MANUAL TÉCNICO

GRAFICADORA

Introducción

El presente documento describe de forma gráfica las especificaciones de los códigos más importantes que se utilizaron en la práctica 2, del curso de Introducción a la Programación y Computación 1, denominada: “GRAPH-USAC”.

Ciudad de Guatemala, Guatemala 23/03/2022

Elaborado por JOSÉ ALEXANDER LÓPEZ LÓPEZ

Objetivos

General

Que el estudiante pueda aplicar los conocimientos del curso para crear una solución de software aplicando hilos.

Específicos

- Que el estudiante pueda identificar una solución acorde a una necesidad.
- Que el estudiante pueda realizar una toma de requerimientos basado en una necesidad previamente identificada.
- Que el estudiante pueda implementar una solución a través de un lenguaje de programación.
- Entender la importancia de la interacción del software con el entorno real.

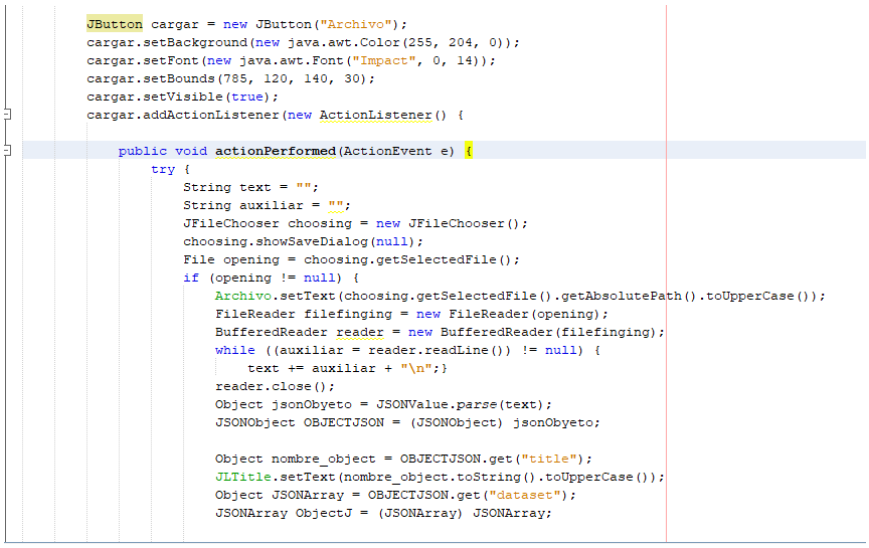
Lógica del Programa

1. Llamada al método principal

```
package PROGRAMA;

/**
 *
 * @author iosea
 */
public class App {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
            Window OBJECT= new Window();

    }

}
```

El método main funciona cómo punto de partida para la ejecución del programa, en el mismo se establece que se llama a la clase Window donde se encuentran las siguientes funciones del programa.

2. Acción botón cargar



```
JButton cargar = new JButton("Archivo");
cargar.setBackground(new java.awt.Color(255, 204, 0));
cargar.setFont(new java.awt.Font("Impact", 0, 14));
cargar.setBounds(785, 120, 140, 30);
cargar.setVisible(true);
cargar.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {
        try {
            String text = "";
            String auxiliar = "";
            JFileChooser choosing = new JFileChooser();
            choosing.showSaveDialog(null);
            File opening = choosing.getSelectedFile();
            if (opening != null) {
                Archivo.setText(choosing.getSelectedFile().getAbsolutePath().toUpperCase());
                FileReader filefinging = new FileReader(opening);
                BufferedReader reader = new BufferedReader(filefinging);
                while ((auxiliar = reader.readLine()) != null) {
                    text += auxiliar + "\n";}
                reader.close();
                Object jsonObyeto = JSONValue.parse(text);
                JSONObject OBJECTJSON = (JSONObject) jsonObyeto;

                Object nombre_object = OBJECTJSON.get("title");
                JLTitle.setText(nombre_object.toString().toUpperCase());
                Object JSONArray = OBJECTJSON.get("dataset");
                JSONArray ObjectJ = (JSONArray) JSONArray;
```

En la imagen anterior se visualiza el código necesario para establecer las propiedades visuales del botón que realizará la acción de abrir el explorador de archivos de donde se extraerá el archivo JSON que contiene los datos necesarios para la graficación.

3. Acción botón ordenar

```
        Inicia = new JButton("Ordenar");
        Inicia.setBounds(785, 160, 140, 30);
        Inicia.setBackground(new java.awt.Color(255, 204, 0));
        Inicia.setFont(new java.awt.Font("Impact", 0, 14));
        Inicia.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        Inicia.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {

                try {
                    new Thread(graphics).start();
                } catch (Exception ex) {
                    System.out.println(ex.toString());
                }
            }
        });
        JFrame2.add(Archivo);
        JFrame2.add(graphics);
        JFrame2.add(JLTitle);
        JFrame2.add(titulol);
        JFrame2.add(cargar);
        JFrame2.add(Inicia);
        JFrame2.setVisible(true);
    }

}
```

En la imagen anterior se visualiza el establecimiento de las propiedades del botón que realizará la acción de comenzar el ordenamiento a través del método de burbuja, además se visualiza que se agregan los componentes que se visualizarán en la ventana.

4. Método de Ordenamiento

```
// BURBUJA
for (int i = 0; i < (n - 1); i++) {
    for (int j = 0; j < (n - 1); j++) {
        if (datal[j] > datal[j + 1]) {
            temp = datal[j];
            datal[j] = datal[j + 1];
            datal[j + 1] = temp;
            Thread.sleep(70);
            graficar(datal);
            steps++;
            titulo.setText("pasos:" + steps);
        }
    }
}
```

En la imagen anterior se visualiza la lógica en sí, del método que realiza la acción de ordenar los valores cargados al programa.

5. Método de Ordenamiento

```
    @Override
    public void run() {
        setForeground(Color.BLACK);
        try {
            int minutes = 0;
            while (true) {
                Threads obj = new Threads();
                tiempo++;
                if (tiempo >= 60) {
                    tiempo = 0;
                    minutes++;
                    obj.time2++;
                }
                int seconds = tiempo;
                setText(String.format("%d:%02d", minutes, seconds));
                Thread.sleep(750);
            }
        } catch (InterruptedException e) {
        }
    }
}
```

Con este código se establece el flujo continuo y secuencial del tiempo del ordenamiento sincronizado con la graficación continua de las barras.