Índice

Ejercicios Unidad 20 - Programación Funcional

Descargar estos ejercicios

> **Antes de empezar**
>
> Para realizar estos ejercicios, deberás descargar los recursos del enlace de proyecto_poo. Como puedes ver, la solución está compuesta de varios proyectos. Cada uno de ellos corresponde con un ejercicio, deberás implementar todo el código, tanto de la Main como de los métodos que se piden en cada ejercicio. Cada proyecto contiene el test correspondiente, que deberás pasar para comprobar que has hecho el ejercicio correctamente.

Ejercicio 1. Delegado Operación

Para practicar un primer uso de los delegados y entender el concepto.

```
Ejercicio 1. Delegado Operación

Selecciona número entero: 4
[1] Cuadrado
[2] Cubo
[3] Cambiar número
[4] Salir
Selecciona una opción: 2
Resultado: 64
```

Requisitos

Crea un tipo delegado denominado `Operacion` 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. Delegados con parámetros variados

Vamos a practicar con delegados que aceptan múltiples parámetros de diferentes tipos para realizar cálculos de lógica de negocio.

```
Ejercicio 2. Delegados con parámetros variados

Ejercicio 2. Delegados con parámetros variados

Pedido 1: Importe=1000, Cantidad=12, VIP=True
-> Usando DescuentoPorCantidad:
   Descuento aplicado: 150 (Precio Final: 850)
-> Usando DescuentoVip:
   Descuento aplicado: 200 (Precio Final: 800)
-> Usando DescuentoCombinado:
   Descuento aplicado: 250 (Precio Final: 750)

Pedido 2: Importe=500, Cantidad=3, VIP=False
-> Usando DescuentoPorCantidad:
   Descuento aplicado: 0 (Precio Final: 500)
-> Usando DescuentoCombinado:
   Descuento aplicado: 0 (Precio Final: 500)
Pulsar una tecla para finalizar...
```

Requisitos

Crea un tipo delegado denominado `CalculoDescuento` que acepte los siguientes parámetros:

- `importe` (double): El importe total de la compra.
- `cantidad` (int): El número de artículos comprados.
- `esVip` (bool): Indica si el cliente es VIP.
- Retorno: `double` (el importe del descuento a aplicar).

En una clase estática `ReglasDescuento`, implementa tres métodos que cumplan con la firma del delegado:

1. `DescuentoPorCantidad`: Aplica un 15% si la cantidad es > 10, un 5% si es > 5, y 0 en otro caso.
2. `DescuentoVip`: Aplica un 20% si es VIP, 0 en caso contrario.
3. `DescuentoCombinado`: Si es VIP y cantidad > 5 aplica un 25%. Si solo es VIP un 15%. Si solo cantidad > 5 un 10%.

Crea un método en el programa principal llamado `ProcesaPedido` que reciba:

- Los datos del pedido (importe, cantidad, esVip).
- Un delegado de tipo `CalculoDescuento`.

El método debe calcular el descuento usando el delegado, restar el descuento al importe y mostrar por consola el precio final y el descuento aplicado.

En el `Main`, prueba `ProcesaPedido` con diferentes combinaciones de datos y pasando los distintos métodos de descuento.

Ejercicio 3. Delegado genérico Comparador

Una vez comprendido el concepto de delegados, se va generalizar creando un primer delegado genérico. Para ello deberás:

```
Ejercicio 3. Delegado genérico Comparador

     SAL     AGUA  AZUCAR     VINO
    COLA     CAFE    ZUMO   LECHE

       3        4        5
     2,4      4,4        5

Pulsar una tecla para finalizar...
```

Requisitos

Crea una aplicación con un método estático genérico `Mostrar`, al que le pases una dentada del tipo `T` y la muestre con una correcta tabulación. Posteriormente, prueba este método con diferentes tipos.

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

Ejercicio 4. Refactorizando ejercicio 1 y ejercicio 2

Modifica el ejercicio 1 y 2 para que ahora utilicen los delegados genéricos de la BCL

```
Ejercicio 4. Refactorización con Delegados Genéricos

1. Ejecutar Ejercicio 1 (Operación)
2. Ejecutar Ejercicio 2 (Descuentos)
3. Salir
Selecciona una opción:
```

Ejercicio 5. Practicando con delegados predefinidos

```
Ejercicio 5. Practicando con delegados predefinidos

AddTwoNumbers --> 15
SquareANumber --> 64
GetTopSpeed --> 108,4
MultiploCinco --> False
ProcedimientoDesconocido -->5 7 9 7,400000095367432 10,400000095367432 12
Introduce Valor 1
5
Introduce Valor 2
3
Calcula --> 15

Pulsa una tecla para continuar...
```

Requisitos

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 6. Sustitución de interfaces por delegados

Partiendo de la propuesta de solución (o la tuya) al **ejercicio 6 de genéricos (unidad 18)**, donde implementábamos diferentes formas de calcular la media y obtener la temperatura usando interfaces a modo de '*estrategia*'. Realiza las siguientes modificaciones:

```
Ejercicio 6. Delegados genéricos con múltiples parámetros

De cuantas provincias quieres recoger la temperatura: 5
Introduce la provincia nº1:  Alicante
Introduce la provincia nº2:  Valencia
Introduce la provincia nº3:  Castellón
Introduce la provincia nº4:  Murcia
Introduce la provincia nº5:  Albacete

Provincias con temperatura máxima mayor que la media:
Valencia
Murcia

Provincias con temperatura mínima menor que la media:
Valencia
Albacete

Provincias con temperatura mínima igual que la media:

Pulsa una tecla para finalizar...
```

Requisitos

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.