INStock

Description

Peter 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 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.
- **`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`**.
- **`FindByLabel(string)`**
  – Returns the product with a given label, throws **`ArgumentException`** if no such product is in stock.
- **`FindAllInPriceRange(decimal, decimal)`**
  – Returns all products within given **price** range (lower end and higher end are inclusive). Keep in mind that they should be returned in descending order. If there are no such products, return empty enumeration (collection).
- **`FindAllByPrice(decimal)`**
  – Returns all products in stock with given **price** or **empty** collection if none were found.
- **`FindMostExpensiveProducts(int)`**
  – Returns the most expensive product in stock.
- **`FindAllByQuantity(int)`**
  – Returns all products in stock with given remaining **quantity**. If there is no product with identical quantity, return empty enumeration.
- **`GetEnumerator<Product>()`**
  – Returns all products in stock.
- **`this[int index]`**
  - Indexer

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).**