****



**TECNOLÓGICO NACIONAL DE MÉXICO**

**Instituto Tecnológico de León**

**INGENIERIA EN SISTEMAS COMPUTACIONALES**

**TOPICOS AVANZADOS DE PROGRAMACION**

**REPORTE – SWINGWORKER**

**ALUMNO:**

LEONEL ALEJANDRO AGUIRRE SERRANO

**PROFESOR**

ING. LUIS EDUARDO GUTIERREZ AYALA

**LEÓN, GUANAJUATO A 26 DE MAYO DEL 2020**

**REDACCION DEL PROBLEMA:**

El problema presentado en este reporte consiste en la creación de un programa que calcule números de la serie de Fibonacci en el que se haga uso de la clase **SwingWorker**, esto para permitir que nuestra interfaz de grafica no se congele cuando realicemos un calculo que pueda tardar una cantidad considerable de tiempo.

**CODIGO FUENTE:**

**Clase FibonacciGUI**

package com.milkyblue;

import java.awt.*Color*;

import java.awt.*GridLayout*;

import java.awt.event.*ActionEvent*;

import java.awt.event.*ActionListener*;

import javax.swing.*JButton*;

import javax.swing.*JFrame*;

import javax.swing.*JLabel*;

import javax.swing.*JPanel*;

import javax.swing.*JTextField*;

import javax.swing.border.*LineBorder*;

import javax.swing.border.*TitledBorder*;

// FibonacciGUI class. Models the GUI.

public class *FibonacciGUI* {

  private *int* n1, n2, count;

  private *JFrame* mainFrame;

  private *JPanel* workerPanel, threadEventsPanel;

  private *JTextField* txtN;

  private *JButton* btnStart, btnNextN;

  private *JLabel* lblFibonacci, lblN, lblNFibonacci;

  // Class constructor.

  public FibonacciGUI() {

    n1 = 0;

    n2 = 1;

    count = 1;

    mainFrame = **new** JFrame("Fibonacci Swing Worker");

    workerPanel = **new** JPanel(**new** GridLayout(2, 2, 5, 5));

    threadEventsPanel = **new** JPanel(**new** GridLayout(2, 2, 5, 5));

    txtN = **new** JTextField();

    btnStart = **new** JButton("Start");

    btnNextN = **new** JButton("Next number");

    lblFibonacci = **new** JLabel();

    lblN = **new** JLabel("Fibonacci of 1: ");

    lblNFibonacci = **new** JLabel(*String*.valueOf(n2));

    // Main methods are called.

    addAttributes();

    addListeners();

    build();

    launch();

  }

  // Adds attributes to elements in the class.

  private *void* addAttributes() {

    mainFrame.setLayout(**new** GridLayout(2, 1, 10, 10));

    mainFrame.setSize(275, 200);

    // Sets a border with a title on each panel.

    workerPanel.setBorder(**new** TitledBorder(**new** LineBorder(*Color*.BLACK), "With SwingWorker"));

    threadEventsPanel.setBorder(**new** TitledBorder(**new** LineBorder(*Color*.BLACK), "Without SwingWorker"));

    mainFrame.setDefaultCloseOperation(*JFrame*.EXIT\_ON\_CLOSE);

    mainFrame.setResizable(false);

  }

  // Adds listeners to elements in GUI.

  private *void* addListeners() {

    // Calculates the desired nth fibonacci number with a FibonacciBackground

    // instance based in a SwingWorker, when its done, lblFibonacci is updated.

    btnStart.addActionListener(**new** ActionListener() {

      public *void* actionPerformed(*ActionEvent* *e*) {

*int* n;

        try {

          n = *Integer*.parseInt(txtN.getText());

        } catch (*NumberFormatException* *error*) {

          lblFibonacci.setText("Input an Integer number.");

          return;

        }

        lblFibonacci.setText("Processing...");

*FibonacciBackground* task = **new** FibonacciBackground(n, lblFibonacci);

        task.execute();

      }

    });

    // Calculates the next fibonacci number based on the previous.

    btnNextN.addActionListener(**new** ActionListener() {

      public *void* actionPerformed(*ActionEvent* *e*) {

*int* temp = n1 + n2;

        n1 = n2;

        n2 = temp;

        ++count;

        lblN.setText("Fibonacci of " + count + ": ");

        lblNFibonacci.setText(*String*.valueOf(n2));

      }

    });

  }

  // Builds the GUI.

  private *void* build() {

    workerPanel.add(**new** JLabel("Get fibonacci of: "));

    workerPanel.add(txtN);

    workerPanel.add(btnStart);

    workerPanel.add(lblFibonacci);

    threadEventsPanel.add(lblN);

    threadEventsPanel.add(lblNFibonacci);

    threadEventsPanel.add(btnNextN);

    mainFrame.add(workerPanel);

    mainFrame.add(threadEventsPanel);

  }

  // Launches mainFrame by setting its visible value to true.

  private *void* launch() {

    mainFrame.setVisible(true);

    mainFrame.pack();

    mainFrame.setLocationRelativeTo(null);

  }

}

**Clase FibonacciBackground**

package com.milkyblue;

import java.util.concurrent.*ExecutionException*;

import javax.swing.*JLabel*;

import javax.swing.*SwingWorker*;

// FibonacciBackground class. Extends from SwingWorker, calculates

// the nth fibonacci number and updates a JLabel object.

public class *FibonacciBackground* extends *SwingWorker*<*String*, *Object*> {

  private final *int* n;

  private final *JLabel* lblResult;

  // Class constructor, takes the desired nth fibonacci number and a reference to

  // the target

  // JLabel.

  public FibonacciBackground(*int* *n*, *JLabel* *lblResult*) {

*this*.n = n;

*this*.lblResult = lblResult;

  }

  // Generates the desired nth fibonacci number in background.

  protected *String* doInBackground() throws *Exception* {

*long* nFib = fibonacci(n);

    return *String*.valueOf(nFib);

  }

  // When doInBackground is done, lblResult is updated to nth fibonacci number's

  // value.

  protected *void* done() {

    try {

      lblResult.setText(get());

    } catch (*InterruptedException* *e*) {

      lblResult.setText("Interrupted while waiting for results.");

    } catch (*ExecutionException* *e*) {

      lblResult.setText("An error was found.");

    }

  }

  // Fibonacci recursive method.

  public *long* fibonacci(*long* *n*) {

    if (n == 0 || n == 1)

      return n;

    else

      return fibonacci(n - 1) + fibonacci(n - 2);

  }

}

**Clase App**

package com.milkyblue;

// App class.

public class *App* {

    // Creates an anonymous instance of FibonacciGUI.

    public static *void* main(*String*[] *args*) {

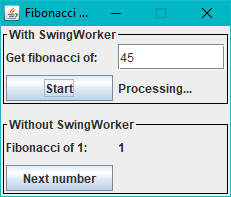
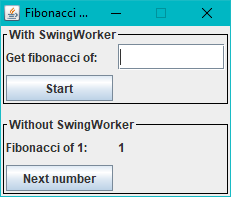
**new** FibonacciGUI();

    }

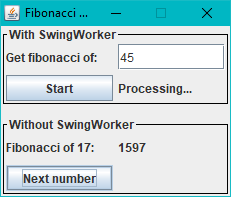
}

**CAPTURAS:**

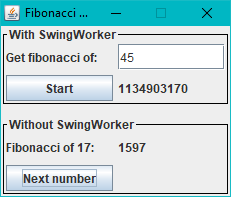
**ESTADO INICIAL DE LA INTERFAZ.**



**CALCULO RECURSIVO ES EJECUTADO EN EL SWINGWORKER.**



**MIENTRAS TANTO, LAS OTRAS PARTES DE LA INTERFAZ PUEDEN SER USADAS SIN PROBLEMA.**



**FINALMENTE, EL PROCESO DEL SWINGWORKER TERMINA Y REALIZA CAMBIOS EN LA INTERFAZ.**

**NOTAS:**

* Puede encontrar el repositorio de este proyecto en mi cuenta de github en el siguiente enlace: <https://github.com/NoisyApple/AdTopics-19.SwingWorker>