



Ejercicios Genericos

[Descargar estos ejercicios](#)

Índice

- [Ejercicio 1](#)
- [Ejercicio 2](#)
- [Ejercicio 3](#)
-  [Ejercicio 4](#)
-  [Ejercicio 5](#)
- [Ejercicio 6](#)

Ejercicio 1

Crea una clase parametrizada **Fraccion** , que tendrá dos propiedades públicas de solo lectura **Numerador** y **Denominador** y un constructor para inicializar los datos. Prueba el siguiente programa con la clase creada:

```
class Program
{
    public static void Main()
    {
        Fraccion<int> fraccionEnteros = new Fraccion<int>(4, 5);
        Console.WriteLine($"La fracción es: {fraccionEnteros.Numerador}/{fraccionEnteros.Denominador}");
        Fraccion<long> fraccionDecimales = new Fraccion<long>(4, 2);
        Console.WriteLine($"La fracción es: {fraccionDecimales.Numerador}/{fraccionDecimales.Denominador}");
    }
}
```

Ejercicio 2

Partiendo de la siguiente definición de clase parametrizada...

```
class A<T, U>
{
    private T clave;
    private U valor;
    ...
}
```

- Define un constructor que reciba los dos atributos como parámetro.
- Crea 2 propiedades que te permitirán devolver los dos atributos.
- Prueba la clase en la **Main** con una clave de tipo entero y un valor de tipo cadena.

Nota: Sin usar el código 'autogenerado' por el IDE.

Ejercicio 3

Crea una clase estática genérica llamada **Comparador<T>** que posea a su vez dos métodos de utilidad estáticos llamados **Mayor** y **Menor** . Ambos recibirán dos parámetros del tipo genérico, y devolverán true o false en el caso de que el primer parámetro sea mayor que el segundo y viceversa, **¿qué problemas has encontrado?**

La mejor forma de solucionarlo, es obligando a que el parámetro genérico implemente la interface `Comparable<T>` .

Crea una clase programa que te permita probar estos métodos, mandando diferentes elementos `int` , `string` , `float` , etc.

Ejercicio 4

Partiendo del ejercicio anterior, crea una clase **Persona** que tenga solo dos propiedades: **Nombre** y **Edad** .

Comprueba si funcionan los métodos **Mayor** y **Menor** con ella, ¿qué ocurre?. Ahora haz que la clase derive de `Comparable<Persona>` y de `Cloneable` y que invalide el `ToString()` .

Crea un programa que te permita saber, de dos objetos **Persona** distintos, cual es el mayor. Clona una persona y prueba los clones con el método estático **Menor** .

✓ Ejercicio 5

Vamos utilizar interfaces para utilizar algo **similar al patrón estrategia** del caso de estudio. Pero a través de métodos estáticos en lugar de clases.

Para ello, vamos a definir en primer lugar la clase **TemperaturasXProvincia** que contendrá el nombre de una provincia y sus temperaturas máxima y mínima respectivamente.

```
class TemperaturasXProvincia
{
    public string Provincia { get; }
    public float TemperaturaMaxima { get; }
    public float TemperaturaMinima { get; }
    public TemperaturasXProvincia(string provincia, float temperaturaMaxima, float tempe
    {
        Provincia = provincia;
        TemperaturaMaxima = temperaturaMaxima;
        TemperaturaMinima = temperaturaMinima;
    }
}
```

Definiremos el interfaz **IObténTemperatura** que obligará a implementar una '*estrategia*' de obtención de temperatura sobre un objeto de tipo **TemperaturasXProvincia**. Esto es, dado un objeto de tipo **TemperaturasXProvincia** me devolverá una de las temperaturas que contiene. En este caso la máxima o la mínima pero piensa que en el futuro este tipo de objetos podría contener una propiedad **TemperaturaMedia**.

Además, vamos a definir un interfaz parametrizado **ICumplePredicado** que obligue a implementar un método **bool Predicado(T o1, T o2)** al que le lleguen dos objetos y me devuelva true si cumplen un determinado predicado.

En la clase del programa principal, tendremos este método de utilidad que pedirá nombres de provincia y asignará aleatoriamente ambas temperaturas devolviéndome un array de **TemperaturasXProvincia**.


```

static TemperaturasXProvincia[] RecogeTemperaturasPorProvincia()
{
    Console.Write("De cuantas provincias quieres recoger la temperatura: ");
    var temperaturasPorProvincia = new TemperaturasXProvincia[int.Parse(Console.ReadLine)];
    Random seed = new Random();
    for (int i = 0; i < temperaturasPorProvincia.Length; i++)
    {
        Console.Write($"Introduce la provincia nº{i + 1}: ");
        string provincia = Console.ReadLine();
        float temperaturaMaxima = seed.Next(17, 25);
        float temperaturaMinima = seed.Next(-5, 17);
        Console.Write("\n\n");
        temperaturasPorProvincia[i] = new TemperaturasXProvincia(
            provincia,
            temperaturaMaxima,
            temperaturaMinima);
    }
    return temperaturasPorProvincia;
}

```

Se pide:

1. Implementar en la clase principal un método llamado **MediaTemperaturas** al que le pasemos el array de **TemperaturasXProvincia** y un objeto que implemente la estrategia definida en **IObténTemperatura** . De tal manera que, sin cambiar el método, pueda calcular la media de las máximas, de las mínimas o en un futuro de las medias.
2. Implementar en la clase principal un método llamado **MuestraProvincias** al que le pasemos el array de **TemperaturasXProvincia** un valor de **temperatura** , un objeto que implemente la estrategia definida en **IObténTemperatura** y un objeto que implemente un predicado definido en **ICumplePredicado** . De tal manera que me muestre aquellas provincias cuya temperatura obtenida por **IObténTemperatura** cumpla un determinado predicado.
3. Crea un programa principal que usando los métodos definidos anteriormente...
 1. Muestre las provincias cuya máxima sea mayor a la media de las máximas.
 2. Muestre las provincias cuya mínima sea menor a la media de las mínimas.
 3. Muestre las provincias cuya mínima sea igual a la media de las mínimas.

 **Pista:** Puedes definir los siguientes tipos/clases públicas para usar en el **Main** que implementen las estrategias de obtención de temperatura y los predicados necesarios dentro de la case **TemperaturasXProvincia**

- class **ObténMaxima** que me permita obtener la temperatura máxima.
- class **ObténMinima** que me permita obtener la temperatura máxima.
- class **MayorQue** que me si una temperatura es mayor que la otra.
- class **MenorQue** que me si una temperatura es menor que la otra.
- class **IgualQue** que me si dos temperaturas son iguales.

Ejercicio 6

Crea una clase genérica **Lista** que contenga:

1. Un array parametrizado de tipo T privado.
2. Un constructor para inicializar el array a 0 elementos.
3. Un método **Add** al que le llegue un dato de tipo parametrizado, redimensione el array y lo añada al final de este.
4. Definiremos un **indizador público** para la clase y así poder acceder y modificar el elemento correspondiente en el array.

Prueba la clase creada con el siguiente programa:

```
class Program
{
    public static void Main()
    {
        Lista<int> lista=new Lista<int>();
        lista.Add(5);
        Console.WriteLine(lista[0]);
        lista.Add(8);
        Console.WriteLine(lista[1]);
        lista[1]=10;
        Console.WriteLine(lista[1]);
        //Prueba la lista con string
    }
}
```