Ejercicios Colecciones de la BCL

Descargar estos ejercicios

Índice

1. Ejercicio 1
2. ✅ Ejercicio 2
3. ✅ Ejercicio 3
4. ✅ Ejercicio 4
5. Ejercicio 5

Ejercicio 1

En este ejercicio se va a practicar el uso de la lista genérica de la BCL, para ello deberás implementar los siguientes métodos y un programa para poder textear el funcionamiento de estos.

1. Implementa un método llamado `BorraEnteros` que reciba como parámetros una lista genérica de enteros (`List<int>` , que deberás inicializar previamente) y un número entero.
   Lo que hará será modificar la lista borrando los números que coincidan con el entero indicado.
2. Implementa una función llamada `Mezcla` que reciba como parámetro dos listas de enteros (ya ordenadas), y devuelva como resultado otra lista donde se unan las dos anteriores, pero con los números también en orden.
3. Implementa un método llamado `ImprimeInverso` que reciba como parámetros una lista de Personas y una posición (entero), e imprima por pantalla en orden inverso los nombres de la lista desde esa posición hasta el principio.
   Puedes utiliza la clase persona de otros ejercicios.
4. Implementa un procedimiento llamado `OrdenaCadenas` que reciba como parámetro una lista de strings la ordene alfabéticamente. Deberás usar algún algoritmo de ordenación y el método `CompareTo` para comparar las cadenas antes de realizar los intercambios.

✅ Ejercicio 2

Crea una clase `Automovil` con los datos básicos de los automóviles (marca, modelo, cilindrada, año de fabricación, etc) y los métodos necesarios para poder usarla con comodidad.
Crea una clase program con una serie de métodos que nos permitan trabajar con una lista genérica de automóviles.

- Necesitaremos un Método `AñadeAutomovil` que a partir de una lista y un automóvil, añadirá este a la lista.
- `EliminaAutomovil` que eliminará el automóvil con el índice X que se haya pasado como argumento.
- Crea un método `AutomovilesPorAñoFabricacion` , que te permita encontrar en la lista los coches con una determinada fecha de fabricación y que retorne una nueva lista con esos datos.
- Otro método `AutomovilesPorAñoFabricacionYColor` que devuelva una sublista con los coches de la lista que sean de un determinado color y una fecha pasados ambos como parámetros.
- Método que permita mostrar el contenido de la lista.

✅ Ejercicio 3

Utilizando la clase genérica `Dictionary<K, V>` definida en `System.Collections.Generic` , implementa un sencillo programa de consola que pida nombres por teclado hasta que introduzcamos la cadena **"fin"**.
En ese momento mostraremos los nombres introducidos y cuantas veces se ha introducido cada uno.
Para mostrar el resultado, deberás mostrar por un lado las claves y posteriormente el par clave valor. Para ello tendrás que recorre el diccionario 2 veces, una con un `foreach` para las claves y otra con un `foreach` para el diccionario obteniendo pares clave valor con la siguiente clase `KeyValuePair<K, V>`

💡 **Pista**: Tienes que guardar los nombres como clave en el diccionario y el número de veces que se ha introducido como valor.

✅ Ejercicio 4

Crea una clase **`Polinomio`** que guarde los datos de un polinomio en un diccionario genérico ordenado.

$$9x^7 - 3x^3 - 7x + 5$$

Por ejemplo el polinomio se guardaría en una propiedad privada automática de forma equivalente a esta ...

```
private SortedDictionary<int, int> Monomios { get; set; }
// Guardando el polinomio en un diccionario de la siguiente forma ...
Monomios = new SortedDictionary<int, int>() { {0, 5}, {1, -7}, {3, -3}, {7, 9} };
```

Donde los monomios se guardan en orden inverso, la "*llave*" es el **exponente** y el "*valor*" el **coeficiente**.

El constructor de la clase polinomio lo recibirá como una cadena de la siguiente forma: **`9x7-3x3-7x+5`** y su método string **`ToString()`** lo mostrará de forma idéntica.

Define un método de clase estático y publico llamado **`Suma`** donde entren dos objetos polinomios en forma devuelva otro con el resultado de la suma de tal manera que si en el programa principal quiero hacer la siguiente suma ...

$$(9x^7 - 3x^3 - 7x + 5) + (4x^2 - 1) = 9x^7 - 3x^3 + 4x^2 - 7x + 4$$

Tendré que escribir el siguiente código ...

```
Polinomio suma = Polinomio.Suma(new Polinomio("9x7-3x3-7x+5"), new Polinomio("4x2-1"));
Console.WriteLine(suma);
```

y su ejecución mostrará ... **`9x7-3x3+4x2-7x+4`**
Recuerda que internamente tendremos en cada objeto polinomio una colección equivalente a las siguientes...

```
// polinomio 1
new SortedDictionary<int, int>(){{ 0, 5 }, { 1, -7 }, { 3, -3 }, { 7, 9 }};
// polinomio 2
new SortedDictionary<int, int>(){{ 0, -1 }, { 2, 4 }};
// suma
new SortedDictionary<int, int>() {{ 0, 4 }, { 1, -7 }, { 2, 4 }, { 3, -3 }, { 7, 9 }};
```

Ejercicio 5

Crea una clase llamada **DatosContacto** con los siguientes campos:

- El DNI de la persona (string)
- El nombre completo de la persona (string)
- La dirección completa de la persona (string)
- Telefono (string)

En la clase del programa principal:

- Crea un diccionario llamado `agendaContactos` con una clave de tipo string y el valor de la clase `DatosContacto` .
- Añade un método estático `CreaContactos` que devuelva el diccionario rellenado por el usuario, hasta que se introduzca un DNI vacío.
- Añade el método estático `BorraContacto` que reciba como parámetros un DNI (como cadena) y el diccionario. Borrará del mismo, el dato cuya clave coincida con el DNI que se le pasa.
- Añade un método estático `AñadeContacto` que reciba como parámetro un dato de tipo DatosContacto, y el diccionario con la agenda de contactos, y añada a la tabla la persona indicada.
- Crea `MuestraAgenda` que reciba como parámetro el diccionario y muestre su contenido.
- Crea un programa principal que te permita probar todos los métodos.

Nota: Como clave para cada contacto utiliza su DNI. Deberás controlar si las claves están en el diccionario anteriores de acceder a el, para evitar posteriores excepciones.