Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики и обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №6**

«Створення прикладного додатку з графічним інтерфейсом

користувача на основі SWING»

Виконав: ст. гр. ІО-92

Петрук В.О.

Перевірив: Болдак А. А.

Київ 2011

**Мета**: Ознайомлення з структурою і функціональними можливостями бібліотеки легких компонентів SWING. Отримання базових навичок з побудови графічного інтерфейсу користувача.

**Завдання**

1. Детально ознайомитись з призначенням і структурою пакету javax.swing. Знати його відмінності від java.awt. Вільно оріентуватись в компонентах бібліотеки SWING.

2. Ознайомитись з концепцією панелей (Panes) в SWING. Знати призначення і вміти використовувати Root Pane, Content Pane, Layered Pane, Glass Pane.

3. Ознайомитись з концепцією менеджерів компоновки (Layout Managers). Вільно володіти існуючими менеджерами

4. Модифікувати, якщо це потрібно, дизайн графічного інтерфейсу користувача (з ЛР-4) для використання компонентів бібліотеки SWING.

5. Скласти таблицю відповідності елементів дизайну інтерфейса Java класам з бібліотеки SWING.

6. Реалізувати розроблений інтерфейс у вигляді програмного додатку.

**Лістинг проекту**

**Класс CSVProcessor**

**package** com.lab111.lab6;

**import** java.util.ArrayList;

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.FileOutputStream;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.io.ObjectInputStream;

**import** java.io.ObjectOutputStream;

**import** java.io.Serializable;

/\*\*

\*A class for the reading, writing CSV-files

\*from the array and serializing/deserializing objects

\* **@author** yozhik

\*/

@SuppressWarnings("serial")

**public** **class** CSVProcessor **implements** Serializable{

/\*\*

\* array for store strings of CSV-file

\*/

**private** ArrayList<String> lines=**new** ArrayList<String>();

/\*\*

\* array of columns from CSV file

\*/

**private** String[] columns;

/\*\*

\* array of main data from CSV file

\*/

**private** Integer[][] mainData;

/\*\*

\* A method for the reading CSV-files

\* **@param** nameOfFile

\* **@throws** IOException

\*/

**public** **void** readFromFile(String nameOfFile) **throws** FileNotFoundException,IOException{

BufferedReader file=**new** BufferedReader(**new** FileReader(nameOfFile));

String line="";

**while** ((line=file.readLine())!=**null**){

lines.add(line);

}

**if** (file!=**null**)

file.close();

}

/\*\*

\* The method for the writing CSV-files

\* from the array

\* **@param** nameOfFile

\* **@throws** IOException

\*/

**public** **void** writeToFile(String nameOfFile)**throws** FileNotFoundException,IOException{

BufferedWriter file=**new** BufferedWriter(**new** FileWriter(nameOfFile));

**for**(**int** i=0; i<lines.size();i++){

file.write(lines.get(i)+"\n");

}

**if**(file!=**null**){

file.close();

}

}

/\*\*

\* Method for the serializing object to the file

\* **@param** nameOfFile

\* **@throws** IOException

\*/

**public** **void** serializeObject(String nameOfFile) **throws** FileNotFoundException,IOException{

ObjectOutputStream object=**new** ObjectOutputStream(**new** FileOutputStream(nameOfFile));

object.writeObject(lines);

**if**(object!=**null**)

object.close();

}

/\*\*

\* The method for the deserializing object from the file

\* **@param** nameOfFile

\* **@throws** IOException, ClassNotFoundException

\*/

@SuppressWarnings("unchecked")

**public** **void** deserializeObject(String nameOfFile)**throws** FileNotFoundException, IOException, ClassNotFoundException{

ObjectInputStream file=**new** ObjectInputStream(**new** FileInputStream(nameOfFile));

ArrayList<String> object=(ArrayList<String>)file.readObject();

**for**( **int** i=0; i<object.size();i++){

lines.add(object.get(i));

}

**if**(file!=**null**)

file.close();

}

/\*\*

\* method for the parsing data form CSVFile's lines

\* to the double array 'data'

\*/

**public** **void** parse() **throws** CSVParseException{

/\*\*

\* separator in CSV-file's lines

\*/

String[][] data;

String separator=";";

String[] headlines=lines.get(0).split(separator);

data=**new** String[lines.size()][headlines.length];

data[0]=headlines;

**for**(**int** i=1; i<lines.size(); i++){

data[i]=lines.get(i).split(separator);

**if** ( (data[i].length)!=(data[i-1].length))

**throw** **new** CSVParseException("Wrong format of CSV-file!");

}

columns = **new** String[data[0].length];

**for**(**int** i=0; i<data[0].length; i++){

**this**.columns[i]=data[0][i];

}

mainData=**new** Integer[data.length-1][data[0].length];

**for**(**int** i=0; i<lines.size()-1; i++){

**for** (**int** j=0; j<data[0].length; j++){

mainData[i][j]=Integer.*parseInt*(data[i+1][j]);

}

}

}

/\*\*

\* Receiving array header table

\* **@return** array of table's columns (clone)

\*/

**public** String[] getColumns(){

**return** columns.clone();

}

/\*\*

\* Receiving the main data table

\* **@return** double array of data (clone)

\*/

**public** Integer[][] getData(){

**return** mainData.clone();

}

}

**Класс TestIO**

**package** com.lab111.lab6;

**import** java.io.BufferedReader;

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

/\*\*

\* Class implements the user interaction and

\* class CSVProcessor. All exceptions are

\* processed in this class.

\* **@author** yozhik

\*/

**public** **class** TestIO{

/\*\*

\* Main class

\* **@param** args

\* **@throws** IOException

\* **@throws** ClassNotFoundException

\*/

**public** **static** **void** main (String[] args) **throws** FileNotFoundException,IOException, ClassNotFoundException, CSVParseException{

CSVProcessor csv=**null**;

File file=**new** File("serialized.dat");

**if** (file.exists()){

csv=**new** CSVProcessor();

System.*out*.print("Deserializing file 'serialized.dat' ... ");

**try**{

csv.deserializeObject("serialized.dat");

}**catch**(FileNotFoundException e1){

System.*out*.println("FileNotFoundException!!! The object was not deserialized!!!");

**throw** e1;

}

**catch**(ClassNotFoundException e2){

System.*out*.println("ClassNotFoundException!!! The object was not deserialized!!");

**throw** e2;

}

**catch**(IOException e3){

**throw** e3;

}

System.*out*.println("Done!");

}

**else**{

BufferedReader consoleIn=**new** BufferedReader(**new** InputStreamReader(System.*in*));

**int** counter=0;

**boolean** done=**true**;

**while**(done){

**try**{

csv=**new** CSVProcessor();

System.*out*.println("Enter name of CSV-file:");

String nameOfFile = consoleIn.readLine();

System.*out*.print("Reading data from file '"+nameOfFile+" ... ");

csv.readFromFile(nameOfFile);

System.*out*.println("Done!");

System.*out*.print("Serializing of array to the file 'serialized.dat' ... ");

csv.serializeObject("serialized.dat");

System.*out*.println("Done!");

done=**false**;

}**catch**(FileNotFoundException e1){

System.*out*.println("Failed!");

System.*out*.println("Wrong name of file! Please try again");

counter++;

**if** (counter==3)

**throw** e1;

}

**catch**(IOException e2){

**throw** e2;

}

}

}

**if** (csv!=**null**){

}

// parsing lines from CSV-file

**final** CSVProcessor parseCSV=csv;

Runnable parseRun = **new** Runnable() {

@Override

**public** **void** run() {

**try** {

System.*out*.print("Parsing the array ... ");

parseCSV.parse();

System.*out*.println("Done!");

}**catch** (CSVParseException e) {

System.*out*.println("Failed!");

System.*out*.println("Wrong format of CSV-file. Please, point right CSV-file");

}

}

};

Thread parseThread = **new** Thread(parseRun);

parseThread.start();

**try** {

parseThread.join();

} **catch** (InterruptedException e) {

System.*out*.println("Interrupted!");

}

DiagramDrawer diagramDrawer=**new** DiagramDrawer();

diagramDrawer.draw(csv.getColumns(),csv.getData());

}

}

**Клас CSVParseException**

**package com.lab111.Lab3;**

/\*\*

\* Signals that a data format error has occurred.

\* **@author** yozhik

\*/

@SuppressWarnings("serial")

**public** **class** CSVParseException **extends** Exception{

CSVParseException(){

**super**();

}

CSVParseException(String massage){

**super**(massage);

}

}

**Клас TestDraw**

**package** com.lab111.lab6;

/\*\*

\* main class for the

\* drawing GUI

\* **@author** yozhik

\*/

**public** **class** TestDraw {

/\*\*

\* Main method for the drawing GUI

\* **@param** args

\*/

**public** **static** **void** main (String[] args){

TestIO testCSVfile =**new** TestIO();

}

}

**Клас MenuBar**

**package** com.lab111.lab6;

**import** java.awt.event.ActionEvent;

**import** javax.swing.AbstractAction;

**import** javax.swing.ImageIcon;

**import** javax.swing.JMenu;

**import** javax.swing.JMenuBar;

**import** javax.swing.JMenuItem;

/\*\*

\* Class for the creating main menu bar of frame

\* **@author** yozhik

\*/

@SuppressWarnings("serial")

**public** **class** MenuBar **extends** JMenuBar{

/\*\*

\* menu bar

\*/

**private** JMenuBar menuBar = **new** JMenuBar();

/\*\*

\* create and add standart menu bar elements to the frame

\* **@return** menu bar

\*/

**public** JMenuBar addMenuBar(){

JMenu fileMenu = **new** JMenu("File") ;

JMenuItem openItem = **new** JMenuItem("Open");//548

JMenuItem closeItem = **new** JMenuItem("Close");

JMenuItem saveItem = **new** JMenuItem("Save",**new** ImageIcon("save\_icon.jpg"));

JMenuItem saveAsItem = **new** JMenuItem("Save as...");

AbstractAction exitAction = **new** AbstractAction("Exit") // Пункт меню "Exit".

{

**public** **void** actionPerformed(ActionEvent event)

{

// Фрагмент программы, реализующий действие.

System.*exit*(0);

}

};

JMenuItem exitItem = fileMenu.add(exitAction);

exitItem.setToolTipText("Exit") ;

fileMenu.add(openItem);

fileMenu.add(saveItem);

fileMenu.add(saveAsItem);

fileMenu.addSeparator();

fileMenu.add(closeItem);

fileMenu.addSeparator();

fileMenu.add(exitItem);

JMenu editMenu = **new** JMenu("Edit") ;

JMenuItem cutItem = **new** JMenuItem("Cut");

JMenuItem copyItem = **new** JMenuItem("Copy");

JMenuItem pasteItem = **new** JMenuItem("Paste");

editMenu.add(cutItem);

editMenu.add(copyItem);

editMenu.add(pasteItem);

JMenu helpMenu = **new** JMenu("Help") ;

JMenuItem helpItem = **new** JMenuItem("Help Contents");

JMenuItem aboutItem = **new** JMenuItem("About");//стор. 526, 538!!!

helpMenu.add(helpItem);

helpMenu.addSeparator();

helpMenu.add(aboutItem);

menuBar.add(fileMenu);

menuBar.add(editMenu);

menuBar.add(helpMenu);

**return** menuBar;

}

/\*\*

\* add new menu to the menu bar

\* **@param** menu - new menu

\*/

**public** **void** addMenu(JMenu menu){

menuBar.add(menu);

}

}

**Клас TablePanel**

**package** com.lab111.lab6;

**import** java.awt.BorderLayout;

**import** javax.swing.BorderFactory;

**import** javax.swing.JPanel;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTable;

**import** javax.swing.border.Border;

/\*\*

\* Class for the creating panel with the table

\* **@author** yozhik

\*/

@SuppressWarnings("serial")

**public** **class** TablePanel **extends** JPanel{

/\*\*

\* columns of table

\*/

**private** String[] columns;

/\*\*

\* main data of table

\*/

**private** Integer[][] data;

/\*\*

\* constructor for initialization table

\* **@param** columns

\* **@param** data

\*/

**public** TablePanel(String[] columns, Integer[][] data){

**this**.columns=**new** String[columns.length];

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

**this**.columns[i]=columns[i];

}

**this**.data = **new** Integer[data.length][data[0].length];

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

**for** (**int** j=0; j<data[0].length; j++){

**this**.data[i][j]=data[i][j];

}

}

}

/\*\*

\* draw table with data from CSV file

\* **@return** panel with table

\*/

**public** JPanel drawTable(){

JTable table=**new** JTable(data,columns);

add(**new** JScrollPane(table),BorderLayout.*CENTER*);

Border etched = BorderFactory.*createLoweredBevelBorder*();;

Border titled = BorderFactory.*createTitledBorder*(etched, "Table Form");

setBorder(titled);

**return** **this**;

}

}

**Клас ChartPanel**

**package** com.lab111.lab6;

/\*\*

\* Class for the creating panel with buble chart

\*/

**import** java.awt.BasicStroke;

**import** java.awt.Color;

**import** java.awt.GradientPaint;

**import** java.awt.Graphics;

**import** java.awt.Graphics2D;

**import** java.awt.Insets;

**import** java.awt.geom.Ellipse2D;

**import** java.awt.geom.Line2D;

**import** java.awt.geom.Point2D;

**import** javax.swing.BorderFactory;

**import** javax.swing.JPanel;

**import** javax.swing.border.Border;

@SuppressWarnings("serial")

**public** **class** ChartPanel **extends** JPanel{

/\*\*

\* main data for chart in integer format

\*/

**private** **int**[][] data;

/\*\*

\* labels of axis

\*/

**private** String[] columns;

/\*\*

\* array of bubbles for the chart with all parameters(coordinates)

\*/

**private** Ellipse2D[] bubbles;

/\*\*

\* constructor for the initialization fields of class

\* and creating borders of this chart panel

\* **@param** columns

\* **@param** data

\*/

**public** ChartPanel(String[] columns,Integer[][] data){

**this**.data=**new** **int**[data.length][data[0].length];

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

**for** (**int** j=0; j<data[0].length; j++){

**this**.data[i][j]=(**int**)data[i][j];

}

}

**this**.columns=**new** String[columns.length];

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

**this**.columns[i]=columns[i];

}

Border etched = BorderFactory.*createLoweredBevelBorder*();

Border titled = BorderFactory.*createTitledBorder*(etched, "Buble Chart");

setBorder(titled);

}

/\*\*

\* draw bubble chart on panel appropriately

\* to the incoming tabular data

\* **@param** Graphics

\*/

**public** **void** paintComponent(Graphics g){

**super**.paintComponents(g);

Graphics2D g2=(Graphics2D)g;

Insets ins = getInsets();

**int** height = getHeight();

**int** width = getWidth();

**int** height\_const=height/10;

**int** width\_const=width/10;

Point2D left\_top\_point=**new** Point2D.Float(ins.left+width\_const,ins.top+height\_const);

Point2D left\_bottom\_point=**new** Point2D.Float(ins.left+width\_const,height-ins.bottom-height\_const);

Point2D right\_bottom\_point=**new** Point2D.Float(width-ins.right-width\_const,height-ins.bottom-height\_const);

Line2D x\_axis = **new** Line2D.Float(left\_bottom\_point,right\_bottom\_point);

Line2D y\_axis = **new** Line2D.Float(left\_top\_point,left\_bottom\_point);

g2.setStroke(**new** BasicStroke(2.0f));

g2.setPaint(**new** GradientPaint(left\_top\_point,Color.*CYAN*,right\_bottom\_point,Color.*GREEN*));

**int** x\_max = data[0][0] + (data[0][2]/2);

**int** y\_max = data[0][1] + (data[0][2]/2);

**int** x\_min = data[0][0] + (data[0][2]/2);

**int** y\_min = data[0][1] + (data[0][2]/2);

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

**if** ( (data[i][0] + (data[i][2]/2) ) > x\_max)

x\_max = data[i][0] + (data[i][2]/2);

**if** ( (data[i][1] + (data[i][2]/2) ) > y\_max )

y\_max = data[i][1] + (data[i][2]/2);

**if** ( (data[i][0] + (data[i][2]/2) ) < x\_min)

x\_min = data[i][0] - (data[i][2]/2);

**if** ( (data[i][0] + (data[i][2]/2) ) < y\_min)

y\_min = data[i][1] - (data[i][2]/2);

}

**int** x\_max\_on\_axis=x\_max+10;

**int** y\_max\_on\_axis=y\_max+10;

**int** x\_min\_on\_axis=x\_min-10;

**int** y\_min\_on\_axis=y\_min-10;

System.*out*.println("xmx= "+x\_max\_on\_axis+" ymx= "+y\_max\_on\_axis+" xmn= "+x\_min\_on\_axis+" ymn= "+y\_min\_on\_axis);

**double** kx = (**double**)(right\_bottom\_point.getX() - left\_bottom\_point.getX())/(x\_max - x\_min);

**double** ky = (**double**)(left\_bottom\_point.getY() - left\_top\_point.getY())/(y\_max - y\_min);

bubbles=**new** Ellipse2D[data.length];

**for** (**int** i=0; i<**this**.bubbles.length;i++){

bubbles[i]=**new** Ellipse2D.Double(

(left\_bottom\_point.getX()+kx\*Math.*abs*(x\_min)+kx\*data[i][0]-**this**.data[i][2]/2),

(left\_bottom\_point.getY()-ky\*Math.*abs*(y\_min)-ky\*data[i][1]-**this**.data[i][2]/2),

**this**.data[i][2],**this**.data[i][2]);

}

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

g2.fill(bubbles[i]);

}

g2.setPaint(Color.*BLACK*);

g2.draw(x\_axis);

g2.draw(y\_axis);

**int** x\_step = (**int**)(right\_bottom\_point.getX()-left\_bottom\_point.getX())/10;

**int** y\_step = (**int**)(left\_bottom\_point.getY()-left\_top\_point.getY())/10;

**int** next\_x = (**int**)left\_bottom\_point.getX();

**int** next\_y = (**int**)left\_bottom\_point.getY();

**for** (**int** i=1; i<10; i++){

next\_x+=x\_step;

next\_y-=y\_step;

Line2D x\_stroke = **new** Line2D.Float(next\_x,(**int**)left\_bottom\_point.getY()-2,

next\_x ,(**int**)left\_bottom\_point.getY()+2);

g2.draw(x\_stroke);

g2.drawString(Integer.*toString*(x\_min+i\*(x\_max-x\_min)/10 ),

next\_x, (**int**)left\_bottom\_point.getY()+15);

Line2D y\_stroke = **new** Line2D.Float((**int**)left\_bottom\_point.getX()-2,next\_y,

(**int**)left\_bottom\_point.getX()+2 ,next\_y);

g2.draw(y\_stroke);

g2.drawString(Integer.*toString*(y\_min+i\*( y\_max-y\_min)/10 ),

(**int**)left\_bottom\_point.getX()-30, next\_y) ;

}

next\_x+=x\_step;

next\_y-=y\_step;

g2.drawString(**this**.columns[0],

next\_x, (**int**)left\_bottom\_point.getY()+15);

g2.drawString(**this**.columns[1],

(**int**)left\_bottom\_point.getX()-30, next\_y) ;

}

}

**Клас MainFrame**

**package** com.lab111.lab6;

**import** java.awt.Dimension;

**import** java.awt.Image;

**import** java.awt.Toolkit;

**import** javax.swing.JFrame;

/\*\*

\* Class for the creating frame with

\* menu bar, table and bubble chart

\* **@author** yozhik

\*/

@SuppressWarnings("serial")

**public** **class** MainFrame **extends** JFrame{

/\*\*

\* constructor for the creating

\* frame with parameters which depends on

\* display options

\*/

**public** MainFrame(){

Toolkit kit = Toolkit.*getDefaultToolkit*() ;

Dimension screenSize = kit.getScreenSize() ;

**int** screenWidth = screenSize.width;

**int** screenHeight = screenSize.height ;

Image img = kit.getImage("ddd.jpg") ;

setIconImage(img);

setTitle("CSV\_file - > Table -> Buble Chart");

setBounds(screenWidth/8, screenHeight/8,screenWidth\*3/4,screenHeight\*3/4);

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

}

}

**Клас DiagramDrawer**

**package** com.lab111.lab6;

**import** java.awt.BorderLayout;

/\*\*

\* Class for the drawing all elements of

\* GUI (frame and its elements)

\* **@author** yozhik

\*/

**public** **class** DiagramDrawer{

/\*\*

\* draw frame and add all panels to frame

\* **@param** columns

\* **@param** data

\*/

**public** **void** draw(String[] columns,Integer[][] data){

MainFrame mainFrame=**new** MainFrame();

MenuBar menuBar=**new** MenuBar();

mainFrame.setJMenuBar(menuBar.addMenuBar()) ;

TablePanel tablePanel = **new** TablePanel(columns,data);

mainFrame.add(tablePanel.drawTable(),BorderLayout.*WEST*);

ChartPanel chartPanel = **new** ChartPanel(columns,data);

mainFrame.getContentPane().add(chartPanel,BorderLayout.*CENTER*);

mainFrame.setVisible(**true**);

}

}

**Таблиця відповідності елементів дизайну та класів з бібліотеки SWING**

|  |  |
| --- | --- |
| MainFrame | javax.swing.JFrame |
| MenuBar | javax.swing.JMenuBar |
| Меню: "File" , "Open", "Close", "Save", "Save as...", "Exit", "Edit", "Cut", "Copy", "Paste", "Help", "Help Contents", "About", | javax.swing.JMenu; |
| TablePanel | javax.swing.JTable; |
| ChartPanel | java.awt.Graphics2D; |
| Border’s | javax.swing.BorderFactory |