

Accueil ► SI - Sciences Informatiques ► SI3 ► Intro POO ► Stuff to do - unevaluated ► Grouping objects - Products

<b>Commencé le</b>	mercredi 18 octobre 2017, 17:48
<b>État</b>	Terminé
<b>Terminé le</b>	mercredi 18 octobre 2017, 19:36
<b>Temps mis</b>	1 heure 48 min
<b>En retard</b>	1 heure 48 min
<b>Points</b>	7,00/7,00
<b>Note</b>	20,00 sur 20,00 (100%)

**Description**

Open the *product* project and complete the **StockManager** class through this and the next exercises.

**StockManager** uses an **ArrayList** to stock **Product** items. Its **addProduct** method already adds a product to the collection, but the following methods need completing: **delivery**, **findProduct**, **printProductDetails**, and **numberInStock**.

Each product sold by the company is represented by an instance of the **Product** class, which records a product's ID name, and how many of the product are currently in stock. The **Product** class defines the **increaseQuantity** method to record increases in the stock level of the product. The **sellOne** method records that one item of that product has been sold by reducing the quantity field level by 1. **Product** has been provided for you, and you should not need to change it.

**Question 1**

Correct

Note de 1,00 sur  
1,00

**Product** has been provided for you, and you should not need to change it.

Start by implementing the **StockManager#printProductDetails** method to ensure that you are able to iterate over the collection of products. Just print out the details of each **Product** returned by calling its **toString** method.

Paste just your **StockManager** code into the Answer box and Check your work.

For example:

Test	Result
<pre>Product[] stdProducts = new Product[3]; IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i] = new Product(i, "Prod" + i)); StockManager manager = new StockManager(); // two different ways of accessing stored products Arrays.asList(stdProducts).forEach(p -&gt; manager.addProduct(p)); IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i].increaseQuantity(i + 1)); manager.printProductDetails();</pre>	<pre>0: Prod0 stock level: 1 1: Prod1 stock level: 2 2: Prod2 stock level: 3</pre>

Réponse:

```
1 package products;
2
3 import java.util.ArrayList;
4
5 /**
6  * Manage the stock in a business. The stock is described by zero or more
7  * Products.
8  *
9  * @author (your name)
10 * @version (a version number or a date)
11 */
12 class StockManager {
13     // A list of the products.
14     private final ArrayList<Product> stock;
15
16     /**
17      * Initialise the stock manager.
18      */
19     ...
```

Vérifier

	Test	Expected	Got
✓	<pre>Product[] stdProducts = new Product[3]; IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i] = new Product(i, "Prod" + i)); StockManager manager = new StockManager(); // two different ways of accessing stored products Arrays.asList(stdProducts).forEach(p - &gt; manager.addProduct(p)); IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i].increaseQuantity(i + 1)); manager.printProductDetails();</pre>	<pre>0: Prod0 stock level: 1 1: Prod1 stock level: 2 2: Prod2 stock level: 3</pre>	<pre>0: Prod0 stock level: 1 1: Prod1 stock level: 2 2: Prod2 stock level: 3</pre>

Passed all tests! ✓

**Correct**

Note pour cet envoi : 1,00/1,00.

**Question 2**

Correct

Note de 1,00 sur

Implement the **StockManager#findProduct(int id)** method. This should look through the collection for a product whose ID field matches the ID argument of this method. If a matching product is found it should be returned as the method's result. If no matching product is found, return **null**.

1,00

This differs from the **printProductDetails** method in that it will not necessarily have to examine every product in the collection before a match is found.

When looking for a match, you will need to call the **getID** method on a **Product**.

Paste just your **StockManager** code into the Answer box and Check your work.

For example:

Test	Result
<pre>Product[] stdProducts = new Product[10]; IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i] = new Product(i, "Prod" + i)); StockManager manager = new StockManager(); // two different ways of accessing stored products Arrays.asList(stdProducts).forEach(p -&gt; manager.addProduct(p)); IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i].increaseQuantity(i + 1)); System.out.println(stdProducts[3] == manager.findProduct(3));</pre>	true

Réponse:

```
1 package products;
2
3 import java.util.ArrayList;
4
5 /**
6  * Manage the stock in a business. The stock is described by zero or more
7  * Products.
8  *
9  * @author (your name)
10 * @version (a version number or a date)
11 */
12 class StockManager {
13     // A list of the products.
14     private final ArrayList<Product> stock;
15
16     /**
17      * Initialise the stock manager.
18      */
19     public StockManager() {
20         stock = new ArrayList<>();
21     }
22
23     /**
24      * Add a product to the stock.
25      *
26      * @param product the product to add
27      */
28     public void addProduct(Product product) {
29         stock.add(product);
30     }
31
32     /**
33      * Find a product in the stock.
34      *
35      * @param id the id of the product to find
36      * @return the product if found, null otherwise
37      */
38     public Product findProduct(int id) {
39         for (Product product : stock) {
40             if (product.getID() == id) {
41                 return product;
42             }
43         }
44         return null;
45     }
46
47     /**
48      * Increase the quantity of a product in the stock.
49      *
50      * @param id the id of the product to increase
51      * @param quantity the quantity to increase by
52      */
53     public void increaseQuantity(int id, int quantity) {
54         Product product = findProduct(id);
55         if (product != null) {
56             product.increaseQuantity(quantity);
57         }
58     }
59 }
```

Vérifier

	Test	Expected	Got	
✓	<pre>Product[] stdProducts = new Product[10]; IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i] = new Product(i, "Prod" + i)); StockManager manager = new StockManager(); // two different ways of accessing stored products Arrays.asList(stdProducts).forEach(p -&gt; manager.addProduct(p)); IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i].increaseQuantity(i + 1)); System.out.println(stdProducts[3] == manager.findProduct(3));</pre>	true	true	✓
✓	<pre>Product[] stdProducts = new Product[10]; IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i] = new Product(i, "Prod" + i)); StockManager manager = new StockManager(); // two different ways of accessing stored products Arrays.asList(stdProducts).forEach(p -&gt; manager.addProduct(p)); IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i].increaseQuantity(i + 1)); System.out.println(stdProducts[3] == manager.findProduct(7));</pre>	false	false	✓
✓	<pre>Product[] stdProducts = new Product[10]; IntStream.range(0, stdProducts.length).forEach(i -</pre>	null	null	✓

```
> stdProducts[i] = new Product(i, "Prod" + i));
StockManager manager = new StockManager();
// two different ways of accessing stored products
Arrays.asList(stdProducts).forEach(p -> manager.addProduct(p));
IntStream.range(0, stdProducts.length).forEach(i -
> stdProducts[i].increaseQuantity(i + 1));
System.out.println(manager.findProduct(17));
```

Passed all tests! ✓

**Correct**

Note pour cet envoi : 1,00/1,00.

### Question 3

Correct

Note de 1,00 sur  
1,00

Implement the **numberInStock** method. This should locate a product in the collection with a matching ID, and return the current quantity of that product as a method result. If no product with a matching ID is found, return zero.

Paste just your **StockManager** code into the Answer box and Check your work.

**For example:**

Test	Result
<pre>Product[] stdProducts = new Product[10]; IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i] = new Product(i, "Prod" + i)); StockManager manager = new StockManager(); // two different ways of accessing stored products Arrays.asList(stdProducts).forEach(p -&gt; manager.addProduct(p)); IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i].increaseQuantity(i + 1)); System.out.println(manager.numberInStock(2));</pre>	3

Réponse:

```
1 package products;
2
3 import java.util.ArrayList;
4
5 /**
6  * Manage the stock in a business. The stock is described by zero or more
7  * Products.
8  *
9  * @author (your name)
10 * @version (a version number or a date)
11 */
12 class StockManager {
13     // A list of the products.
14     private final ArrayList<Product> stock;
15
16     /**
17      * Initialise the stock manager.
18      */
19     public StockManager() {
```

Vérifier

	Test	Expected	Got	
✓	<pre>Product[] stdProducts = new Product[10]; IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i] = new Product(i, "Prod" + i)); StockManager manager = new StockManager(); // two different ways of accessing stored products Arrays.asList(stdProducts).forEach(p -&gt; manager.addProduct(p)); IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i].increaseQuantity(i + 1));</pre>	3	3	✓

	System.out.println(manager.numberInStock(2));			
✓	<pre>Product[] stdProducts = new Product[10]; IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i] = new Product(i, "Prod" +i)); StockManager manager = new StockManager(); // two different ways of accessing stored products Arrays.asList(stdProducts).forEach(p -&gt; manager.addProduct(p)); IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i].increaseQuantity(i + 1)); System.out.println(manager.numberInStock(17));</pre>	0	0	✓

Passed all tests! ✓

**Correct**

Note pour cet envoi : 1,00/1,00.

**Question 4**

Correct

Note de 1,00 sur  
1,00

Implement the **delivery** method using a similar approach to that used for `numberInStock`. It should find the product with the given ID in the collection of products and then call its **increaseQuantity** method.

Paste just your **StockManager** code into the Answer box and Check your work.

**For example:**

Test	Result
<pre>Product[] stdProducts = new Product[10]; IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i] = new Product(i, "Prod" + i)); StockManager manager = new StockManager(); // two different ways of accessing stored products Arrays.asList(stdProducts).forEach(p -&gt; manager.addProduct(p)); IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i].increaseQuantity(i + 1)); manager.delivery(5, 7); System.out.println(manager.numberInStock(5));</pre>	13

Réponse:

```
1 package products;
2
3 import java.util.ArrayList;
4
5 /**
6  * Manage the stock in a business. The stock is described by zero or more
7  * Products.
8  *
9  * @author (your name)
10 * @version (a version number or a date)
11 */
12 class StockManager {
13     // A list of the products.
14     private final ArrayList<Product> stock;
15
16     /**
17      * Initialise the stock manager.
18      */
19     ...
```

Vérifier

	Test	Expected	Got	
✓	<pre>Product[] stdProducts = new Product[10]; IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i] = new Product(i, "Prod" + i)); StockManager manager = new StockManager(); // two different ways of accessing stored products Arrays.asList(stdProducts).forEach(p -&gt; manager.addProduct(p)); IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i].increaseQuantity(i + 1)); manager.delivery(5, 7); System.out.println(manager.numberInStock(5));</pre>	13	13	✓

Passed all tests! ✓

**Correct**

Note pour cet envoi : 1,00/1,00.

**Question 5**

Correct

Note de 1,00 sur  
1,00

Implement a method **StockManager#fewer(int numbers)** to print details of all products with stock levels below a given value (passed as a parameter to the method).

Paste just your **StockManager** code into the Answer box and Check your work.

**For example:**

Test	Result
<pre>Product[] stdProducts = new Product[10]; IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i] = new Product(i, "Prod" + i)); StockManager manager = new StockManager(); // two different ways of accessing stored products Arrays.asList(stdProducts).forEach(p -&gt; manager.addProduct(p)); IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i].increaseQuantity(i + 1)); manager.fewer(5);</pre>	<pre>0: Prod0 stock level: 1 1: Prod1 stock level: 2 2: Prod2 stock level: 3 3: Prod3 stock level: 4</pre>

Réponse:

```
1 package products;
2
3 import java.util.ArrayList;
4
5 /**
6  * Manage the stock in a business. The stock is described by zero or more
7  * Products.
8  *
9  * @author (your name)
10 * @version (a version number or a date)
11 */
12 class StockManager {
13     // A list of the products.
14     private final ArrayList<Product> stock;
15
16     /**
17      * Initialise the stock manager.
18      */
19     public StockManager() {
```

Vérifier

	Test	Expected	Got
✓	<pre>Product[] stdProducts = new Product[10]; IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i] = new Product(i, "Prod" + i)); StockManager manager = new StockManager(); // two different ways of accessing stored products Arrays.asList(stdProducts).forEach(p - &gt; manager.addProduct(p)); IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i].increaseQuantity(i + 1)); manager.fewer(5);</pre>	<pre>0: Prod0 stock level: 1 1: Prod1 stock level: 2 2: Prod2 stock level: 3 3: Prod3 stock level: 4</pre>	<pre>0: Prod0 stock level: 1 1: Prod1 stock level: 2 2: Prod2 stock level: 3 3: Prod3 stock level: 4</pre>

Passed all tests! ✓

**Correct**

Note pour cet envoi : 1,00/1,00.

**Question 6**

Correct

Note de 1,00 sur  
1,00

Modify the **addProduct** method so that a new product cannot be added to the product list with the same ID as an existing one.

You might find it useful to add another **StockManager** method **int numberOfProducts()** which returns the number of different types of product.

Paste just your **StockManager** code into the Answer box and Check your work.

For example:

Test	Result
<pre>Product[] stdProducts = new Product[10]; IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i] = new Product(i, "Prod" +i)); StockManager manager = new StockManager(); // two different ways of accessing stored products Arrays.asList(stdProducts).forEach(p -&gt; manager.addProduct(p)); IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i].increaseQuantity(i + 1)); int before = manager.numberOfProducts(); manager.addProduct(new Product(666, "Awesome thingy")); System.out.println(manager.numberOfProducts() == before + 1);</pre>	true

Réponse:

```
1 package products;
2
3 import java.util.ArrayList;
4
5 /**
6  * Manage the stock in a business. The stock is described by zero or more
7  * Products.
8  *
9  * @author (your name)
10 * @version (a version number or a date)
11 */
12 class StockManager {
13     // A list of the products.
14     private final ArrayList<Product> stock;
15
16     /**
17      * Initialise the stock manager.
18      */
19     public StockManager() {
20         stock = new ArrayList<>();
21     }
22
23     /**
24      * Add a product to the stock.
25      * @param product the product to add
26      */
27     public void addProduct(Product product) {
28         stock.add(product);
29     }
30
31     /**
32      * Increase the quantity of a product in the stock.
33      * @param productId the id of the product
34      * @param quantity the quantity to increase
35      */
36     public void increaseQuantity(int productId, int quantity) {
37         Product product = stock.get(productId);
38         product.increaseQuantity(quantity);
39     }
40
41     /**
42      * Get the number of products in the stock.
43      * @return the number of products
44      */
45     public int numberOfProducts() {
46         return stock.size();
47     }
48 }
```

Vérifier

	Test	Expected	Got	
✓	<pre>Product[] stdProducts = new Product[10]; IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i] = new Product(i, "Prod" +i)); StockManager manager = new StockManager(); // two different ways of accessing stored products Arrays.asList(stdProducts).forEach(p -&gt; manager.addProduct(p)); IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i].increaseQuantity(i + 1)); int before = manager.numberOfProducts(); manager.addProduct(new Product(666, "Awesome thingy")); System.out.println(manager.numberOfProducts() == before + 1);</pre>	true	true	✓
✓	<pre>Product[] stdProducts = new Product[10]; IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i] = new Product(i, "Prod" +i)); StockManager manager = new StockManager(); // two different ways of accessing stored products Arrays.asList(stdProducts).forEach(p -&gt; manager.addProduct(p)); IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i].increaseQuantity(i + 1)); int before = manager.numberOfProducts(); manager.addProduct(new Product(8, "Awesome thingy")); System.out.println(manager.numberOfProducts() == before);</pre>	true	true	✓



Passed all tests! ✓

Correct

Note pour cet envoi : 1,00/1,00.

## Question 7

Correct

Note de 1,00 sur

1,00

Add to **StockManager** a method that finds a product from its name rather than its ID

Product findProduct(String name)

In order to do this, you need to know that two **String** objects, **s1** and **s2**, can be tested for equality using the boolean expression **s1.equals(s2)**.

Paste just your **StockManager** code into the Answer box and Check your work.

For example:

Test	Result
<pre>Product[] stdProducts = new Product[10]; IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i] = new Product(i, "Prod" + i)); StockManager manager = new StockManager(); // two different ways of accessing stored products Arrays.asList(stdProducts).forEach(p -&gt; manager.addProduct(p)); IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i].increaseQuantity(i + 1)); System.out.println(manager.findProduct("Prod8") == stdProducts[8]);</pre>	true

Réponse:

```
1 package products;
2
3 import java.util.ArrayList;
4
5 /**
6  * Manage the stock in a business. The stock is described by zero or more
7  * Products.
8  *
9  * @author (your name)
10 * @version (a version number or a date)
11 */
12 class StockManager {
13     // A list of the products.
14     private final ArrayList<Product> stock;
15
16     /**
17      * Initialise the stock manager.
18      */
19     public StockManager() {
20         stock = new ArrayList<>();
21     }
22
23     /**
24      * Add a product to the stock.
25      *
26      * @param p the product to add
27      */
28     public void addProduct(Product p) {
29         stock.add(p);
30     }
31
32     /**
33      * Increase the quantity of a product in the stock.
34      *
35      * @param id the product ID
36      * @param quantity the quantity to increase
37      */
38     public void increaseQuantity(int id, int quantity) {
39         Product p = findProduct(id);
40         p.increaseQuantity(quantity);
41     }
42
43     /**
44      * Find a product by its ID.
45      *
46      * @param id the product ID
47      * @return the product or null if not found
48      */
49     public Product findProduct(int id) {
50         for (Product p : stock) {
51             if (p.getId() == id) {
52                 return p;
53             }
54         }
55         return null;
56     }
57
58     /**
59      * Find a product by its name.
60      *
61      * @param name the product name
62      * @return the product or null if not found
63      */
64     public Product findProduct(String name) {
65         for (Product p : stock) {
66             if (p.getName().equals(name)) {
67                 return p;
68             }
69         }
70         return null;
71     }
72 }
```

Vérifier

	Test	Expected	Got	
✓	<pre>Product[] stdProducts = new Product[10]; IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i] = new Product(i, "Prod" + i)); StockManager manager = new StockManager(); // two different ways of accessing stored products Arrays.asList(stdProducts).forEach(p -&gt; manager.addProduct(p)); IntStream.range(0, stdProducts.length).forEach(i - &gt; stdProducts[i].increaseQuantity(i + 1)); System.out.println(manager.findProduct("Prod8") == stdProducts[8]);</pre>	true	true	✓
✓	<pre>Product[] stdProducts = new Product[10];</pre>	null	null	✓

```
IntStream.range(0, stdProducts.length).forEach(i -  
> stdProducts[i] = new Product(i, "Prod" +i));  
StockManager manager = new StockManager();  
// two different ways of accessing stored products  
Arrays.asList(stdProducts).forEach(p -> manager.addProduct(p));  
IntStream.range(0, stdProducts.length).forEach(i -  
> stdProducts[i].increaseQuantity(i + 1));  
System.out.println(manager.findProduct("Whatever"));
```

Passed all tests! ✓

**Correct**

Note pour cet envoi : 1,00/1,00.