D07

Object-Oriented Programming (Java II)

AEC Programming and Internet Technologies

Practical Work – E-Shop Website

Weighting :

25 % of your final grade

Directions :

- Follow and include the same directory structure that was used in class.
- Compress your finished work into a .zip file and submit it to Dropbox.
- Submit the SQL file of you database with the rest of your work.
- If necessary, submit a README.TXT file with your project's configurations.
- ***Due on the last day of class.***

Description:

Write a web application that allows a user to register to your site and order items from your E-Shop.

Your website should be able to:

- Register on your website (a login with at least a password),
- View the list of products,
- Display products by category,
- Fill a shopping cart with selected items,
- View the contents of this shopping cart,
- Change the quantity of selected items,
- Remove a selected item from the shopping cart,
- Proceed to checkout,
- “Send the order by mail” and register this in a database.

Your shop's catalogue of products must contain at least **4 categories.** Use a Mysql database on your web server.

During an online shopping session, each user and their respective shopping cart will be stored in HttpSession objects.

Since a user may choose any number of items to their shopping cart, you will need to implement a dynamic data structure to store their selected items. Luckily Java has several dynamic data structures to choose from!

You **must** use a HashMap data structure to store client shopping carts.

A HashMap is a type of dynamic table that uses a key value pair, where the key and value are objects.

Non-exhaustive list of methods of the **HashMap** object:
(Refer to the API documentation for complete details regarding this class. It is recommended to do some testing with the HashMap class before trying to implement it into a more complex context)

a) Create a new **HashMap** object using the **HashMap()** constructor
   Ex: Hashtable cart = new Hashtable();
b) Add an object to the Hashtable with the put() method
   Ex 1 : cart.put(productCode, quantity);
   Ex 2 : cart.put(productCode, product);
   //product being an object of type Product

c) Get an object from the Hashtable with the get() method:
   Ex : String quantity = (String)cart.get(productCode);

d) Remove an object from the Hastable withe remove() method:
   Ex : cart.remove(productCode);

e) Get the list of all keys in the Hashtable by calling the keys() method which returns an Enumeration:
   Enumeration productKeys = cart.keys();

f) You can recover values from the Enumeration with the hasMoreElements() and nextElement() methods.

```
while( productKeys.hasMoreElements() ) {
    productCode = (String) productKeys.nextElement();
    quantity  = (String) cart.get(productCode);
    out.println( "Product code:" + codeProduit +
                        "Quantity:"     + quantity);
}
```

Remarks :

When you are not familiar with a data structure (for example, the **Hashtable**), do some simple tests to confirm that you understand it before you try to implement the data structure into you application.

Before writing any Java code, first write the steps you will need to follow in sentences. These sentences will become the comments of your program and you will write code following your instructions. This methodology forces you to **first**

conceptualize what your application will need to accomplish without needing to worry about the syntactic details of the language. This should make it easier for you to design your application, and you wont have to do the boring job of commenting your code afterwards.

You **cannot** use templates or bootstrap for the Practical Work assignment.

The layout of your website should be done using the css property display:flex and not display:inline-block.

Links:
http://www.w3schools.com/css/css3_flexbox.asp

https://openclassrooms.com/courses/apprenez-a-creer-votre-site-web-avec-html5-et-css3/la-mise-en-page-avec-flexbox

Any improvements to the website above and beyond will be strongly taken in consideration.

Improvement suggestions:

- Administration options for the merchant to use.
- More detailed user information.
- Using new classes or methods which were not covered during the course.
- Added client-side validation (JavaScript).
- Added AJAX Query to enhance user experience.
- Advance cookie use: last article viewed, encoded cookies, suggested articles based on articles viewed.