

Índice

- [Ejercicio 1. Gestión en Biblioteca](#)
- [Ejercicio 2. Coincidencia en lista de cadenas](#)
- [Ejercicio 3. Múltiplos en lista de números](#)
- [Ejercicio 4. Operaciones con funciones Lambda](#)
- [Ejercicio 5. Carrito compra con funcional](#)

Ejercicios Unidad 20 - Programación Funcional - Lambda

[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. Gestión en Biblioteca

A partir del código que se proporciona como recurso y que contiene un `Value Object Libro` y el método que nos proporciona una lista con los datos de una serie de libros. Vamos a realizar el código necesario para gestionar nuestra biblioteca:

Ejercicio 1. Gestión en Biblioteca

```
Prestando ... { DNI = 22111333, Título = Cien años de soledad, ISBN = 9788420471839 }
Prestando ... { DNI = 22111333, Título = Los mejores cuentos de Clarín, ISBN = 97884315
True
False
Título: Los mejores cuentos de Clarín, Autor: Leopoldo Alas Clarín, Editorial: De Vecch
ISBN: 9788431533441, Nº Páginas: 145, Reseña: Una cuidadosa selección que nos muestra l
riqueza de los recursos estilísticos de este gran escritor del s. XIX.En el volumen se
Doña Berta, Benedictino, Manín de Pepa José, Zurita, Cambio de luz, y la Conversión de
3
Título: El camino, Autor: Miguel Delibes, Editorial: Espasa,
ISBN: 9788467023664, Nº Páginas: 187, Reseña: Una de las más importantes obras de Migue
la historia de un niño , Daniel el Mochuelo, que tiene que trasladarse a la ciudad para
Una noche antes de partir Daniel recordará todo lo que le ha ocurrido en este lugar, su
y descubrirá que su camino está en esa aldea, unido a lo que ha sido hasta ese momento
Nostálgica novela realista a través de la cual podemos aprender que nunca sabemos lo qu
se nos ha escapado.

TítuloAutor { Título = Los mejores cuentos de Clarín, Autor = Leopoldo Alas Clarín }

pulsa una tecla para finalizar...
```

Requisitos

Crear una clase denominada `Biblioteca` que tendrá como propiedades: un `Nombre` de tipo string, una `lista de Libros` y una lista de string `Prestamos` .

- Su constructor recibirá el nombre y la lista de libros.
- Crea un método público denominado `BuscaPorISBN` que reciba una cadena con el ISBN y devuelva el primer libro con ese ISBN o null si no lo encuentra.

```
public Libro BuscaPorISBN(String isbn)
{
    [Tipo] TieneISBN = [funcion-λ];
    return Libros.Find(TieneISBN);
}
```

- Ahora vamos a practicar con el concepto '*de tipos anónimos*' para guardar solo los datos referentes a un préstamo de un **Libro** en la lista Prestamos. **Sin definir una nueva clase o tipo Préstamo.**
 - Para ello, vamos a crear un método público en la clase Biblioteca denominado **Presta**. Este método recibirá dos string: **dni** del socio **ISBN** del libro a prestar.
 - Buscaremos el **ISBN** en la lista de libros y si lo encontramos, crearemos un nuevo objeto anónimo var préstamo con las propiedades **DNI**, **Título** e **ISBN**.
- ✦ **Nota:** En caso de no existir el ISBN en la biblioteca generaremos una **BliblitecaException** con el mensaje correspondiente.
- Por último, añadiremos a la lista de **Prestamos** la cadena resultado de pasar a **ToString** este objeto anónimo.
- Crea otro método público **EstaPrestado** al que se le pasará el **ISBN** de un libro y devolverá un Booleano que nos indicará si el libro se encuentra prestado o no,
 - Crea un método público **CuentaLibrosConNumeroDePaginasMenorA** que reciba un valor entero y te devuelva la cantidad de libros con un número de páginas menor a ese valor entero.
 - Crea un método público **EliminaPorAutor** que reciba el nombre de un autor y borre de la biblioteca aquellos libros de ese autor.
 - Redefine el método **ToString()** en **Libro** **creando un tipo anónimo** con **Libro**, **Autor**, e **ISBN** y devolviendo su **ToString**.
 - Redefine un método **AutorTitulo** al que le llega un ISB y devuelve el libro de la biblioteca que coincida con el ISB, pero esta vez solo nos interesará sacar la información de Título y Autor, por lo que se creará un **Tipo Record** para hacerlo.

Ejercicio 2. Coincidencia en lista de cadenas

Crea una aplicación que sirva para **buscar coincidencias en una lista de cadenas**.

Ejercicio 2. Coincidencia en lista de cadenas

Lista: rosa mesa flor ventana blanco perro sillón azul melón
Introduce una cadena a buscar: an

Buscar coincidencias de an - Usando Clausuras

Pulsa una tecla...

ventana blanco

Buscar coincidencias de an - Sin usar Clausuras

Pulsa una tecla...

ventana blanco

Pulsa una tecla...

Requisitos

- Para ello, asignaremos a **unos métodos** públicos y estáticos llamados:

`CoincidenciasCadena_UsandoClausuras`

`CoincidenciasCadena_SinUsarClausuras`

La funcionalidad necesaria de forma que mediante **funciones-λ** y a partir de una lista y una cadena obtengan el resultado de una sublista con las palabras que contienen la cadena. Puedes usar sobre la lista el método `FindAll` para realizar la búsqueda de la cadena.

Ten en cuenta que el método `FindAll` necesitará un predicado para el cual utilizaremos otra función-λ para formarlo.

✦ **Nota:** Puede serte de utilidad la operación `Contains` sobre cadenas.

- Hazlo de dos forma diferentes, usando clausura sobre la cadena a buscar y sin clausura, cada forma se asignará al objeto delegado correspondiente.
 - Por lo que al método que no usa clausuras le llegará una lista y una cadena y devolverá el resultado ya resuelto.
 - Mientras que al método que si usa clausuras le llegará solo una cadena, por lo que tendrá que devolver el resultado en forma de landa, para que sea resuelto a posteriori.



Aviso

Puede darte una pista que devolverá el delegado `CoincidenciasCadena_SinUsarClausuras` si te fijas en el for y en el delegado a completar **busquedaClausura** de la Main

Al final de la main deberás mostrar, con el método `public void ForEach (Action<T> action);` definido en `List<T>`, la lista resultante que devuelva la invocación del delegado.

Ejercicio 3. Múltiplos en lista de números

Crea una aplicación que a partir de una Lista de enteros, te muestre los múltiplos de un número introducido por teclado que existan en la lista, usando **funciones-λ** y el operador `?:`. Realiza todo el código en la Main usando funciones lambda y los siguientes métodos

Requisitos

- Hazlo de dos forma diferentes, usando clausura sobre el número introducido y sin clausura.
 - Crea un método público y estatico `EsMultiploDe_ConClausura` al que le llegue un parámetro de entrada entero y devuelva un delegado que capture el parámetro de entrada y compruebe su multiplicidad con un parámetro del propio delegado.
 - Crea otro método similar al anterior `EsMultiploDe_SinClausura` con la funcionalidad necesaria, ahora sin clausura.

Ejercicio 3. Múltiplos en lista de números

Lista: 2 4 12 3 18 4 7 6 21 33 17 30 27
Introduce un número: 6

Múltiplos de 6 - Usando Clausuras

Pulsa una tecla...

12 18 6 30

Múltiplos de 6 - Sin usar Clausuras

Pulsa una tecla...

12 18 6 30

Pulsa una tecla...

Ejercicio 4. Operaciones con funciones Lambda

Programa para seguir practicando las funciones lambda con operaciones sencillas.

Ejercicio 4. Operaciones con funciones Lambda

50,26548245743669

True

la palabra Esto tiene una longitud de 4

la palabra es tiene una longitud de 2

la palabra una tiene una longitud de 3

la palabra prueba tiene una longitud de 6

la palabra para tiene una longitud de 4

la palabra crear tiene una longitud de 5

la palabra el tiene una longitud de 2

la palabra diccionario tiene una longitud de 11

Pulsa una tecla para finalizar...

Requisitos

- Escribir una función **VolumenEsfera** que devuelva un Delegado con el valor de este, a partir de un radio.
- Escribe un método **EsCapitular**, que devuelva un Delegado indicando si la primera letra de una palabra es mayúscula. Comprueba que la palabra no es nula.
- Escribe un método **DiccionarioDePalabras**, que reciba una frase y devuelva un diccionario con las palabras que contiene y la longitud de estas.

Ejercicio 5. Carrito compra con funcional

Programa que permite crear un carrito de compra usando funciones lambda combinadas.

Ejercicio 5. Carrito compra con funcional

para un precio de 4euros y un iva de 3% el total final será: 4,12

para un precio de 4euros y un descuento de 3% el total final será: 3,88

el total del carrito con un descuento de de 3% será: 58,6

Pulsa una tecla para finalizar...

Requisitos

- Escribe un método **AplicaIva**, que devuelva un Delegado con el precio total de un producto, a partir de aplicar un IVA al precio inicial.

- Escribe un método `AplicaDescuento` , que devuelva un Delegado con el precio total de un producto a partir de aplicar un descuento a un precio inicial.
- Escribe un método `CarritoCompra` , que devuelva un Delegado con el precio total de una lista de la compra. La lista de la compra se recibirá como una lista de tuplas o pares de valores, donde el primer valor es el precio y el segundo el porcentaje (valor que se referirá al posible descuento o IVA aplicado). El delegado también recibirá una de las funciones anónimas predefinidas anteriormente (descuento o IVA).