DesignPattern-SER5155-Fall22-SauravSarkar

Installation
1. Clone repository from the given github [Repository link
https://github.com/sauravsrkr/DesignPattern-SER5155-Fall22-SauravSarkar] or from zip file.
2. Open the project in IntelliJ IDE.
3. Click on SS in the left side menu and click on src and search for Main class.
4. Select the Main class from the project menu on the left side.
5. Select Build and click on Build project in the top navigation menu.
6. Select run and click on run "Main.java" file.
7. The output will be displayed in the terminal below.

Command Line Compilation and Execution
1. Clone repository from the given github [Repository link
https://github.com/sauravsrkr/DesignPattern-SER5155-Fall22-SauravSarkar] or extract zip file.
2. Change directory to the path of project "cd
DesignPattern-SER5155-Fall22-SauravSarkar/src/".
3. Generate the Class Files of all java files using the command "javac Main.java".
4. Execute the program using the command "java Main".

Patterns Used
Facade Pattern -
I have used facade pattern to access modules through interface layer. In my system Facade
acts as a central interface that enables Login, Reminder, ProductMenu, TradingMenu and
Offering Menu.

Bridge Pattern -
I have used bridge pattern to decouple an abstraction from the implementation. For example,
ProductMenu is dynamically assigned to type of meat and produce menu.

Factory Method -
I have used factory method to create dynamic class to be instantiated. For example, When
deciding the productMenu factory helps to decide MeatProductMenu and ProduceProductMenu.

Iterator Pattern -
I have used iterator pattern to list all products available for the current user in their
corresponding menu. This can also help in multiple variations in traversal of an aggregate.

Visitor Pattern -
I have used visitor pattern to encapsulate the view offerings and get trading logic without the
need to change the classes of the elements. This has helped me to decouple the classess for
the data structure and algorithms used.