INSTITUTO TECNOLÓGICO DE COSTA RICA
CURSO: DISEÑO DE SOFTWARE
PROFESOR: ANDRÉS VÍQUEZ VÍQUEZ
ACTIVIDAD: PRUEBAS UNITARIAS

1. Se le proporciona una clase simple: **Database** que almacena enteros. El constructor recibe un arreglo de números, que se añade a un campo privado. La clase tiene la funcionalidad de añadir, eliminar y recuperar todos los elementos almacenados. Su tarea es probar la clase, en otras palabras, escriba las pruebas para asegurarse de que sus métodos funcionan como se espera.

   Restricciones:

   - La capacidad del arreglo de almacenamiento debe ser exactamente de 16 enteros.
     - Si el tamaño del arreglo no es de 16 enteros, se lanza una excepción InvalidOperationException.
   - La operación Add() debe añadir un elemento en la siguiente celda libre (como una pila).
     - Si hay 16 elementos en la Base de Datos y se intenta añadir un 17º, se lanza una excepción InvalidOperationException.
   - La operación Remove() debe permitir solo la eliminación de un elemento en el último índice (como una pila).
     - Si intentas eliminar un elemento de una Base de Datos vacía, se lanza una excepción InvalidOperationException.
   - Los constructores deben tomar solo enteros y almacenarlos en el arreglo.
   - El método Fetch() debe devolver los elementos como un arreglo.

   Consejo: no olvide probar los constructores: ¡también son métodos!

2. La clase **Database** la hemos modificado y añadido más funcionalidad en una nueva clase llamada **Extended Database**. Ahora soporta añadir, eliminar y encontrar Personas. Hay dos tipos de métodos de búsqueda: primero: FindById (long id) y el segundo: FindByUsername (string username). Como puede suponer, cada persona tiene un id único y un nombre de usuario único. Su tarea es probar la clase proporcionada.

   Restricciones: la base de datos debe tener métodos

   - Add
     - Si ya hay usuarios con este nombre de usuario, se lanza una excepción InvalidOperationException.
     - Si ya hay usuarios con este id, se lanza una excepción InvalidOperationException.
   - Remove
   - FindByUsername
     - Si no hay un usuario presente por ese nombre de usuario, se lanza una excepción InvalidOperationException.

    - Si el parámetro de nombre de usuario es nulo, se lanza una excepción ArgumentNullException.
    - Los argumentos son CaseSensitive.
- FindById
    - Si no hay un usuario presente por ese id, se lanza una excepción InvalidOperationException.
    - Si se encuentran ids negativos, se lanza una excepción ArgumentOutOfRangeException.

Consejo: no olvide probar los constructores ¡también son métodos! Además, tenga en cuenta que toda la funcionalidad de ejercicio anterior sigue existiendo y necesita probarla de nuevo.

3. Se le proporciona una clase llamada **Car**. La clase proporcionada es simple, su objetivo principal es representar algunas de las funcionalidades de un auto. Cada auto contiene información sobre su marca, modelo, consumo de combustible, cantidad de combustible y capacidad de combustible. Además, cada auto puede añadir combustible a su tanque al repostar y puede recorrer una distancia al conducir. Para ser conducido, nuestro auto necesita tener suficiente combustible.

   La clase proporcionada funciona perfectamente bien y no debe cambiarla. Su tarea es probar exactamente cada parte del código dentro de la clase:

   - Debe probar todos los constructores.
   - Debe probar todas las propiedades (getters y setters).
   - Debe probar todos los métodos y validaciones dentro de la clase.