# Data Structures Exam: INStock

## Description

Pesho has been struggling lately. He is a major shareholder at one of the largest product manufacturers in the world. As such he is always looking for new ways to improve his game and stay on the top. He has hired you (because you are from SoftUni, therefore you are a grandmaster programmer) to create a product tracking system for him. He has a lot of products in stock so you have to make the system really fast.

* Add(Product) – Add the new manufactured Product in stock. You will need to implement the Contains() method as well.
* Contains(Product) – Checks if a particular product is in stock. \*Keep in mind that only labels are unique.
* Count – Returns the number of products currently in stock
* Find(int) – Return the **N**-th product that was added in stock. The index is based on insertion order in the data structure. If such index is not present, throw IndexOutOfRangeException/OutOfBounds in java
* ChangeQuantity(string,int) –Changes the quantity of a given product by n-amount. If the particular product is not in stock, throw ArgumentException
* FindByLabel(string) – Returns the product with a given label, throws ArgumentException if no such product is in stock
* FindFirstByAlphabeticalOrder(int) – Returns the first **N**-th products in stock ordered by label in alphabetical order or returns an empty enumeration if the passed argument is out of range.
* FindAllInPriceRange(double,double) – Returns all products within given **price** range (lower end is exclusive, higher end is inclusive). Keep in mind that they should be returned in descending order. If there are no such products, return empty enumeration (collection)
* FindAllByPrice(double) – Returns all products in stock with given **price** or **empty** collection if none were found
* FindFirstMostExpensiveProducts(int) – Returns the first **N** **products** with **highest** price in stock or throw ArgumentException if less than **count** exist
* FindAllByQuantity(int) – Returns all products in stock with given remaining **quantity**. If there is no product with identical quantity, return empty enumeration. The last product should be the one with last changed quantity or last added ( in order of change ).
* GetEnumerator<Product>() – Returns all products in stock.

Duplicates of the product class **are allowed**. That means that the price and quantity of multiple objects might be the same **(It is acceptable for the quantity to be 0).** There will be no duplicate references or labels. **The input will always be valid**.

**Note** – For java use **IndexOutOfBoundsException** and **IllegalArgumentException** classes

## Input / Output

You are given a **Visual Studio C# project skeleton** (unfinished project) / **IntelliJ Java project** holding the interface IProductStock, the unfinished classes ProductStock and Product. **Tests** covering the ProductStock **functionality** and **performance**.

Your task is to **finish this class** to make the tests run correctly.

* You are **not allowed to change the tests**.
* You are **not allowed to change the interface**.
* You can add to the Product class, but don't remove anything.
* You can edit the ProductStock class if it implements the IProductStock interface.

## Interface

The interface IProductStock in C# looks like the code below:

|  |
| --- |
| public interface IProductStock : IEnumerable<Product>  {  int Count { get; }  bool Contains(Product product);  void Add(Product product);  void ChangeQuantity(string product, int quantity);  Product Find(int index);  Product FindByLabel(string label);  IEnumerable<Product> FindFirstByAlphabeticalOrder(int count);  IEnumerable<Product> FindAllInRange(double lo, double hi);  IEnumerable<Product> FindAllByPrice(double price);  IEnumerable<Product> FindFirstMostExpensiveProducts(int count);  IEnumerable<Product> FindAllByQuantity(int quantity);  } |

The interface ProductStock in Java looks like the code below:

|  |
| --- |
| public interface ProductStock extends Iterable<Product>{  int getCount();  boolean contains(Product product);  void add(Product product);  void changeQuantity(String product, int quantity);  Product find(int index);  Product findByLabel(String label);  Iterable<Product> findFirstByAlphabeticalOrder(int count);  Iterable<Product> findAllInRange(double lo, double hi);  Iterable<Product> findAllByPrice(double price);  Iterable<Product> findFirstMostExpensiveProducts(int count);  Iterable<Product> findAllByQuantity(int quantity);  } |

## Submission

Submit an archive (.zip) of the source code. Your code **mustn't** contain namespaces/packages.

## Scoring

Each implemented method brings you a specific amount of points, some of the points are awarded for correct behavior, others for performance. **The performance tests might not work on your PC**. You need to cover all tests in each group to receive points. Bellow is a breakdown of all points by methods:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Correct Behavior | Performance | Total |
| Overall | 100 | 100 | 200 |