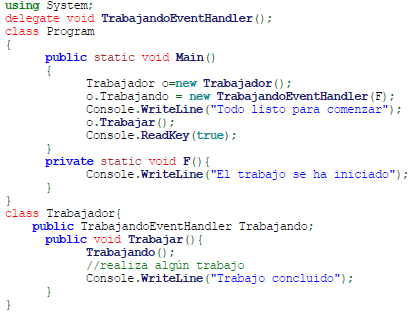
**Seminario de Lenguajes (.NET)**

**Práctica 7**

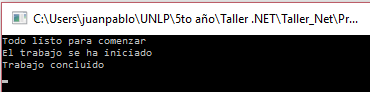
1) Cree una nueva solución en el SharpDevelop y codifique el siguiente programa:



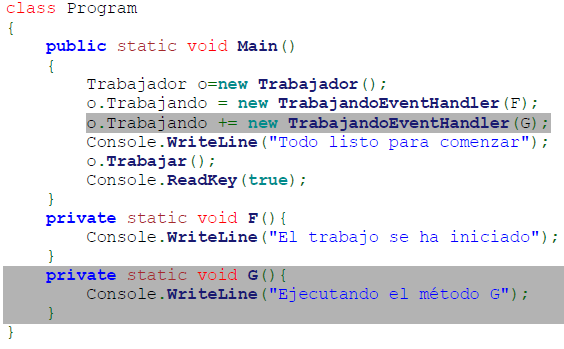
a) Ejecute paso a paso el programa y observe cuidadosamente su funcionamiento.

b) ¿Qué salida produce por Consola?

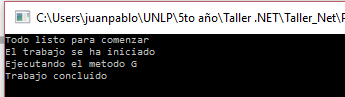
1. B- Salida por consola:



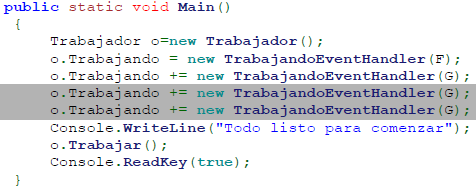
1. Borrando la línea “o.Trabajando = **new TrabajandoEventHandler”:**
   1. Se produce un error en el programa y este se interrumpe durante su ejecución.
   2. “El campo 'Ej1.Trabajador.Trabajando' nunca se asigna y siempre tendrá el valor predeterminado null”.
   3. Agregando el **if**(Trabajando != **null**) Trabajando(); se soluciona el problema.
2. Modifique la clase Program de la siguiente manera (se ha sombreado las líneas de código agregadas):

  
Modificando el programa en las líneas definidas en el enunciado:

* 1. Observe el paso a paso del programa.
  2. Qué salida produce?

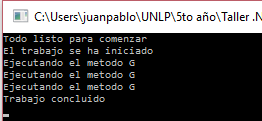


* 1. En qué orden se invocan los métodos contenidos en el delegado?  
     Se invocan siguiendo el orden en que fueron agregados al delegado (orden FIFO).

1. Modificando el programa en las líneas definidas:  
   
   1. Observe el paso a paso del programa.
   2. ¿Intentar que un delegado contenga más de una instancia del mismo método produce algún error o es una posibilidad perfectamente válida?

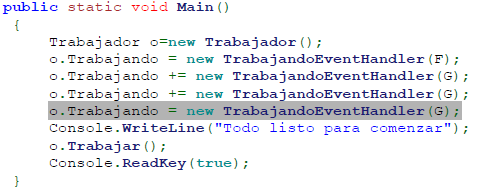
Que un delegado contenga más de una instancia del mismo método no produce ningún error, es una opción válida.

* 1. Salida por consola?



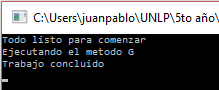
* 1. Borre la instrucción “o.Trabajando =new TrabajandoEventHandler(F);” del método Main() y ejecute el programa nuevamente. Observe que no se produce error alguno. ¿Por qué no? Piense cuál es el valor de o.Trabajando en la primera instrucción ¿Puede decir entonces que, cuando uno de los operandos es un delegado, el operador “+”está sobrecargado?

1. Compruebe si “o.Trabajando = o.Trabajando + new TrabajandoEventHandler(G)” es equivalente a la forma abreviada “o.Trabajando += new TrabajandoEventHandler(G)” Codifique el método Main() de la siguiente manera:

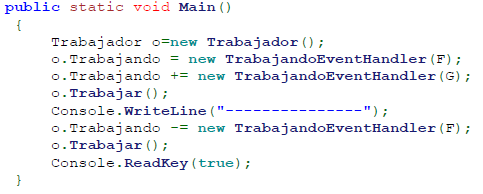


* Las siguientes líneas son equivalentes:
  + 1. o.Trabanado = o.Trabando + New TrabajandoEventHandler(G)
    2. .Trabajando += new TrabajandoEventHandler(G);
* Salida por consola ingresando la línea en el enunciado:

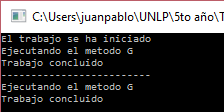
o.Trabanado = New TrabajandoEventHandler(G)



1. Codifique el método Main() de la siguiente manera

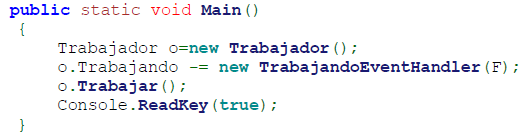


* 1. Salida por consola:

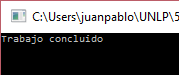


* 1. Que hace el operador “-=”?  
     El operador “-=” borra el evento definido del eventHandler, en este caso el evento será el método F.
  2. De que otra forma puede escribir la línea “o.Trabajando -= new TrabajandoEventHandler(F)”?  
     Podria reemplazar la línea “o.Trabajando -= new TrabajandoEventHandler(F)” por: “o.Trabajando-=F”.
  3. En el caso que el delegado o.Trabajando posea varias instancias del método F() ¿Cuántas son eliminadas con la siguiente instrucción?   
     “o.Trabajando -= new TrabajandoEventHandler(F);”  
     Si el eventHandler posee varias instancias de un método, la operación de resta solo borrara 1 sola de dichas instancias.

1. Modificando el programa:

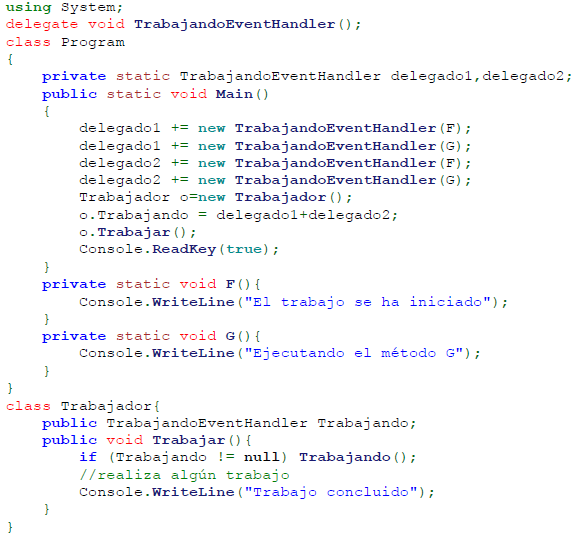
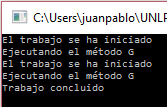


* 1. Salida por consola:



* 1. El operador “-“ está sobrecargado.

1. Hasta ahora utilizamos el operador “+” para agregar un único método a la lista de métodos de un delegado. Cuál es el resultado de la aplicación del operador “+” entre dos delegados que poseen una lista con varios métodos. Para ello, codifique, ejecute y analice el siguiente programa:

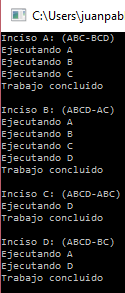
  
Salida por consola:  
  
Por lo tanto delegado1+delegado2 = F+G + F+G

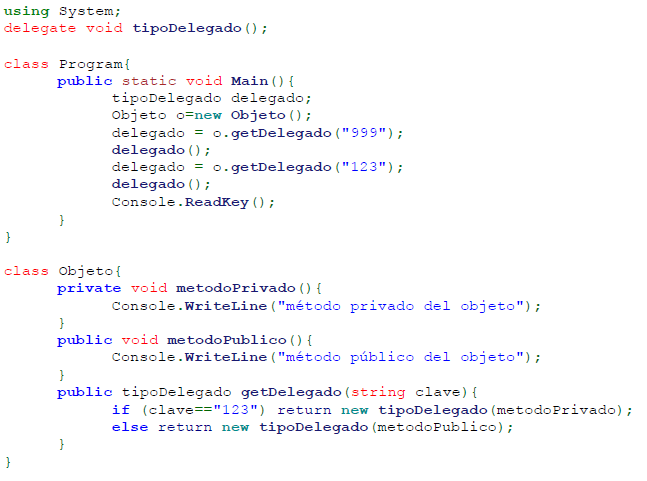
1. Hasta ahora utilizamos el operador “-” para eliminar un único método de la lista de métodos de un delegado. Cuál es el resultado de la aplicación del operador “-” entre dos delegados que poseen una lista con varios métodos cada uno. Para ello, codifique, ejecute y analice un programa adecuado contestando las siguientes preguntas:

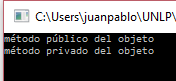
a) Si delegado1 posee los métodos A, B y C (en ese orden) y delegado2 posee los métodos B, C y D (en ese orden), qué métodos serán asignados a delegado3 al ejecutar la siguiente instrucción delegado3=delegado1-delegado2;

b) Si delegado1 posee los métodos A, B, C y D (en ese orden) y delegado2 posee los métodos A y C (en ese orden), qué métodos serán asignados a delegado3 al ejecutar la siguiente instrucción delegado3=delegado1-delegado2;

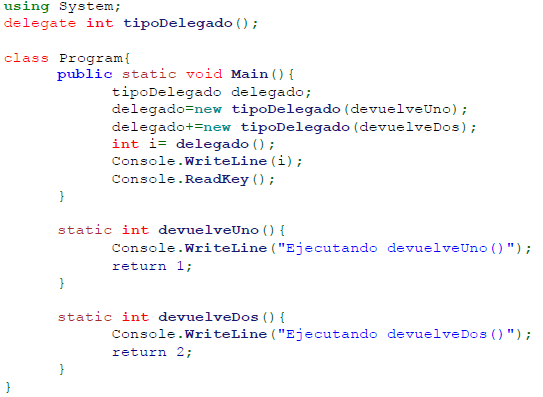
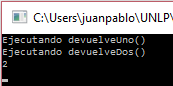
c) Si delegado1 posee los métodos A, B, C y D (en ese orden) y delegado2 posee los métodos A, B y C (en ese orden), qué métodos serán asignados a delegado3 al ejecutar la siguiente instrucción delegado3=delegado1-delegado2;

d) Si delegado1 posee los métodos A, B, C y D (en ese orden) y delegado2 posee los métodos B y C (en ese orden), qué métodos serán asignados a delegado3 al ejecutar la siguiente instrucción delegado3=delegado1-delegado2;  
Salida por consola:  


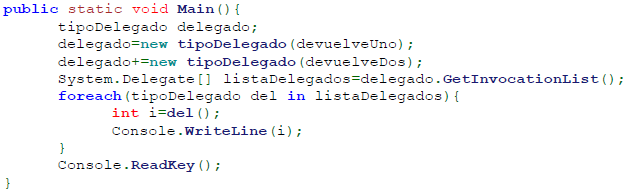
1. Cree y codifique el siguiente programa:

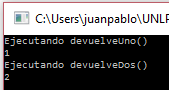
Salida por consola:  
  
Obsérvese que, cuando se realiza una llamada a través de un objeto delegado no se tienen en cuenta los modificadores de visibilidad de los métodos que se ejecutarán, lo que permite llamar desde un tipo a métodos privados de otros tipos que estén almacenados en un delegado.

1. Cree y codifique el siguiente programa:

  
Salida por consola:  
  
  
delegado=**new** **tipoDelegado**(devuelveUno);  
delegado+=**new** **tipoDelegado**(devuelveDos);  
**int** i= **delegado**(); 🡪 Se le asigna el 2  
  
Nota: Observe que en la instrucción “delegado=new tipoDelegado(devuelveUno);” no se ha utilizado el operador “+=” como veníamos haciendo en ejemplos anteriores. Esto se debe a que delegado se ha definido como una variable local en el método Main(), por lo tanto el compilador no permite expresiones de lectura sobre esta variable verificando que nunca fue asignada.

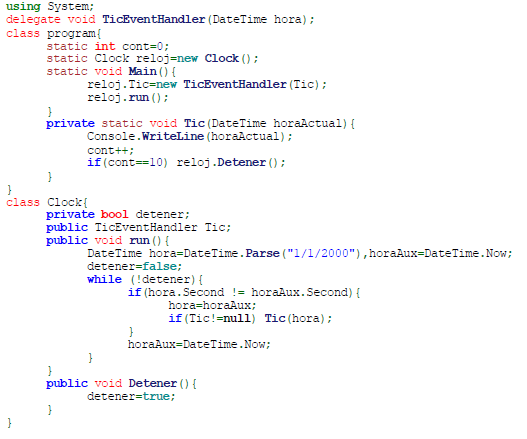
1. Reemplazar el main por el siguiente:

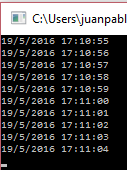


* 1. Salida por consola:  
     
  2. Que hace el Delegado.**GetInvocationList()?**

Retorna un array con los delegados asignados a la variable que la llama. En el ejericio retorna un array de dimensión 2:  
array[0] 🡪 devuelveUno  
array[1] 🡪 devuelveDos

* 1. Serviria este método para poder ejecutar los métodos encolados en un delegado en un orden distinto al de la cola? Si.

1. Cree y codifique el siguiente programa:  
   
   1. Que hace el programa? Cuenta 10 segundos y muestra fecha y hora por consola. Una vez finalizado los 10 segundos se detiene a través de un evento.
   2. Salida por consola



1. Modifique el código del ejercicio 13 para cumplir con la convención vista en la teoría respecto de los parámetros que debe llevar un manejador de eventos (sender de tipo object y e de tipo EventArgs o clase derivada).  
   Correcciones:  
   - delegate void TicEventHandler(TicEvenArgs e);  
   - class TicEvenArgs:EventArgs  
       {  
        public DateTime horaActual{get;set;}  
       }  
   - TicEvenArgs e  
    e.horaActual = hora;  
     Tic(e);