Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №4

Виконав:

студент групи ІП-22

Борисов О.В.

Перевірив:

Невдащенко М. В.

Номер залікової книжки:

2204

Київ 2014

Тема: Розширення можливостей графічного інтерфейсу користувача.

Мета: Ознайомлення з реалізацією MVC-патерну в бібліотеці SWING. Отримання

навичок з побудови інтерактивного інтерфейсу користувача. Поглиблене

вивчення функціональних можливостей бібліотек SWING та Java2D.

Завдання

1. Визначити C2 як остачу від ділення номера залікової книжки студента на 7.

2. В залежності від C2 визначити тип редагування:Редагування за допомогою миші в графічній області з синхронізацією табличної

області. Наприклад: виділити мишею точку на графіку і за допомогою

перемістити її (натисканням миші чи за допомогою drag’n’drop) – відповідно

таблична частина має змінити координати для цієї точки.

3. Модифікувати додаток таким чином, щоб забезпечити: можливість модифікації

завантажених даних згідно варіанту (п. 2); синхронізацію графічної і табличної

частин додатку; можливість збереження діаграми у файлі формату JPEG.

4. Код повинен відповідати стандартам JCC та бути детально задокументований.

Опис програми

Дана програма є збіркою попередніх лабораторних робіт, що спрощує її виконання. Для повної реалізації завдання знадобилося створити метод(CustomListener), який реагує на активність мишки користувача на діаграмі. Також для більшої зручності додано графічну відповідь програмою на дії користувача по зміні значень в таблиці та на діаграмі(drawSelectedPart). Одним із завдань було створення картинки діаграми у форматі jpg для цього на форму було додано кнопку, яка відповідає за цю дію.

Код програми

MyExcel.java

package lab4;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.ArrayList;

public class MyExcel {

// Attachment class which make new thread for parsing csv file

private static class Mythread extends Thread{

private String table[][];

private CSVProcessor csvprocessor;

String[][] GetTable(){

return table;

}

Mythread(final CSVProcessor csvprocessor){

this.csvprocessor = csvprocessor;

}

@Override

public void run() {

try {

table = csvprocessor.parse();

} catch (CSVParseException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

// Main method

public static void main(String[] args) throws Exception {

// TODO Auto-generated method stub

CSVProcessor csvProcessor = new CSVProcessor();

String pathCSV, pathDat = "serialized.dat";

ArrayList<String> list = null;

if(csvProcessor.ExistsFile(pathDat))

{

list = csvProcessor.DatFileRead(pathDat);

}

else

{

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

byte kol = 0;

do

{

try{

System.out.println("Put path to file");

pathCSV = reader.readLine();

list = csvProcessor.CsvRead(pathCSV);

}

catch(Exception e){

kol++;

}

}

while (list == null && kol != 3);

reader.close();

if(kol == 3)

throw new Exception("Enter path right!");

csvProcessor.DatFileWrite(pathDat);

}

// Put on console data from csv file

for(String line: list)

System.out.println(line);

System.out.println();

Mythread thread = new Mythread(csvProcessor);

thread.start();

thread.join(); // Stop main thread until daughter thread not ended

String table[][] = thread.GetTable();

// Put on console data from csv file in format table

for(String line[]: table){

for(String item: line)

System.out.format("%30s", item);

System.out.println();

}

// Start draw graphics interface

Interface inter = new Interface();

inter.putData(table);

//inter.setDaemon(true);

inter.start();

}

}

Interface.java

package lab4;

import java.awt.AWTException;

import java.awt.BorderLayout;

import java.awt.Dimension;

import java.awt.Point;

import java.awt.Rectangle;

import java.awt.Robot;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JScrollPane;

import javax.swing.JTable;

// Class which make graphic interface for data from csv file

public class Interface extends Thread{

private String data[][];

private JFrame frame;

private JTable table;

private DiagramDrawer diagram;

public void putData(String data[][]){

this.data = data;

}

// Make canvas and put on it table, button, diagram

@Override

public void run() {

// Table Creation

String columns[], items[][] = new String[data.length - 1][];

columns = data[0];

for(int i = 1; i < data.length; ++i)

items[i - 1] = data[i];

float values[] = new float[data.length - 1]; // Values for diagram

String names[] = new String[data.length - 1]; // Names for columns

for(int i = 0; i < items.length; i++){

values[i] = Float.parseFloat(items[i][1]); // Gets values from diagram from data

try{

if(values[i] < 0)

throw new Exception("Signed number" + values[i]);

}catch(Exception e){

values[i] = 0;

items[i][1] = e.getMessage();

}

names[i] = data[i + 1][0]; // Gets names for table columns

}

table = new JTable(items, columns);

table.getTableHeader().setReorderingAllowed(false); // No changes for position of table columns

JScrollPane scrollPane = new JScrollPane(table);// Make scroll panel for table

scrollPane.setPreferredSize(new Dimension(500, 100));

// Diagram creation

diagram = new DiagramDrawer();

diagram.setPreferredSize(new Dimension(500, 500));

diagram.putValues(names, values);

// Button creation

JButton button = new JButton("Make image");

button.addActionListener(new ActionListener(){ // Make jpg of diagram when button pressed

public void actionPerformed(ActionEvent e) {

try {

BufferedImage bufferedImage = new Robot().createScreenCapture(new Rectangle(3 + frame.getX(), // Capture picture

25 + frame.getY(), diagram.getWidth() - 2, diagram.getHeight()));

ImageIO.write(bufferedImage, "jpg", new File("Diagram.jpg")); // Save picture as jpg file

} catch (AWTException e1) {

e1.printStackTrace();

} catch (IOException e2) {

e2.printStackTrace();

}

/\*try

{

BufferedImage image = new BufferedImage(diagram.getWidth(), diagram.getHeight(), BufferedImage.TYPE\_INT\_RGB);

Graphics graphics = image.createGraphics();

diagram.paint(graphics);

ImageIO.write(image,"jpeg", new File("D:/we.jpeg"));

}

catch(Exception exception)

{

//code

}\*/

}

});

// Creation canvas

frame = new JFrame("MiniExcel");

// Added on canvas elements

frame.add(diagram, BorderLayout.NORTH);

frame.add(scrollPane, BorderLayout.AFTER\_LINE\_ENDS);

frame.add(button, BorderLayout.AFTER\_LAST\_LINE);

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

frame.pack();

frame.setResizable(false); // No resizable for canvas

frame.setVisible(true);

changesValue(names, values);// Check for updates data

}

// Method for updates data

private void changesValue(String names[], float values[]){

int x = -1, y = -1, colum = -1; // Response for changes in table

int indexChangeValue = -1, angleBegin = -1, angleEnd = -1; // Response for changes in diagram

CustomListener mouseEvent = new CustomListener(); // Class which response for mouse events on diagram

boolean changeValue = false, mouseDrag = false;

diagram.addMouseListener(mouseEvent);

for(;frame != null;){

diagram.makeCavas();

try {

Thread.sleep(30);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

//Look in table changes

colum = table.getSelectedColumn();

if(!changeValue && colum == 1 ){ // User activated on of the table item

y = table.getSelectedRow();

x = colum;

changeValue = true;

}

if(changeValue){

diagram.drawSelectedPart(y, 0, values[y]); // Shows selected item on diagram

if (y != table.getSelectedRow() || colum != 1){

try{

values[y] = Float.parseFloat(table.getValueAt(y, x).toString()); // Try to save new value

}catch(Exception e){

values[y] = 0;

table.setValueAt("Error ####", y, x);

}

try{

if(values[y] < 0)

throw new Exception("Signed number " + values[y]);

}catch(Exception e){

values[y] = 0;

table.setValueAt(e.getMessage(), y, x);

}

changeValue = false;

diagram.putValues(names, values); // Updates values in table

}

}

//Changes on diagram by mouse

if(mouseEvent.getMouseStatus() && !mouseDrag){ // Mouse press on diagram.

indexChangeValue = 0;

angleBegin = getAngleFromDot(mouseEvent.getStartMousePosition()); // Remember start angle.

// Get index of selected element of diagram

for(int startAngle = 0; indexChangeValue <

values.length; indexChangeValue++){

if(angleBegin >= startAngle

&& angleBegin < startAngle + diagram.getAngles()[indexChangeValue])

break;

startAngle += diagram.getAngles()[indexChangeValue];

}

mouseDrag = true; // Set that mouse drag

}

if(mouseDrag){ // Mouse release on diagram

try{

angleEnd = getAngleFromDot(diagram.getMousePosition()); // Get angle where mouse was release.

angleEnd = diagram.drawSelectedPart(indexChangeValue, angleBegin, angleEnd); // Draw active line on selected part of diagram

if(!mouseEvent.getMouseStatus()){

mouseDrag = false;

if(angleEnd >= 360) // Protection of unacceptable errors of angleEnd calculation

angleEnd = 359;

if(diagram.getAngles()[indexChangeValue] != angleEnd)

values[indexChangeValue] = angleEnd \* (diagram.getSum() - values[indexChangeValue]) / (360 - angleEnd); // Put new value to diagram

table.setValueAt(String.valueOf(values[indexChangeValue]), indexChangeValue, 1); // Set new value in table

diagram.putValues(names, values);// Set new value in diagram

}

}

catch(Exception e){

mouseDrag = false;

}

}

diagram.drawDiagram(); // Update diagram graphic

}

}

// Method calculate angle from dot position to position of dot of circle beginning

private int getAngleFromDot(Point dot){

double a, b, c;

int angle;

a = dot.distance(diagram.getCenter());

b = diagram.getCenter().distance(new Point(diagram.getCenter().x +

diagram.getBigRadiusW(), diagram.getCenter().y));

c = dot.distance(new Point(diagram.getCenter().x +

diagram.getBigRadiusW(), diagram.getCenter().y));

angle = (int) Math.floor((180 / Math.PI) \* Math.acos((c \* c - a \* a - b \* b) / (2 \* a \* b)));

if(dot.y > diagram.getCenter().y)

angle = 180 + angle;

else angle = 180 - angle;

return angle;

}

// Class response for mouse events

public class CustomListener implements MouseListener {

private boolean mouseStatus = false;

private Point beginMousePosition;

public boolean getMouseStatus(){

return mouseStatus;

}

public Point getStartMousePosition(){

return beginMousePosition;

}

public void mouseEntered(MouseEvent e) {

}

public void mouseExited(MouseEvent e) {

mouseStatus = false;

}

// Method calculate if mouse pressed on diagram circle

public void mousePressed(MouseEvent e) {

Point mousePosition = diagram.getMousePosition();

if(mousePosition != null && mousePosition.distance(diagram.getCenter()) < diagram.getBigRadiusW() - 10 &&

mousePosition.distance(diagram.getCenter()) > diagram.getSmallRadiusW() + 10){

mouseStatus = true;

beginMousePosition = mousePosition;

}

}

public void mouseReleased(MouseEvent e) {

mouseStatus = false;

}

@Override

public void mouseClicked(MouseEvent arg0) {

// TODO Auto-generated method stub

}

}

}

DiagramDrawing.java

package lab4;

import java.awt.Canvas;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Graphics;

import java.awt.Point;

import java.awt.image.BufferStrategy;

// Class uses to draw diagram

public class DiagramDrawer extends Canvas{

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

private int x0, y0, rBig1, rBig2, rSmall1, rSmall2; // Position and radius of diagram circle

float sum = 0; // Show summary of values

private int angles[] = {360};

private String names[] = {""}; // Nemes of diagram columns

private Graphics g;

private BufferStrategy bs;

public float getSum(){

return sum;

}

public Point getCenter(){

return new Point(x0, y0);

}

public int[] getAngles(){

return angles;

}

public int getBigRadiusW(){

return rBig1;

}

public int getBigRadiusH(){

return rBig2;

}

public int getSmallRadiusW(){

return rSmall1;

}

public int getSmallRadiusH(){

return rSmall2;

}

// Method draw line around active element of diagram(table event)

public void drawSelectedPart(int index, float changeValueOn, float value){

int angleBegin = 0, angleEnd =(int) (1 + changeValueOn / value) \* angles[index];

for(int i = 0; i < index; i++)

angleBegin += angles[i];

g.setColor(Color.magenta);

g.fillArc(x0 - rBig1 - 6, y0 - rBig2 - 6, rBig1 \* 2 + 12,

rBig2 \* 2 + 12, angleBegin, angleEnd);

}

// Method draw line around active element of diagram(mouse event)

public int drawSelectedPart(int index, int angleChengBegin, int angleChengEnd){

int newAngleBegin = 0, newAngleEnd = 0;

for(int i = 0; i < index; i++)

newAngleBegin += angles[i];

if(angleChengBegin > 180 && angleChengEnd < newAngleBegin) // New angle for diagram element

newAngleEnd = angles[index] + angleChengEnd + (360 - angleChengBegin);

else newAngleEnd = angles[index] + angleChengEnd - angleChengBegin;

g.setColor(Color.blue);

g.fillArc(x0 - rBig1 - 4, y0 - rBig2 - 4, rBig1 \* 2 + 8,

rBig2 \* 2 + 8, newAngleBegin, newAngleEnd);

return newAngleEnd;

}

// Set Size of diagram, position of center and radius

public void setPreferredSize(Dimension dimension){

y0 = dimension.height / 2;

x0 = dimension.width / 2;

rBig1 = (int) ((x0 + (x0 \* 0.05)) / 2);

rBig2 = (int) ((y0 + (y0 \* 0.05)) / 2);

rSmall1 = (int) (rBig1 \* 0.5);

rSmall2 = (int) (rBig2 \* 0.5);

super.setPreferredSize(dimension);

}

//Generate special color for diagram elements

private Color newColor(int r, int b, int g){

if(r > 255)

r = r % 255;

if(b > 255)

b = b % 255;

if(g > 255)

g = g % 255;

return new Color(r,b,g);

}

// Transform values of table into angles of diagram elements

public void putValues(String names[], float values[]){

int max = 0, allAng = 0;

float minAngles = 0, valueOnAngle = 0;

angles = new int[values.length + 1];

sum = angles[values.length] = 0;

for(float value: values)

sum += value;

valueOnAngle = 360 / sum; // Percent factor

for(int i = 0;i < values.length;i++){

if(values[max] < values[i])

{

max = i;

}

angles[i] = (int) (values[i] \* valueOnAngle);

if(angles[i] == 0){

minAngles += values[i];

}

allAng += angles[i];

}

if(minAngles > 0.0)// Values which angle < then 1(make special diagram element)

{

angles[values.length] = (int) Math.ceil(minAngles \* valueOnAngle);

allAng += angles[values.length];

}

angles[max] += 360 - allAng; // Add unallocated angle to the most large diagram element

this.names = names;

}

// Set connection to canvas

public void makeCavas(){

bs = getBufferStrategy();

if (bs == null) {

createBufferStrategy(2); //создаем BufferStrategy для нашего холста

requestFocus();

bs = getBufferStrategy();

}

g = bs.getDrawGraphics();

g.setColor(Color.white);

g.fillRect(0, 0, this.getWidth(), this.getHeight());

}

// Method draw diagram

public void drawDiagram() {

int x1, y1, x2 = 0, y2 = 0, angleStr = 330, startAngle = 0, comentAngle;

for(int i = 0; i < angles.length; i++){ // Each element of diagram draw separately

if(angles[i] != 0){

comentAngle = (startAngle + angles[i]/2);

g.setColor(newColor(20 \* i + 40, 15 \* i + 50, 30 \* i + 40)); // Generate color

// Calculate position of name and message lines of element

x1=(int) (x0 + rBig1 \* Math.cos((Math.PI / 180) \*

(360 - startAngle - angles[i] / 2)));

y1=(int) (y0 + rBig2 \* Math.sin((Math.PI / 180) \*

(360 - startAngle - angles[i] / 2)));

// Look in what dimension of diagram will be diagram element

if((comentAngle) >= 90 && (comentAngle) < 180)

angleStr = 230;

if((comentAngle) >= 180 && (comentAngle) < 270)

angleStr = 130;

if((comentAngle) >= 270 && (comentAngle) < 360)

angleStr = 30;

x2=(int) (x1 + rSmall1 / 2 \* Math.cos((Math.PI / 180) \* angleStr));

y2=(int) (y1 + rSmall2 / 2 \* Math.sin((Math.PI / 180) \* angleStr));

g.drawLine(x1, y1, x2, y2); // Draw message line

if((comentAngle) < 90 || (comentAngle) > 270){

g.drawLine(x2, y2, (int) (x2 + x0 \* 0.4), y2);

if(i == angles.length - 1){

g.setColor(Color.cyan);

g.drawString(String.valueOf((int) (angles[i] / 360. \* 100000) /

1000.) +"% Інші", x2, y2);

}

else g.drawString(String.valueOf((int) (angles[i] / 360. \* 100000) /

1000.) + "% " + names[i], x2, y2);

}

else{

g.drawLine(x2, y2, (int) (x2 - x0 \* 0.4), y2);

g.drawString(String.valueOf((int) (angles[i] / 360. \* 100000) /

1000.) + "% " + names[i], (int) (x2 - x0 \* 0.4), y2);

}

g.fillArc(x0 - rBig1, y0 - rBig2, rBig1 \* 2, rBig2 \* 2, startAngle, angles[i]);// Draw part of diagram circle

startAngle += angles[i];

}

}

g.setColor(Color.white);

g.fillArc(x0 - rSmall1, y0 - rSmall2, rSmall1 \* 2, rSmall2 \* 2, 0, 360); // Make from circle ring

g.dispose();

bs.show();

}

}

CSVParseException.java і CSVProcessor.java в лабораторній роботі 2

Висновок

У процесі виконання лабораторної роботи я навчився використовувати swing бібліотеку, а саме її елементи JFrame, JButton, JScrollPane, JTable. Навчився оброблювати події мишки. Також зумів покращити графічний інтерфейс програми, що надало їй більшої інтерактивності.

Приклад роботи

