Candy Recipe Creation System

Submitted by Syed Aziz Ullah Hussaini

Candy Recipe Creation System

As the final project I am proceeding with the idea given in spread sheet i.e., “Candy Recipe Creation System” in which I am using below four design patterns

1. Factory Design Pattern
2. Iterator Design Pattern
3. State Design Pattern
4. Builder Design Pattern

The procedure/story that I have chosen is straight forward i.e., When a customer wants to walk into the Candy Store it would have to go through the state pattern i.e., either it is open or closed in the same fashion when he/she places the order the other states will follow. While placing the order, menu will be displayed using the Iterator pattern and once the order is placed then for delivering the same the internal system which Candy Store follows is being used as the Factory Design pattern for getting the various candies from Candy Stores in turn displaying the recipe according to the candy chosen by the customer and so on.

I am still working on the implementation of above logic and might trim some things if the situation demands as per the execution of the program to make it stable and executable over the IDE of other machines as well. To brief about the test cases, as you said I have planned to include 2-unit test cases minimum by trying to execute any of the above 2 patterns. State pattern will be surely tested and will try to utilize the opportunity to get unit test cases for all the Design Patterns that am using if possible.

I am referring the Headfirst Design Patterns textbook if am stuck at any point however am not getting any great clues so that I can easily decouple the patterns and get it done. Making it work is getting a bit challenging, but I know that I can do it for sure if I focus on getting it in a real time manner.

Below are the brief highlights about each pattern that is being used:

1. Factory Design Pattern:

   I am using two different Candy Stores which has its own Candy Flavours and different ingredients which are being displayed using the Interfaces with their implementations and a couple of Classes like CandyStore that has 2 child classes also for the Flavours

2. Iterator Pattern:

   This pattern is used for displaying the Menu in each store along with their rates, once the Customer is in the store I mean if the State is open then the Menu will be displayed prompting the customer to choose

2. State Pattern:

   This will transition the customer from one state to another like if the state is open then customer is allowed to see menu and order the candy, likewise the customer will be moved into the winnerState() if he/she is the lucky customer and two candies will be given for the winner and if the candies are finished the Store will be moved to closedState()

3. Builder Pattern:

   This pattern is being used in the same way as we used it earlier i.e., by asking the customer to fille the form if he wins and there'll be some optional fields that if customer doesn't fills then Storekeeper will take it as default values and finally the Candy gets delivered to the customer

The unit test cases will be tested automatically as the program executes i.e., by employing the necessary function calls or object creations in main()