# Core Java - Cafe Projects

**Project 1**

Create a new package in your exercise project named com.perscholas.cafe. Create a class named Product inside this package with four private attributes (name, price, description, quantity). The class should also include a no-arg constructor and a constructor which takes three arguments for name, price and description. Next, create getters and setters and include a method named calculateProductTotal( ) which calculates the product subtotal for the order. Create a driver class (i.e., class that includes a main method) named CafeApp which creates three Product instances named "coffee", "espresso" and "cappuccino". Assign descriptions and prices to each product instance. Use the Scanner class to prompt the user for the quantity of each product and then print the name, description and product subtotal for each after each prompt. Print the sales subtotal, sales tax and sales total before exiting the application.

**Project 2**

**Inheritance & Abstract Classes**

Make the Product class in the previous cafe exercise an abstract class and make the method calculateProductSubtotal in the Product class an abstract method. Now create three new classes (Coffee, Espresso, Cappuccino) which each extend the Product class. You will get an error in the new classes once you extend them due to the missing implementation of the calculateProductSubtotal( ) method. If you hover over the class name with this error you will be given the option to add the method. Go ahead and click the link to add the method (alternatively, you can simply write the method yourself). You should end up with an empty method with the @Override annotation. Do this for each class.

- In the Coffee class, add the boolean attributes, sugar and milk. Create a no-arg constructor which calls the super constructor and assigns this.milk and this.sugar to false. Create a constructor which accepts five attributes (three from Product and two from Coffee) and passes three arguments to the super constructor and assign this.milk and this.sugar to the arguments passed to the constructor. Create getters and setters for the two added attributes. Implement the calculateProductSubtotal method.
- In the Espresso class, add the boolean attributes extraShot and macchiato and complete the process as with the Coffee class. In the calculateProductSubtotal method, add $2.00 to each item with an extra shot and add $1.00 to each item made in macchiato style.
- In the Cappuccino class, add the boolean attributes peppermint and whippedCream and complete the rest of the class as with the Coffee and Espresso classes. In the calculateProductSubtotal method, add $2.00 to each item with peppermint and add $1.00 to each item with whipped cream.
- Rewrite the CafeApp class to utilize the Scanner class to accept orders for each product and, as before, print the name, description, quantity, and product subtotal for each item along with the sales subtotal, sales tax, and sales total.

**Project 3**

We will continue the Cafe concept for this exercise.

- Create two new abstract methods in the Product class: addOptions() and printOptions(); Implement these methods in each of the subclasses.
- Write a program that prompts the user to select products from the cafe and select options attached to those products (e.g, "Would you like sugar with the coffee?"), but this time include store and shopping cart classes and include classes for each item (you may use the classes created in previous exercises and add these products to lists in the store and shopping cart classes). Options may or may not incur additional cost, but you should have at least one product that has options which do incur additional cost. Have the user select items from the store and specify any options. Add the items to the shopping cart. Print out a list of each item with price, quantity and item subtotal along with the purchase subtotal, sales tax and purchase total after the user is done selecting items (i.e., there should be a choice on the menu to “check out” at which time the purchase totals are displayed and the application quits.

Optional: Create and implement a custom exception

Sample Output:

```
Please select from the following menu:
1: Coffee
2: Cappuccino
3: Espresso
4: Check Out
4

Proceed to checkout.

Item:   Espresso        Price: 3.99     Qty: 1   Subtotal: 4.99
        Extra Shot: No  Machiatto: Yes (Add $1)
Item:   Cappuccino      Price: 4.99     Qty: 1   Subtotal: 7.99
        Peppermint: Yes (Add $2)        Whipped Cream: Yes (Add $1)
Item:   Coffee  Price: 3.49     Qty: 1   Subtotal: 3.49
        Sugar: Yes      Milk: Yes
Purchase Subtotal: 16.47
Sales Tax: 1.03
Purchase Total: 17.50
```

Resources

Output Formatting:

**https://docs.oracle.com/javase/tutorial/essential/io/formatting.html**
**(https://docs.oracle.com/javase/tutorial/essential/io/formatting.html)**

**https://docs.oracle.com/javase/tutorial/java/data/numberformat.html**
**(https://docs.oracle.com/javase/tutorial/java/data/numberformat.html)**