Joao Paulo Dos Santos Ferreira

Fundamentals of Programing II – CSIT 112\_04

Professor Dajin Wang

April 22, 2018

**Programing Project 4/16/2018**

**PP 8.8 -** Design and implement an application that reads a sequence of up to 25 pairs of names and postal (ZIP) codes for individuals. Store the data in an object designed to store a first name (string), last name (string), and postal code (integer). Assume each line of input will contain two strings followed by an integer value, each separated by a tab character. Then, after the input has been read in, print the list in an appropriate format to the screen.

import java.util.Scanner;  
import java.io.\*;  
  
public class Contacts {  
   
 private String firstName,lastName, address, city, state, line;  
 private int zip, contactList;  
 private long phone;  
   
 // Constructor can be used for both contacts input file  
 public Contacts(Scanner scan) throws IOException {  
 // Stores the whole scanned line in a String  
 line = scan.nextLine();  
   
 // Based on line length, we know which file is being scanned  
 if (line.length() < 26)   
 {   
 // Assign which file is being looked at   
 // to be used later on the toString method  
 contactList = 1;  
   
 // Creates a new scan to scan each line individually  
 Scanner scan2 = new Scanner(line);  
   
 // scans the line  
 firstName = scan2.next();  
 lastName = scan2.next();  
 zip = scan2.nextInt();  
 }  
   
 else   
 {  
 // Assign which file is being looked at   
 // to be used later on the toString method  
 contactList = 2;  
   
 // Creates a new scan to scan each line individually  
 // and changes the delimiter   
 Scanner scan2 = new Scanner(line);  
 scan2.useDelimiter(";");  
   
 // scans the line  
 firstName = scan2.next();  
 lastName = scan2.next();  
 address = scan2.next();  
 city = scan2.next();  
 state = scan2.next();  
 zip = scan2.nextInt();  
 phone = scan2.nextLong();  
 }  
 }  
   
 public String toString() {  
 String result = "";   
   
 // Outer If-else statements tells the program which list was being looked at  
 // Inner If-else statements organizes the output in a nice way  
 if (contactList == 1)   
 {   
 if (lastName.length() + firstName.length() > 13)  
 result = lastName + ", " + firstName + "\t" + zip;  
 else   
 result = lastName + ", " + firstName + "\t\t" + zip;  
 }  
   
 else   
 {  
 if (lastName.length() + firstName.length() > 13)  
 result = lastName + ", " + firstName + "\t" + address + ", " + city + ", " + state;  
 else   
 result = lastName + ", " + firstName + "\t\t" + address + ", " + city + ", " + state;  
   
 if ((address.length() + city.length()) > 25)  
 result += "\t" + zip + "\t" + phone;  
 else   
 result += "\t\t" + zip + "\t" + phone;  
 }  
   
 return result;  
 }  
}

import java.util.ArrayList;  
import java.util.Scanner;  
import java.io.\*;  
  
public class ZipCodes {  
 public static void main(String[] args) throws IOException {  
   
 // Creates an array to store up to 25 Contacts  
 Contacts[] contacts = new Contacts[25];  
   
 // Scans the contacts file  
 Scanner scan = new Scanner(new File("contacts.dat"));  
   
 // Loop stores the information in the ArrayList  
 // and prints it in a nice way  
 System.out.println("Name: \t\t\tZIP Code:");  
 for (int i=0; i < 22; i++)   
 {  
 Contacts contact = new Contacts(scan);  
   
 contacts[i] = contact;  
   
 System.out.println(contacts[i]);  
 }  
 }  
}

**PP 8.9 -** Design and implement an application that reads a sequence of up to 25 pairs of names and postal (ZIP) codes for individuals. Store the data in an object designed to store a first name (string), last name (string), and postal code (integer). Assume each line of input will contain two strings followed by an integer value, each separated by a tab character. Then, after the input has been read in, print the list in an appropriate format to the screen.

import java.util.Scanner;  
import java.io.\*;  
  
public class Contacts {  
   
 private String firstName,lastName, address, city, state, line;  
 private int zip, contactList;  
 private long phone;  
   
 // Constructor can be used for both contacts input file  
 public Contacts(Scanner scan) throws IOException {  
 // Stores the whole scanned line in a String  
 line = scan.nextLine();  
   
 // Based on line length, we know which file is being scanned  
 if (line.length() < 26)   
 {   
 // Assign which file is being looked at   
 // to be used later on the toString method  
 contactList = 1;  
   
 // Creates a new scan to scan each line individually  
 Scanner scan2 = new Scanner(line);  
   
 // scans the line  
 firstName = scan2.next();  
 lastName = scan2.next();  
 zip = scan2.nextInt();  
 }  
   
 else   
 {  
 // Assign which file is being looked at   
 // to be used later on the toString method  
 contactList = 2;  
   
 // Creates a new scan to scan each line individually  
 // and changes the delimiter   
 Scanner scan2 = new Scanner(line);  
 scan2.useDelimiter(";");  
   
 // scans the line  
 firstName = scan2.next();  
 lastName = scan2.next();  
 address = scan2.next();  
 city = scan2.next();  
 state = scan2.next();  
 zip = scan2.nextInt();  
 phone = scan2.nextLong();  
 }  
 }  
   
 public String toString() {  
 String result = "";   
   
 // Outer If-else statements tells the program which list was being looked at  
 // Inner If-else statements organizes the output in a nice way  
 if (contactList == 1)   
 {   
 if (lastName.length() + firstName.length() > 13)  
 result = lastName + ", " + firstName + "\t" + zip;  
 else   
 result = lastName + ", " + firstName + "\t\t" + zip;  
 }  
   
 else   
 {  
 if (lastName.length() + firstName.length() > 13)  
 result = lastName + ", " + firstName + "\t" + address + ", " + city + ", " + state;  
 else   
 result = lastName + ", " + firstName + "\t\t" + address + ", " + city + ", " + state;  
   
 if ((address.length() + city.length()) > 25)  
 result += "\t" + zip + "\t" + phone;  
 else   
 result += "\t\t" + zip + "\t" + phone;  
 }  
   
 return result;  
 }  
}

**PP 8.13 -** Design a class that represents the visual representation of a car. Use polylines and polygons to draw the car in any graphics context and at any location. Create a main driver to display the car.

import javax.swing.JFrame;  
  
public class Car  
{  
 //-----------------------------------------------------------------  
 // Creates and presents the program frame.  
 //-----------------------------------------------------------------  
 public static void main (String[] args)  
 {  
 JFrame frame = new JFrame ("Car");  
 frame.setDefaultCloseOperation (JFrame.EXIT\_ON\_CLOSE);  
  
 CarPanel panel = new CarPanel();  
 frame.getContentPane().add(panel);  
  
 frame.pack();  
 frame.setVisible(true);  
 }  
}

import javax.swing.JPanel;  
import java.awt.\*;  
  
public class CarPanel extends JPanel {  
   
 // Sets the coordinates of each point  
 private int[] xCar = {10, 10, 30, 70, 115, 150, 180, 180};  
 private int[] yCar = {140, 95, 95, 60, 60, 95, 95, 140};  
 private int[] xWindow = {85, 115, 115, 85};  
 private int[] yWindow = {70, 70, 90, 90};  
 private int radius = 45;  
 private int y = 115;  
 private int x1 = 20;  
 private int x2 = 125;  
   
 //-----------------------------------------------------------------  
 // Constructor: Sets up the basic characteristics of this panel.  
 //-----------------------------------------------------------------  
 public CarPanel()   
 {  
 setBackground (Color.white);  
 setPreferredSize (new Dimension(200, 200));  
 }  
   
 //-----------------------------------------------------------------  
 // Draws a car using polygons and polylines.  
 //-----------------------------------------------------------------  
 public void paintComponent (Graphics page){  
 super.paintComponent (page);  
 page.setColor (Color.blue);  
 page.fillPolygon (xCar, yCar, xCar.length);  
 page.setColor (Color.yellow);  
 page.fillPolygon (xWindow, yWindow, xWindow.length);  
 page.setColor (Color.black);  
 page.fillOval(x1, y, radius, radius);  
 page.setColor (Color.black);  
 page.fillOval(x2, y, radius, radius);   
   
 }  
}

**PP 8.15 -** Modify the *QuoteOptions* program from Chapter 5 so that it provides three additional quote options. Use an array to store all of the quote strings.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// QuoteOptions.java Author: Lewis/Loftus  
//  
// Demonstrates the use of radio buttons.  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
import javax.swing.JFrame;  
  
public class QuoteOptions  
{  
 //-----------------------------------------------------------------  
 // Creates and presents the program frame.  
 //-----------------------------------------------------------------  
 public static void main (String[] args)  
 {  
 JFrame frame = new JFrame ("Quote Options");  
 frame.setDefaultCloseOperation (JFrame.EXIT\_ON\_CLOSE);  
  
 QuoteOptionsPanel panel = new QuoteOptionsPanel();  
 frame.getContentPane().add(panel);  
  
 frame.pack();  
 frame.setVisible(true);  
 }  
}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// QuoteOptionsPanel.java Author: Lewis/Loftus  
//  
// Demonstrates the use of radio buttons.  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.\*;  
  
public class QuoteOptionsPanel extends JPanel  
{  
 private JLabel quote;  
 private JRadioButton comedy, philosophy, carpentry, inspirational, heartBreak, happy;  
 private String comedyQuote, philosophyQuote, carpentryQuote, inspirationalQuote, heartBreakQuote, happyQuote;  
 private String[] quotes = {"dasdasd", "asadsadas", "dasdascae"};  
  
 //-----------------------------------------------------------------  
 // Sets up a panel with a label and a set of radio buttons  
 // that control its text.  
 //-----------------------------------------------------------------  
 public QuoteOptionsPanel()  
 {  
   
 quotes = new String[6];  
 comedyQuote = "Take my wife, please.";  
 philosophyQuote = "I think, therefore I am.";  
 carpentryQuote = "Measure twice. Cut once.";  
 inspirationalQuote = "I believe you can do it!";  
 heartBreakQuote = "Your heart is my piñata.";  
 happyQuote = "Be happy for this moment";  
   
 quotes[0] = comedyQuote;  
 quotes[1] = philosophyQuote;  
 quotes[2] = carpentryQuote;  
 quotes[3] = inspirationalQuote;  
 quotes[4] = heartBreakQuote;  
 quotes[5] = happyQuote;  
   
 quote = new JLabel (comedyQuote);  
 quote.setFont (new Font ("Helvetica", Font.BOLD, 24));  
  
 comedy = new JRadioButton ("Comedy", true);  
 comedy.setBackground (Color.green);  
 philosophy = new JRadioButton ("Philosophy");  
 philosophy.setBackground (Color.green);  
 carpentry = new JRadioButton ("Carpentry");  
 carpentry.setBackground (Color.green);  
 inspirational = new JRadioButton ("Inspirational");  
 inspirational.setBackground (Color.green);  
 heartBreak= new JRadioButton ("Heart break");  
 heartBreak.setBackground (Color.green);  
 happy = new JRadioButton ("Happy");  
 happy.setBackground (Color.green);  
   
   
  
 ButtonGroup group = new ButtonGroup();  
 group.add (comedy);  
 group.add (philosophy);  
 group.add (carpentry);  
 group.add(inspirational);  
 group.add(heartBreak);  
 group.add(happy);  
  
 QuoteListener listener = new QuoteListener();  
 comedy.addActionListener (listener);  
 philosophy.addActionListener (listener);  
 carpentry.addActionListener (listener);  
 inspirational.addActionListener(listener);  
 heartBreak.addActionListener(listener);  
 happy.addActionListener(listener);  
   
 add (quote);  
 add (comedy);  
 add (philosophy);  
 add (carpentry);  
 add (inspirational);  
 add (heartBreak);  
 add (happy);  
  
 setBackground (Color.green);  
 setPreferredSize (new Dimension(300, 100));  
 }  
  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 // Represents the listener for all radio buttons  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 private class QuoteListener implements ActionListener  
 {  
 //--------------------------------------------------------------  
 // Sets the text of the label depending on which radio  
 // button was pressed.  
 //--------------------------------------------------------------  
 public void actionPerformed (ActionEvent event)  
 {  
 Object source = event.getSource();  
  
 if (source == comedy)  
 quote.setText (quotes[0]);  
 else if (source == philosophy)  
 quote.setText (quotes[1]);  
 else if (source == carpentry)  
 quote.setText (quotes[2]);  
 else if (source == inspirational)  
 quote.setText(quotes[3]);  
 else if (source == heartBreak)  
 quote.setText (quotes[4]);  
 else if (source == happy)  
 quote.setText (quotes[5]);  
   
 }  
 }  
}