Anthony Dushaj

Professor Arias

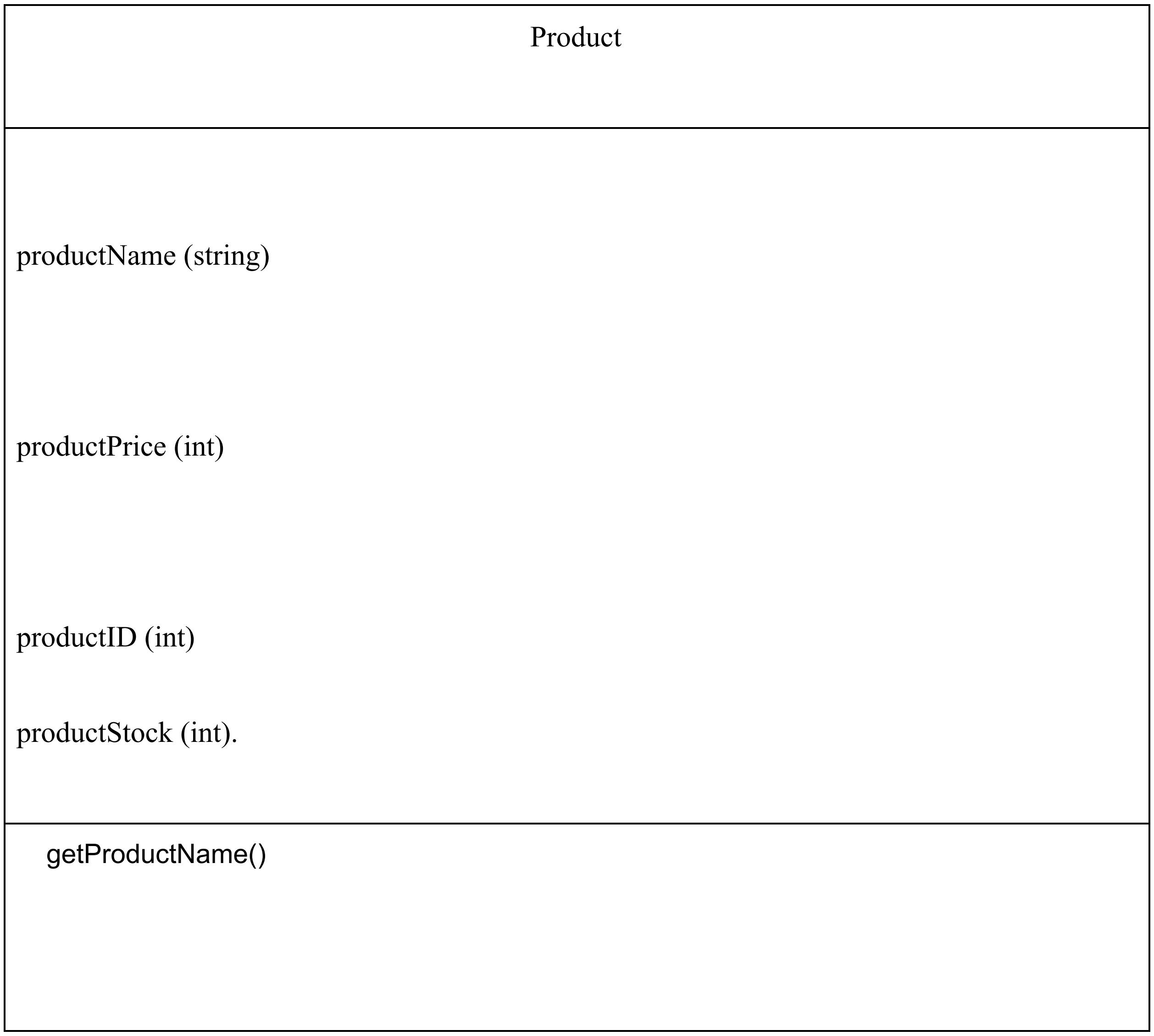
Software Development I

26 April 2019

Project 2

My Java project has the primary goal of generating a virtual online shopping like experience, this will be done with the main goal of having a shopping cart, which can imitate or mimic the true shopping experience, through a java programming project. My java project creates four different classes that will ultimately allow the user to have similar to an online shopping experience. The four different classes will consist of the Product Class, Products Class, Shopping Cart, and finally the User Interface Class. The main purpose of my project is to allow the user to feel as if they were shopping, and to facilitate their shopping experience in a plethora of ways through methods, functions, and algorithms.

The product class is the class that creates an individual product and assigns the attributes of private String productName, private double product Price, private int productID, private int productStock. Moreover the product class will consist of four parameters, and there will be getter and setter methods for each variable. In addition to the properties, the Product Class will also have to Override the hashCode() and equals() methods. I had to include this part of the program for a couple of different reasons. As I was in the process of completing my project, I realized how important it is to override these methods for Object Comparison. Without these overrides, one problem in my project was that I was not able to remove an item (product) from the array of products, I soon realized that there was a problem of object equality, after overriding the hashCode() method, and the equals() , the problem was quickly solved. To wrap up the first class, the UML diagram is shown below.



getProductPrice()

getProductID()

getProductStock()

setProductName()

setProductPrice()

setProductID()

setProductStock()

hashCode()

equals(object o)

The next class is the Products Class, this is the class that will provide with store products. Moreover, this class will initialize initial values in the method called initialize-Store-Items(). Three different arrays, consisting of the product name (String), product Price (double), and lastly product Stock (Integer). A for loop that is in the Products class, will combine the properties of a product, and then create a new instance of product and add it to a list of products. Product ID was not included as an array, because the for loop initializes the value of it to 1 and then accumulates accordingly. For example, the first product will be number one, and the second will be number two, and this product ID is what later allows the user to perform many desired tasks that include, adding or removing a product by its specific ID number. To summarize, the main goal of this class is to create and store a list of objects from the previous class (product), so that the user can add what they want to the list, or in contrary, remove what they want from the list. I have provided the Product class inside the <> braces as the new ArrayList(), this will signal the compiler that the list is of type Product and it can only contain an item which belongs to the specific type Product. This demonstrates a direct relationship between the first two classes: product and products.

Products
Products (objects)
Initialize-Store-Items()

getProducts()

The next class is called the Shopping Cart class. This class is similar to a virtual shopping cart, and this class will store items temporarily. The Shopping Cart Class carries all of the methods that an ideal shopping cart would carry. For example, the methods include: adding an item by product ID, removing a item by product ID, checking the amount of items in ones cart, and checking the total price of all the items in the users shopping cart. Using an arraylist was done to keep the program more simple and subtle, as the arraylists carry methods like add(), remove(), size(), therefore these built in methods facilitate unnecessary work for both the programmer, and the user. Similarly to the last class, this class will create an arraylist of the type product as well. This list called Cart Items will hold the items that the user selects to add, and in contrary can also remove the product as well if the user pleases. In conclusion, the shopping cart carries some of the most important methods of the whole entire Java Virtual Shopping Cart project.

Shopping Cart

List<Product> cartItems = new ArrayList<Product>()

getProductByProductID(int productID)
 	
addToCart(Product product)
 	
removeProductByProductID(int productID)
 	
removeFromCart(Product product)
 	
totalPrice()
 	
getTotalItemCount()
 	
getMostExpensiveItem()
 	
printCartItems()

Lastly, the last class in my Java project will be the User Interface class in which the user will do most of the work at this point. The user will be inputting their values onto the console. The User Interface class will call those methods from previous classes. The first menu will print three different options for the user to enter. The first option is to display store products (user will decide what they want to add or not), the second option is to display the cart (view what is currently in the cart), and the last option will be to exit. This allows the user to add items, then check to see their cart to make sure they have everything they need, and make sure they don't have anything they do not need. The get Total Cost() method is one example of addressing a problem that occurs in real life. Many times people fill up their shopping carts, and sometimes overspend or underspend, therefore this method can allow the user to access the total price of all the products that the user had added in their own shopping cart without having to do math or add up each thing individually. The method initializes the total to 0 at first, then it has a for loop that will check each product in the shopping cart, and add up their price (getProductPrice) and returns it to the user if necessary. There were be a single variable that is responsible for all of this, as sometimes adding many different variables can sometimes make things confusing and more tedious. The get user input will throw a NumberFormatException, and after the user enters their number, it will enter through the case, and when the user is done they can choose to keep shopping, or simply exit by typing in Zero. The User Interface class is what allows for the main interaction between the user and the application. UML:

User Interface

displayScreen()
 	
itemsMenu()
 	
Menu()
 	
displayStoreProducts()
 	
Choice1Inner()
AddProductToCart()
RemoveProductFromCart()
showCart()

printItemCount()
 	
getUserInput()

The System should be used by first viewing the products in the store that are available, the user will then enter in the product ID to add that product / item respectively. After adding products that the user wants, they can check their cart to make sure they have everything they leave before shopping. Once the user has completely finished looking through, and adding products they can then start to check out and begin to finish the shopping experience.

This Java Project accomplished many things that would help people in the real world today, and also helped me learn more about polymorphism, and ultimately helped me learn that the importance of many different vital java programming techniques.