Investigación Hilos

Una thread es un único flujo de control dentro de un programa. Algunas veces es llamado contexto de ejecución porque cada thread debe tener sus propios recursos, como el program counter y el stack de ejecución, como el contexto de ejecución. Sin embargo, toda thread en un programa aún comparte muchos recursos, tales como espacio de memoria y archivos abiertos. Threads también son llamadas procesos livianos (lightweight process).

Creación y ejecución de Threads

Hay dos formas de hacer una tarea correr concurrentemente con otra: crear una nueva clase como subclase de la clase Thread o declarar una clase e implementar la interfaz Runnable.

Uso de Subclase

Cuando se crea una subclase de Thread, la subclase debería definir su propio método run() para sobre montar el método run() de la clase Thread. La **tarea concurrente es desarrollada en este método run**().

Ejecución del método run()

Una instancia de la subclase es creada con new, luego llamamos al método start() de la thread para hacer que la máquina virtual Java ejecute el método run(). Ojo para iniciar la concurrencia invocamos a start(), así invocamos a run() en forma indirecta. Si invocamos a run() directamente, se comportará como el llamado a cualquier método llamado dentro de un mismo hilo (sin crear uno independiente).

Implementación de la Interfaz Runnable

La interfaz Runnable requiere que sólo un método sea implementado, el método run(). Primero creamos una instancia de esta clase con new, luego creamos una instancia de Thread con otra sentencia new y usamos el objeto recién creado en el constructor. Finalmente, llamamos el método start() de la instancia de Thread para iniciar la tarea definida en el método run().

Control de la Ejecución de una Thread

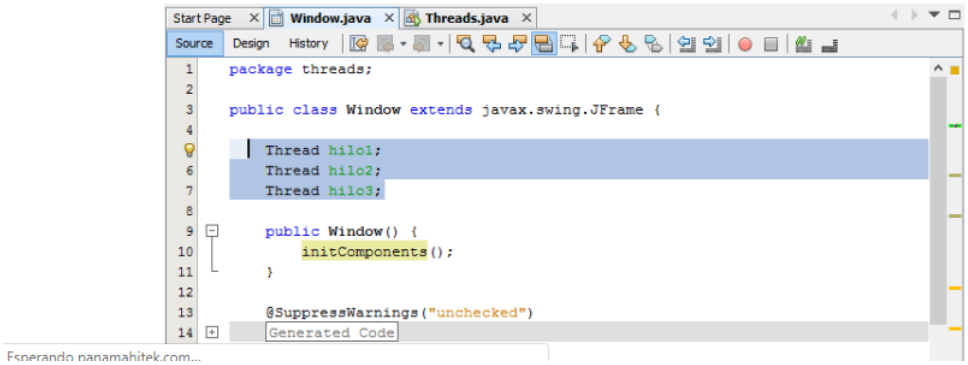
Varios métodos de la clase java.lang.Thread controlan la ejecución de una thread.

Métodos de uso común:

void start(): usado para iniciar el cuerpo de la thread definido por el método run().
void sleep(): pone a dormir una thread por un tiempo mínimo especificado.
void join(): usado para esperar por el término de la thread sobre la cual el método es invocado, por ejemplo por término de método run().
void yield(): Mueve a la thread desde el estado de corriendo al final de la cola de procesos en espera por la CPU.

Ahora vamos al código y declaramos 3 variables tipo Thread.

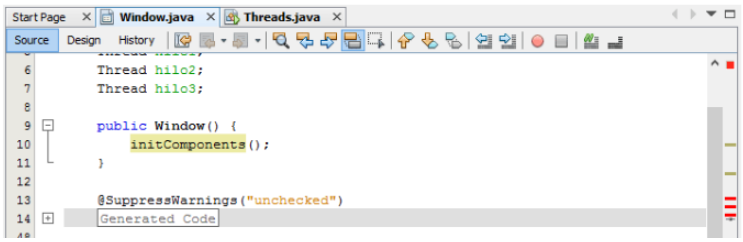
```
Thread hilo1;

Thread hilo2;

Thread hilo3;
```



Ahora vamos a programar el botón que colocamos en el JFRame.

Dentro del código del botón iniciaremos los threads. Colocamos lo siguiente:

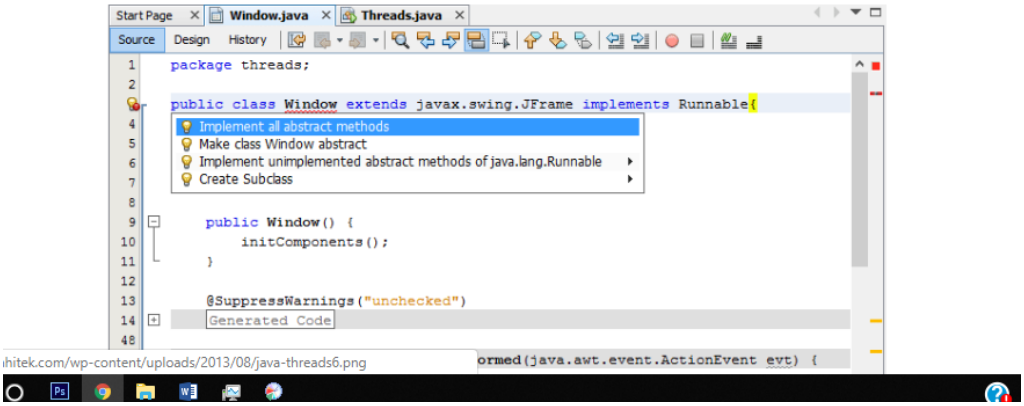
```
hilo1 = new Thread(this);
hilo2 = new Thread(this);
hilo3 = new Thread(this);

hilo1.start();
hilo2.start();
hilo3.start();
```

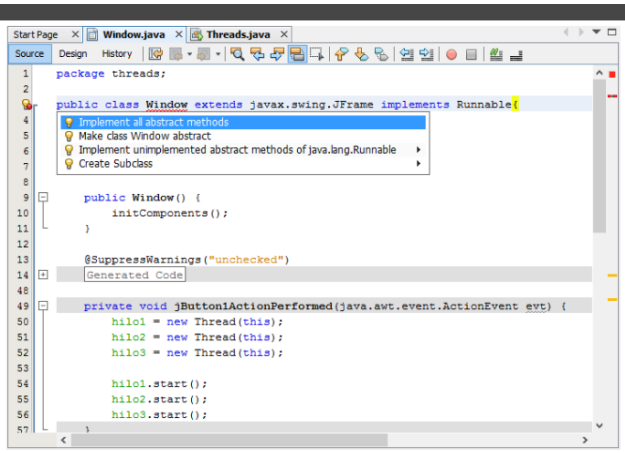


Como podemos observar, Netbeans marca 3 errores con la inicialización de los threads. Esto se debe a que necesitamos implementar la interfaz Runnable en la Clase de nuestra JFrame.

Image

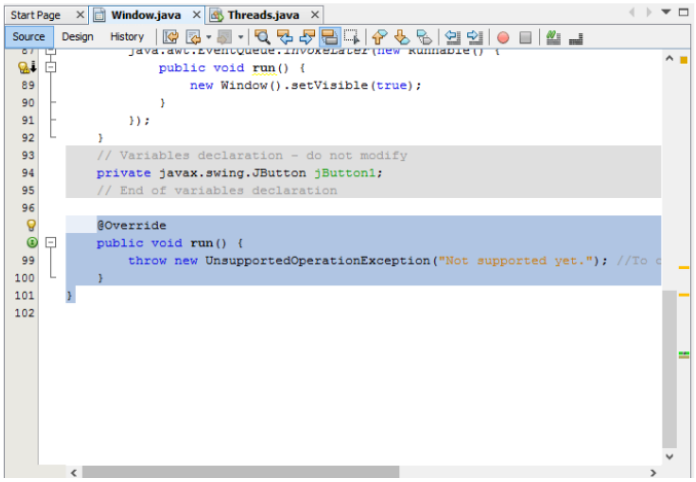
Aún así Netbeans marca un error. Le damos clic en el ícono de la izquierda para corregir el error. Nos aparecerá la opción “Implement all abstract methods”.





Al hacer clic desaparecerá el error. Al final del código se agregará lo siguiente:

Al hacer clic desaparecerá el error. Al final del código se agregará lo siguiente:



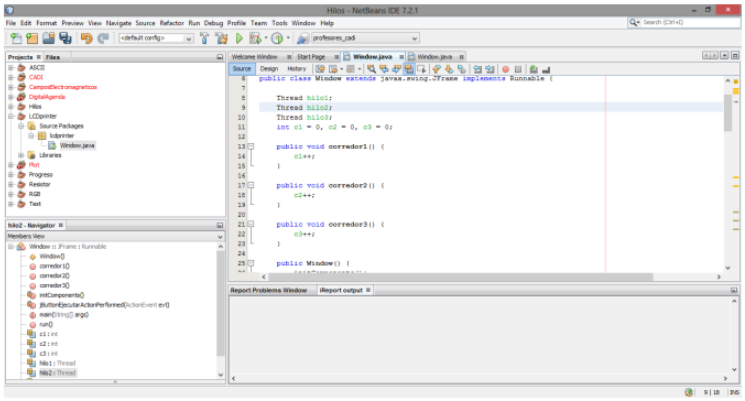
En este nuevo método que se ha agregado es donde se coloca el código que ejecutarán los threads.

Para ver como se ejecutan los threads, veremos una carrera de 3 procesos. Primero declararemos 3 variables, c1, c2 y c3. Serán del tipo entero y las inicializaremos en 0.

Luego introduciremos los siguientes métodos:

```
public void corredor1() {
c1++;
}

public void corredor2() {
c2++;
}

public void corredor3() {
c3++;
}
```

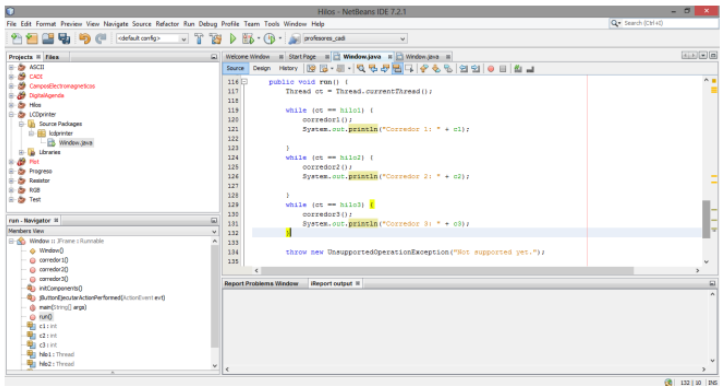
Estos métodos lo que harán es que le agregarán una unidad a cada una de las variables, es decir, c1, c2 y c3.



Ahora nos dirigimos al final del programa, donde está el método **public void run().**

Allí colocamos lo siguiente:

```
Thread ct = Thread.currentThread();

while (ct == hilo1) {

corredor1();

System.out.println("Corredor 1: " + c1);

}
while (ct == hilo2) {

corredor2();

System.out.println("Corredor 2: " + c2);

}
while (ct == hilo3) {

corredor3();

System.out.println("Corredor 3: " + c3);

}
```



Programa en Java reloj con hilos.

https://infoxbosque.wordpress.com/2012/01/07/reloj-en-java-como-los-hilos-pueden-cambiar-nuestra-vida/

http://diogenesamaury.blogspot.com/2013/11/reloj-en-tiempo-real-en-java.html