**C#**

1. Implement Singleton in multithreading environment (all known implementations)
2. You have class that represent information about currency and amount: **Price.** You need to implement some methods to be able to use it as a key in Dictionary like: **Dictionary<Price, SomeOtherType>.**
3. Please suggest simple implementation for extension method **Where** for **IEnumerable<TSource>**
4. Check this code snippet:

**public** class Test

**{**

**public** IList**<**ProductRubPrice**>** ComposeProductRubPrices**(**IList**<**Product**>** products**,** IList**<**Price**>** prices**)**

**{**

var productPrices **=** **new** List**<**ProductRubPrice**>();**

**foreach** **(**var product **in** products**)**

**{**

var filteredPrices **=** prices**.**Where**(**p **=>** p**.**ProductId **==** product**.**Id **&&** p**.**Currency **==** "RUB"**)**

**.**Select**(**p **=>** **new** ProductRubPrice **{** Amount **=** p**.**Amount**,** ProductName **=** product**.**Name **})**

**.**ToList**();**

**if** **(**filteredPrices**.**Any**())**

**{**

productPrices**.**AddRange**(**filteredPrices**);**

**}**

**}**

**return** productPrices**.**Distinct**().**ToList**();**

**}**

**}**

**public** class Product

**{**

**public** int Id**;**

**public** string Name**;**

**}**

**public** class Price

**{**

**public** int ProductId**;**

**public** decimal Amount**;**

**public** string Currency**;**

**}**

**public** class ProductRubPrice

**{**

**public** string ProductName**;**

**public** decimal Amount**;**

**}**

Please suggest an optimized version of the code that solves the same task

1. Please suggest implementation for extension method that split async enumerable source per batches of certain size

IAsyncEnumerable<List<T>> ToBatchesAsync<T>(this IAsyncEnumerable<T> source, int batchSize, [EnumeratorCancellation] CancellationToken cancellationToken = default)