KeyEvent.VK\_P);

radApple.addItemListener(new ItemListener() {

public void itemStateChanged(ItemEvent e) {

statusLabel.setText("Apple RadioButton: " + (e.getStateChange() == 1 ? "checked" : "unchecked"));

}

});

/\*\*

\* Lambda implementation of ItemListener

\*/

radMango.addItemListener(

(e) -> statusLabel.setText("Mango RadioButton: " + (e.getStateChange() == 1 ? "checked" : "unchecked"))

);

//

// radMango.addItemListener(new ItemListener() {

// public void itemStateChanged(ItemEvent e) {

// statusLabel.setText("Mango RadioButton: " + (e.getStateChange() == 1 ? "checked" : "unchecked"));

// }

// });

radPear.addItemListener(new ItemListener() {

public void itemStateChanged(ItemEvent e) {

statusLabel.setText("Pear RadioButton: " + (e.getStateChange() == 1 ? "checked" : "unchecked"));

}

});

// Group the radio buttons.

ButtonGroup group = new ButtonGroup();

group.add(radApple);

group.add(radMango);

group.add(radPear);

/\*\*

\* add each GUI control to a JPanel

\* (just for GUI organization)

\*/

controlPanel.add(headerLabel);

controlPanel.add(radApple);

controlPanel.add(radMango);

controlPanel.add(radPear);

controlPanel.add(statusLabel);

/\*

\* create pop-up window (JFrame with a JScrillPanel) for this GUI

\*/

simplePopUpWindow("Fruit Radio Button Example", controlPanel);

// simplePopUpWindow("Fruit Radio Button Example", new JScrollPane(controlPanel));

}

public static void demo() {

System.out.println("\n\t" + SimpleSwingDemo3.class.getName() + ".demo()...");

SimpleSwingDemo3 obj = new SimpleSwingDemo3();

// obj.demoRadioButtonDemo(); // radio buttons

// obj.demoFruitCombobox(); // combo box

// obj.demoDateComboboxDemo(); // combo box

// obj.myPersonTableDemo(); // table of MyPerson objects

// obj.itemTableDemo(); // table of Item objects

obj.filteredMyPersonTableDemo();

// obj.runThySelf(); // demonstrate Thread and Runnable

/\*\*

\* Instantiate an object from an inner class from this static method

\*

\* static demo method means demo is not an object instance method

\* i.e. no object exists...yet

\* MUST create an object, e.g. "obj" and

\* MUST use an object, e.g. "obj" to create object from inner class

\* e.g.

\* MyPerson innerObject = obj.new MyPerson();

\*/

MyPerson innerObject1 = obj.new MyPerson(); // use obj to create object from inner class

System.out.println("Inner class MyPerson: " + innerObject1);

SimpleSwingDemo3.MyPerson innerObject2 = obj.new MyPerson();

System.out.println("Inner class MyPerson: " + innerObject2);

System.out.println("\n\t" + SimpleSwingDemo3.class.getName() + ".demo()... done!");

}

}