Problem 1:

ShakeShack should create milkshakes of the following types, Chocolate_Shake, Coffee_Shake, Strawberry_Shake, Vanilla_Shake, and Zero_Shake.

As a customer of ShakeShack you don't know any of this unnecessary detail, you just order your preferred shake using the produceShake function of ShakeShack.

All these shakes have milk, and all shakes other than Zero_Shake have sugar. The additional ingredients are: chocolate_syrup and chocolate icecream for Chocolate_Shake, coffee and jello for Coffee_Shake, strawberry_syrup and strawberry_icecream for strawberry shake, vanilla_flavoring and jello for vanilla shake. Zero_shake has sweetener instead of sugar, and vanilla_flavoring with sugar_free_jello.

All of the shakes can be made lactose-free, by substituting almond milk instead of milk. However, making the shake lactose-free adds Tk 60 to the base price. Moreover, all of these shakes can have candy added on top (adding Tk 50 to the base price), or cookie added on top (adding Tk 40 to the base price). Please ensure that your object creation handles these cases.

The base prices of the shakes are:

Chocolate_Shake: Tk 230

Coffee_Shake: Tk 230

Strawberry_Shake: Tk 200

Vanilla_Shake: Tk 190

Zero_Shake: Tk 240

After the order placing is done, the code should output the names of the shakes ordered, the base and added ingredients, and the individual prices (in case of a price increase, mention why the pricing increased). You have to perform this using a print function. Also, output the total price of the order.

Please note that there can be multiple orders during a run. However, the orders are sequential. If 'O' is pressed, an order opens, and if 'E' is pressed, the order closes. If one tries to open another order while a current order is ongoing, you should show an error message, and ask if he wants to include something else in the previous order. Once placed, an order cannot be changed. Moreover, ensure that there is at least one item in an order before closing it. Think of a few more boundary cases and include those in your code.

Tasks:

Identify the design pattern that can best capture the scenario above (only the shakes are to be implemented following a design pattern, not the ordering mechanism).
Implement the scenario in Java.
Marks will depend on how the code is implemented. The code can run as instructed, and yet fail to use the appropriate pattern. No marks will be awarded in that case.

Problem 2:

You have been given the task of designing a system to display and print shapes from a database. The resolution required to display and print the shapes depends on the computer that the system is currently running on, i.e., the CPU speed and the amount of memory available. Your system must be careful about how much demand it is placing on the computer.

The shapes are: Circle, Square, Rectangle, and Triangle. Also, write functions for printing the surface area and perimeter of these shapes.

There can be 3 types of computers, ComputerA, ComputerB, and ComputerC. All of these have their own CPU and MMU. The resolutions of the images produced by the computers are 200x200, 350x250, and 550x430, respectively.

You have to input the Computer name, and the shape that you are trying to display, along with the parameters for the shape (you don't have to take any input from the screen, designing a few test cases should be okay). The code should initialize the proper CPU and MMU (as customary according to the design pattern), and finally print the name of the shape, the resolution, and the surface area and perimeter (obviously, to calculate the surface area and perimeter, you will need the parameters for the shapes).

While coding, think of the necessary boundary conditions according to the unique situation of your code.

Tasks:

Identify the design pattern that can best capture the scenario above (only the computer specs are to be considered while identifying the design pattern, not the shapes).

Implement the scenario in Java.

Marks will depend on how the code is implemented. The code can run as instructed, and yet fail to use the appropriate pattern. No marks will be awarded in that case.

Submission instructions:

Enclose all the files inside a single folder named after your 7 digit student id. For example, 1605001 will enclose all the files in a folder named 1605001. Then, zip the folder (the name of the zipped file will also be the same as the folder, i.e., 1605001.zip).

Submit the zipped file by 14 June 2019 (11 PM).