DAD-1: Hilos

Definición

Un Hilo es la secuencia única de control de flujo dentro del programa. Suele ser la unidad de código más pequeña que se puede ejecutar.

Un hilo se puede implementar heredando la clase Thread o implementando la interfaz run().

Heredando Clase Thread

Por ejemplo, este hilo que vamos a implementar lo que hará será que cada vez que sea llamando repetir múltiples veces algo con cierto retraso entre ejecución.

```
public class SimpleThread extends Thread{
  //Atributos:
  protected long retraso;
  protected int veces;
  //constructor del hilo, indicamos cuantas veces se repite y cada cuanto
  public SimpleThread (long retraso, int veces){

    System.out.println("Constructor del Hilo Simple");
    this.retraso = retraso;
    this.veces = veces;
  }
  //Definimos que hará el hilo durante su ejecución:
  public void run(){
    try{
      for(int i = veces; veces >= 0; veces--){
        System.out.println("Cooldown de: " + retraso);
        sleep(retraso);
      }
      System.out.println("Fin de la ejecucion del hilo")
    }catch(Exception e){
      System.out.println("Error en el Hilo: " + e);
    }
  }
}
```

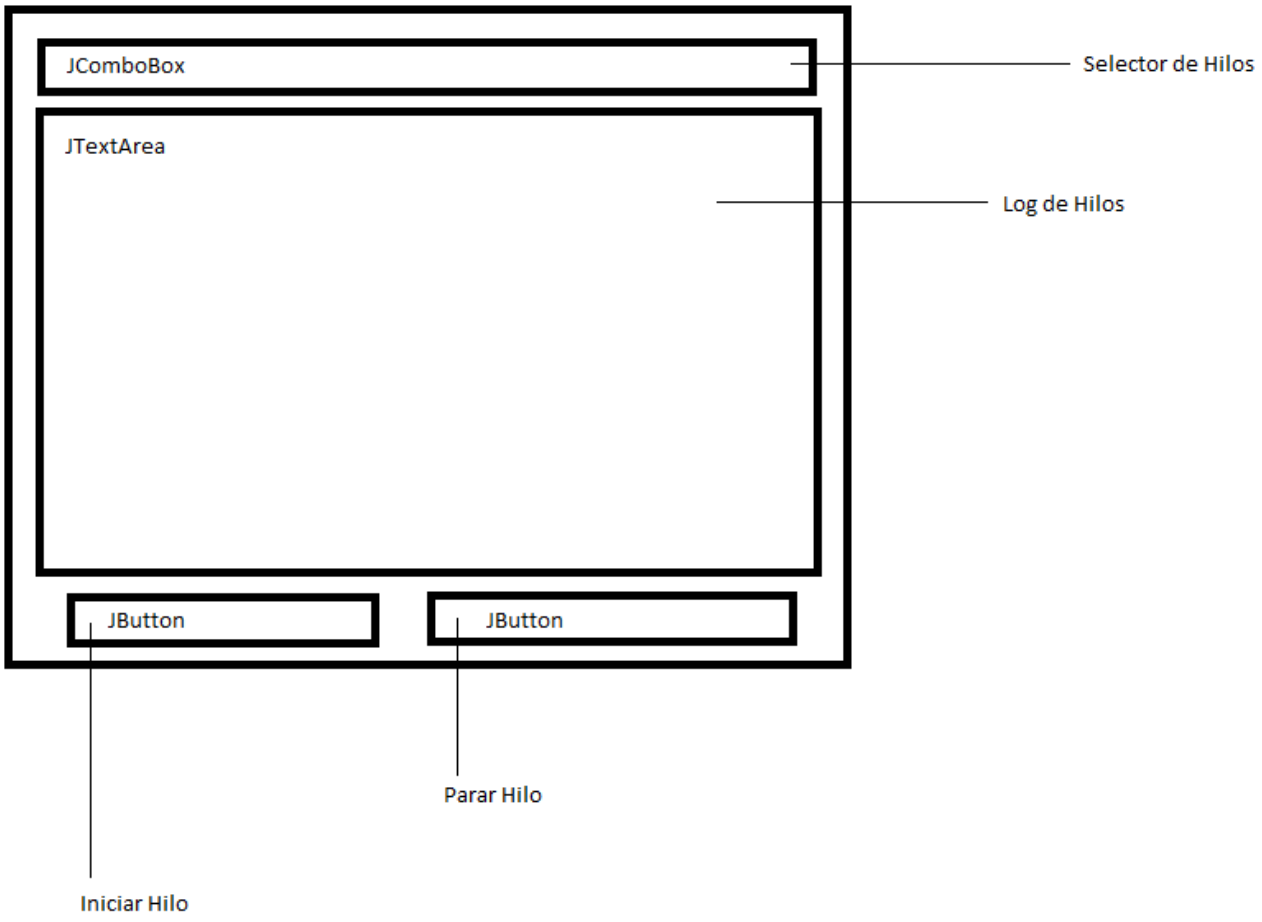
OJO: Para ejecutar el hilo usamos .start(), no .run().

En la práctica

Un hilo es una unidad única de control, para implementar un hilo necesitamos un objeto de la clase

thread.

El Programa que queremos crear deberá poder arrancar y parar hilos, de forma que podamos seleccionar el hilo a arrancar con un JComboBox y muestre en un log como van los hilos dentro de un JTextArea



Declaración de Hilos

```
private void initialize() {
    SimpleThread st = new SimpleThread(2000, SimpleThread.FOR_EVER);//Iniciamos hilo infinito con delay de 2 segundos
    frame = new JFrame();
```

Ejemplo de Funcionalidad Botón Lanzar Hilo

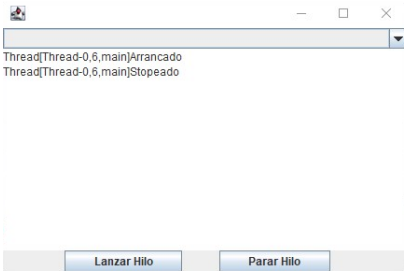
```java
JButton btnNewButton = new JButton("Lanzar Hilo");
btnNewButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        System.out.println("Iniciando Hilo...");//Mostramos mensaje en consola
        try {
            st.start();//Arrancamos Hilo
            textArea.append(st + "Arrancado");//Mostramos en el log el arranque del Hilo
        }catch(Exception e1){
            textArea.append(st + "Fallo de Arranque");//En caso de que falle esto en el log
        }

    }
});
```

Ejemplo de Funcionalidad Botón Parar Hilo

```java
JButton btnNewButton_1 = new JButton("Parar Hilo");
btnNewButton_1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        System.out.println("Parando hilo");
        try {
            st.interrupt();//Paramos hilo
            textArea.append("\n" + st + "Stopeado\n");
        }catch(Exception e1){
            textArea.append("\n" + st + "Error en la Parada\n");
        }

    }
});
```



Ejemplo de Funcionalidad ComboBox

```java
JComboBox<String> comboBox = new JComboBox<>(NombreHilos.getNombreHilosArray());//comboBoxTipos de Hilo
comboBox.setBounds(10, 11, 322, 32);
contentPane.add(comboBox);
```

NombreHilos sería:

```java
public enum NombreHilos {
    HILO_1("MiHilo"), HILO_2("MiHilo2");

    private final String nombreHilo;

    private NombreHilos(String nombreHilo) {
        this.nombreHilo = nombreHilo;
    }

    public String getNombreHilo() {
        return nombreHilo;
    }

    public static String[] getNombreHilosArray() {
        NombreHilos[] values = NombreHilos.values();
        String[] nombresHilosArray = new String[values.length];

        for (int i = 0; i < values.length; i++) {
            nombresHilosArray[i] = values[i].getNombreHilo();
        }

        return nombresHilosArray;
    }

}
```

From:
https://knoppia.net/ - **Knoppia**

Permanent link:
https://knoppia.net/doku.php?id=dad1:hilos

Last update: **2023/11/16 07:53**