PPA Assignment 8

This coursework is designed to test the content from Weeks 1-5 and 7-9.

Do you think there is a problem with any of the content below? Let us know immediately at programming@kcl.ac.uk.

Read through this brief carefully before starting your attempted solution and make sure you follow it as closely as possible, paying particular attention to the names required for elements of your code. Also ensure that you comment your code, as part of the assignment documentation.

The total marks available for this assignment are 22.

For this week's assessment, consider the following scenario, and then complete the tasks that follow it:

After obtaining a fortune of gold coins from their treasure hunt, our pirate has now docked at a port and is headed into town to purchase products from a shop. Once inside the shop, the customer can view the products in stock at the shop, add products to their shopping basket or remove products from their shopping basket. Finally, they can purchase all the products in their shopping basket, which transfers the money to the shop and then transfers ownership of the products to the customer.

1. Model this scenario based on the following requirements:

1. All financial transactions use `GoldCoin`s, specified in the previous coursework. (1 mark)
2. A `Product` has a `name` and a `price`, indicating how many whole gold coins are required to be exchanged in order to purchase the product. When we create an instance of this class, we should be able to provide values to these fields. Furthermore, there should be a default String representation of objects of this class using standard conventions, e.g., "ClassName" ["field1name"="field1value", "field2name=field2value", ...repeat for all fields...]. (1 mark)
3. A `Shop` has a `name`, and a number of collections of elements, specified below, for which you must select the appropriate data structure. When we create an object of this class, we should be able to specify a value for the `name` and the collections should be initialised. The collections are as follows:
    1. `products` which contains the `Product`s that are currently in stock and available for purchase. Duplicates are allowed in this collection, because the shop deals very few, but rare and high value, products. In the shop, there is the ability to `addProduct` to this collection, using a supplied `Product`. There is also the ability to `removeProduct` from the collection, which takes a supplied `Product`, removes it from the collection and returns a positive result if the product was succesfully removed. Finally, it is possible to `searchProduct`, which takes a supplied product name and returns the `Product` with that name in this collection. (2 marks)

2. `coinBox` which contains the `GoldCoin`s that are currently in the box of money stored at the shop. The shop prefers to index each coin in their coin box. A `GoldCoin` can be deposited into the box via the ability `addGoldCoin`. (2 marks)

4. A `Customer` is the entity that can browse the shop and buy products. It has a `name` and a number of collections, detailed below. You must identify the appropriate data structure to use for each collection. When we create an instance of this class, we should be able to pass a value for the name as well as initialise the following collections that are also attributes of this class:

    1. `shoppingBasket` which contains the `Product`s that the customer has selected as the ones they intend to purchase. A customer can `addToShoppingBasket`, which takes a supplied `Product` and adds it to this collection, or `removeFromShoppingBasket`, which also takes a supplied `Product` but removes it from this collection and then returns a positive result (or negative if the product was not found). Finally, a customer can `searchShoppingBasket`, which takes a supplied name of a product and retrieves the product with that name from the shopping basket. (2 marks)

    2. `ownedProducts` which contains the `Product`s that the customer owns currently. A customer can `addOwnedProduct`, which takes a supplied `Product` and adds it to this collection. (2 marks)

    3. `purse` which contains the `GoldCoin`s that the customer currently possesses. A customer can `addCoin`, which takes a supplied `GoldCoin` and adds it to this collection. (2 marks)

5. The `Shop` has one last collection: `customerTotalSpend` which associates the name of a customer with the total amount of gold coins that they have spent in the shop, over multiple transactions. The Shop has the ability to `updateTotalSpend`, which takes a supplied `Customer` and an amount of coins, then updates the total spend associated with that customer's name by adding on the supplied amount of coins. (2 marks)

6. Lastly for the `Shop`, ensure that it can be converted by default into a String representation, which presents the name of the shop and all the details of each product currently in stock. (1 mark)

7. Lastly for the `Customer`, it can `purchaseProducts`, which takes a supplied `Shop` and purchases the items in the current shopping basket. This returns either a positive or negative result to indicate whether the transaction went through successfully. This works as follows:

    1. The total cost of the shopping basket is calculated. If this is *higher* than the amount of coins in the customer's `purse`, then the purchase fails and a negative result is returned. Otherwise the purchase continues.

    2. The required number of coins are transferred, one by one, from the customer's `purse` to the shop's `coinBox`.

    3. The products in the shopping basket are transferred to the user's `ownedProducts` and then the shopping basket is emptied.

    4. The customer's total overall spend in the shop is then updated with the spend undertaken in this transaction. A positive result is then returned.

    (2 marks)

2. Create a class `ShoppingTrip`, which can be compiled and run from the command line. Use this class to do the following (in order), using the classes and methods you have created for Question 1.

1. Create the following products, then print their details to the terminal:

    - Product 1:
        - the `name` as *Diamond*
        - the `price` as *40*
    - Product 2:
        - the `name` as *Crown Jewels*
        - the `price` as *100*
    - Product 3:
        - the `name` as *Silver Locket*
        - the `price` as *60*

    (1 mark)

2. Create a shop. Give it the name 'Hidden Hideaway' and put 125 gold coins in its coin box. Add the products to from the previous question to the shop's products. Print the shop's details to the terminal, including its name, the number of gold coins in the coin box and its products. (1 mark)

3. Create a customer, give them the name 'BlackBeard' and give them 100 gold coins. Print their details to the terminal. (1 mark)

4. Create the main shopping visit using the following guidelines:

    - The first output should be an introductory message to the user (assuming the user takes the role of the customer), welcoming them to the shop.
    - Next, repeatedly present the products in the shop, the shopping basket and the total number of coins in the customer's wallet to the user. The user should be asked for their input each time the code repeats. The code should stop repeating when the user enters 'exit'.
    - If the user enters 'add product', they should then be prompted to input the name of a product from the shops product list, which should then be added to their shopping basket. This product must also be removed from the shop's products.
    - If the user enters 'remove product', they should be prompted to input the name of the product which they wish to remove from their shopping basket. The entered product should then be removed from their shopping basket and added back into the shop's products.
    - If the user enters 'purchase', all the products in the shopping basket must be purchased.

    (2 marks)

Once completed, submit your assignment using the link marked `Assignment 8: Nexus Submission Link' on KEATS.

You must complete the plagiarism and collusion training before submitting this assignment.

You must also submit complete documentation of your solution. You will find a sample piece of documentation in the Support section on KEATS marked `Sample Assignment Documentation'. Submit your documentation using the link marked `Assignment 8: Documentation Submission' on KEATS.

Students who do not submit documentation along with their code, or vice-versa, will receive a mark of zero.

Any submitted code or documentation that is found to be unduly similar to the code or documentation submitted by any other student(s), will result in a penalty for those involved.

Provisional mark for your code will be released on KEATS within one week of submission. Final assignment grades will be submitted to the exam board at the end of the semester, and will take into consideration the quality of your documentation and the quality of the comments written into your code directly.

For all other queries, see the Support section on KEATS, specifically the document marked `Introduction'.