Delegados

Ejercicio 1

- Crea un tipo delegado denominado **Operación** de parámetro entero y de retorno entero.
- En una clase estática **Cálculos** añade dos métodos, con esa misma signatura, llamados **Cuadrado** y **Cubo**, que te devuelvan el cuadrado y el cubo de un número introducido respectivamente.
- Crea una aplicación en el programa principal que dependiendo de la opción seleccionada, te muestre el resultado del cálculo (lógicamente utilizando el objeto de tipo delegado).

Ejercicio 2

- Crea un tipo delegado denominado **Infinitivo** para métodos que no reciban, ni devuelvan nada.
- Define ahora los métodos **Ser**, **Correr**, **Ver**, **Pensar**, **Comer** los cuales mostrarán por consola los infinitivos en inglés de dichos verbos.
- Crea un programa principal donde se instancie un objeto del delegado definido y …
  1. Le asociemos **Ser**, **Correr** y **Ver** y realicemos una llamada al delegado.
  2. Desasocia ahora **Ser** y **Ver** y asocia **Pensar** y **Comer** y vuleve ha hacer una llamada.

Eventos

Ejercicio 1

- Vamos realizar un programa que detecte mediante **eventos** la modificación del nombre de un archivo en un determinado directorio.
- Para ello, vamos a utilizar la clase **FileSystemWatcher** que tiene un evento **Renamed** que es desencadenado al cambiar el nombre de un archivo.
- Deberemos suscribir a este evento un método estático que me diga por consola si ha sido modificado un archivo, el nombre anterior y el nuevo nombre.

  Nota: Para que nuestro objeto **FileSystemWatcher** desencadene los eventos deberemos establecer la propiedad **EnableRaisingEvents** a true.

Ejercicio 2

- Supongamos que queremos que un banco quiere que cada vez que se retire dinero de sus cajeros se envíe un eMail y un SMS al usuario que ha realizado la transacción.
- Desde el equipo informático se ha diseñado una clase **Cajero** que tendrá:
  - Un campo entero que será el número de cajero.
  - Un evento al que podremos suscribirnos denominado **RetiradaDeEfectivo** y cuya signatura enviará a los suscriptores el cajero donde se ha producido el evento, el dni del usuario que está realizando la retirada y la cantidad.
  - Un método público **RetirarEfectivo** que recibirá el dni del usuario que hace la operación y la cantidad. Además, internamente el método mostrará por la

pantalla del cajero (la consola) el mensaje "*Realizando operación...*", se enperará 2 segundos y mostrá "*Operación realizada con éxito.*" y a continuación desencadenará el evento informando a todos los suscriptores de la operación realizada.

- Crea dos clases **EnvioEMail** y **EnvioSMS** que implementen el método **EnviarAvisoRetiradaDeEfectivo** mediante el cual se suscriban a la publicación del evento por parte del objeto cajero y muestren por pantalla.
  - "*Buscando datos usuario <dni>...*"
  - "*Enviando <eMain | SMS> al usuario <dni> de retirada de <cantidad> el día <fecha> a las <hora> horas.*"
- Por último, crea un programa principal donde instáncies objetos de las 3 clases anteriores y pruebes que el evento es enviado correctamente.

Ejercicio 3

Cuando en un programa de consola pulsamos **Control + C** se produce un evento de cancelación y un programa finaliza. Además, esta combinación de teclas no es detectada por un ***Console.ReadKey(...)*** o un ***Console.ReadLine()***, etc.

Existe un evento especial de Consola llamado **CancelKeyPress**, al que podremos suscribirnos para detectar dicha pulsación.

- Vamos a crear un clase denominada **ConsoleKeyLog** que contendrá un lista de cadenas privada. Donde cada cadena contendrá una pulsación de tecla.
- Además definirá un evento **GestionTeclasPulsadasTrasFinalizar** al que podremos suscribirnos y que se generará cuando el usuario pulse **Control + C** para finalizar el programa. Dicho evento pasará al suscriptor la lista de cadenas con las pulsaciones de tecla hasta que se pulso Control + C.
- Definiremos un método público denominado **BucleLogConsola** que se suscribirá al evento de cancelación de la consola mencionado al principio y que irá leyendo pulsaciones de tecla en la consola y las añadirá a la lista, a través del siguiente método de utilidad.

```
static string TextoTecla(ConsoleModifiers m, ConsoleKey t) {
    string txt = "";
    if ((m & ConsoleModifiers.Alt) == ConsoleModifiers.Alt)
        txt += "Alt + ";
    if ((m & ConsoleModifiers.Control) == ConsoleModifiers.Control)
        txt += "Control + ";
    if ((m & ConsoleModifiers.Shift) == ConsoleModifiers.Shift)
        txt += "Shift + ";
    txt += t.ToString();
    return txt;
}
```

Crea un programa principal que, a través de un objeto instanciado de ConsoleKeyLog leerá la teclas pulsadas del teclado mediante el método BucleLogConsola y además definirá un manejador para el evento GestionTeclasPulsadasTrasFinalizar que guarde en un fichero de texto denominado teclas.log la lista con las teclas pulsadas. (Una por línea).

Ejercicio 4 Ampliación

Invéntate una situación similar a la del ejercicio 2 donde varios objetos suscriptores se suscriban a otro publicador, realizando una acción determinada cuando este último genere un evento.

Por ejemplo, el disparo de un evento por parte de un sensor de calor en el núcleo de una central nuclear que ponga en marcha una turbina y genere una aviso en la consola del operador.

Delegados Parametrizados Genéricos

Ejercicio 1

1. Crea una aplicación con un método estático genérico **`Mostrar`**, al que le pases una matriz del tipo T y la muestre con buena tabulación. Prueba este método con diferentes tipos.

   Posible ejemplo de una llamada con float y otra con string ...

```
      3        4        5
    2,4      4,4        5

    SAL     AGUA   AZUCAR     VINO
   COLA     CAFE     ZUMO    LECHE
```

2. Añade a la aplicación un método genérico que devuelva el **mayor** de dos valores pasados como parámetros.
   - Deberemos comprobar el mayor con el método CompareTo.
   - Comprueba el funcionamiento con diferentes tipos.

3. Crea un **delegado predefinido genérico** para los métodos Mayor y Mostrar y comprueba su funcionamiento.

Ejercicio 2

Crea delegados predefinidos genéricos para los siguientes métodos y prueba su funcionamiento.

```
static int AddTwoNumbers(int n1, int n2)
{
   return n1 + n2;
}
```

```
int SquareANumber(int number)
{
   return number * number;
}
```

```
static double GetTopSpeed()
{
   return 108.4;
}
```

```
static bool MultiploCinco(int n)
{
   if ((n % 5) != 0) return false;
   else return true;
}
```

```
static double fcalcul(
   string tipo, string nom1, string nom2)
{
   Console.WriteLine($"Introduce {nom1}");
   string temp1 = Console.ReadLine();
   int var1 = Int32.Parse(temp1);
   Console.WriteLine($"Introduce {nom2}");
   string temp2 = Console.ReadLine();
   int var2 = Int32.Parse(temp2);
   if(tipo=="Potencia") return (var1*var2);
   else return (var1 / var2);
}
```

```
static void Nosegue(
   double[,] x, int[] y, String z)
{
   if(x.Length==y.Length)
   {
       int p=0;
       foreach (double c in x){
          z += (c + y[p]);
          z += " ";
          p++; }
    }
   Console.WriteLine(z);
}
```