Manual Técnico

1. Clase Grafica

En esta clase se crearon dos atributos, los cuales son nombre y cantidad en los cuales se almacenarán todos los datos de los archivos .csv que se cargarán en el programa para ser graficados.

```java
package Ventanas;


public class Grafica {
    private String nombre;
    private int cantidad;



    public Grafica(String nombre, String cantidad) {
        this.nombre = nombre;
        this.cantidad = Integer.parseInt(cantidad);
    }




    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getCantidad() {
        return cantidad;
    }

    public void setCantidad(int cantidad) {
        this.cantidad = cantidad;
    }


}
```

2. Método AbrirArchivo

Este método es utilizado para que abra automáticamente el reporte generado al momento de finalizar la ejecución del ordenamiento.

```
public AbrirArchivo (String archivo) {

    try {

        File objetofile = new File(archivo);
        Desktop.getDesktop().open(objetofile);

    } catch (IOException ex) {

        System.out.println(ex);

    }

}
```

3. Método validar Entero

Este método se utiliza para comprobar si los datos ingresados por medio del archivo csv correspondían a números.

```
public static boolean validaNumeroEntero_Exp(String texto) {
    return texto.matches("^-?[0-9]+$");
}
```

4. Clase Pasos

Esta clase es utilizada para ir contando el numero de pasos que se realizan para el ordenamiento de las gráficas, en esta clase se implementa un hilo con un contador que ira aumentando conforme la gráfica vaya cambiando.

```
public class Pasos extends Thread {

    JLabel eti;
    static int y = 0;

    Pasos(JLabel pasos) {
        VentanaPrincipal.steps = 0;
        this.eti = pasos;

    }

    public void run() {

        try {
            int x = 0;
            y=0;
            while (VentanaPrincipal.iniciaPasos) {
                //Thread.sleep(10);
                //System.out.println(x);
                //ejecutarHiloCronometro(x);
                ejecutarPasos(x);
               // System.out.println(y);
                //y++;
            }

        } catch (Exception e) {
            System.out.println("Error hilo" + e.getMessage());
        }

    }

    private synchronized void ejecutarPasos(int x) {
        if (y==+1) {
             VentanaPrincipal.steps++;
        }

        String text = "";
        text += y;
        eti.setText(text);
    }
```

5. Método para leer archivo .csv y generar gráfica

En este método se lee el archivo .csv pidiendo como parámetro la ruta del archivo, el panel donde se colocará la gráfica y el titulo que se le desea colocar a la gráfica que se generará. Además de esto guardara grafica en una imagen que posteriormente será colocada en el reporte final y también ira creando la estructura de tablas de los datos originales en formato HTML para ser agregados en el reporte.

```
public MetodosSueltos(String ruta, JPanel panel, String titulo) {
    Static.elementos = new Grafica[700];
    Static.contadorElementos = 0;
    DefaultCategoryDataset data = new DefaultCategoryDataset();
    String texto = "";

    try {
        BufferedReader br = new BufferedReader(new FileReader(ruta));
        String linea;
        while ((linea = br.readLine()) != null) { //leer

            String columnas[] = linea.split(",");


            if (MetodosSueltos.validaNumeroEntero_Exp(columnas[1])) {
                int numEntero = Integer.parseInt(columnas[1]);
                data.setValue(numEntero, columnas[0], " ");
                Grafica a = new Grafica(columnas[0], columnas[1]);
                Static.elementos[Static.contadorElementos] = a;
                Static.contadorElementos++;
            } else if (!MetodosSueltos.validaNumeroEntero_Exp(columnas[1])) {
                nombrebarras = columnas[0];
                nombrenumeracion = columnas[1];

            }

        }
        String a = "";

         a = "<div style=\"text-align:center;\">\n"
                + " <table border=\"1\" style=\"margin: 0 auto;\">";
        a += "\n";
          a+= "      <TR><TH>" + nombrebarras + "</TH>\n";
        for (int i = 0; i < Static.contadorElementos; i++) {
            a +=
                    "              <TD>"+Static.elementos[i].getNombre()+"</TD> \n";



        }
        a+="        <TR><TH>" + nombrenumeracion + "</TH>\n";
        for (int i = 0; i < Static.contadorElementos; i++) {
            a+= "           <TD>"+Static.elementos[i].getCantidad()+"</TD> \n";
        }
        a+="</TABLE>";
```

```
                a +=
                        "            <TD>"+Static.elementos[i].getNombre()+"</TD> \n";



            }
            a+="        <TR><TH>" + nombrenumeracion + "</TH>\n";
            for (int i = 0; i < Static.contadorElementos; i++) {
                a+= "           <TD>"+Static.elementos[i].getCantidad()+"</TD> \n";
            }
            a+="</TABLE>";
            tablaNormal = a;
            System.out.println(a);
            br.close(); //cerrar el flujo y liberar recursos
        } catch (IOException e) {
            //en caso de error
        }

        //  for (int i = 0; i < Static.contadorElementos; i++) {
        // System.out.println(Static.elementos[i]+"   ");
        //  }
        JFreeChart barras = ChartFactory.createBarChart(titulo, nombrebarras, nombrenumeracion, data, PlotOrientation.VERTICAL, true, true, false);
        ChartPanel panel1 = new ChartPanel(barras);
        //panel1.setMouseWheelEnabled(true);
        panel.setPreferredSize(new Dimension(500, 540));

        panel.setLayout(new BorderLayout());
        panel.add(panel1, BorderLayout.CENTER);
        panel.validate();
        //pack();
        //panel.repaint();
        try {
            final File file = new File("A:\\Programas Java\\Practica2_Graficas\\IPC1_Practica2_202006666\\name.png"); //Definición del archivo con nombre y extensión
            ChartUtilities.saveChartAsPNG(file, barras, 800, 500); //Generar gráfica en formato PNG
            System.out.println("imagengenerada");
        } catch (IOException e) {
            System.out.println("Imagen no generada.");
        }

    }
```

6. Clase cronometro

En esta clase se implementa un hilo que creará un cronometro el cual estará en milisegundos, segundos y minutos que servirá para monitorear el tiempo de ejecución del ordenamiento elegido por el usuario.

```java
public class Cronometro extends Thread {

    JLabel eti;
    static int x = 0;
    static String timer= "";
    Cronometro(JLabel cronometro) {
        VentanaPrincipal.hora = 0;
        VentanaPrincipal.segundo = 0;
        VentanaPrincipal.minuto = 0;
        this.eti = cronometro;

    }

    public void run() {
        try {
            x = 0;

            while (VentanaPrincipal.iniciaHilo) {
                Thread.sleep(10);
                //System.out.println(x);
                ejecutarHiloCronometro(x);
                x++;
            }

        } catch (Exception e) {
            System.out.println("Error hilo" + e.getMessage());
        }
    }

    private synchronized void ejecutarHiloCronometro(int x) {

        // System.out.println(x+ " - " + Thread.currentThread().getName());
        VentanaPrincipal.segundo++;
        if (VentanaPrincipal.segundo > 59) {
            VentanaPrincipal.segundo = 0;
            VentanaPrincipal.minuto++;

        }
        if (VentanaPrincipal.minuto > 59) {
            VentanaPrincipal.minuto = 0;
            VentanaPrincipal.hora++;
```

```
    private synchronized void ejecutarHiloCronometro(int x) {

        // System.out.println(x+ " - " + Thread.currentThread().getName());
        VentanaPrincipal.segundo++;
        if (VentanaPrincipal.segundo > 59) {
            VentanaPrincipal.segundo = 0;
            VentanaPrincipal.minuto++;

        }
        if (VentanaPrincipal.minuto > 59) {
            VentanaPrincipal.minuto = 0;
            VentanaPrincipal.hora++;

        }
        String textoSeg = "", textMin = "", textHora = "";

        if (VentanaPrincipal.segundo < 10) {
            textoSeg = "0" + VentanaPrincipal.segundo;
        } else {
            textoSeg = "" + VentanaPrincipal.segundo;
        }

        if (VentanaPrincipal.minuto < 10) {
            textMin = "0" + VentanaPrincipal.minuto;
        } else {
            textMin = "" + VentanaPrincipal.minuto;
        }

        if (VentanaPrincipal.hora < 10) {
            textHora = "0" + VentanaPrincipal.hora;
        } else {
            textHora = "" + VentanaPrincipal.hora;
        }

        //textoSeg+=VentanaPrincipal.segundo;
        //textMin+=VentanaPrincipal.minuto;
        String reloj = textHora + ":" + textMin + ":" + textoSeg;
        timer=reloj;
        eti.setText(reloj);

    }
}
```

7. Método para actualizar gráfica y generar imagen

Este método se utiliza cada vez que un elemento de la gráfica cambie de posición en el arreglo donde está guardado y así ir mostrando cuando se mueve de posición en la gráfica, y si ya ha terminado de completar el ordenamiento se generara automáticamente una imagen con la gráfica ordenada que será colocada posteriormente en un archive HTML para el reporte final.

```java
public void grafica_barras(JPanel panel) {
    try {
        DefaultCategoryDataset datos = new DefaultCategoryDataset();  // guardaremos la informacion para la grafica
        for (int i = 0; i < Static.contadorElementos; i++) {
            datos.setValue(Static.elementos[i].getCantidad(), Static.elementos[i].getNombre(), "");
        }
        JFreeChart barras = ChartFactory.createBarChart(VentanaPrincipal.titulos //titulo de grafica
                ,
                 MetodosSueltos.nombrebarras // nombre de eje x
                ,
                 MetodosSueltos.nombrenumeracion, // nombre eje y
                datos, // informacion que usará
                PlotOrientation.VERTICAL,// orientacion de la grafica
                true,// lengenda de barras individuales por color
                true,// herramientas
                false);// url de grafico

        ChartPanel panel_grafica = new ChartPanel(barras);
        panel.setPreferredSize(new Dimension(500, 540));

        panel.setLayout(new BorderLayout());
        panel.add(panel_grafica, BorderLayout.CENTER);
        //panel.validate();
        if (VentanaPrincipal.iniciaHilo == false && VentanaPrincipal.iniciaPasos == false) {
            final File file = new File("A:\\Programas Java\\Practica2_Graficas\\IPC1_Practica2_202006666\\ordenada.png"); //
            ChartUtilities.saveChartAsPNG(file, barras, 800, 500); //Generar gráfica en formato PNG
            System.out.println("imagengenerada");

        }
    } catch (Exception e) {
        System.out.println(e);
    }
}
```

8. Métodos para los distintos tipos de ordenamiento

Estos métodos son utilizados dependiendo del tipo de ordenamiento que el usuario desea que se realice, en estos métodos se implementa hilos para que el ordenamiento se de junto con la actualización de la grafica en tiempo real, para ello se necesitan dos parámetros los cuales son el panel donde se ira actualizando la grafica y la velocidad con que se desea que se realice el ordenamiento. También en este método se generará una variable que contendrá los datos ya ordenados en tablas en formato HTML para que sean agregadas en el reporte final.

```java
public class MetodoBurbujaDescendente extends Thread {

    JPanel panel;
    JLabel grafica_personas;
    int velocidad;
    boolean ordenado = false;

    public MetodoBurbujaDescendente(JPanel panel, int velocidad) {
        this.panel = panel;
        this.velocidad = velocidad;
        grafica_barras(panel);
    }

    @Override
    public void run() {
        try {
            //aqui empieza el metodo burbuja
            Grafica aux;

            for (int i = 0; i < Static.contadorElementos - 1; i++) {
                for (int j = 0; j < Static.contadorElementos - i - 1; j++) {
                    if (Static.elementos[j + 1].getCantidad() > Static.elementos[j].getCantidad()) {
                        aux = Static.elementos[j];
                        Static.elementos[j] = Static.elementos[j + 1];
                        Static.elementos[j + 1] = aux;
                        // grafica_barras(panel,grafica_personas);
                        grafica_barras(panel);

                        Pasos.y++;

                        Thread.sleep(velocidad);
                    }

                }

            }
            VentanaPrincipal.iniciaHilo = false;
            VentanaPrincipal.iniciaPasos = false;
            grafica_barras(panel);

        }
        VentanaPrincipal.iniciaHilo = false;
        VentanaPrincipal.iniciaPasos = false;
        grafica_barras(panel);

        String a = "";

        a = "<div style=\"text-align:center;\">\n"
                + " <table border=\"1\" style=\"margin: 0 auto;\">";
        a += "\n";
        a += "      <TR><TH>" + MetodosSueltos.nombrebarras + "</TH>\n";
        for (int i = 0; i < Static.contadorElementos; i++) {
            a
                    += "            <TD>" + Static.elementos[i].getNombre() + "</TD> \n";

        }
        a += "      <TR><TH>" + MetodosSueltos.nombrenumeracion + "</TH>\n";
        for (int i = 0; i < Static.contadorElementos; i++) {
            a += "          <TD>" + Static.elementos[i].getCantidad() + "</TD> \n";
        }
        a += "</TABLE>";
        MetodosSueltos.tablaOrdenada = a;
        System.out.println(MetodosSueltos.tablaOrdenada);
        Reporte report = new Reporte(VentanaPrincipal.velocidadReporte, VentanaPrincipal.tipoReporte, VentanaPrincipal.ordenamientoReporte, Pasos.y, Cronometro.timer);

        AbrirArchivo openFile = new AbrirArchivo("A:\\Programas Java\\Practica2_Graficas\\IPC1_Practica2_202006666\\reporte.html");

    } catch (InterruptedException e) {
        System.out.println("Error al Ordenar");
    }
}
```

9. Método para generar reporte

En este método se creará un documento .HTML en el directorio indicado el cual contendrá los distintos datos recolectados en la ejecución del ordenamiento de las gráficas, también tendrá una tabla y una imagen del estado inicial y final de los datos.

```
public Reporte(String velocidad, String tipo, String ordenamiento, int pasos, String timer) {
    try (BufferedWriter bw = new BufferedWriter(new FileWriter("A:\\Programas Java\\Practica2_Graficas\\IPC1_Practica2_202006666\\reporte.html"))) {

        String primeraParte = "<head>\n"
                + "    <title> Reporte </title>\n"
                +"<meta http-equiv=\"Content-Type\" content=\"text/html\" charset=\"UTF-8\" />"
                + "    <meta charset=\"UTF-8\">\n"
                + "    <meta name=\"viewport\" content=\"widht>\n"
                + "</head>\n"
                + "<body>\n"
                + "    <p style=\"text-align: center;\"> </p>\n"
                + "<h1 style=\"text-align: center; color: #3f7320;\">Elvis Joseph V&aacute;squez Villatoro</h1>\n"
                + "<h1 style=\"text-align: center; color: #3f7320;\">202006666</h1>\n"
                + "<hr /><!-- Este comentario es visible solo en el editor fuente -->\n"
                + "<h2 style=\"text-align: center;\"><strong>Detalle de Ordenamiento                  </strong><strong>Det
                + "<p style=\"text-align: center;\"> </p>\n"
                + "<p style=\"text-align: center;\"><strong>Algoritmo: </strong>" + ordenamiento + "                    &nbs
                + "<p style=\"text-align: center;\"><strong>Tipo: </strong>" + tipo + "                        &nb
                + "<p>                                            &nb
                + "<p style=\"text-align: center;\">                              <strong> 
                + "<hr />\n"
                + "<h2 style=\"text-align: center;\"><strong>Estado Inicial</strong></h2>\n"
                + "<br>\n"
                + "    <p style=\"text-align: center;\"> </p>\n"
                + "<h1 style=\"text-align: center; color: #3f7320;\">Elvis Joseph V&aacute;squez Villatoro</h1>\n"
                + "<h1 style=\"text-align: center; color: #3f7320;\">202006666</h1>\n"
                + "<hr /><!-- Este comentario es visible solo en el editor fuente -->\n"
                + "<h2 style=\"text-align: center;\"><strong>Detalle de Ordenamiento                    </strong><strong>Detalle
                + "<p style=\"text-align: center;\"> </p>\n"
                + "<p style=\"text-align: center;\"><strong>Algoritmo: </strong>" + ordenamiento + "                      &r
                + "<p style=\"text-align: center;\"><strong>Tipo: </strong>" + tipo + "                        &
                + "<p>                                            &
                + "<p style=\"text-align: center;\">                              <strong>   
                + "<hr />\n"
                + "<h2 style=\"text-align: center;\"><strong>Estado Inicial</strong></h2>\n"
                + "<br>\n"
                + "<br>\n"
                + MetodosSueltos.tablaNormal + "\n";

        String segundaParte
                = "<p><img class=\"imageCenter\" style=\"display: block; margin-left: auto; margin-right: auto;\" src=\"name.png\"  width=\"600\" height=\"450\" /></p>\n"
                + "<hr />\n"
                + "<h2 style=\"text-align: center;\"><strong>Estado Final</strong></h2>\n"
                +MetodosSueltos.tablaOrdenada
                + "<p><img class=\"imageCenter\" style=\"display: block; margin-left: auto; margin-right: auto;\" src=\"ordenada.png\" alt=\"\" width=\"600\" height=\"450\" /></p>\n"
                + "    </body>\n"
                + "\n"
                + "</html>";

        bw.write(primeraParte + segundaParte);

    } catch (IOException ex) {

    }
```