MANUAL TÉCNICO

Descripción de clases.

1. **CorrerPrograma:** Al ejecutar, lo único que hace es iniciar la ejecución desde la ventana extendida en VentanaPrincipal. Aquí empieza todo.

CorrerPrograma.java

```java
package com.mycompany.practica2codigo;

public class CorrerPrograma {

    public static void main(String[] args) {
        VentanaPrincipal m = new VentanaPrincipal();
        m.VentanaPrincipal();
    }

}
```

Invoca a la clase de VentanaMenuPrincipal.

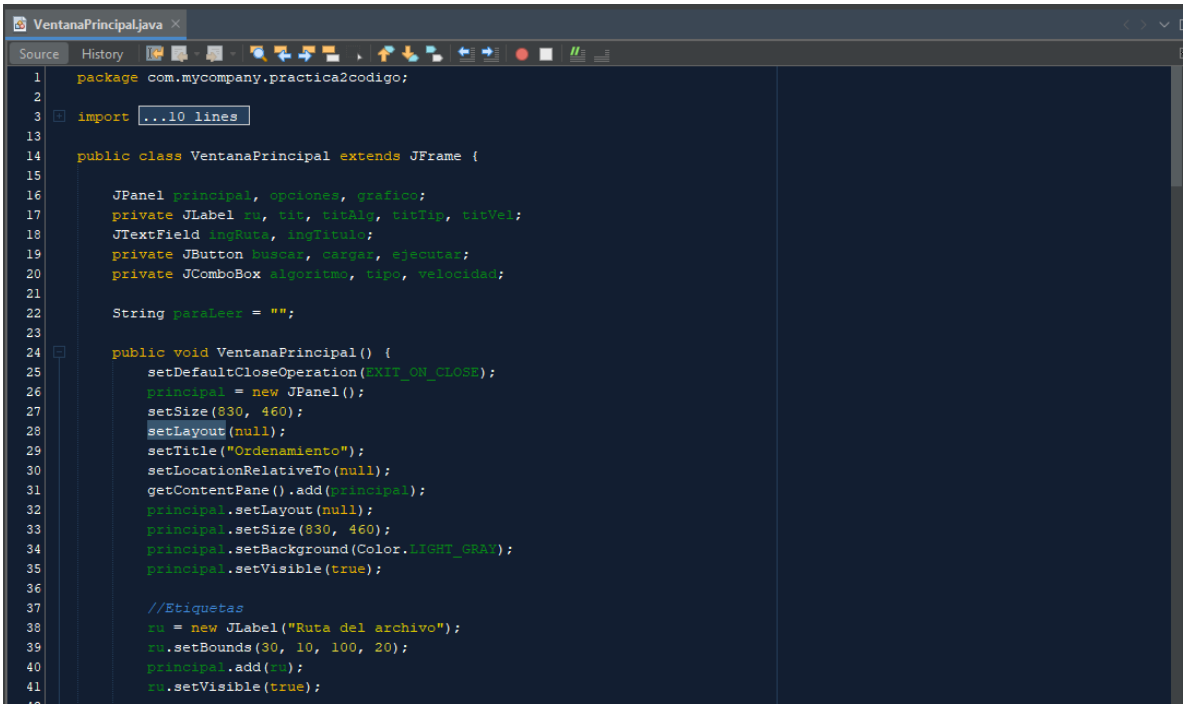
2. **Datos:** Con atributos y métodos para cada dato. Incluye los datos de cursos y notas para trabajar con el almacenamiento de los datos leídos, junto con sus **setters** y **getters**.

Datos.java

```java
package com.mycompany.practica2codigo;

public class Datos {
    private int valores;
    private String curso;

    public Datos (int valores, String curso) {
        this.setValores(valores);
        this.setCurso(curso);
    }

    public Datos () {
    }

    public Datos(String curso, int valores) {
        this.valores = valores;
        this.curso = curso;
    }

    public Datos(String curso, String valores) {
        this.valores = Integer.parseInt(valores);
        this.curso = curso;
    }

    public int getValores() {
        return valores;
    }

    public void setValores(int valores) {
        this.valores = valores;
    }

    public String getCurso() {
        return curso;
    }

    public void setCurso(String curso) {
        this.curso = curso;
    }

}
```

3. **VentanaMenuPrincipal:** Contiene el JFrame principal y sus características. Incluye la selección de archivo y selección de características para visualizar el mismo. Tiene las barras donde al seleccionar un archivo se divide el texto de la ruta del archivo en título con extensión de archivo y el resto de la ruta, se muestran separados en dos JTextFields al

momento de presionar click en aceptar en el cuadro del JFileChooser. En ésta misma se envía el archivo (ruta) a la clase de lectura llamada LeerCSV que será descrita más adelante.




```java
package com.mycompany.practica2codigo;

import ...10 lines

public class VentanaPrincipal extends JFrame {

    JPanel principal, opciones, grafico;
    private JLabel ru, tit, titAlg, titTip, titVel;
    JTextField ingRuta, ingTitulo;
    private JButton buscar, cargar, ejecutar;
    private JComboBox algoritmo, tipo, velocidad;

    String paraLeer = "";

    public void VentanaPrincipal() {
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        principal = new JPanel();
        setSize(830, 460);
        setLayout(null);
        setTitle("Ordenamiento");
        setLocationRelativeTo(null);
        getContentPane().add(principal);
        principal.setLayout(null);
        principal.setSize(830, 460);
        principal.setBackground(Color.LIGHT_GRAY);
        principal.setVisible(true);

        //Etiquetas
        ru = new JLabel("Ruta del archivo");
        ru.setBounds(30, 10, 100, 20);
        principal.add(ru);
        ru.setVisible(true);
```

```java
        tit = new JLabel("Titulo de la gráfica");
        tit.setBounds(30, 60, 150, 20);
        principal.add(tit);
        tit.setVisible(true);

        //TextFields
        ingRuta = new JTextField();
        ingRuta.setBounds(30, 30, 650, 20);
        principal.add(ingRuta);
        ingRuta.setVisible(true);

        ingTitulo = new JTextField();
        ingTitulo.setBounds(30, 80, 650, 20);
        principal.add(ingTitulo);
        ingTitulo.setVisible(true);

        //Panel de Opciones
        opciones = new JPanel();
        opciones.setLayout(null);
        opciones.setBounds(30, 130, 250, 270);
        Border bordejpanel = new TitledBorder(new EtchedBorder(), "Panel de Opciones");
        opciones.setBorder(bordejpanel);
        opciones.setBackground(Color.LIGHT_GRAY);
        opciones.setVisible(true);
        principal.add(opciones);

        //Panel Gráfico
        grafico = new JPanel();
        grafico.setLayout(null);
        grafico.setBounds(305, 130, 480, 270);
        Border bordeg = new TitledBorder(new EtchedBorder(), "Gráfica");
        grafico.setBorder(bordeg);
```

```
        grafico.setBackground(Color.LIGHT_GRAY);
        grafico.setVisible(true);
        principal.add(grafico);

        //Label
        titAlg = new JLabel("Algoritmo: ");
        titAlg.setBounds(30, 50, 100, 20);
        opciones.add(titAlg);
        titAlg.setVisible(true);

        titTip = new JLabel("Tipo: ");
        titTip.setBounds(30, 100, 100, 20);
        opciones.add(titTip);
        titTip.setVisible(true);

        titVel = new JLabel("Velocidad: ");
        titVel.setBounds(30, 150, 100, 20);
        opciones.add(titVel);
        titVel.setVisible(true);

        //Botones
        buscar = new JButton("Buscar");
        buscar.setBounds(695, 30, 90, 20);
        principal.add(buscar);
        buscar.setVisible(true);

        //Abrir FileChooser
        ActionListener filechooser = new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
```

```
                AbrirRutaArchivo k = new AbrirRutaArchivo();
                k.leerarchivo();
                ingTitulo.setText(k.1);
                ingRuta.setText(k.1);
                //Asignando variable a la ruta para que lo lea el LeerCSV.java
                paraLeer = k.leerarchivo();
            }
        };
        buscar.addActionListener(filechooser);

        cargar = new JButton("Cargar");
        cargar.setBounds(695, 80, 90, 20);
        principal.add(cargar);
        cargar.setVisible(true);

        //Leer y mostrar .CSV
        ActionListener leermostrar = new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String archivo1 = paraLeer;
                System.out.println(archivo1);
                try {
                    LeerCSV m = new LeerCSV();
                    m.LeerCSV1(archivo1);

                    System.out.println(m);
                    GenerarGrafico y = new GenerarGrafico();
                    Datos i = new Datos();
                    y.GenerarGrafico(i.getValores());
```

```
                } catch (IOException ex) {
                    Logger.getLogger(VentanaPrincipal.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
        };
        cargar.addActionListener(leermostrar);

        ejecutar = new JButton("Ejecutar");
        ejecutar.setBounds(70, 210, 100, 20);
        opciones.add(ejecutar);
        ejecutar.setVisible(true);

        //ComboBox
        String[] j = {"Burbuja", "Inserción", "Selección"};
        algoritmo = new JComboBox(j);
        algoritmo.setBounds(110, 50, 100, 20);
        opciones.add(algoritmo);
        algoritmo.setVisible(true);

        String[] m = {"Ascendente", "Descendente"};
        tipo = new JComboBox(m);
        tipo.setBounds(110, 100, 100, 20);
        opciones.add(tipo);
        tipo.setVisible(true);

        String[] p = {"Rápida", "Media", "Lenta"};
        velocidad = new JComboBox(p);
        velocidad.setBounds(110, 150, 100, 20);
        opciones.add(velocidad);
        velocidad.setVisible(true);

        setVisible(true);
```

```
    }
}
```

4. **LeerCSV:** Se encarga de recibir el archivo, dividir y guardar el texto para más adelante usarla en los gráficos. Lee usando BufferedReader y obteniendo ruta del archivo seleccionado en la ventana de VentanaPrincipal. Contiene un método para generar el gráfico inicial, el gráfico con el archivo original.

LeerCSV.java

```
package com.mycompany.practica2codigo;

import java.io.*;
import javax.swing.JOptionPane;

public class LeerCSV {

    VentanaPrincipal yy = new VentanaPrincipal();
    public static Datos[] datosn;
    public static String[] titulos;
    public static String[] columnas;
    String x, y;
    String nombres;
    String valores;
    GenerarGrafico tt;

    public void LeerCSV1(String ruta) throws FileNotFoundException, IOException {
        File t = null;
        t = new File(ruta);

        try {
            BufferedReader lector = new BufferedReader(new FileReader(t));
            String line = "";
            String texto = "";

            while ((line = lector.readLine()) != null) {
                texto += line + "\n";
            }

            String[] lineas = texto.split("\n");
            titulos = lineas[0].split(",");
            x = titulos[0];
```

```
            x = titulos[0];
            y = titulos[1];

            System.out.println(x + y);

            datosn = new Datos[lineas.length - 1];

            for (int i = 1; i < lineas.length; i++) {
                columnas = lineas[i].split(",");
                datosn[i - 1] = new Datos(columnas[0], columnas[1], titulos[0], titulos[1]);

            }

            x = titulos[0];
            y = titulos[1];

            MostrarGraficoInicial();

            for (Datos dato : datosn) {
                System.out.println("Curso: " + dato.getCurso() + " Nota: " + dato.getValores() + " Titulo x: " + dato.getX()
            }

        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "El sistema no puede encontrar la ruta especificada.");
        }

    }

    public void MostrarGraficoInicial() {
        tt = new GenerarGrafico();
        tt.GenerarGrafico(datosn);
    }
```

5. **AbrirRutaArchivo:** Es la clase encargada de obtener la ruta leída por LeerCSV y la divide en dos partes, la URL sin el título y el título. Envía ambas a las cajas de texto de la VentanaPrincipal para poder enviarlas a la clase LeerCSV.

AbrirRutaArchivo.java

Source History

```
package com.mycompany.practica2codigo;

import java.io.File;
import javax.swing.JFileChooser;
import javax.swing.JPanel;

public class AbrirRutaArchivo {

    String ruta = "";
    JPanel lectura = new JPanel();
    String j = "";
    String l = "";

    public String leerarchivo() {

        JFileChooser fc = new JFileChooser();
        int op = fc.showOpenDialog(lectura);
        if (op == JFileChooser.APPROVE_OPTION) {

            File pRuta = fc.getSelectedFile();

            //Identificador de titulo en la ruta
            ruta = pRuta.getAbsolutePath();
            String o = "";

            for (int i = ruta.length(); i >= 1; i--) {
                String p = String.valueOf(ruta.charAt(i - 1));
                if (p.equals("\\")) {
                    break;
                } else {
                    o = p + o;
                }
```

```
                }
            }

            //Aquí en o ya tengo el nombre del archivo. Ahora, ¿cómo lo elimino?
            j = ruta.replaceAll(o, "");

            String m = "";

            for (int i = ruta.length(); i >= 1; i--) {
                String p = String.valueOf(ruta.charAt(i - 1));
                if (p.equals("\\")) {
                    break;
                } else {
                    m = p + m;
                }
            }
            l = m;

        } else if (op == JFileChooser.CANCEL_OPTION) {

        }
        return ruta;
    }
}
```

6. **GenerarGrafico:** La clase que genera el gráfico del archivo original, obtenido de la lectura hecha por la clase LeerCSV.

GenerarGrafico.java

Source History

```
package com.mycompany.practica2codigo;

import java.awt.Color;
import javax.swing.*;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.chart.ChartPanel;

public class GenerarGrafico extends JFrame{

    String cursosN, notasN;
    ChartPanel grafico;
    JPanel principal1= new JPanel();;

    LeerCSV t = new LeerCSV();
    Datos m = new Datos();
    JFreeChart graficoBar;
    VentanaPrincipal ven = new VentanaPrincipal();
    DefaultCategoryDataset datosg;

    int contador = 0;
    String x, y;
    String titX, titY;
    String CursoN;
    String NOMBRECURSO;
    public static Datos [] datosn;
    int notaN;
    int notaCurso;
    int NOTAOBTENIDA;
    Datos [] notasCursos;
```

```
    Datos [] NotasCursos;
    public void GenerarGrafico (Datos [] datosn) {

        this.datosn = datosn;


        try {
        datosg = new DefaultCategoryDataset();

        for (Datos nota : datosn) {
         CursoN = nota.getCurso();
         NOTAOBTENIDA = nota.getValores();
            System.out.println(" notas: " + CursoN + " nombre: " + NOTAOBTENIDA + " " + t.titulos[0] + " " + t.titulos[1]);
        datosg.addValue(nota.getValores(), nota.getCurso(), contador + "");
        contador++;
        }
//        }

        } catch (Exception e){

        }
        graficoBar = ChartFactory.createBarChart("Calificaciones", t.titulos[0], t.titulos[1], datosg, PlotOrientation.VERTICAL, true, true, false);



        setSize(550, 400);
        setLayout(null);
        setLocationRelativeTo(null);
        getContentPane().add(principal1);
        principal1.setLayout(null);
        principal1.setSize(500, 300);
        principal1.setVisible(true);
```

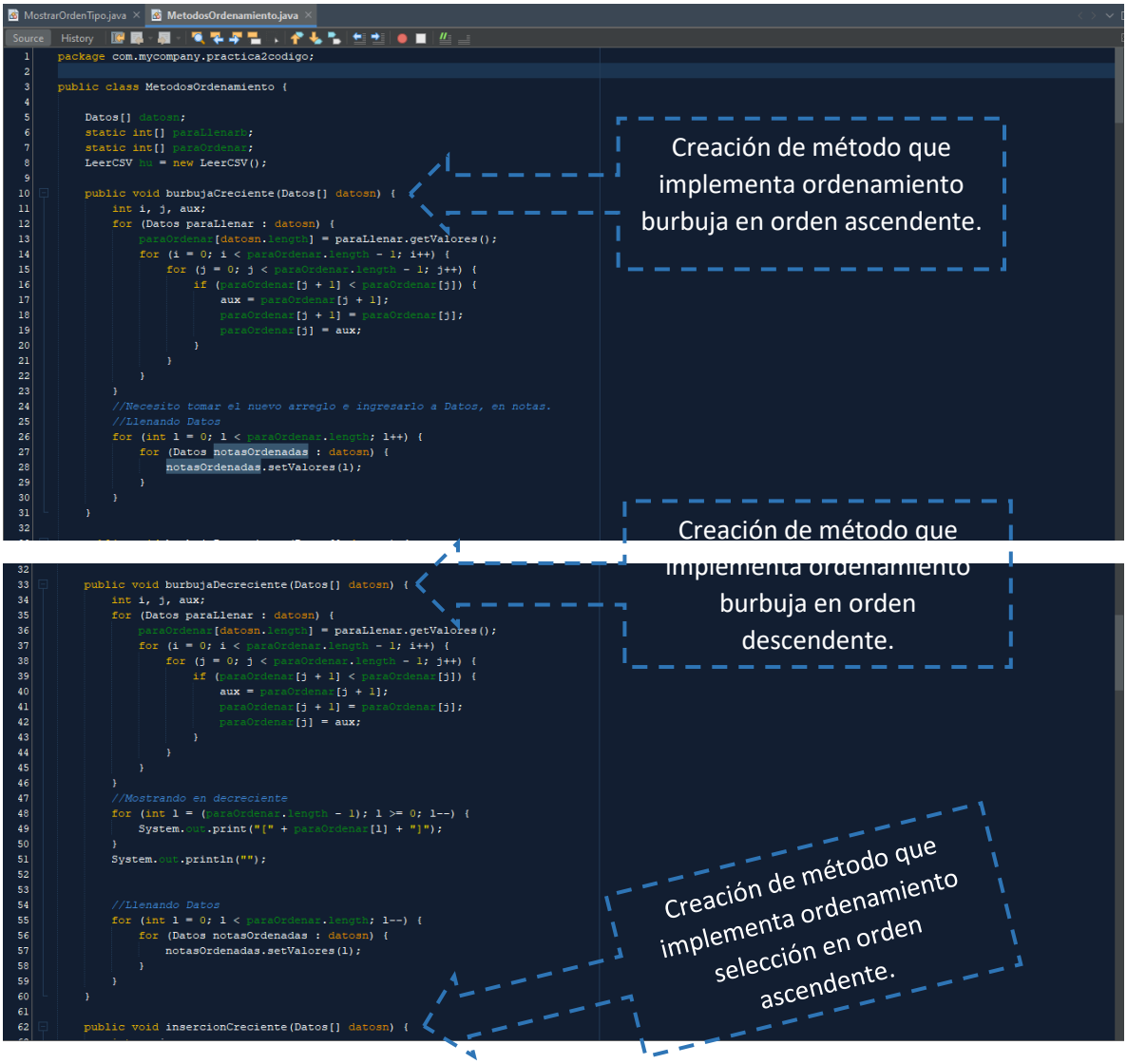
```
        principal1.setVisible(true);

        grafico = new ChartPanel(graficoBar);
        grafico.setSize(500, 300);
        grafico.setBackground(Color.LIGHT_GRAY);
        principal1.add(grafico);
        grafico.setVisible(true);
        setDefaultCloseOperation(HIDE_ON_CLOSE);
        setVisible(true);

    }

}
```

7. **Timer:** Clase intento para hacer un reloj. No fue aplicada en el proyecto, porque no era la manera correcta de implementarlo, pero era parte de las clases, así que igual está siendo añadida.

Timer.java

```
package com.mycompany.practica2codigo;
public class Timer implements Runnable {

    Thread t;
    String nombre;

    public void rapido() {
        t = new Thread(this, "");
        t.start();
    }

    public void rapido(String nombre) {
        this.nombre = nombre;
        t = new Thread(this, "");
        t.start();
    }

    public void run() {
        try {
            for (int i = 0; i < 60; i++) {
                if (i < 10) {
                    System.out.println("Tiempo transcurrido: " + ":00" + ":00:0" + i);
                } else if (i >= 10 && i < 60) {
                    System.out.println("Tiempo transcurrido: " + ":00" + ":00:" + i);
                }

                Thread.sleep(500);
            }
        } catch (InterruptedException e) {
        };
    }
}
```

8. **MetodosOrdenamiento:** Clase donde se desarrollaron los algoritmos de ordenamiento, también se implementó el tipo, si sería ascendente o descendente. Incluye diferentes métodos para ambas de las opciones.




```java
package com.mycompany.practica2codigo;

public class MetodosOrdenamiento {

    Datos[] datosn;
    static int[] paraLlenarb;
    static int[] paraOrdenar;
    LeerCSV hu = new LeerCSV();

    public void burbujaCreciente(Datos[] datosn) {
        int i, j, aux;
        for (Datos paraLlenar : datosn) {
            paraOrdenar[datosn.length] = paraLlenar.getValores();
            for (i = 0; i < paraOrdenar.length - 1; i++) {
                for (j = 0; j < paraOrdenar.length - 1; j++) {
                    if (paraOrdenar[j + 1] < paraOrdenar[j]) {
                        aux = paraOrdenar[j + 1];
                        paraOrdenar[j + 1] = paraOrdenar[j];
                        paraOrdenar[j] = aux;
                    }
                }
            }
        }
        //Necesito tomar el nuevo arreglo e ingresarlo a Datos, en notas.
        //Llenando Datos
        for (int l = 0; l < paraOrdenar.length; l++) {
            for (Datos notasOrdenadas : datosn) {
                notasOrdenadas.setValores(l);
            }
        }
    }

    public void burbujaDecreciente(Datos[] datosn) {
        int i, j, aux;
        for (Datos paraLlenar : datosn) {
            paraOrdenar[datosn.length] = paraLlenar.getValores();
            for (i = 0; i < paraOrdenar.length - 1; i++) {
                for (j = 0; j < paraOrdenar.length - 1; j++) {
                    if (paraOrdenar[j + 1] < paraOrdenar[j]) {
                        aux = paraOrdenar[j + 1];
                        paraOrdenar[j + 1] = paraOrdenar[j];
                        paraOrdenar[j] = aux;
                    }
                }
            }
        }
        //Mostrando en decreciente
        for (int l = (paraOrdenar.length - 1); l >= 0; l--) {
            System.out.print("[" + paraOrdenar[l] + "]");
        }
        System.out.println("");


        //Llenando Datos
        for (int l = 0; l < paraOrdenar.length; l--) {
            for (Datos notasOrdenadas : datosn) {
                notasOrdenadas.setValores(l);
            }
        }
    }

    public void insercionCreciente(Datos[] datosn) {
```

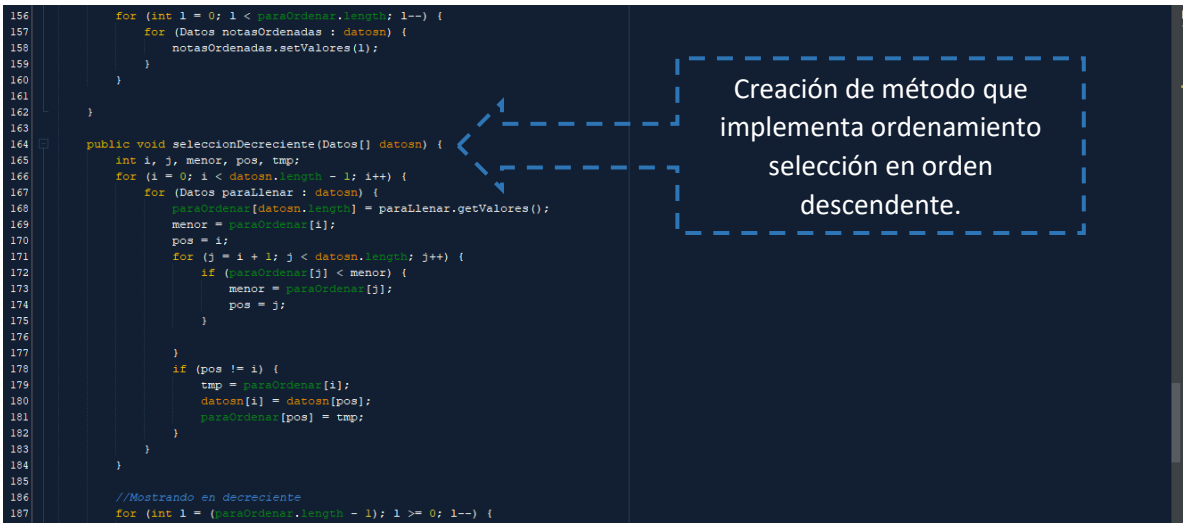
```java
public void insercionCreciente(Datos[] datosn) {
    int p, j;
    int aux;
    for (Datos paraLlenar : datosn) {
        paraOrdenar[datosn.length] = paraLlenar.getValores();
        for (p = 1; p < paraOrdenar.length; p++) {

            aux = paraOrdenar[p];
            j = p - 1;
            while ((j >= 0) && (aux < paraOrdenar[j])) {

                paraOrdenar[j + 1] = paraOrdenar[j];
                j--;
            }
            paraOrdenar[j + 1] = aux;
        }
    }
    //Creciente
    for (int l = 0; l < paraOrdenar.length; l++) {
        System.out.print("[" + paraOrdenar[l] + "]");
    }
    System.out.println("");

    for (int l = 0; l < paraOrdenar.length; l++) {
        for (Datos notasOrdenadas : datosn) {
            notasOrdenadas.setValores(l);
        }
    }
    System.out.println("");

    //Llenando Datos
    for (int l = 0; l < paraOrdenar.length; l--) {
```

```java
        for (Datos notasOrdenadas : datosn) {
            notasOrdenadas.setValores(l);
        }
    }

}

public void insercionDecreciente(Datos[] datosn) {
    int p, j;
    int aux;
    for (Datos paraLlenar : datosn) {
        paraOrdenar[datosn.length] = paraLlenar.getValores();
        for (p = 1; p < paraOrdenar.length; p++) {

            aux = paraOrdenar[p];
            j = p - 1;
            while ((j >= 0) && (aux < paraOrdenar[j])) {

                paraOrdenar[j + 1] = paraOrdenar[j];
                j--;
            }
            paraOrdenar[j + 1] = aux;
        }
    }

    //Mostrando en decreciente
    for (int l = (paraOrdenar.length - 1); l >= 0; l--) {
        System.out.print("[" + paraOrdenar[l] + "]");
    }
    System.out.println("");

}
```

Creación de método que implementa ordenamiento inserción en orden descendente.

```java
}

public void seleccionCreciente(Datos[] datosn) {
    int i, j, menor, pos, tmp;
    for (i = 0; i < datosn.length - 1; i++) {
        for (Datos paraLlenar : datosn) {
            paraOrdenar[datosn.length] = paraLlenar.getValores();
            menor = paraOrdenar[i];
            pos = i;
            for (j = i + 1; j < datosn.length; j++) {
                if (paraOrdenar[j] < menor) {
                    menor = paraOrdenar[j];
                    pos = j;
                }

            }
            if (pos != i) {
                tmp = paraOrdenar[i];
                datosn[i] = datosn[pos];
                paraOrdenar[pos] = tmp;
            }
        }
    }

    //Creciente
    for (int l = 0; l < paraOrdenar.length; l++) {
        System.out.print("[" + paraOrdenar[l] + "]");
    }
    System.out.println("");

    //Llenando Datos
    for (int l = 0; l < paraOrdenar.length; l--) {
```

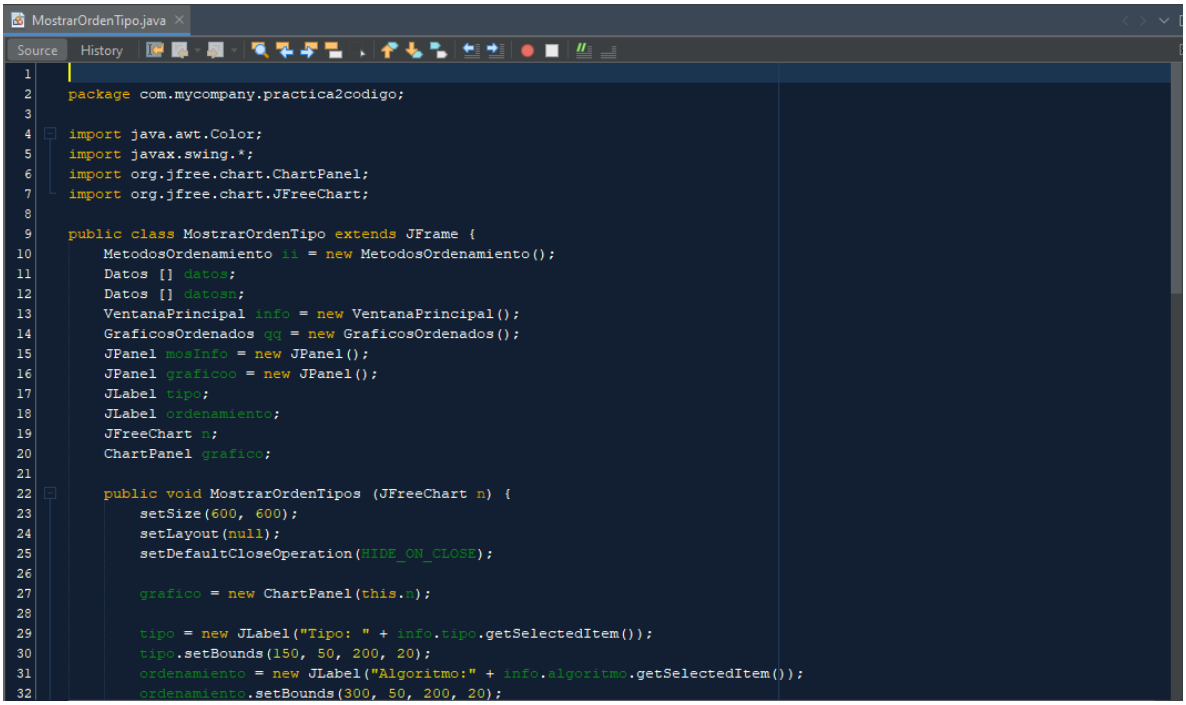
Creación de método que implementa ordenamiento selección en orden ascendente.




```
        for (int l = 0; l < paraOrdenar.length; l--) {
            for (Datos notasOrdenadas : datosn) {
                notasOrdenadas.setValores(l);
            }
        }

    }

    public void seleccionDecreciente(Datos[] datosn) {
        int i, j, menor, pos, tmp;
        for (i = 0; i < datosn.length - 1; i++) {
            for (Datos paraLlenar : datosn) {
                paraOrdenar[datosn.length] = paraLlenar.getValores();
                menor = paraOrdenar[i];
                pos = i;
                for (j = i + 1; j < datosn.length; j++) {
                    if (paraOrdenar[j] < menor) {
                        menor = paraOrdenar[j];
                        pos = j;
                    }

                }
                if (pos != i) {
                    tmp = paraOrdenar[i];
                    datosn[i] = datosn[pos];
                    paraOrdenar[pos] = tmp;
                }
            }
        }

        //Mostrando en decreciente
        for (int l = (paraOrdenar.length - 1); l >= 0; l--) {
```

```
        for (int l = (paraOrdenar.length - 1); l >= 0; l--) {
            System.out.print("[" + paraOrdenar[l] + "]");
        }
        System.out.println("");

        //Llenando Datos
        for (int l = 0; l < paraOrdenar.length; l--) {
            for (Datos notasOrdenadas : datosn) {
                notasOrdenadas.setValores(l);
            }
        }

    }

}
```

9. **MostrarOrdenTipo:** En esta clase fue construida para mostrar los ordenamientos realizados por la clase de MetodosOrdenamiento, se trasladaban los datos a Datos y se llamaban en esta clase. Incluye if's para determinar si los combobox tenían seleccionados datos específicos del panel en VentanaPrincipal y mostrarlos en la ventana que se había creado en esta clase "MostrarOrdenTipo".



```
MostrarOrdenTipo.java
Source  History

package com.mycompany.practica2codigo;

import java.awt.Color;
import javax.swing.*;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;

public class MostrarOrdenTipo extends JFrame {
    MetodosOrdenamiento ii = new MetodosOrdenamiento();
    Datos [] datos;
    Datos [] datosn;
    VentanaPrincipal info = new VentanaPrincipal();
    GraficosOrdenados qq = new GraficosOrdenados();
    JPanel mosInfo = new JPanel();
    JPanel graficoo = new JPanel();
    JLabel tipo;
    JLabel ordenamiento;
    JFreeChart n;
    ChartPanel grafico;

    public void MostrarOrdenTipos (JFreeChart n) {
        setSize(600, 600);
        setLayout(null);
        setDefaultCloseOperation(HIDE_ON_CLOSE);

        grafico = new ChartPanel(this.n);

        tipo = new JLabel("Tipo: " + info.tipo.getSelectedItem());
        tipo.setBounds(150, 50, 200, 20);
        ordenamiento = new JLabel("Algoritmo:" + info.algoritmo.getSelectedItem());
        ordenamiento.setBounds(300, 50, 200, 20);
```

```
33              tipo.setVisible(true);
34              ordenamiento.setVisible(true);
35
36              mosInfo.setBackground(Color.LIGHT_GRAY);
37              mosInfo.setBounds(0, 10, 500, 100);
38              mosInfo.add(tipo);
39              mosInfo.add(ordenamiento);
40              getContentPane().add(mosInfo);
41              mosInfo.setVisible(true);
42
43              setVisible(true);
44
45              if (info.tipo.getSelectedItem().equals("Ascendente") && info.algoritmo.getSelectedItem().equals("Burbuja")) {
46                  ii.burbujaCreciente(datos);
47                  qq.GenerarGraficoOrdenado(datosn);
48              }
49
50              else if (info.tipo.getSelectedItem().equals("Descendente") && info.algoritmo.getSelectedItem().equals("Burbuja")) {
51                  ii.burbujaDecreciente(datos);
52                  qq.GenerarGraficoOrdenado(datosn);
53              }
54
55              else if (info.tipo.getSelectedItem().equals("Ascendente") && info.algoritmo.getSelectedItem().equals("Inserción")) {
56                  ii.burbujaDecreciente(datos);
57                  qq.GenerarGraficoOrdenado(datosn);
58              }
59
60              else if (info.tipo.getSelectedItem().equals("Descendente") && info.algoritmo.getSelectedItem().equals("Inserción")) {
61                  ii.insercionDecreciente(datos);
62                  qq.GenerarGraficoOrdenado(datosn);
63              }
64
```

Validaciones para ver qué ordenamiento de la clase anterior se va a elegir.

```
63              }
64
65              else if (info.tipo.getSelectedItem().equals("Ascendente") && info.algoritmo.getSelectedItem().equals("Selección")) {
66                  ii.seleccionCreciente(datos);
67                  qq.GenerarGraficoOrdenado(datosn);
68              }
69
70              else if (info.tipo.getSelectedItem().equals("Descendente") && info.algoritmo.getSelectedItem().equals("Selección")) {
71                  ii.seleccionDecreciente(datos);
72                  qq.GenerarGraficoOrdenado(datosn);
73              }
74
75          }
76      }
77
```