PRACTICA 2

Ordenamiento de Datos

Universidad de San Carlos de Guatemala
Facultad de ingeniería
Escuela de Ciencias y Sistemas
Laboratorio de Introducción a la Programación 1
Sección D
Aux. José Orlando Wannan Escobar
Manual Técnico

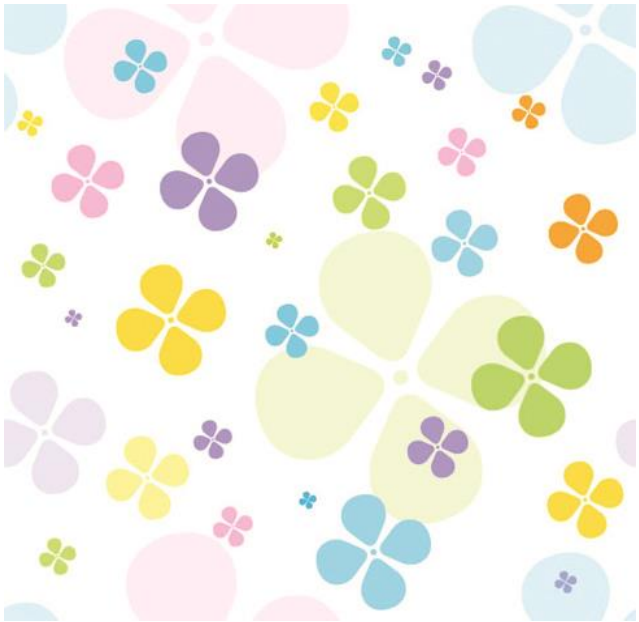
CONTENIDO

1. Introducción

En el presente manual se pretende explicar de forma detallada la lógica del programa, es decir sus métodos, clases, atributos, botones, entre otras cosas.

La funcionalidad del programa consiste en realizar un ordenamiento de datos mostrados en una gráfica por medio de uno de los ordenamientos vistos en clase, el utilizado en esta practica es el ordenamiento por inserción.

Al finalizar dicho ordenamiento se genera un reporte en formato .pdf y .html.



2. Objetivo

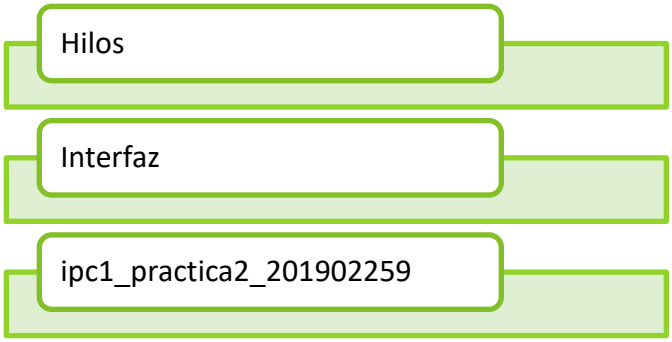
Permitir a personas, dentro del ámbito de programación, conocer la funcionalidad del proyecto mediante su código y estructura. De esta manera podrán observar e imaginar la lógica del programa a través del paradigma del autor.

REQUERIMIENTOS

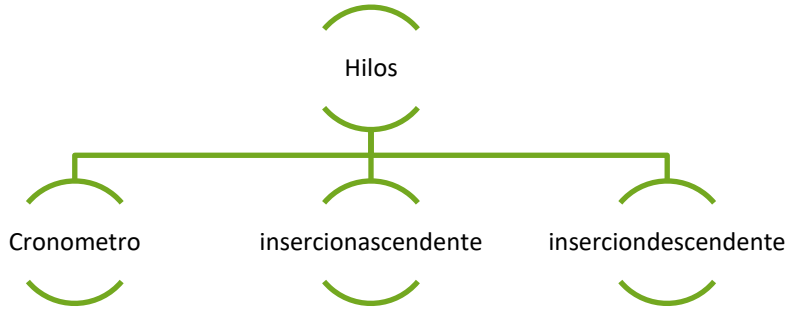
CPU:	Intel de 64 bits, Recomendado: Intel Core i3 o superior
RAM:	4GB, Recomendado: 8 GB o más
Espacio disponible en disco:	8GB
Sistema Operativo:	Windows 10
Resolución de Pantalla:	Básico: pantalla de 1280 x 800
Programas Adicionales	Apache Netbeans IDE 12.6

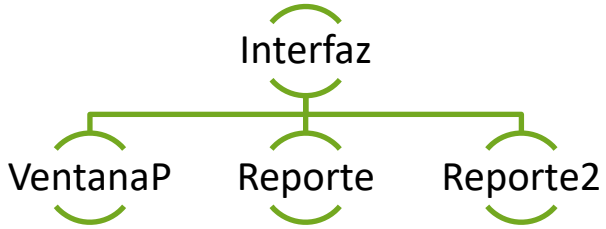
3. Estructura

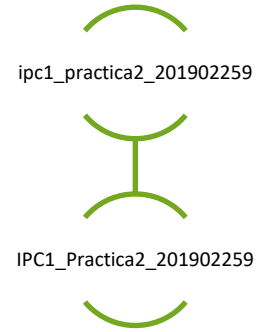
El programa se divide en los siguientes paquetes:



Cada paquete se subdivide en clases:







4. Paquete Hilos

4.1. Cronometro

En esta clase se encuentra la programación del hilo del cronometro, contiene la función run y la función parar.

4.1.1. Run

Dentro de la función run() se pueden encontrar diferentes sentencias, como la que permite contar los minutos como 60 segundos, es decir el cronometro no podrá contar 61 segundos. También se incluyó por estética una sentencia que permita poner un 0 antecedente de los segundos del 1 al 9.

```
public void run() {
    if (ban) {
        try {
            while (this.minutos != 100) {
                this.segundos = 0;
                while (this.segundos != 60) {
                    if (this.minutos < 10) {
                        if (this.segundos < 10) {
                            System.out.println("0" + this.minutos + ":0" + this.segundos);
                            cronom.setText("0" + this.minutos + ":0" + this.segundos);

                        } else {
                            System.out.println("0" + this.minutos + ":" + this.segundos);
                            cronom.setText("0" + this.minutos + ":" + this.segundos);

                        }
                    } else {

                        if (this.segundos < 10) {
                            System.out.println(this.minutos + ":0" + this.segundos);
                            cronom.setText( this.minutos + ":0" + this.segundos);

                        } else {
                            System.out.println(this.minutos + ":" + this.segundos);
                            cronom.setText(this.minutos + ":" + this.segundos);

                        }
                    }
                    this.segundos++;
```

4.2. insercionascendente

Dentro de esta clase se encuentra el ordenamiento por inserción de manera ascendente, adicional la función para la gráfica de barras, del mismo modo, ascendente.

```
public void Insercion(int[] A) throws InterruptedException {
    int aux;
    this.cr.start();
    for (int i = 1; i < A.length; i++) {
        aux = A[i];
        int j = i - 1;
        while ((j >= 0) && (aux < A[j])) {
            A[j + 1] = A[j];
            j--;
            /*this.panel.removeAll();
            grafica(A);
            this.panel.repaint();*/
            sleep(500);

        }
        A[j + 1] = aux;
        this.panel.removeAll();
        grafica(A);
        this.movimientos++;
        this.panel.repaint();
        cmov.setText(String.valueOf(this.movimientos));
    }
    parar();
}
```

```
public void grafica(int[] numeros) {
    //grafica de barras de prestamos
    String c = "";
    DefaultCategoryDataset datos2 = new DefaultCategoryDataset();
    for (int i = 0; i < numeros.length; i++) {
        datos2.addValue(numeros[i], String.valueOf(numeros[i]), "");
    }

    JFreeChart barras = ChartFactory.createBarChart("", "numeros", "", datos2, PlotOrientation.VERTICAL, true, true, false);
    barras.setBackgroundPaint(Color.PINK);
    ChartPanel p2 = new ChartPanel(barras);
    p2.setBounds(0, 0, 600, 350);
    p2.setVisible(true);
    this.panel.add(p2);

}
```

También se encuentra una función que permite ejecutar el hilo haciendo el llamado al algoritmo de ordenamiento y cuando dicho algoritmo pare genere el reporte.

```
public void run() {
    if (state) {
        try {
            Insercion(this.arreglo);
            this.cr.stop();
            Reporte r = new Reporte();
            r.reporte();
        } catch (InterruptedException ex) {

        }
    }

}
```

4.3. inserciondescendente

Al igual que la clase anterior, dentro de esta clase se encuentra el ordenamiento por inserción pero de manera descendente y la función para la gráfica de barras, del mismo modo, descendente. Adicional el algoritmo de generación del reporte.

```
public void Insercion(int[] A) throws InterruptedException {
    int aux;
    this.cr.start();
    for (int i = 1; i < A.length; i++) {
        aux = A[i];
        int j = i - 1;
        while ((j >= 0) && (aux > A[j])) {
            A[j + 1] = A[j];
            j--;
            /*this.panel.removeAll();
            grafica(A);
            this.panel.repaint();*/
            sleep(500);

        }
        A[j + 1] = aux;
        this.panel.removeAll();
        grafica(A);
        this.movimientos++;
        this.panel.repaint();
        cmov.setText(String.valueOf(this.movimientos));
    }
    parar();
}
```

```
public void run() {
    if (state) {
        try {
            Insercion(this.arreglo);
            this.cr.stop();
            Reporte2 r = new Reporte2();
            r.reporte2();
        } catch (InterruptedException ex) {

        }
    }

}
```

5. Paquete Interfaz

5.1. VentanaP

Dentro de esta clase se encuentra la programación de los elementos de la ventana principal. Se encuentran los botones, radiobutton, textfield…

```
JButton examinar, generarg, ordenar;
JRadioButton asc, desc, alg1, alg2, alg3;
ButtonGroup ascydesc, algoritmos;
JTextField titulo, ruta;
public JPanel graf;
public ChartPanel p2;
public static JLabel cronom, tiempo, mov, cmov;
```

Función para realizar la carga del archivo y obtener la ruta.

```
public void cargararchivo() {
    try {
        JFileChooser busc = new JFileChooser();
        int o = busc.showOpenDialog(this);
        if (o == JFileChooser.APPROVE_OPTION) {
            System.out.println(busc.getSelectedFile());
            archivo = busc.getSelectedFile();
            System.out.println(archivo);
            ruta.setEditable(true);
            ruta.setText(String.valueOf(busc.getSelectedFile()));
            ruta.setEditable(false);
        }

    } catch (Exception e) {
        System.out.println("Hubo un error :c");

    }
}
```

Función para almacenar los datos del archivo en un arreglo.

```
public void leerarchivo() {
    try {
        textcont = "";
        lector = new FileReader(archivo);
        buff = new BufferedReader(lector);
        String contline;
        while ((contline = buff.readLine()) != null) {
            textcont += contline;
        }
        System.out.println(textcont);
        JsonParser JSONValue = new JsonParser();
        Object objeto = JSONValue.parse(textcont);

        JsonObject ob = (JsonObject) objeto;
        String Titulo = ob.get("title").getAsString();
        titulo.setEditable(true);
        titulo.setText(Titulo);
        titulo.setEditable(false);
        Object datos = ob.get("dataset");
        JsonArray arreglo = (JsonArray) datos;
        numeros = new int[arreglo.size()];
        numerosd = new int[arreglo.size()];
        for (int i = 0; i < arreglo.size(); i++) {
            System.out.println("numero " + i + " : " + arreglo.get(i).getAsInt());
            numeros[i] = arreglo.get(i).getAsInt();
            numerosd[i] = arreglo.get(i).getAsInt();
        }
```

Funcionalidad de los botones

```
@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == examinar) {
        cargararchivo();
    } else if (e.getSource() == generarg) {
        leerarchivo();
        grafica();
        graf.repaint();
    } else if (e.getSource() == ordenar) {
        Cronometro c = new Cronometro();
        if (asc.isSelected() == true) {
            insercionascendente i = new insercionascendente(graf, numeros, c);
            i.start();
        }
        else if (desc.isSelected()==true) {
            insercionDescendente d = new insercionDescendente(graf, numeros, c);
            d.start();
        }
    }
```

5.2. Reporte

Esta clase fue creada para obtener el reporte del ordenamiento ascendente.

Función para generarlo en HTML

```
public void reporte() {
    String nombreReporte;
    File reporte;
    FileWriter fw;
    BufferedWriter br;
    String cadenaHTML;

    try {
        DateTimeFormatter dtf3 = DateTimeFormatter.ofPattern("yyyy_MM_dd_HH_mm_ss");
        String fecha = dtf3.format(LocalDateTime.now());
        nombreReporte = fecha + ".html";
        reporte = new File(nombreReporte);
        fw = new FileWriter(reporte);
        br = new BufferedWriter(fw);

        cadenaHTML = "<!doctype html>\n"
                + "<html lang=\"en\">\n"
                + "  <head>\n"
                + "    <!-- Required meta tags -->\n"
                + "    <meta charset=\"utf-8\">\n"
                + "    <meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">\n"
                + "\n"
                + "    <!-- Bootstrap CSS -->\n"
```

Función para generarlo en PDF

```
public void crearPdf(String contenido, String nombre) {
    try {

        Document document = new Document(PageSize.LETTER);
        PdfWriter.getInstance(document, new FileOutputStream(nombre + ".pdf"));

        document.open();
        document.addAuthor("Patty");
        document.addCreator("Patty");
        document.addSubject("Reporte de Grafica");
        document.addCreationDate();
        document.addTitle("Reporte de Grafica");

        HTMLWorker htmlWorker = new HTMLWorker(document);
        htmlWorker.parse(new StringReader(contenido));

        document.close();
        abrirarchivo(nombre + ".pdf");
        abrirarchivo(nombre + ".html");
    } catch (Exception e) {

    }
```

5.3. Reporte2

Esta clase fue creada para obtener el reporte del ordenamiento descendente, con la misma lógica que Reporte.

```
public class Reporte2 {
    public void reporte2() {
        String nombreReporte;
        File reporte2;
        FileWriter fw;
        BufferedWriter br;
        String cadenaHTML;

        try {
            DateTimeFormatter dtf3 = DateTimeFormatter.ofPattern("yyyy_MM_dd_HH_mm_ss");
            String fecha = dtf3.format(LocalDateTime.now());
            nombreReporte = fecha + ".html";
            reporte2 = new File(nombreReporte);
            fw = new FileWriter(reporte2);
            br = new BufferedWriter(fw);

            cadenaHTML = "<!doctype html>\n"
                    + "<html lang=\"en\">\n"
                    + "  <head>\n"
                    + "    <!-- Required meta tags -->\n"
                    + "    <meta charset=\"utf-8\">\n"
                    + "    <meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">\n"
                    + "\n"
                    + "    <!-- Bootstrap CSS -->\n"
```

6. Paquete ipc1_practica2_201902259

6.1. IPC1_Practica2_201902259

```java
package ipc1_practica2_201902259;

import Hilos.Cronometro;
import Interfaz.VentanaP;



public class IPC1_Practica2_201902259 {



    public static void main(String[] args) {
        new VentanaP();

    }

}
```