



UAEM

Universidad Autónoma
del Estado de México

CENTRO UNIVERSITARIO UAEM ZUMPANGO

Ingeniería en computación



Programación Avanzada

LIRA ALANIS ESTEBAN

#Cuenta : 1624256

Profesor: Asdrúbal López Chau

3 / Noviembre / 2018



UAEM

Universidad Autónoma
del Estado de México

Índice

Índice	2
Introducción	2
Desarrollo	3
Conclusiones.	8
Referencias	8

Introducción

La carrera de labels es un proyecto que da lugar a un trabajo de lleno de componentes, métodos y trucos, que llevan un cierto orden con variables que tienen un motivo en específico con respecto a los threads como lo son las funciones sincronizadas entre si con una cierta cantidad de variables y haciendo uso de las llamadas barreras cíclicas de java para Threads clásicos de java swing.

Java pone lo mejor de lo que tiene a sus proyectos y lo interesantes que son los hilos el caso de poder tener varios por decirlo así main's para que hagan cada uno una función específica en palabras comunes "que cada quien haga su chamba" y para el uso de la barrera cíclica "el que termine primero espera que los demás terminen".

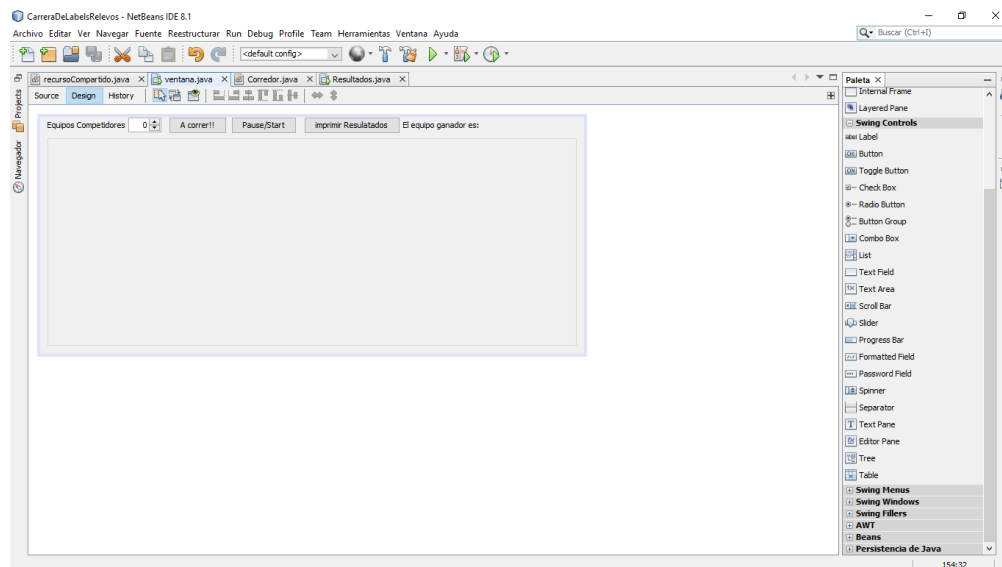


UAEM

Universidad Autónoma
del Estado de México

Desarrollo

Primero fue necesario crear el proyecto de netbeans y con una Clase JFrame para la GUI y que esta carrera sea emocionante viendo cómo se rebasan una y otra pero para ello es necesario de vez en cuando usar y bueno es más cómodo usar la paleta de componentes pero en el ámbito grafico solo colocaremos los controles



Estos constan de un par de labels para decirle al usuario de que está manipulando y tres botones que están destinados para lanzar los hilos que controlan las labels y otro para detener la carrera y poder reanudarla y otro para mostrar una tabla con las posiciones una vez terminada la carrera



UAEM

Universidad Autónoma
del Estado de México

```
132 private void jButtonActionPerformed(java.awt.event.ActionEvent evt) {  
133     recursoCompuesto recurso = new recursoCompuesto(jLabel1);  
134     RecursoRelevo recursoR = new RecursoRelevo(jLabel1);  
135  
136     a = (int) jSpinner1.getValue();  
137     CyclicBarrier barrier = new CyclicBarrier(a+1);  
138     tr = new Corredor(a);  
139     jPanel1.removeAll();  
140     jPanel1.setLayout(null);  
141     JLabel[] Jc = new JLabel[a];  
142     JLabel[] J2 = new JLabel[a];  
143     for (int i = 0; i < a; i++) {  
144         Jc[i] = new JLabel("Labels" + (i + 1));  
145         Jc[i].setSize(new Dimension(100, 40));  
146         Jc[i].setLocation(0, i * 15);  
147         jPanel1.add(Jc[i]);  
148  
149         J2[i] = new JLabel("Labels" + (i + 1));  
150         J2[i].setSize(new Dimension(100, 40));  
151         J2[i].setLocation(((this.getWidth() - 105) / 2), i * 15);  
152         jPanel1.add(J2[i]);  
153     }  
154     jPanel1.updateUI();  
155  
156     for (int i = 0; i < Jc.length; i++) {  
157         tr[i] = new Corredor(Jc[i], J2[i], recurso, recursoR, (this.getWidth() - 105) / 2, barrier);  
158     }  
159     for (int i = 0; i < Jc.length; i++) {  
160         tr[i].Start();  
161     }  
162 }  
163  
164 private void jButtonActionPerformed(java.awt.event.ActionEvent evt) {
```

El botón que genera la llamada de los Threads es “A correr” que es este Tramo de código crea los hilos y los echa a andar claro antes creamos las labels que controlaran y le asignamos el equipo de dos Threads al arreglo de objetos de la clase entonces ahora será un arreglo de equipos de Threads Corredor como ya se dijo asignando sus recursos como el que ocupa para el relevo y para que obtenga la bandera de ganador y que sepa cuan larga es la carrera y por supuesto la barrera cíclica.

```
31 public class Corredor implements Runnable {  
32  
33     /**  
34      * variables de necesarias para los Threads  
35      */  
36  
37     int w;  
38     int a = 0;  
39     volatile boolean pausa = false;  
40     JLabel label;  
41     JLabel label2;  
42     recursoCompuesto RecursosCompuesto;  
43     RecursoRelevo recursoRelevo;  
44     private int ancho;  
45     int auxAvance1, auxAvance2;  
46     CyclicBarrier b;  
47     static Resultados r;  
48  
49     /**  
50      * Constructor de los Hilos  
51      * @param label  
52      * @param label2  
53      * @param RecursosCompuesto  
54      * @param recursoRelevo  
55      * @param ancho  
56      * @param b  
57      */  
58  
59     public Corredor(JLabel label, JLabel label2, recursoCompuesto RecursosCompuesto, RecursoRelevo recursoRelevo, int ancho, CyclicBarrier b) {  
60         this.label = label;  
61         this.label2 = label2;  
62         this.RecursosCompuesto = RecursosCompuesto;  
63         this.recursoRelevo = recursoRelevo;  
64         this.ancho = ancho;
```



UAEM

Universidad Autónoma
del Estado de México

Una vez creado la GUI se continúa con la parte más importante de estos proyectos como lo es programar el Thread.

Primero Creamos el Constructor con todos sus variables que necesitamos de la GUI además se incorporan unas variables como una JFrame de Resultado donde se alojan los resultados y un par de contadores que se ocupan para definir unos bucles for asi como un variable boolean que es la que se encarga de detener o reanudar un Thread.

La parte que más causa conflictos como es la función run la cual reescribimos con el proceso que el Thread va a ejecutar y pues para no hacerla tan robusta se creó una función llamada proceso que contiene todo lo que el Thread necesita hacer como lo es crear una variable random para que de manera aleatoria se mueva la label una cierta distancia asi mismo que el hilo que se esté ejecutando duerma un momento para que los demás puedan avanzar y cuando termine de llegar a la meta definida por ancho esté llegue y tome la FlagWinner de la variable sincronizada y en la barrea ciclica ahí espere a los demás hilos .

```
37  
38  
39  
40  
41 public recursoCompartido(JLabel jLabel) {  
42     this.jLabel = jLabel;  
43 }  
44  
45 public int Lugar() {  
46     return 0;  
47 }  
48  
49 public boolean isFlagWinner() {  
50     return flagWinner;  
51 }  
52  
53 public synchronized void setFlagWinner(boolean flagWinner) {  
54     this.flagWinner = flagWinner;  
55 }  
56 public synchronized void setPau(boolean pau) {  
57     this.pau = pau;  
58 }  
59 public boolean isPau() {  
60     return pau;  
61 }  
62  
63 public String getNameWinner() {  
64     return NameWinner;  
65 }  
66  
67 public synchronized void setNameWinner(String NameWinner) {  
68     if (!isFlagWinner()) {  
69         this.NameWinner = NameWinner;  
70         flagWinner = true;  
71     }  
72 }
```

En la Parte de la Clase de RecursoCompartido es la clase que aloja a la función y variable que están sincronizadas con los hilos en ejecución incluso con el hilo principal utilizadas para hacer exclusión mutua y que solo haya un ganador.

De la misma clase se obtiene en orden el lugar y nombre en el que las labels llegan a la meta y estos datos son ocupados para enviárselos a la clase Resultados para mostrarlos en la tabla de posiciones.



UAEM

Universidad Autónoma
del Estado de México

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
```

```
/**
 *
 * @author esteb
 */
public class RecursoRelevo {

    private boolean flagWinner = false;
    private String NameWinner;
    private JLabel jLabel;

    public RecursoRelevo(JLabel jLabel) {
        this.jLabel = jLabel;
    }

    public boolean isFlagRelevo() {
        return flagWinner;
    }

    public synchronized void setFlagRelevo(boolean flagWinner) {
        this.flagWinner = flagWinner;
    }

    public String getRelevo() {
        return NameWinner;
    }

    public synchronized void setRelevo(String NameWinner) {
        flagWinner = true;
    }
}
```

La clase recursoRelevo hace algo parecido a RecursoCompartido ya que esta solo les dice a los hilos que su pareja ya llego y que ya pueden avanzar para esta Clase se ocupó la misma estructura de la clase RecursoCompartido ya que hacen cosas parecidas.

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116
```

```
public void resultados() {
    initComponents();
}

/** funcion para agregar las posiciones a la tabla
 *
 * @param a
 * @param b
 */

public void UP(String a,int b){
    DefaultTableModel o1 = (DefaultTableModel)jTable1.getModel();
    o1.addRow(new Object[]{a,b});
}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings({"unchecked"})
Generated Code

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)
    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            //new Resultados().setVisible(true);
        }
    });
}
```



UAEM

Universidad Autónoma
del Estado de México

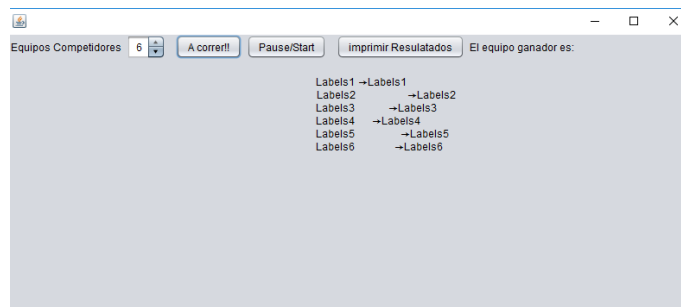
En la clase Resultados se reciben de la clase ventana desde el botón de Imprimir resultado el nombre del equipo y su puesto y son agregados a la tabla con la función UP con un string y un int de parámetros.

Funciona así:

Colocamos 6 Labels en el Jspinner y a continuación damos clic en “A correr”.



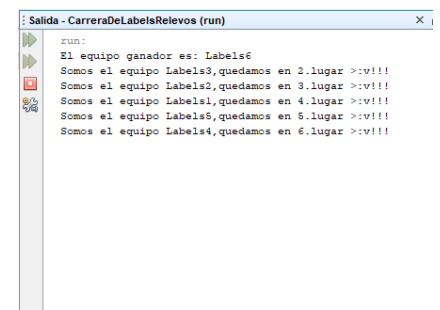
Así comienza el relevo sale el segundo grupo.



Y gano solo una y fue el equipo de labels6 y así son las estadísticas en la tabla de posiciones y en la consola de Netbeans.



Equipo	Lugar
Labels6	1
Labels3	2
Labels2	3
Labels1	4
Labels5	5
Labels4	6





UAEM

Universidad Autónoma
del Estado de México

Conclusiones.

Es interesante la forma de trabajar con Threads y con CyclicBarriers así como con exclusión mutua que en hilos de programación avanzada bueno son unos trucos que facilitan en cierto modo la programación paralela de subprocesos que son bastante eficientes si son usados de la manera más correcta y bueno el uso de esto es como ya lo vimos un juego y como no puede dar lugar a mas juegos con threads aleatorios.

Incluso mientras uno esta ejecutando una tarea otros ya hicieron dos procesos cabe decir que es muy curiosos que la clase main es un Thread y en las barreras cíclicas debe tomarse en cuenta. La exclusión mutua también tiene múltiples usos como en bancos, en cuentas de clubes de ocio, juegos y aplicaciones móviles.

Referencias

<https://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/>

<https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html>