** **

*“Año de la unidad, la paz y el desarrollo”*

**INFORME DE PROGRAMACIÓN DE PARALELISMO, CONCURRENCIA Y SISTEMAS DISTRIBUIDOS**

**INFORME N° 002-2023-UNAP/FIMEES-EPIS-GAEV**

**PRESENTADO POR :** Gomez Alcos Elmer Valerio. .

**PARA :** Ing. Edwin Mamani Calderon.

**ASUNTO :** Practica de hilos en C# y Operaciones básicas con Threads en C#

**LUGAR Y FECHA** **:** Puno, 26 de enero del 2023.

1. **REPOSITORIO**

<https://github.com/ValerioGomez/thread-simulator-graphics>

1. **VIDEO**

<https://paralelismo-2023.web.app>

1. **INTRODUCCION**

En este informe se discutirá el desarrollo de una aplicación utilizando hilos en Windows Form de C#, utilizando el entorno de desarrollo SharpDevelop 5.1. Se describirán los pasos necesarios para diseñar cuatro botones y un objeto, y se explicará cómo se utilizan los hilos para mover el objeto en los ejes X e Y. También se detallará cómo se detienen los hilos y cómo se logra que el hilo termine junto con la aplicación utilizando una variable. Este informe será de gran ayuda para aquellos desarrolladores interesados en aprender cómo utilizar hilos en aplicaciones de Windows Form de C#.

Daremos una breve introducción a los materiales necesarios para comprender y trabajar con hilos.

1. **C#.** C# (C Sharp) es un lenguaje de programación orientado a objetos desarrollado por Microsoft para su uso en la plataforma .NET. Es un lenguaje de programación multiparadigma, que admite programación orientada a objetos, programación imperativa y programación funcional. C# es utilizado para desarrollar aplicaciones de escritorio, aplicaciones web, aplicaciones móviles, juegos y servicios web. Es compatible con Windows, Linux y macOS.
2. **SharpDevelop 5.1.** SharpDevelop es un entorno de desarrollo integrado (IDE) de código abierto para el lenguaje de programación C#. SharpDevelop 5.1 es una versión específica de este IDE. Es compatible con las versiones más recientes de C# y .NET Framework. Además de ofrecer las funciones básicas de un IDE, como el resaltado de sintaxis y el depurado, SharpDevelop también ofrece funciones avanzadas como el refactorizado de código y la generación automática de código. Es una herramienta popular entre los desarrolladores, especialmente los que trabajan en proyectos de código abierto o pequeños proyectos de escritorio.
3. **DESARROLLO**
4. **Formulación del Proyecto.**

Para realizar una aplicación utilizando Hilos en Windows Form de C# con las características mencionadas, se deben seguir los siguientes pasos:

* 1. Diseñar 4 botones y un objeto en el formulario de la aplicación utilizando herramientas de diseño de interfaz gráfica de usuario (GUI)
  2. Utilizar el método Thread.Start() para crear un hilo y asignarle un método que hará mover el objeto por el eje X cuando se presione el botón “iniciar hilo 1”
  3. Utilizar el método Thread.Start() para crear un hilo y asignarle un método que hará mover el objeto por el eje Y cuando se presione el botón “iniciar hilo 2”
  4. Utilizar el método Thread.Abort() para detener el hilo 1 cuando se presione el botón “abortar hilo1”, Utilizar el método Thread.Abort() para detener el hilo 2 cuando se presione el botón “abortar hilo2”
  5. Utilizar una variable para controlar el estado de la aplicación y en el método de cierre de la aplicación, utilizar el método Thread.Join() para esperar que los hilos terminen antes de cerrar la aplicación.

Diagrama

Descripción generada automáticamente

1. **Sobre el Proyecto** 
   1. Realizamos el Diseño en la Interfaz gráfica. Obteniendo lo siguiente:

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

2.2 La función "Mover\_en\_X" es un método que se encarga de mover un objeto (en este caso, un botón con la referencia "button1") en el eje X. Utiliza un ciclo "for" que se ejecuta 339 veces, hallado de la posición adecuada (Captura1), en cada iteración, el valor de la posición X del objeto se incrementa en 1 y se actualiza su posición en la pantalla utilizando el método "BeginInvoke". El método "Thread.Sleep(5)" hace que el hilo se detenga durante 5 milisegundos entre cada iteración, lo que permite ver el movimiento del objeto en la pantalla. Hacemos lo mismo en *Mover\_en\_Y()*

Texto

Descripción generada automáticamente

2.3 Usaremos variables globales para la mejor manipulación de los hilos,



2.4 Asignamos el evento del boton2 iniciando el *\_hilo1* lo iniciamos con el método Start()

Texto

Descripción generada automáticamente con confianza media

2.5 De Igual manera en el botón del \_*hilo2*

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

2.6 La función Button5Click es un evento que se ejecuta cuando se presiona el botón "abortar hilo2". Dentro de esta función, se utiliza el método Abort() para detener el hilo 2 (que se especifica en la variable \_hilo2) y se habilita el botón "iniciar hilo 2" (especificado en la variable button3) para que el usuario pueda volver a iniciar el hilo 2 si lo desea.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

2.7 Se establece una nueva instancia de un objeto "Point" con coordenadas X = 90 y Y = 39. El objeto "Point" es una estructura de datos en C# que representa un punto en un plano bidimensional con coordenadas X e Y. En este caso, la variable "currentPosition" está siendo inicializada con las coordenadas (90, 39) para representar la posición inicial del objeto en la aplicación.



2.8 Para el Problema: **Lograr que el hilo termine junto con la aplicación, utilizando una variable, y utilizando la opción de hilos.**

***Primer Método y más fiable Opción de Hilos***

- Tenemos en cuenta de que el *\_hilo1* es el hilo principal, entonces el movimiento principal es en el eje X.

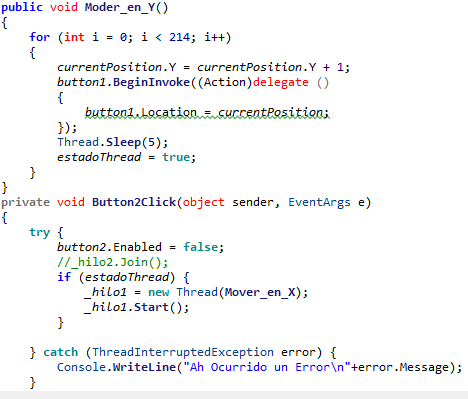
- Al Iniciar el *\_hilo2* el *\_hilo1* debe esperar usando el método Join() hasta que el *\_hilo2* termine, por consecuente el *\_hilo1* reanudara suproceso hasta su fin.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Es importante mencionar que el hilo principal (el que ejecuta la interfaz de usuario) no se bloqueara, ya que está usando el método BeginInvoke(), así que seguirá mostrando la animación en pantalla.

***Segundo Método Variable.*** Para ver el estado del \_hilo2 con la variable global booleana estadoThread, puede agregar una línea de código en el método Moder\_en\_Y para actualizar el valor de estadoThread cada vez que se ejecuta. Por ejemplo, agregue la siguiente línea de código al inicio del método Moder\_en\_Y:



Sin embargo, ten en cuenta que el uso de una variable global compartida puede causar problemas de sincronización de hilos, es mejor utilizar el método Thread.IsAlive() para verificar si un hilo está activo o no.

***Segundo Método Variable.*** Para verificar el estado de un hilo utilizando el método Thread.IsAlive(), se debe utilizar el siguiente código:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

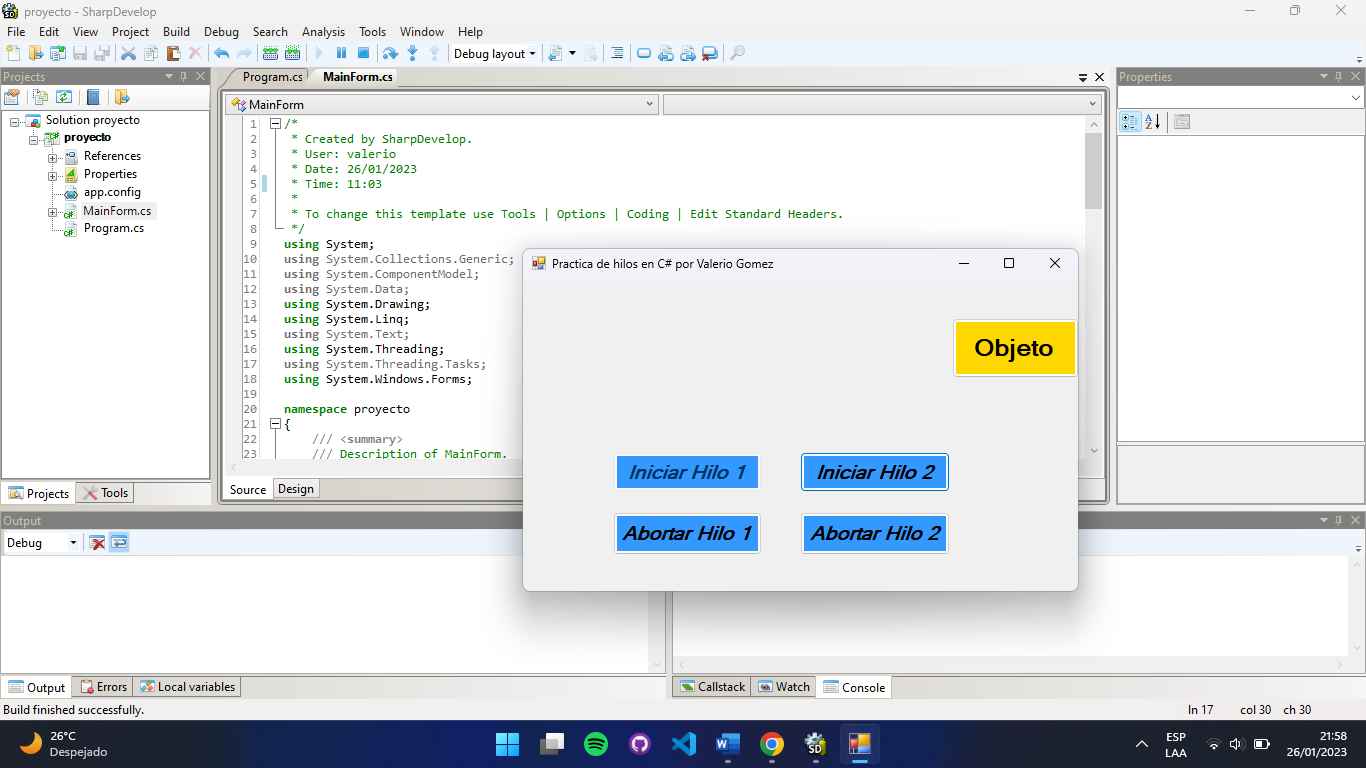
En este caso, se está verificando si el hilo \_*hilo2* no se encuentra en ejecución, entonces se incia el *\_hilo1* . Caso contrario si el método IsAlive() devuelve true, usamos el metodo Abort() con el *\_hilo1* significa que el hilo se encuentra en ejecución, y si devuelve false, significa que el hilo no se encuentra en ejecución.

2. **Código**
3. */\**
4. *\* Created by SharpDevelop.*
5. *\* User: valerio*
6. *\* Date: 26/01/2023*
7. *\* Time: 11:03*
8. *\**
9. using System;
10. using System.Collections.Generic;
11. using System.ComponentModel;
12. using System.Data;
13. using System.Drawing;
14. using System.Linq;
15. using System.Text;
16. using System.Threading;
17. using System.Threading.Tasks;
18. using System.Windows.Forms;
19. namespace proyecto
20. {
21. public partial class MainForm : Form
22. {
23. Thread \_hilo1;
24. Thread \_hilo2;
25. *//static bool estadoThread = false;*
26. public MainForm()
27. {
28. InitializeComponent();
29. }
30. Point currentPosition = new Point(90, 39); *// Global variable*
31. public void Mover\_en\_X()
32. {
33. *for* (int i = 0; i < 339; i++)
34. {
35. currentPosition.X = currentPosition.X + 1;
36. button1.BeginInvoke((Action)delegate ()
37. {
38. button1.Location = currentPosition;
39. });
40. Thread.Sleep(5);
41. }
42. }
43. public void Moder\_en\_Y()
44. {
45. *for* (int i = 0; i < 214; i++)
46. {
47. currentPosition.Y = currentPosition.Y + 1;
48. button1.BeginInvoke((Action)delegate ()
49. {
50. button1.Location = currentPosition;
51. });
52. Thread.Sleep(5);
53. }
54. }
55. private void Button2Click(object sender, EventArgs e)
56. {
57. *try* {
58. button2.Enabled = false;
59. \_hilo1 = new Thread(Mover\_en\_X);
60. \_hilo1.Start();
61. } *catch* (ThreadInterruptedException error) {
62. Console.WriteLine("Ah Ocurrido un Error\n"+error.Message);
63. }
64. }
65. private void Button3Click(object sender, EventArgs e)
66. {
67. *try* {
68. button3.Enabled = false;
69. \_hilo2 = new Thread(Moder\_en\_Y);
70. \_hilo2.Start();
71. *//\_hilo2.Join();*
72. } *catch* (ThreadInterruptedException error) {
73. Console.WriteLine("Ah Ocurrido un Error\n"+error.Message);
74. }
75. }
76. private void Button4Click(object sender, EventArgs e){
77. \_hilo1.Abort();
78. button2.Enabled = true;
79. }
80. private void Button5Click(object sender, EventArgs e){
81. \_hilo2.Abort();
82. button3.Enabled = true;
83. }
84. }
85. }
86. **CAPTURAS**
    1. **Captura1**

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

* 1. **Captura2**



* 1. **Captura 3**

Una captura de pantalla de una computadora

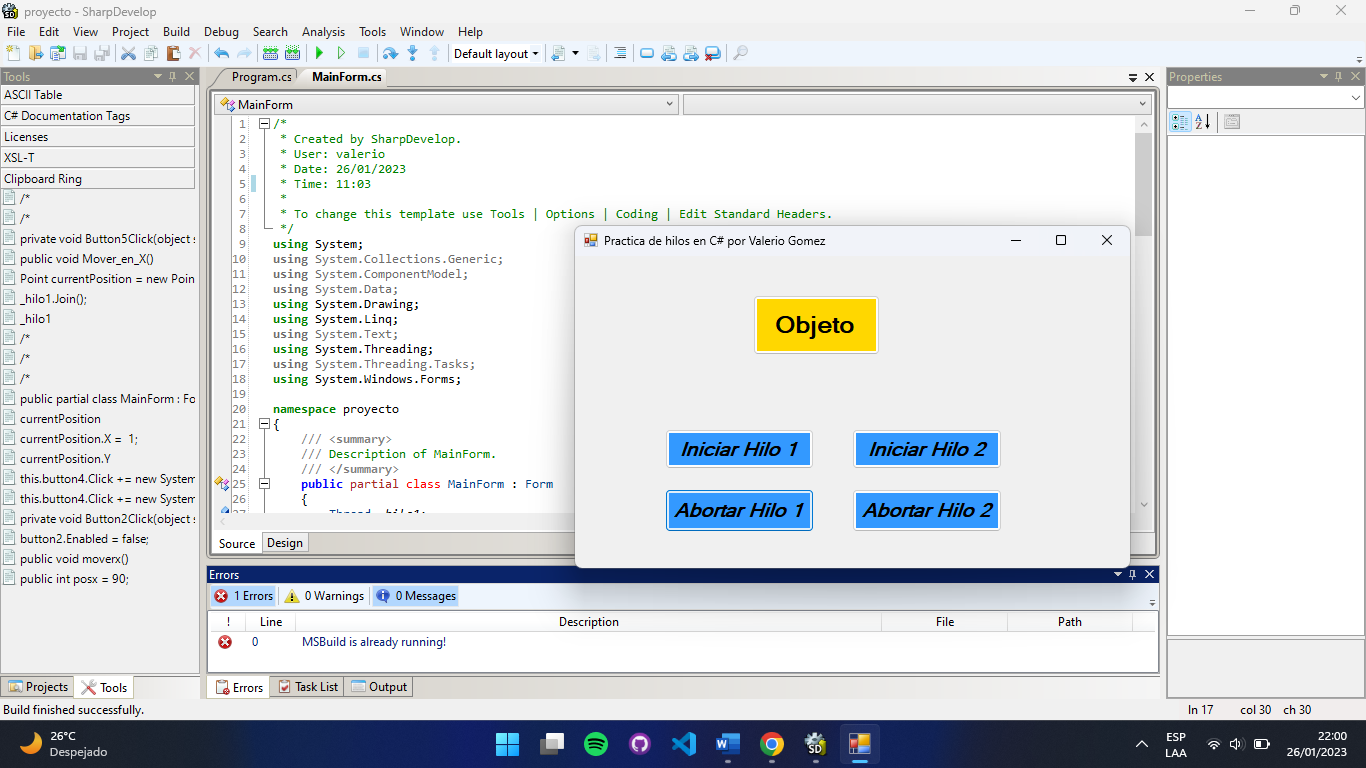
Descripción generada automáticamente

* 1. **Captura 4**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

* 1. **Captura5**



* 1. **Captura6**

