PPA Assignment 8

Wonjoon Seol, Computer Science with Intelligent Systems, K1631098

December 5, 2016

1 Introduction

In this assignment we simulate transaction activities between a shop and a customer. The shop has number of products which a given customer can search, put or remove requested item from his shopping basket and also able to purchase every item in the the basket. This transfer necessary gold coins from the customer's purse to the coin box in the shop. After each purchase the shop updates the customer's spending record. I need to demonstrate my ability to select appropriate data types including ArrayList and TreeMap with multiple object types and able to make use of them. (Topics from week 1,. . . ,5 and 7,. . . ,9).

2 Pseudocode

2.1 Class GoldCoin

Pseudocode 1: This class models a gold coin.

```
1  Initialise private String coinNumber
2  Initialise private static integer totalCoin
3  Define GoldCoin
4      Set coinNumber to be totalCoin + 1
5  Define getCoinNumber
6      Return coinNumber
```

2.2 Class Product

Pseudocode 2: This class represents characteristics of a product.

```
1  Initialise private String name
2  Initialise private Int price
3  Define Product
4      Set name
5      Set price
6  Define toString
7      Return "Product" + "[name=" + name + ", price="+ price + "]"
8  Define getName
9      Return name
10 Define getPrice
11     Return price
```

2.3 Class Customer

Pseudocode 3: This class represents the characteristics of a customer.

```
1  Initialise private String name
2  Initialise private ArrayList<Product> shoppingBasket
3  Initialise private ArrayList<Product> ownedProducts
4  Initialise private ArrayList<GoldCoin> purse
5  Define Customer
6      Set name
7      Intialise ArrayList shoppingBasket type Product
8      Intialise ArrayList ownedProducts type Product
9      Intialise ArrayList purse type GoldCoin
10 Define toString
11     Return name and amount of the gold coins in the purse
12 Define addToShoppingBasket
13     Add a Product to the ArrayList shoppingList
14 Define removeFromShoppingBasket
15     for i = 0 to ArrayList size of shoppingBasket - 1 do
16         if product name equals to name of item in shoppingBasket then
17             Remove element at index i from shoppingBasket;
18             Return true;
19         end
20     end
21     Return false;
```

2.4 Class Shop

Pseudocode 4: This class models a shop where customers can buy products.

```
1  Initialise private String name
2  Initialise private ArrayList<Product> products
3  Initialise private ArrayList<GoldCoin> coinBox
4  Initialise TreeMap<String, Integer> customerTotalSpend
5  Define Pirate
6      Set name Initialise ArrayList coinBox type GoldCoin
7      Initialise ArrayList products type Product
8      Initialise TreeMap customerTotalSpend key type String, value type int
9  Define toString
10     Return name and products
11 Define addProduct
12     Add a Product to ArrayList products
13 Define getName
14     return name
15 Define getCoinNumber
16     return number of elements in ArrayList coinBox
17 Define removeProduct
18     for int i = 0 to ArrayList size of products - 1 do
19         if product name equals to name of item in products then
20             Remove element at index i from products;
21             Return true;
22         end
23     end
24     Return false
25 Define searchProduct
26     Initialise boolean isCoinTaken to be false
27     if Supplied String value equals to name of item in products then
28         Return elements at index i of ArrayList products;
29     end
30     Return null;
31 Define addGoldCoin
32     Add a GoldCoin to ArrayList coinBox
33 Define updateTotalSpend
34     if customerTotalSpend contains supplied customer's name as a key then
35         Put customer's name as a key and add supplied coin amount to the current stored
             value
36     else
37         Put customer's name as a key and supplied coin amount as a value
38     end
```

2.5 Class ShoppingTrip (Continued next page)

Pseudocode 5: This class is going to drive our program, continued next page.

```
1  Initialise Scanner in
2  Initialise Product product1 with "Diamond", 40
3  Print product1
4  Initialise Product product2 with "CrownJewels", 100
5  Print product2
6  Initialise Product product3 with "SilverLocket", 60
7  Print product3
8  Initialise Shop shop with "HiddenHideaway"
9  for i = 0 to 124 do
10 |  Initialise GoldCoin and add it to coinBox in shop
11 end
12 Add Product product1 to ArrayList products in shop
13 Add Product product2 to ArrayList products in shop
14 Add Product product3 to ArrayList products in shop
15 Print shop and number of coins in the shop
16 Initialise Customer customer with "Blackbeard"
17 for i = 0 to 99 do
18 |  Initialise GoldCoin and add it to purse in customer
19 end
20 Print customer
21 Print name of shop with welcome statements
22 Initialise String userInput
```

2.6 Class ShoppingTrip (2)

Pseudocode 6: This class is going to drive our program.

```
1  do
2      Print shop and customer
3      Print ask user to input value
4      Set userInput to read next input value
5      if userInput is not equal to "exit" then
6          if userInput is equal to "add product" then
7              Print ask user to input value
8              Set userInput to read next input value
9              Initialise Product item to be product returned after searching the store with
                userInput
10             if item is not null AND removing item from shop returns true then
11                 Add item to ArrayList shoppingBasket in customer
12             else
13                 Print the user input value could not be found
14             end
15         else if userInput is equal to "remove product" then
16             Print ask user to input value
17             Set userInput to read next input value
18             Initialise Product item to be product returned after searching the customer
                shoppingBasket with userInput
19             if item is not null AND removing item from shoppingBasket in customer
                returns true then
20                 Add item to ArrayList products in shop
21             else
22                 Print the user input value could not be found
23             end
24         Print the user input value could not be found
25         else if userInput is equal to "purchase" then
26             if customer successfully purchased products in the shoppingBasket then
27                 Print the user the products are bought
28             else
29                 Print the user did not have enough coins
30             end
31         Print the user input command could not be found
32     end
33 while userInput is not equal to "exit"
34 Close scanner in
```

3 Class Diagram

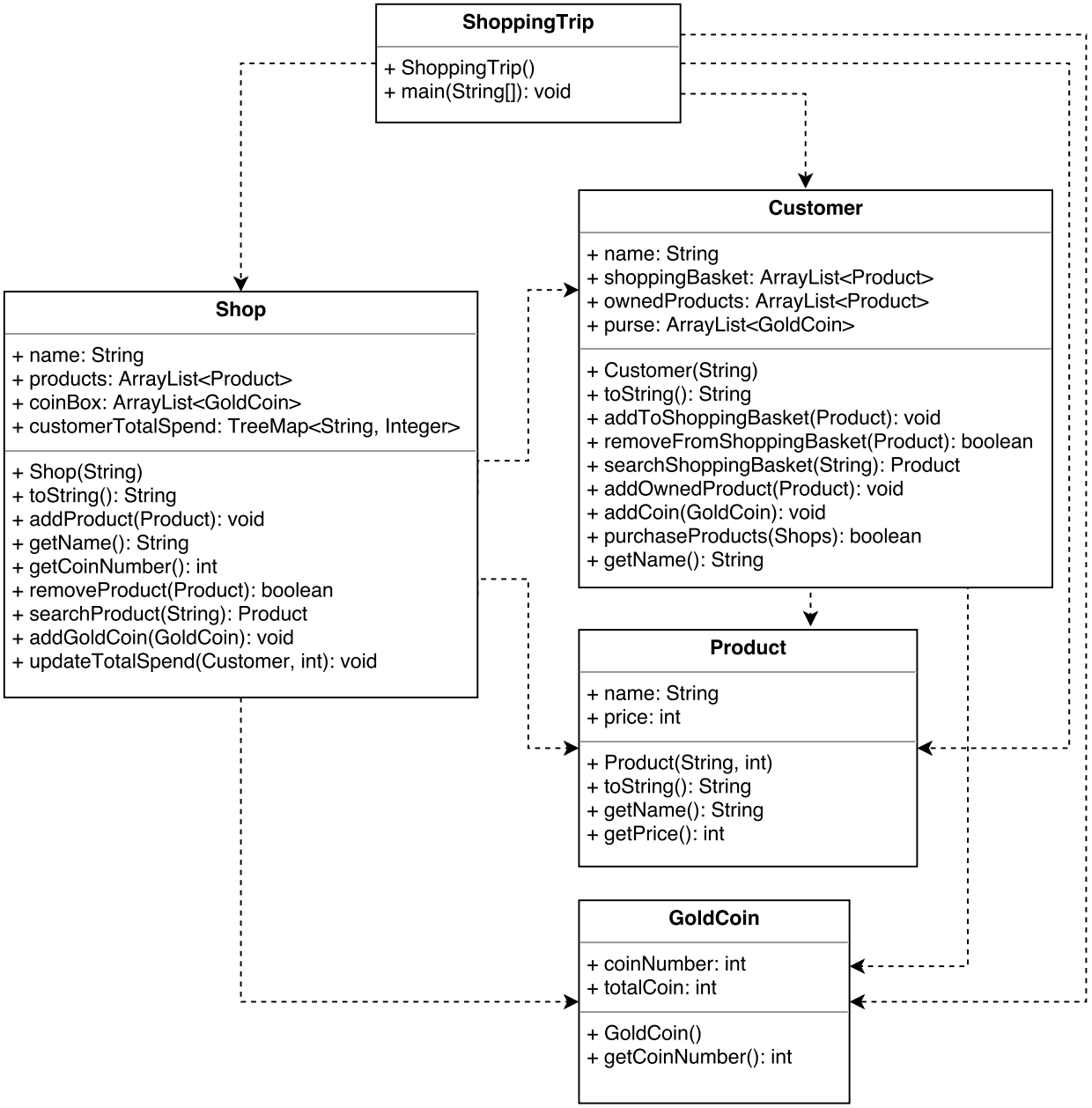

Figure 1: Shop simulation Class Diagram.

4 Description

1. Class GoldCoin
   This class is the same as the previous assignment.

2. Class Product
   The class stores product's name and its price. The *toString* method returns field information similar to JSON format.

3. Class Customer
   The class Customer stores 3 ArrayList and the name of the customer. Each ArrayList represents shopping basket, ownedProducts and purse. These are initialised within the constructor. The toString method returns *name*, size of the purse, and ArrayList *shoppingBasket*. When *shoppingBasket* is returned, it calls the default *toString* method inherited from Class AbstractCollection. This then returns the list of each product's *toString* method written in JSON style. The entire String output is enclosed in square brackets.

   The method *removeFromShoppingBasket* returns a boolean value. This will be used in the condition within the driver class. Instead of two return statements single return method could have been used instead, if a local boolean field was declared. My personal coding preference is to minimise the use of *return* or *break* statements, especially when they are used to control the flow of the code. In this example, however, I have used two returns to end the given method prematurely such that searching entire ArrayList can be avoided. Similar job can be done with a local boolean field and single return statement by adding the boolean field as an extra condition within the loop.

   The *purchaseProducts* method can be made more readable if private methods were created to contain each blocks of the code. When coins in the *purse* or products in *shoppingBaskets* are removed I supplied index 0 instead of i. This is because every time *removed* method is used, the ArrayList shifts every element to the left. So removing the first elements n times, where n = size of the ArrayList is sufficient to remove every elements in the ArrayList. If *size* method was used within the loop condition, the loop will only remove half of every elements because i is incremented by one while the size is decremented by one each loop. Therefore the local variable was necessary to snapshot the initial size of the ArrayList.

4. Class Shop
   Most of the methods were similar to the Class Customer, so please refer to the above paragraph. The method *updateTotalSpend* checks one thing : whether there exists a supplied customer data already within the TreeMap. If it exists then it updates its value by adding number of coins to the already stored value. If the key does not exist, then it adds new entry with the supplied coin number as its value.

5. Class ShoppingTrip
   This is our driver class, the brief was not clear what to do when the customer typed wrong input for the products name. There were two choice: 1. ask the user to type the product name again 2. return to the initial loop where the customer will type command again. I chose the second because the customer might have changed his mind but the first option would still force the user to either add or remove the products. So I believe the second option is better here.