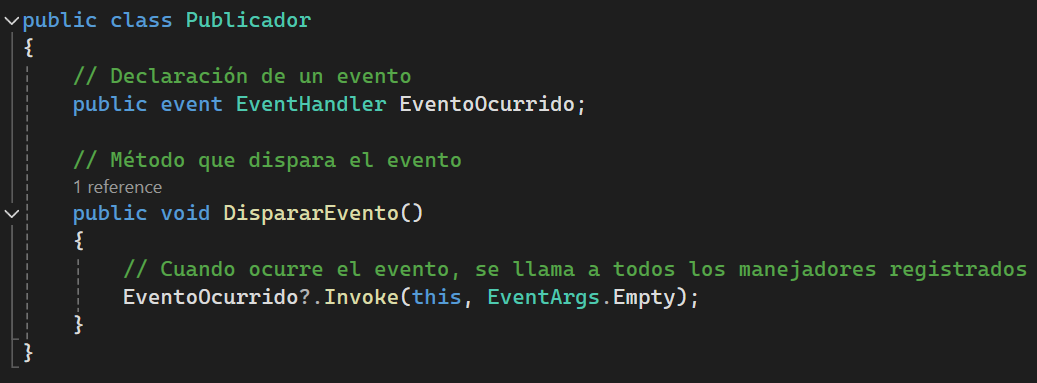
**Tarea 5**

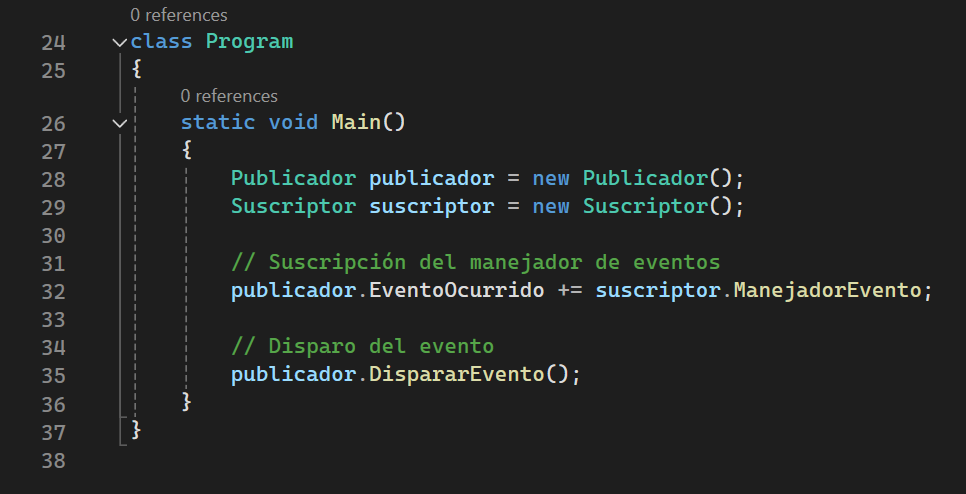
**21/08/2024**

**Alumnos: Briguera, Juarez, Carivalli, Guerrero.**

**Ejemplo 1:**

1. **Diseño de una clase que simula un bucle despachador de eventos.**
2. **Definición de eventos personalizados dentro de la clase.**

La clase publicador simula el bucle despachador de eventos, ya que este reaccionará ante el acontecimiento de algún evento. En el mismo segmento de código, está presente el EventHandler llamado EventoOcurrido, el cual puede ser disparado por otros objetos y es un evento personalizado.

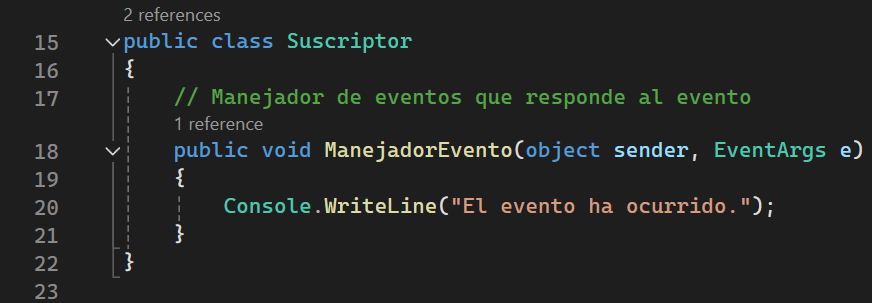
1. **Manipulación de eventos mediante suscripciones y desencadenamientos.**

Aquí el suscriptor (suscriptor.ManejadorEvento) se suscribe al evento EventoOcurrido del publicador.

Además se produce el desencadenamiento se produce *DispararEvento() de la clase publicador* y se invoca utilizando EventoOcurrido?.Invoke(this, EventArgs.Empty);

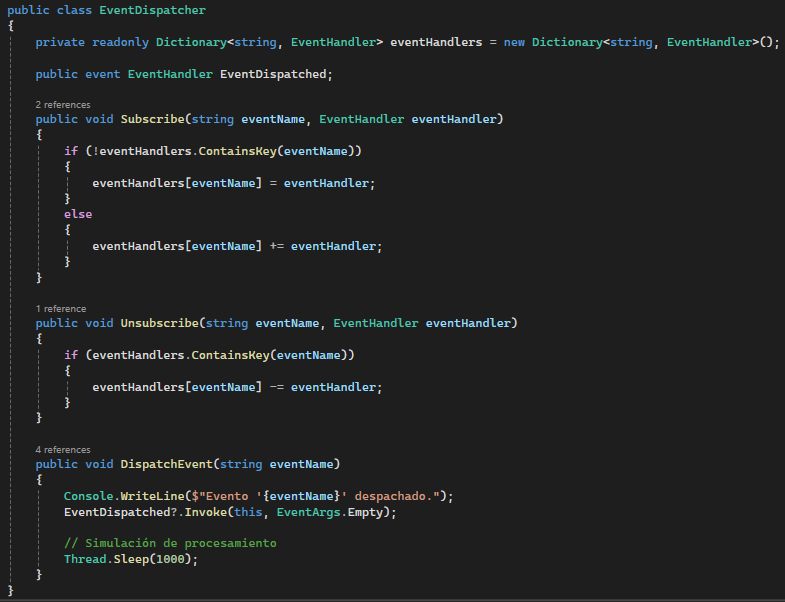
Texto

Descripción generada automáticamente

1. **Simulación de la recepción y procesamiento de mensajes utilizando eventos.**

Esta clase contiene el método ManejadorEvento, el método es llamado cuando el evento EventoOcurrido es disparado en la clase publicador. Aquí se simula la recepción y procesamiendo de un mensaje.

**Ejemplo 2:**

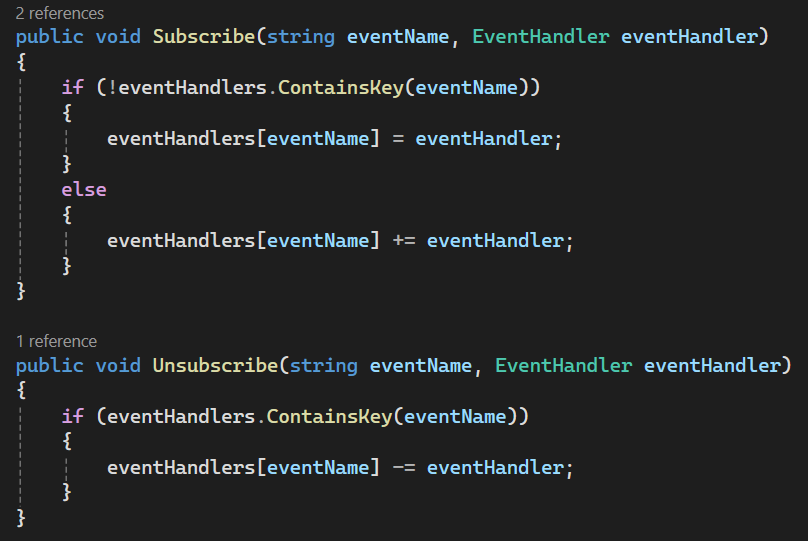
1. **Diseño de una clase que simula un bucle despachador de eventos.**

Esta clase llamada EventDispatcher es la encargada de simular el bucle dispachador de eventos.

1. **Definición de eventos personalizados dentro de la clase.**

En la clase EventDispatcher se encuentra el evento EventDispatched el cual es un evento personalizado. Este se dispara cada vez que un evento es despachado por el eventDispatcher.

1. **Manipulación de eventos mediante suscripciones y desencadenamientos.**

****Los métodos Subscribe y Unsubscribe se encargar de permitir las suscripciones y las de-suscripciones.

El desencadenamiento se da DispatchEvent, el cual desencadena algún método especificado.

1. **Simulación de la recepción y procesamiento de mensajes utilizando eventos.**

La simulación de la recepción y procesamiento de mensajes se realiza en el método DispatchEvent, donde se imprime un mensaje en la consola indicando que un evento ha sido despachado, y luego se realiza una pausa para la simulación.

Los manejadores de eventos (Event1Handler, Event2Handler, GlobalEventHandler) son llamados cuando se despachan los eventos respectivamente, demostrando cómo los eventos y sus mensajes son recibidos y procesados por los suscriptores.