Cronómetro

Alumno: Aarón Gutiérrez Domínguez
Curso: 2oDAM
Modulo: Programación de servicios y procesos
Fecha: 20 de Noviembre 2019

Práctica a resolver:

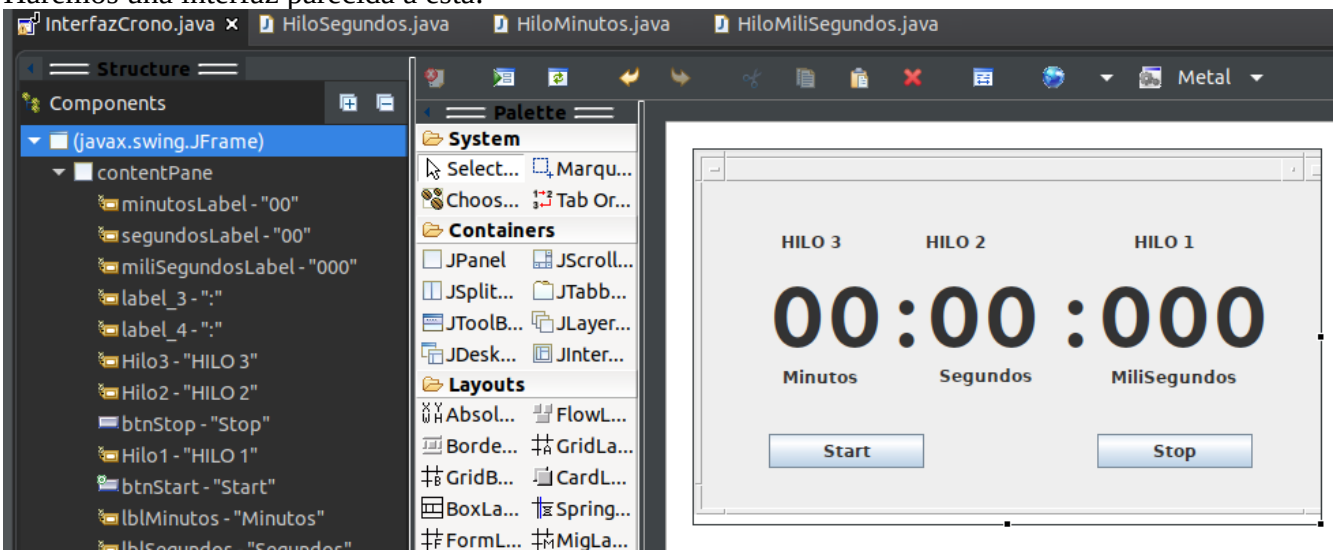
Realiza una especie de reloj o cronómetro en Java, en un entrono gráfico, que utilice la interfaz Runnable para crear los hilos necesarios que se encarguen de actualizar los dígitos que aparecen en pantalla.

Análisis:

Para esta actividad será necesario crear una interfaz. Con el IDE Eclipse 2019 se utilizara Swing para la crear la GUI del programa. Se emplearan las clases de la librería Swing Jframe, Jpanel, Jbutton y el mas importante Jlabel que nos permite cambiar su valor, el cual se peude utlizar para mostrar el tiempo en el cronómetro.
Se empleará también la clase Thread de JAVA para actualizar los digitos que modificaran el Jpanel.
El objetivo es poder con un botón “Start” iniciar el cronómetro y con el “Stop” pararlo.
El problema viene con el “Stop” ya que el método stop() de la Clase Thread esta deprecated ya no se recomienda.
Oracle recomienda utilizar un AtomicBoolean para iniciar/para un Thread utlizándolo como un Flag.
Sera necesario crear nuestros propios métodos stop() y start() que utilizan la Flag para iniziar el proceso y en caso de pararlo lanzara una “InterruptedException” que saltara en el proceso que estamos ejecutando.
A partir de ahí podemos utilizar la excepcion para usar el método Thread.currentThread.Interupt()
De esta manera se elimina el proceso.
Para actualizar los dígitos los Thread pueden utilizar la función sleep() que como parámetro usa un int representado cada mili-segundo.
Se necesitarán tres clases para cada hilo (minuto, segundo , mili-segundo) que tendrán que implementar la interfaz Runnable que obliga implementar

Diseño y desarrollo:

Primero generamos una interfaz creando una clase Jframe dentro de Eclipse (para ello es necesario descargar del MarketPlace de Eclipse la librería WindowBuilder 1.9.1, la cual contiende un diseñador de interfaces que utiliza Swing).
Haremos una interfaz parecida a esta:



En ella tendremos dos botones que dentro del Main que inicia y para el cronómetro. Hay que tener en cuenta que el actionListener del boton stop debe estar dentro del actionListener del start.
Dentro del actionListener del start instanciamos los hilos y con el boton stop hacemos una llamada a las funciones stop() que interrumpen los hilos.
A continuación creamos tres JLabel los cuales le pasaremos al hilo para que controle la salida en la interfaz.
Ahora se tienen que crear 3 clases que implementen la interfaz Runnable. Una para Minutos otra para Segundos y otra para Mili-segundos.

El diseño para las 3 clases es parecido:

Tienen un Thread el cual activamos y desactivamos con el Flag.
Un contador(Min,Seg o MiliSeg).
Un constructor al cual le pasamos un JLabel (se le pasa el de la interfaz para modificarlo desde run()).
El constructor inicia el contador a 0. Después enlaza el JLabel con el Thread.
El booleano Atómico nos sirve ahora para crear los métodos start() y stop() dentro de cada clase.

Luego tiene un método start() en el cual creamos e iniciamos el Hilo.
A continuación se sobrescribe el método run(). Aquí es donde nos viene de utilidad el booleano atómico.
Al principio del run() iniciamos el booleano atómico a true (este tipo de booleano ya tiene getter y setter)y en el bucle del “while” en vez de true utilizamos el método get del booleano.
En el bucle implementamos la función sleep() y actualizamos el valor del Jpanel que pasa pore l constutor.
Es necesario utilizar un try-catch para tratar el hilo en concreto al saltar la excepción “InterruptedException” en el cual podremos para el interrumpir el hilo.
Se crea un método stop() que cambia el valor del flag a false.

Código Fuente:

Clase MAIN con GUI

```
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JButton;
import javax.swing.JLabel;
import java.awt.Font;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class InterfazCrono extends JFrame {
	private JPanel contentPane;
	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					InterfazCrono frame = new InterfazCrono();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}

			}
		});
	}

	/**
	 * Create the frame.
	 */
	public InterfazCrono() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 670, 267);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);
		contentPane.setLayout(null);

		//START inizia los trheads que haran de contadores
		/*
		 *
		 *   Hilo 1 == contador milisegundos
		 *   Hilo 2 == contador segundos
		 *   Hilo 3 == contador minutos
		 *
		 *
		 * */

		JLabel minutosLabel = new JLabel("00");
```

```
                minutosLabel.setFont(new Font("Courier 10 Pitch", Font.BOLD, 70));
                minutosLabel.setBounds(50, 67, 103, 65);
                contentPane.add(minutosLabel);

                JLabel segundosLabel = new JLabel("00");
                segundosLabel.setFont(new Font("Courier 10 Pitch", Font.BOLD, 70));
                segundosLabel.setBounds(165, 67, 103, 65);
                contentPane.add(segundosLabel);

                JLabel miliSegundosLabel = new JLabel("000");
                miliSegundosLabel.setFont(new Font("Courier 10 Pitch", Font.BOLD, 70));
                miliSegundosLabel.setBounds(291, 67, 201, 65);
                contentPane.add(miliSegundosLabel);

                JLabel label_3 = new JLabel(":");
                label_3.setFont(new Font("Bitstream Charter", Font.BOLD, 70));
                label_3.setBounds(139, 66, 66, 55);
                contentPane.add(label_3);

                JLabel label_4 = new JLabel(":");
                label_4.setFont(new Font("Bitstream Charter", Font.BOLD, 70));
                label_4.setBounds(263, 66, 66, 55);
                contentPane.add(label_4);

                JLabel Hilo3 = new JLabel("HILO 3");
                Hilo3.setBounds(59, 30, 66, 25);
                contentPane.add(Hilo3);

                JLabel Hilo2 = new JLabel("HILO 2");
                Hilo2.setBounds(165, 30, 66, 25);
                contentPane.add(Hilo2);

                JLabel Hilo1 = new JLabel("HILO 1");


                //STOP BUTTON
                JButton btnStop = new JButton("Stop");

                btnStop.setBounds(291, 183, 114, 25);
                contentPane.add(btnStop);


                //START BUTTON
                Hilo1.setBounds(318, 30, 66, 25);
                contentPane.add(Hilo1);
                JButton btnStart = new JButton("Start");
                btnStart.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                //Creamos e Iniziamos Hilos del cronometro
                                HiloMiliSegundos milisegundosHilo=new
HiloMiliSegundos(miliSegundosLabel);
                                milisegundosHilo.start();
                                HiloSegundos segundosHilo=new HiloSegundos(segundosLabel);
                                segundosHilo.start();
                                HiloMinutos minutosHilo=new HiloMinutos(minutosLabel);
                                minutosHilo.start();



                                //SI APRETAMOS STOP SE INTERRUMPEN LOS THREADS
                                btnStop.addActionListener(new ActionListener() {
                                        public void actionPerformed(ActionEvent e) {
                                                milisegundosHilo.stop();
                                                segundosHilo.stop();
                                                minutosHilo.stop();
                                        }
                                });

                        }
                });
                btnStart.setBounds(50, 183, 114, 25);
                contentPane.add(btnStart);



        }
}
```

Clase HiloMilisegundos

```
import java.util.concurrent.atomic.AtomicBoolean;

import javax.swing.JLabel;

public class HiloMiliSegundos implements Runnable{
    Thread hiloMilisegundos;
        JLabel labelMiliSegundos;
        int contadorMiliSegundos;
        private final AtomicBoolean running = new AtomicBoolean(false);//FLAG

        boolean ejecutar;
    public HiloMiliSegundos(JLabel labelSegundos) {
        this.labelMiliSegundos=labelSegundos;
        this.contadorMiliSegundos=0;
        }

    //PARA PARAR EL THREAD
    public void stop() {
        running.set(false);
    }
    //Genera nuevo hilo y lo inizia
    public void start() {
        hiloMilisegundos = new Thread(this);
        hiloMilisegundos.start();
    }
        @Override
        public void run() {
                running.set(true);
                while(running.get()) {

                                try {

                                    if(contadorMiliSegundos<100){

labelMiliSegundos.setText("0"+String.valueOf((contadorMiliSegundos++)));
                                        }else {

labelMiliSegundos.setText(String.valueOf((contadorMiliSegundos++)));
                                        }
                                        //Para cada milisegundo
                                        Thread.sleep(1);


                                } catch (InterruptedException e) {
                                        //Cambiando el estado del flag desde el boton Stop obligamos
que salte la expepcion
                                        //Y como esa excepcion ocurre en el mismo trhead actual podemos
utilizar el metodo de interrupt().
                                        Thread.currentThread().interrupt();
                                        e.printStackTrace();

                                }
                                if(contadorMiliSegundos>=1000) {
                                        contadorMiliSegundos=0;
                                }


                }
        }



}
```

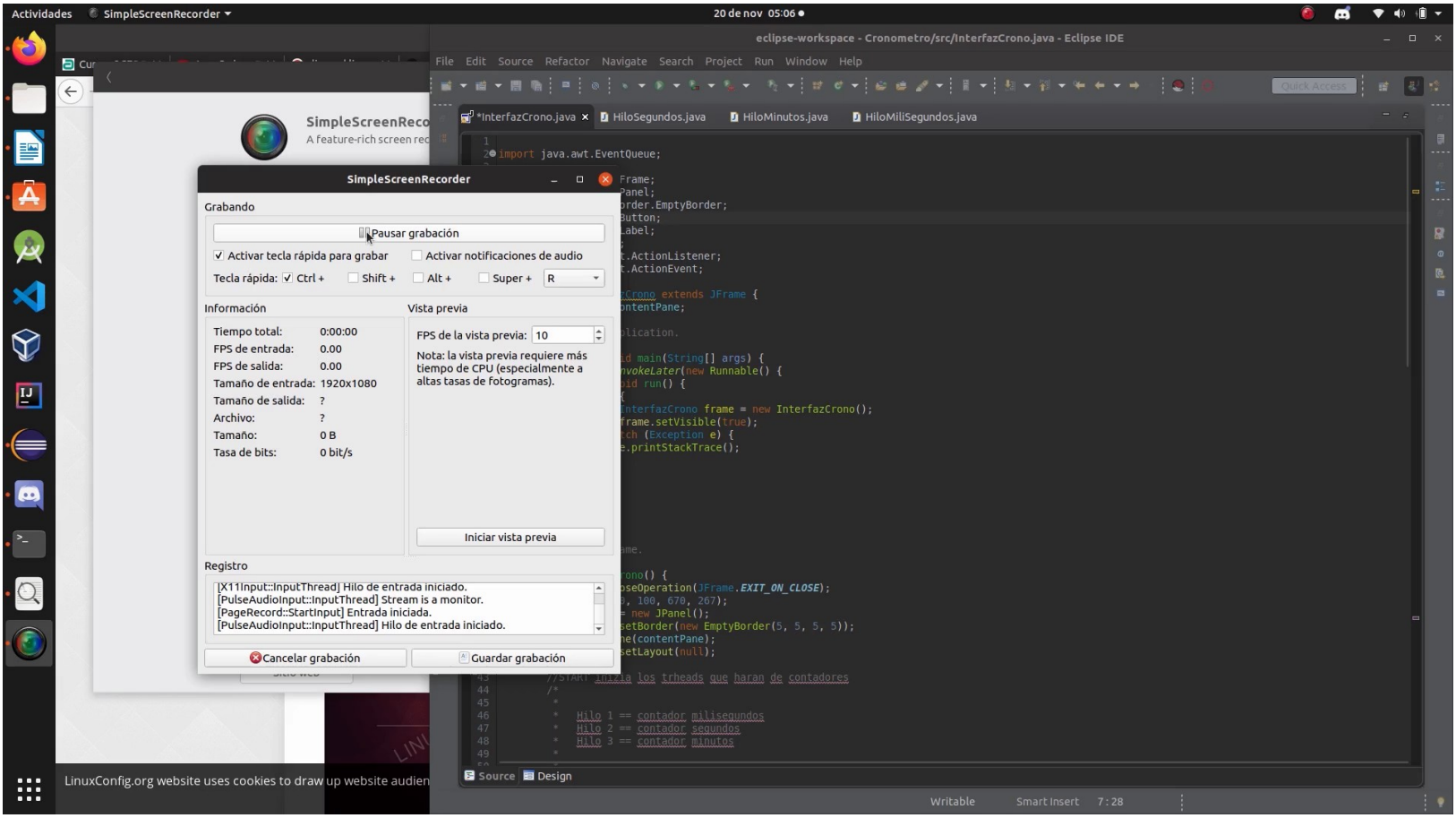
Clase HiloSegundos

```
import java.util.concurrent.atomic.AtomicBoolean;


import javax.swing.JLabel;

public class HiloSegundos implements Runnable{
    Thread hilo1;
        JLabel labelSegundos;
        int contadorSegundos;
        private final AtomicBoolean running = new AtomicBoolean(false);//FLAG

    public HiloSegundos(JLabel labelSegundos) {
        //this.hilo1=new Thread(this,"HiloSegundos");
        this.labelSegundos=labelSegundos;
        this.contadorSegundos=0;
        }

    //PARA PARAR EL THREAD
    public void stop() {
        running.set(false);
    }
    //Genera nuevo hilo y lo inizia
    public void start() {
        hilo1 = new Thread(this);
        hilo1.start();
    }
        @Override
        public void run() {
                running.set(true);//flag
                while(running.get()) {

                                try {
                                        //Para que de 0 a 10 se imprima con 2 digitos
                                        if(contadorSegundos<10) {

labelSegundos.setText("0"+String.valueOf(contadorSegundos++));
                                        }else{
                                                labelSegundos.setText(String.valueOf(contadorSegundos+
+));
                                        }
                                        //Para cada Segundo
                                        Thread.sleep(1000);


                                } catch (InterruptedException e) {
                                        //Cambiando el estado del flag desde el boton Stop obligamos
que salte la expepcion
                                        //Y como esa excepcion ocurre en el mismo trhead podemos
utilizar el metodo de interrupt().
                                        Thread.currentThread().interrupt();//interrumpimos el hilo
                                        e.printStackTrace();

                                }
                                if(contadorSegundos>=60) {
                                        contadorSegundos=0;
                                }


                }
        }



}
```

Clase HiloMinutos

```
import java.util.concurrent.atomic.AtomicBoolean;

import javax.swing.JLabel;

public class HiloMinutos implements Runnable{
    Thread hilo2;
        JLabel labelMinutos;
        int contadorMinutos;
        private final AtomicBoolean running = new AtomicBoolean(false);//FLAG

        boolean ejecutar;
    public HiloMinutos(JLabel labelMin) {
    //  this.hilo2=new Thread(this,"HiloMinutos");
        this.labelMinutos=labelMin;
        this.contadorMinutos=0;
        }

    //PARA PARAR EL THREAD
    public void stop() {
        running.set(false);
    }
    //Genera nuevo hilo y lo inizia
    public void start() {
        hilo2 = new Thread(this);
        hilo2.start();
    }
        @Override
        public void run() {
                running.set(true);
                while(running.get()) {

                                try {
                                        //Para que de 0 a 10 se imprima con 2 digitos
                                        if(contadorMinutos<10) {

labelMinutos.setText("0"+String.valueOf(contadorMinutos++));
                                        }else{
                                                labelMinutos.setText(String.valueOf(contadorMinutos+
+));
                                        }
                                        //Para cada minuto
                                        Thread.sleep(60000);


                                } catch (InterruptedException e) {
                                        //Cambiando el estado del flag desde el boton Stop obligamos
que salte la expepcion
                                        //Y como esa excepcion ocurre en el mismo trhead actual podemos
utilizar el metodo de interrupt().
                                        Thread.currentThread().interrupt();
                                        e.printStackTrace();

                                }
                                if(contadorMinutos>=60) {
                                        contadorMinutos=0;
                                }


                }
        }


}
```

Captura de video



Documentación

https://docs.oracle.com/javase/1.5.0/docs/guide/misc/threadPrimitiveDeprecation.html

https://www.baeldung.com/java-thread-stop

Programas utilizados

Eclipse 2019 + WindowBuilder 1.9.1(Swing) . SimpleScreenRecorder(Ubuntu) parar capturar el resultado.