Lab – Test Driven Development

This document defines the lab for the "Java Advanced" course @ Software University.

INStock

John 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 an index is not present, throw **`IndexOutOfBoundsException`.**
- **`changeQuantity(String, int)`** –Changes the quantity of a given product by n-amount. If the particular product is not in stock, throw **`IllegalArgumentException`.**
- **`findByLabel(String)`** – Returns the product with a given label, throws **`IllegalArgumentException`** if no such product is in stock.
- **`findFirstByAlphabeticalOrder(int)`** – Returns the first **N**-th products in stock ordered by a label in alphabetical order or returns an empty collection if the passed argument is out of range.
- **`findAllInPriceRange(double, double)`** – Returns all products within a given **price** range (the lower end is exclusive, the higher end is inclusive). Keep in mind that they should be returned in descending order. If there are no such products, return the **empty collection**.
- **`findAllByPrice(double)`** – Returns all products in stock with the given **price** or **empty collection** if none were found.
- **`findFirstMostExpensiveProducts(int)`** – Returns the first **N products** with the **highest** price in stock or throws **`IllegalArgumentException`** if less than **counts** exist.
- **`findAllByQuantity(int)`** – Returns all products in stock with the given remaining **quantity**. If there is no product with an identical quantity, return an **empty collection**.
- **`getIterable<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**.

Input / Output

You are given an **IntelliJ Java project** holding the interface **`ProductStock that extends Iterable<Product>`**, the unfinished classes **`Instock`** and **`Product`**.

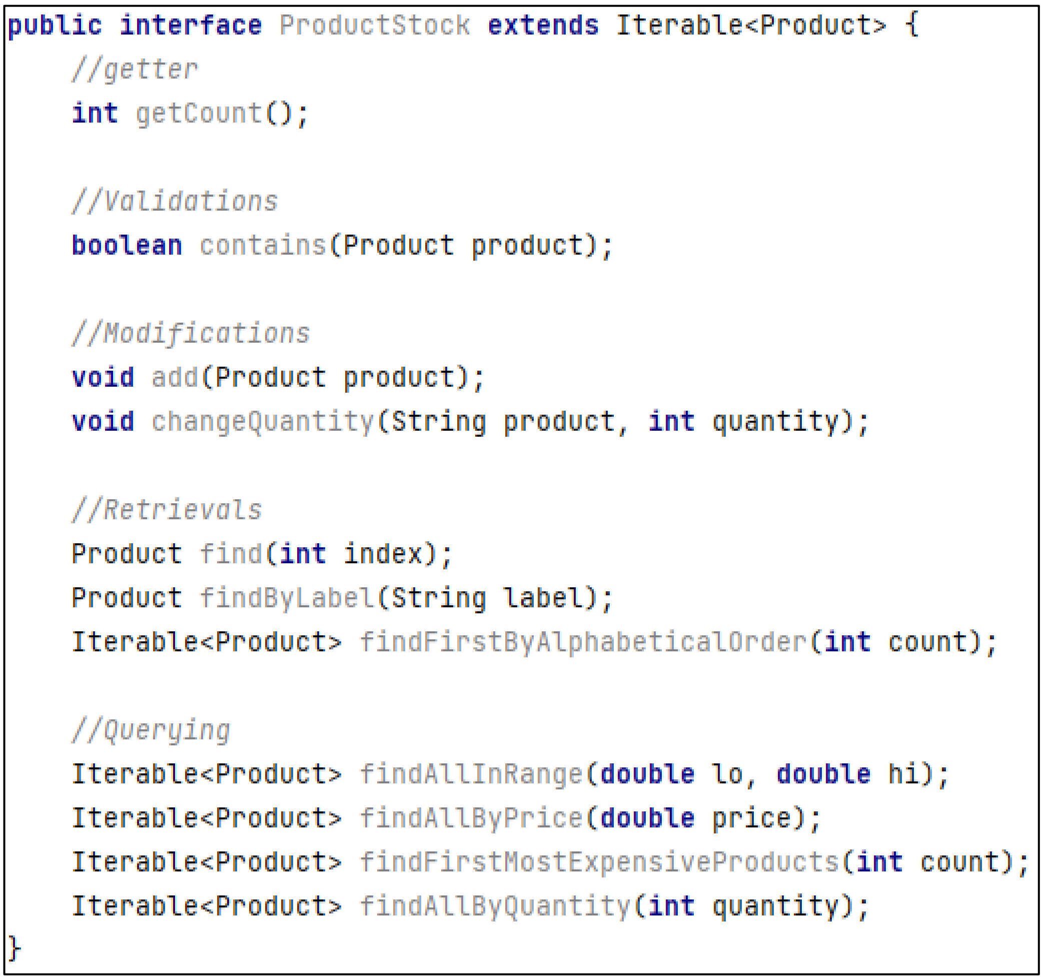
Your task is to **finish this class** and to create tests that run correctly.

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

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

Interface

The interface **ProductStock** looks like the code below:



```
public interface ProductStock extends Iterable<Product> {
    //getter
    int getCount();

    //Validations
    boolean contains(Product product);

    //Modifications
    void add(Product product);
    void changeQuantity(String product, int quantity);

    //Retrievals
    Product find(int index);
    Product findByLabel(String label);
    Iterable<Product> findFirstByAlphabeticalOrder(int count);

    //Querying
    Iterable<Product> findAllInRange(double lo, double hi);
    Iterable<Product> findAllByPrice(double price);
    Iterable<Product> findFirstMostExpensiveProducts(int count);
    Iterable<Product> findAllByQuantity(int quantity);
}
```

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.