CLASE ARCHIVO

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package pspprogram1\_lixander\_felipe\_cadavid\_betancur;

import java.awt.\*;

import java.io.\*;

/\*\*

\*

\* @author lixanderCadavid

\*/

public class Archivo {

/\*\*

\* Este metodo obtiene la ruta del archivo que se desea cargar al programa.

\* @return Retorna una variable de tipo String con la ruta del archivo especificado.

\*/

public String obtenerRuta()

{

javax.swing.JFileChooser j= new javax.swing.JFileChooser();

j.showOpenDialog(j);

try

{

String path = j.getSelectedFile().getAbsolutePath();

return path;

}catch(NullPointerException e)

{

javax.swing.JOptionPane.showMessageDialog(j, "Ha finalizado el programa");

System.exit(0);

}

return null;

}

/\*\*

\* Este método retorna lo que contiene el archivo de texto .txt en una estructura String.

\* @param rutaArchivo Es la ruta del archivo con el cual se va a trabajar, esta variable es de tipo String.

\* @return Retorna el texto almacenado en el .txt como un String.

\*/

public String realizarCalculos(String rutaArchivo)

{

String lectura = "";

File f = new File(rutaArchivo);

try{

FileReader fr = new FileReader(f);

BufferedReader br = new BufferedReader(fr);

String aux;

while((aux = br.readLine())!=null)

{

lectura = lectura + aux;

}

}catch(IOException e){

}

return lectura;

}

/\*\*

\* Este método carga el archivo Ayuda.pdf, el cual contiene las instrucciones para seguir el programa.

\*/

public void abrirAyuda()

{

try {

File path = new File ("Ayuda.pdf");

Desktop.getDesktop().open(path);

}catch (IOException ex) {

ex.printStackTrace();

}

}

/\*\*

\* Este método verifica si el archivo especificado si sea .txt

\* @param ruta Es una variable de tipo String, la cual contiene la ruta del archivo especificado.

\* @return Retorna booleano verdadero si existe en el String mayor, y booleano falso si el String no tiene.

\*/

public boolean comprobarArchivo(String ruta)

{

boolean respuesta;

respuesta = ruta.contains(".txt");

return respuesta;

}

}

CLASE PSPProgram1\_Lixander\_Felipe\_Cadavid\_Betancur

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package pspprogram1\_lixander\_felipe\_cadavid\_betancur;

/\*\*

\*

\* @author lixanderCadavid

\*/

public class PSPProgram1\_Lixander\_Felipe\_Cadavid\_Betancur {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

FramePrincipal objetoFramePrincipal = new FramePrincipal();

objetoFramePrincipal.show();

}

}

CLASE ListaSimpleCircularCabeza

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package pspprogram1\_lixander\_felipe\_cadavid\_betancur;

import javax.swing.JOptionPane;

/\*\*

\*

\* @author Lixandercadavid

\*/

public class ListaSimpleCircularCabeza {

Nodo primero;

Nodo ultimo;

ListaSimpleCircularCabeza otraLista;

public ListaSimpleCircularCabeza(){

primero = ultimo = new Nodo();

primero.AsignaSiguiente(primero);

}

public Nodo primero(){

return(primero.Retorne\_Sig());

}

public Nodo ultimo(){

return(ultimo);

}

public Nodo cabeza(){

return(primero);

}

public boolean EsVacia(){

if(primero.Retorne\_Sig() == primero){

return(true);

}

else{

return(false);

}

}

/\*\*

\* El método LlenarLista llena una lista de tipo ListaSimpleCircularCabeza con los datos de un arreglo de String.

\* @param arregloDatos Es un arreglo de String que contiene los datos a ser procesados.

\* @return Retorna un booleano como falso en caso de error al llenar la lista, o verdadero si se llenó correctamente.

\*/

public boolean LlenarLista(String[] arregloDatos)

{

boolean completado = false;

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

{

Nodo nuevo = new Nodo(arregloDatos[i]);

insertarFinal(nuevo);

}

Nodo x = primero();

return completado;

}

/\*\*

\* Este método inserta un nuevo nodo que es una estructura tipo Nodo al final de la lista que es de tipo ListaSimpleCircularCabeza.

\* @param nuevoNodo Es una variable de tipo Nodo que contiene una liga y un String.

\* @return Retorna un booleano como falso si no se insertó el nuevo nodo en la lista o verdadero en caso de ser insertado correctamente.

\*/

public boolean insertarFinal(Nodo nuevoNodo){

Nodo x = primero();

String nombre = nuevoNodo.retornaDato();

boolean resp = false;

if(EsVacia())

{

primero.AsignaSiguiente(nuevoNodo);

nuevoNodo.AsignaSiguiente(primero);

ultimo = nuevoNodo;

resp = true;

}

else

{

while(x != cabeza())

{

if(x.retornaDato().equals(nombre))

{

return false;

}

else

{

x = x.Retorne\_Sig();

resp = true;

}

}

if(resp)

{

ultimo.AsignaSiguiente(nuevoNodo);

nuevoNodo.AsignaSiguiente(primero);

ultimo = nuevoNodo;

}

}

return resp;

}

/\*\*

\* Este método retorna la cantidad de datos almacenados en la lista.

\* @return Retorna la cantidad de datos como tipo int.

\*/

public int cantidadDatos(){

int cantidad = 0;

Nodo x = primero();

while(x != cabeza())

{

cantidad++;

x = x.Retorne\_Sig();

}

return cantidad;

}

}

CLASE NODO

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package pspprogram1\_lixander\_felipe\_cadavid\_betancur;

/\*\*

\*

\* @author Lixandercadavid

\*/

public class Nodo {

private Nodo sig;

private String nombre = new String();

Nodo()

{

sig = null;

}

Nodo(String n)

{

nombre = n;

sig = null;

}

public void asignaDato(String n){

nombre = n;

}

public String retornaDato()

{

return nombre;

}

public void AsignaSiguiente(Nodo x){

sig = x;

}

Nodo Retorne\_Sig(){

return sig;

}

}

CLASE FRAMEPRINCIPAL

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package pspprogram1\_lixander\_felipe\_cadavid\_betancur;

import javax.swing.JOptionPane;

/\*\*

\*

\* @author lixanderCadavid

\*/

public class FramePrincipal extends javax.swing.JFrame {

/\*\*

\* Creates new form FramePrincipal

\*/

public FramePrincipal() {

initComponents();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jButton1 = new javax.swing.JButton();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

jTextField1 = new javax.swing.JTextField();

jTextField2 = new javax.swing.JTextField();

jLabel3 = new javax.swing.JLabel();

jMenuBar1 = new javax.swing.JMenuBar();

jMenu1 = new javax.swing.JMenu();

jMenuItem1 = new javax.swing.JMenuItem();

jMenuItem2 = new javax.swing.JMenuItem();

jMenu2 = new javax.swing.JMenu();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setTitle("Calcular Promedio y Desviación Estándar");

jButton1.setText("Calcular Datos");

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

jLabel1.setText("El promedio de sus datos es: ");

jLabel2.setText("La desviación estándar de sus datos es:");

jLabel3.setText("Programa Seleccionado");

jMenu1.setText("Archivo");

jMenuItem1.setText("Nuevo Archivo");

jMenuItem1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jMenuItem1ActionPerformed(evt);

}

});

jMenu1.add(jMenuItem1);

jMenuItem2.setText("Salir");

jMenuItem2.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jMenuItem2ActionPerformed(evt);

}

});

jMenu1.add(jMenuItem2);

jMenuBar1.add(jMenu1);

jMenu2.setText("Ayuda");

jMenu2.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jMenu2MouseClicked(evt);

}

});

jMenuBar1.add(jMenu2);

setJMenuBar(jMenuBar1);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel1)

.addGap(18, 18, 18)

.addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, 101, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED\_SIZE, 101, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createSequentialGroup()

.addGap(64, 64, 64)

.addComponent(jButton1))

.addComponent(jLabel3))

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel3)

.addGap(19, 19, 19)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 30, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel2)

.addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(26, 26, 26)

.addComponent(jButton1)

.addContainerGap(43, Short.MAX\_VALUE))

);

pack();

}// </editor-fold>

private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {

Archivo a = new Archivo();

boolean activarBoton = false;

String ruta;

ruta = a.obtenerRuta();

jLabel3.setText(ruta);

this.setSize(this.getWidth() + jLabel3.getWidth(), this.getHeight());

activarBoton = a.comprobarArchivo(ruta);

if(!activarBoton)

{

jButton1.setEnabled(activarBoton);

JOptionPane.showMessageDialog(null, "Cargue un archivo .txt válido.");

}

else

{

jButton1.setEnabled(activarBoton);

}

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

Archivo a = new Archivo();

String []datosVector;

String lectura;

String desviacion, promedio;

lectura = a.realizarCalculos(jLabel3.getText());

datosVector = lectura.split(",");

Operaciones objetoOperaciones = new Operaciones();

ListaSimpleCircularCabeza listaDatos = new ListaSimpleCircularCabeza();

listaDatos.LlenarLista(datosVector);

promedio = objetoOperaciones.calcularPromedioDosDecimales(listaDatos);

desviacion = objetoOperaciones.calcularDesviacionEstandar(listaDatos);

jTextField1.setText(promedio.toString());

jTextField2.setText(desviacion);

}

private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {

Integer confirmacionSalida;

confirmacionSalida = JOptionPane.showConfirmDialog(null, "¿Desea finalizar el programa?");

if(confirmacionSalida == JOptionPane.YES\_OPTION)

{

System.exit(0);

}

}

private void jMenu2MouseClicked(java.awt.event.MouseEvent evt) {

Archivo a = new Archivo();

a.abrirAyuda();

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(FramePrincipal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(FramePrincipal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(FramePrincipal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(FramePrincipal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new FramePrincipal().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JMenu jMenu1;

private javax.swing.JMenu jMenu2;

private javax.swing.JMenuBar jMenuBar1;

private javax.swing.JMenuItem jMenuItem1;

private javax.swing.JMenuItem jMenuItem2;

private javax.swing.JTextField jTextField1;

private javax.swing.JTextField jTextField2;

// End of variables declaration

}

CLASE OPERACIONES

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package pspprogram1\_lixander\_felipe\_cadavid\_betancur;

import javax.swing.JOptionPane;

import java.math.\*;

import java.text.DecimalFormat;

/\*\*

\*

\* @author Usuario

\*/

public class Operaciones {

/\*\*

\* El método lee la lista listaDatos, invoca al método calcularPromedio con parámetro listaDatos e imprime el promedio con el formato decimal "0.00".

\* @param listaDatos Es el listado de los datos que van a ser procesados, se almacena en una estructura de tipo ListaSimpleCircularCabeza.

\* @return Este método retorna el promedio como una estructura String.

\*/

public String calcularPromedioDosDecimales(ListaSimpleCircularCabeza listaDatos)

{

String prom;

DecimalFormat df = new DecimalFormat("0.00");

Double promedio = calcularPromedio(listaDatos);

prom = df.format(promedio);

return prom;

}

/\*\*

\* El método lee los datos de la lista listaDatos y calcula el promedio de los datos.

\* @param listaDatos Es el listado de los datos que van a ser procesados, se almacena en una estructura de tipo ListaSimpleCircularCabeza.

\* @return Este método retorna el promedio de los datos de la lista en una estructura Double.

\*/

public Double calcularPromedio(ListaSimpleCircularCabeza listaDatos)

{

Double promedio, acumulador = 0.0;

Integer cantidadDatos = listaDatos.cantidadDatos();

Nodo x = listaDatos.primero();

while(x != listaDatos.cabeza())

{

acumulador = acumulador + Double.parseDouble(x.retornaDato());

x = x.Retorne\_Sig();

}

promedio = acumulador / cantidadDatos;

return promedio;

}

/\*\*

\* El método calcularDesviacionEstandar lee los datos de la listaDatos para calcular la desviación estándar de estos.

\* @param listaDatos Es el listado de los datos que van a ser procesados, éste parámetro se representa con una estructura ListaSimpleCircularCabeza.

\* @return Este método retorna la desviación estándar de los datos ingresados representada en una estructura String.

\* @see Math Este método utiliza las funciones pow() y sqrt() de la librería Math de Java.

\*/

public String calcularDesviacionEstandar(ListaSimpleCircularCabeza listaDatos)

{

Double desviacionEstandar, acumuladorDivididoCantidad;

String respuesta;

Double promedio = calcularPromedio(listaDatos);

DecimalFormat df = new DecimalFormat("0.00");

Double acumulador = 0.0;

Integer cantidadDatos = listaDatos.cantidadDatos();

Nodo x = listaDatos.primero();

while(x != listaDatos.cabeza())

{

acumulador = acumulador + Math.pow((Double.parseDouble(x.retornaDato()) - promedio),2);

x = x.Retorne\_Sig();

}

acumuladorDivididoCantidad = acumulador/(cantidadDatos-1);

desviacionEstandar = Math.sqrt(acumuladorDivididoCantidad);

respuesta = df.format(desviacionEstandar);

return respuesta;

}

}