Candy Recipe Creation System

Submitted by Syed Aziz Ullah Hussaini

Outcome from the course:

This course was very fascinating I mean the way design patterns helps to resolve any problem or a situation is amazing!

I learned a lot of things during my journey of this course like using the eclipse IDE in an efficient manner, implementing the Junit test cases which was the very first time for me as I had never implemented a Junit test case before and obviously the kind of design patterns that I can employ in more realistic problems in future even though I used interfaces, inheritance before but after learning and doing the assignments I came to know how we can get the most out of the patterns and make our problem simpler.

The problem statement/Final project that I have worked on had various things that racked my brain to get the work done most importantly the decoupling of the different design patterns in one program was bit tricky for me which in turn made me to look out for Headfirst Design patterns to get it in place.

Initializing the object by providing the number candies to getting the candy the steps and the logics involved were having to be very concise so that whatever the situation or the condition is the program must not fail which made me think of the CandyStore in a realistic manner and align my approach towards the conclusion. There were many instances that made me rethink of the idea which I was going forward with and then put some thought on the idea again for making me continue over this application.

Using suitable variable names so that it can be identified easily anywhere in class was important aspect to keep in mind as the variable names plays a vital role in any programming language, I also had my focus to keep the naming convention clear and concise moreover debugging part while implementing the program gave a real hard time to me initially, when I was not getting the expected output then tracing the program like currently which state it is in and which function is being called helped me a lot to learn the debug concepts too

I would like to thank you for providing information and inputs wherever needed and making things achievable easily.

As the resource for helping out if am stuck anywhere then Headfirst Design Pattern was a great reference which never made me look into any other thing as it is everything for patterns in itself, the way examples are described and explained is exemplary which made my work lighter.

Background of the project:

This is the Final project titled "Candy Store Recipe Creation" which is build using four different design patterns given in the Headfirst Design Patterns textbook.

The project works as per below information provided

The candy store is firstly initialized with desired number of candies, here we have taken 5. As soon as the store is initialized with the candies it comes to the open state and checks the count of candies and if those are greater than 0 then the state changes to "open state" and waits for the order.

Once the order is placed then it'll go to the order placed state and checks for the lucky customer using random() function, if the customer is lucky then it'll go to the winner state and delivers two candies to the customer and moves to the delivered state and checks for the number of candies and if those are greater than 0 then it'll move to the open state again and if the candies are not greater than 0 then it'll move to the closed state.
In each state it checks for the number of candies, if at any moment it is 0 then the candy store will come to closed state.

The execution of the program starts from the main class which is "CandyStoreStateTest.java" and below are the classes which plays the most important role in the program

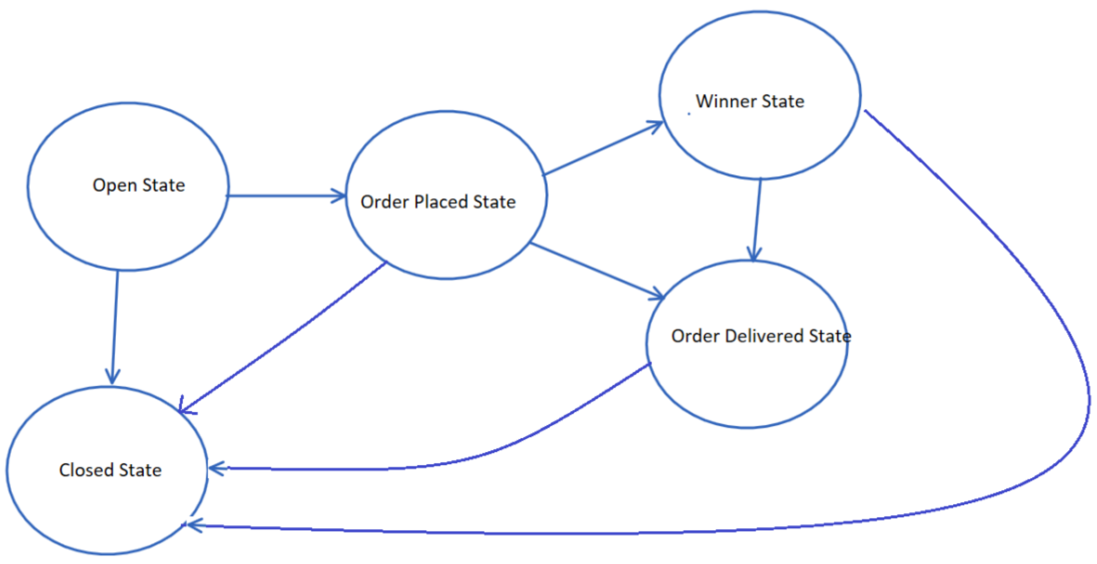
```
Candy.java
CandyStore.java
CandyStoreStateTest.java
ChicagoCandyStore.java
NewHeavenCandyStore.java
```

The patterns used to create the above project are listed below

1. ***State Pattern:*** To depict the states of the candy store, we have 5 states and accordingly the transitions take place.

```
State Pattern
---------------
State.java
OpenState.java
OrderPlacedState.java
WinnerState.java
OrderDeliveredState.java
ClosedState.java
```

There are various functions in the state pattern tat helps it achieves the required states orderNow(), cancelOrder(), placeOrder(), delivered();



State Transitions

2. ***Builder Pattern:*** This is used to get the information of the user if the user wins and displays the details of winner. There are two most important classes of the Builder pattern as given below which employs the setter methods along with getCandy() to achieve the task

```
Builder Pattern
---------------
Winner.java
WinnerStateBuilder.java
```

3. ***Factory Pattern:*** Factory pattern is used to get the different kinds of candies available in the store/Factory of the concerned city. We have taken two major cities as the candy store places namely Chicago, New Haven. There are different flavours of the candies available to in different candy factories as listed below. The candy stores when called will fetch the appropriate flavours as per the factories

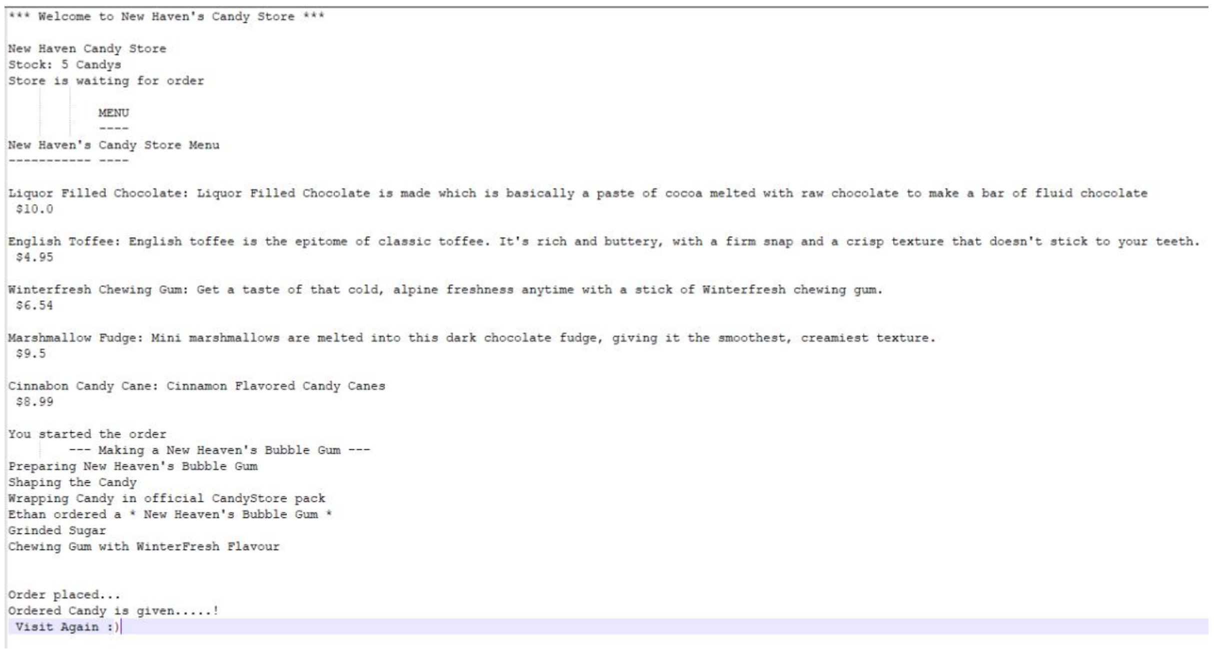
```
Factory Pattern
---------------
CandyIngredientFactory.java
ChicagoIngredientFactory.java
NewHavenIngredientFactory.java
```

The main call will be done for this pattern to be tested is through the CandyStoreStateTest.java

4. ***Iterator Pattern:*** This pattern is used to display the menu of each candy store {Only the customised Iterator class is used}
It displays the menu as per the Candy stores like below, for example it displays the below menu for the Chicago Candy Store

```
                                MENU
                        ----
Chicago’s CandyStore Menu
----------- ----

Milk Chocolate: Milk chocolate is a solid chocolate confectionery containing cocoa, sugar and milk.
 $4.45

Bubble Mint Sugar Free Gum: Bubblemint Sugarfree Chewing Gum is made with a delicious bubblemint flavo
r for a clean and fresh feeling
 $2.1

Veg Marshmallow: Soft and mildly flavoured cubes of marshmallow
100% Vegan Confectionery
 $7.2

Almond Butter Crunch: The sweet buttery crunch of this delicious Buttercrunch Toffee is paired with ch
ocolate and toasted almonds for a winning combination.
 $6.59

Chocolate Mint Candy Cane: peppermint flavor as the originals, but they also sport a festive, chocolat
e stripe.
 $3.25
```

Sample Output post running this program will be as below,



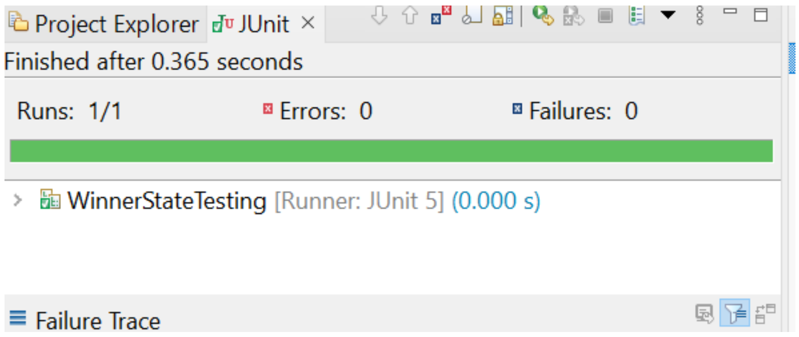
```
*** Welcome to New Haven's Candy Store ***

New Haven Candy Store
Stock: 5 Candys
Store is waiting for order

                MENU
                ----
New Haven's Candy Store Menu
----------- ----

Liquor Filled Chocolate: Liquor Filled Chocolate is made which is basically a paste of cocoa melted with raw chocolate to make a bar of fluid chocolate
 $10.0

English Toffee: English toffee is the epitome of classic toffee. It's rich and buttery, with a firm snap and a crisp texture that doesn't stick to your teeth.
 $4.95

Winterfresh Chewing Gum: Get a taste of that cold, alpine freshness anytime with a stick of Winterfresh chewing gum.
 $6.54

Marshmallow Fudge: Mini marshmallows are melted into this dark chocolate fudge, giving it the smoothest, creamiest texture.
 $9.5

Cinnabon Candy Cane: Cinnamon Flavored Candy Canes
 $8.99

You started the order
        --- Making a New Heaven's Bubble Gum ---
Preparing New Heaven's Bubble Gum
Shaping the Candy
Wrapping Candy in official CandyStore pack
Ethan ordered a * New Heaven's Bubble Gum *
Grinded Sugar
Chewing Gum with WinterFresh Flavour


Order placed...
Ordered Candy is given.....!
 Visit Again :)
```

Junit Testing:

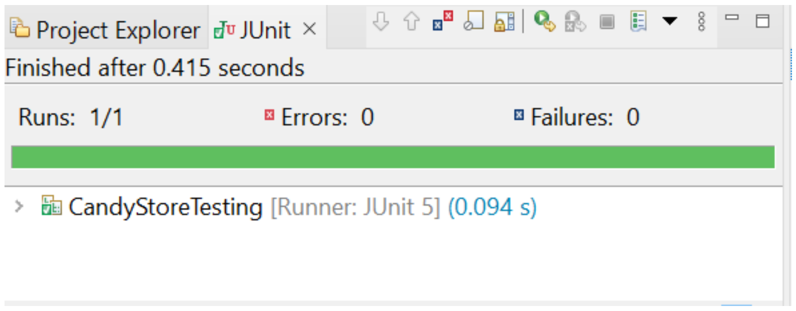
This was a very new thing for me to learn and implement, there have been three junit classes implemented each for one design pattern namely

WinnerStateTesting.java for Builder Pattern
CandyIngredientFactoryTest.java for Factory pattern testing
CandyStoreTesting.java for State pattern testing

As I was getting the output in the form of multiple lines and having been used function which return void most of the times it was not possible to test all the scenarios, but I have done my best to test the important functionalities for the same



```
Finished after 0.365 seconds
Runs:  1/1          Errors:  0          Failures:  0
> WinnerStateTesting [Runner: JUnit 5] (0.000 s)
Failure Trace
```



```
Finished after 0.415 seconds
Runs:  1/1          Errors:  0          Failures:  0
> CandyStoreTesting [Runner: JUnit 5] (0.094 s)
```