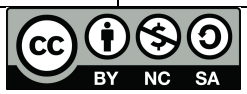
Estructura de Datos y Algoritmos. Ingeniería Informática Universidad Carlos III de Madrid CASO PRÁCTICO. CURSO 2017-2018	uc3m | Universidad Carlos III de Madrid



FASE 1. SharingCar

El objetivo es crear un servicio para compartir vehículo (similar al servicio blablacar).

Como primer paso crea una estructura de datos, **Request**, que te permita definir y almacenar los datos básicos necesarios para que un usuario pueda solicitar un viaje. Por simplificar el problema, sólo consideraremos los siguientes atributos (aunque en una aplicación real debería almacenarse más información):

- source: ciudad de origen (por ejemplo, Madrid).
- target: ciudad de destino (por ejemplo, Barcelona).
- login: id del usuario (por ejemplo, isegura)

Implementa una estructura de datos tipo lista, **LstRequest**, que te permita almacenar todas las peticiones. La estructura debe implementar todos los métodos de la interfaz IList. Elije entre una implementación basada en lista simple o lista doble y explica por qué has elegido esa implementación.
Crea una clase **SharingCar** para gestionar las peticiones. La clase SharingCar debe implementar los siguientes métodos:

1) *mergeAlternateRequests*, que reciba como parámetros dos objetos de tipo LstRequest. El método debe devolver un objeto de tipo LstRequest cuyo contenido sea los elementos de ambas listas de entrada de forma alterna. Veamos un ejemplo con letras:
   {A,C,D,E,F,G} {B,J} -> {A,B,C,J,D,E,F,G}
2) *sharing*, que reciba como parámetros dos objetos de tipo LstRequest, y que devuelva un objeto de tipo LstRequest con todas las peticiones de ambas listas de entrada, cuyas ciudades de origen son iguales, y también sus ciudades de destino. Por ejemplo, si A es una petición que está en la primera lista, con ciudad origen Madrid y destino Barcelona, y a su vez B es una petición de la segunda lista, también con origen Madrid y destino Barcelona, ambas peticiones, A y B, deberán estar en la lista que devuelve el método.
3) *sort*, que reciba como parámetros un objeto de tipo LstRequest y un integer. El método debe devolver un objeto de tipo LstRequest con las peticiones de la lista de entrada, ordenadas en orden alfabético ascendente del atributo ciudad de origen (si el valor del segundo parámetro es 1), en orden alfabético ascendente de la ciudad de destino (si el valor es 2), y en cualquier otro caso, en orden alfabético ascendente del login.

4) *removeDuplicates*, que reciba como parámetro un objeto de tipo LstRequest, y elimine las peticiones duplicadas (deben coincidir en origen, destino y login). El método debe devolver la nueva lista sin duplicados.
5) *searchSource* que reciba como parámetros un objeto de tipo LstRequest y un String y devuelva un objeto de tipo LstRequest con todas las peticiones cuya ciudad de origen coincide con el parámetro de entrada
6) *searchTarget*, que reciba como parámetros un objeto de tipo LstRequest y un String y devuelva un objeto de tipo LstRequest con todas las peticiones cuya ciudad de destine coincide con el parámetro de entrada.
7) Escribe una tabla con las funciones Big-O de cada uno de los métodos anteriores. Incluye una breve descripción para cada método.
8) Añade el código necesario en el método main de la clase SharingCar para probar cada uno de los métodos anteriores.