Ejercicios Delegados

Descargar estos ejercicios

Índice

Ejercicio 1

Crea un **tipo delegado** denominado `Operación` de parámetro **entero** y de retorno **entero**.

- En una clase de utilidad (estática) denominada `Cálculos` añade **dos métodos con esa misma signatura** llamados `Cuadrado` y `Cubo`, que te devuelvan respectivamente el cuadrado y el cubo de un número introducido.
- Crea una aplicación en el programa principal con un menú que asigne un delegado con uno de los dos métodos de utilidad según la elección del usuario. Posteriormente deberás usar ese delegado para realizar la operación.

✅ 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 con el operador `+=` los métodos **Ser**, **Correr** y **Ver** y realicemos una llamada al delegado, para que se llamen los tres métodos de forma consecutiva.
2. Desasocia ahora con el operador `-=` los métodos **Ser** y **Ver** y asocia **Pensar** y **Comer** y vuelva a hacer una llamada.

✅ Ejercicio 3

Crea una aplicación con un método estático genérico **Mostrar**, al que le pases una matriz del tipo `T` y la muestre con una correcta tabulación. Posteriormente, 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
```

Crea un objeto delegado **predefinido genérico** de la BCL para el método `Mostrar` y comprueba su funcionamiento.

Ejercicio 4

Crea un objeto de los **delegados predefinidos genéricos** para los siguientes métodos y prueba su funcionamiento en el programa principal:

```
static int Suma(int n1, int n2) => n1 + n2;
static int CuadradoDe(int number) => number * number;
static double GetVelocidadParada() => 108.4;
static bool EsMultiploDeCinco(int n) => n % 5 == 0;
```

```
static double Calcula(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);
    return tipo == "Potencia" ? var1*var2 : var1 / var2;
}
```

```
static void ProcedimientoDesconocido(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);
}
```

✅Ejercicio 5

Partiendo de la propuesta de solución (o la tuya) al **ejercicio 5 de genéricos**, donde implementábamos diferentes formas de calcular la media y obtener la temperatura usando interfaces a modo de '*estrategia*'. Realiza las siguientes modificaciones:

1. Elimina la definición de los interfaces y las clases que los implementaban.
2. Los métodos que se definían en esas clases ahora serán métodos estáticos dentro de la clase `Program` o una clase estática que los agrupe como métodos de utilidad con la siguiente implementación...

```
static float ObtenTemperaturaMaxima(TemperaturasXProvincia txp)
                    => txp.TemperaturaMaxima;
static float ObtenTemperaturaMinima(TemperaturasXProvincia txp)
                    => txp.TemperaturaMinima;
static bool MayorQue(float t1, float t2) => t1 > t2;
static bool MenorQue(float t1, float t2) => t1 < t2;
static bool IgualQue(float t1, float t2) => t1 == t2;
```

3. Sustituye en los métodos `MediaTemperaturas` y `MuestraProvincias` donde pasábamos una parámetro formal de tipo interfaz **por un tipo delegado genéricos** de tipo `Func<T1, T2, ..., R>`.
4. Sustituye en el `Main` la creación del objeto que implementaban los interfaces que hemos borrado por el métodos que equivalentes que hemos definido en el punto 2, cumplen con el interfaz definido en el delegado y hacen ahora la operación.