CS 102 – Homework 02

Due Date: October 31, 23:55

This assignment is similar to Assignment 1. You will rewrite the Bakery class, and you will implement two additional classes called Product and Branch.
Your assignment can be graded automatically therefore it is your responsibility to implement your code as exactly as possible to the details given below. These details should be enough, in case you encounter any ambiguous part, feel free to email to the instructor. Please also read the submission instructions at the end of this document very carefully. There will be no exception to these rules.

Classes to be implemented:

Product:

- A product has the following fields: *productName*, *productPrice*, *productCost* and *soldProductCount*.
- *productName* is the name of the product. It can be anything like bread, pie, cookies, cake etc.
- *productPrice* is the price of the product that is used to sell them to customers. It needs to be double.
- *productCost* is the cost of cooking the product at the bakery. This cost may include the cost of floor, oil, salt, sugar or any other ingredients needed together with the utility cost to cook it. You don't need to think about these individual costs. Just keep the total cost of the product. *productCost* is double.
- *soldProductCount* is the number of products sold. Initially the *soldProductCount* is 0. As customers visit the bakery and buy this product, its count will increase.
- Implement a constructor for this class. Only get the name, price and cost of the product from the user in this order. These values won't be changed afterwards.
- Implement a *sellProduct* function which takes the number of products sold to a customer as a function argument. Use this number of update the *soldProductCount.*
- Implement a *toString* method for printing all the information about a Product object. The *toString* method is expected to return an output as shown below.

```
Product Name: Pizza
Price: 10.0
Cost: 8.0
This product has been sold 3 times.
```

Branch:

- A branch has the following fields: *branchName*, *branchProducts*, *branchTotalProfit*.
- *branchName* is the name of the branch. It is mostly the location name where the branch is located like “Bagdat branch”, “Nişantaşı şubesi” etc.
- Each branch holds a list of Product objects called *branchProducts*. Over time new products can be inserted to this list or old ones can be removed. Implement *addProduct* and *removeProduct* methods which take a product object as an argument.
    - For *removeProduct* function if product exists within the *branchProducts*, then it is going to be removed. Otherwise nothing will happen.
    - For *addProduct* function if a product with same *productName* does not exist within the *branchProducts*, then it is going to be added to the end of the list. Otherwise, the old product will be replaced with the new one.
- *branchTotalProfit* is the total profit of the branch. Initially it is 0. As more products are sold, this profit will increase.
- Implement a *sellProducts* function for the branch object which is called with two arguments: the name and the number of the product sold. This function will use the *productName* to find the product with the same name and call its *sellProduct* function.
- Implement a *getTotalProfit* function which returns the branchTotalProfit.
- Implement a *toString* method for printing all the information about a Branch object. The *toString* method is expected to return an output as shown below.

```
Branch Name: OzU Subesi
------------------------
Product Name: Pizza
Price: 10.0
Cost: 8.0
This product has been sold 3 times.
------------------------
Product Name: Cola
Price: 4.0
Cost: 1.0
This product has been sold 2 times.
------------------------
Total profit from this branch is 12.0
```

Bakery:

- A bakery has the following fields: *bakeryName*, *bakeryBranches*, *bakeryTotalProfit*.
- *bakeryName* is the name of the bakery. It is like “Starbucks”, “Caffe Nero” etc.
- A bakery has a list of branches in different locations which is being kept at *bakeryBranches*. Initially a bakery has no branches.
- Additional new braches can be opened using the *openBranch* function which takes a branch object and adds it to the end of the *bakeryBranches*.
- A branch can be closed and removed from the list by using the *closeBranch* method which takes the name of the branch. There can be branches with the same name. In such a case close the branch that has been most recently opened. (Think about the location of this branch object within the *bakeryBranches* ArrayList)
- *bakeryTotalProfit* is the sum of the profits of the branches of the bakery.

- Implement a *getTotalProfit* function which returns the bakeryTotalProfit.
- Implement a *toString* method for printing all the information about a Bakery object. The *toString* method is expected to return an output as shown below.

```
Bakery Name : Caffe Nero
-----------------------------------
Branch Name: OzU Subesi
------------------------
Product Name: Pizza
Price: 10.0
Cost: 8.0
This product has been sold 3 times.
------------------------
Product Name: Cola
Price: 4.0
Cost: 1.0
This product has been sold 2 times.
------------------------
Total profit from this branch is 12.0
-----------------------------------
Branch Name: Levent Subesi
------------------------
Total profit from this branch is 0.0
```

Feel free to add any useful class instance to these classes and/or implement any helper functions.

After the implementation, think about if there is any situation where this proposed model will fail. In other words, do you think is there a bug in these classes?

- If not than submit the codes.
- If you think there is a bug, write that specific case and explain the problem clearly, also explain a possible solution to fix it (do not implement the solution, just clearly describe the steps in your words). Submit this as a **pdf** file together with the codes within zip.

Submission Instructions:

- Please **DO NOT** include a main method in your homework. If you have a main method in one of the class please remove it before you submit your homework. You are allowed to test your classes by writing a main for your homework, but make sure to remove them before submission. We have our own Test class to test and grade your homework.
- You will submit this homework via the LMS system. You should follow the file-naming conventions and guidelines below. In case you don't your assignment can fail the automatic grading tools and so you may lose significant points in your grade.
- You should submit your source files as a **ZIP** archive file (**NOT** RAR or other formats). The name of the file should be in format "**<USER-ID>_hw<HOMEWORK-NR>.zip**". For example, if your username is un1234, then the name of the submitted file should be "un1234_hw02.zip". Pay attention that all the letters are in lower-case. ZIP archive is supposed to contain **just the java files**, no folders are allowed by any means.
- Late submissions, missing submissions and source files that do not compile are **not** accepted and will receive 0 as the grade. After submission, download your assignment

file to a location different than where your original assignment codes are. Make sure that the downloaded files contain everything to compile/build your assignment.

- You can resubmit your homework (until the deadline) if you need to.

Final Controls Before Submission:

- Do not upload *.class files. These files can't be graded and you will get 0.
- Only use zip format. (rar, 7z, tar, vs. not allowed)
- Do not zip the whole Eclipse project.
- Do not zip src folder. Select the java files and zip them directly.
- Name your classes "exactly" the same as stated above. Capitalization of letters are important, too. (Do the same for the methods if stated in the pdf)
- Do not upload a main class.