Statement of Work

The customer can choose whatever she/he wants to buy and can see the cost. The customer can pay Bitcoin, Creditcard, or PayPal. If it’s a lucky hour, which means a discount, the cost will be half. If the customer subscribes the site can use this luck.

Explanation of Utilized Design Patterns

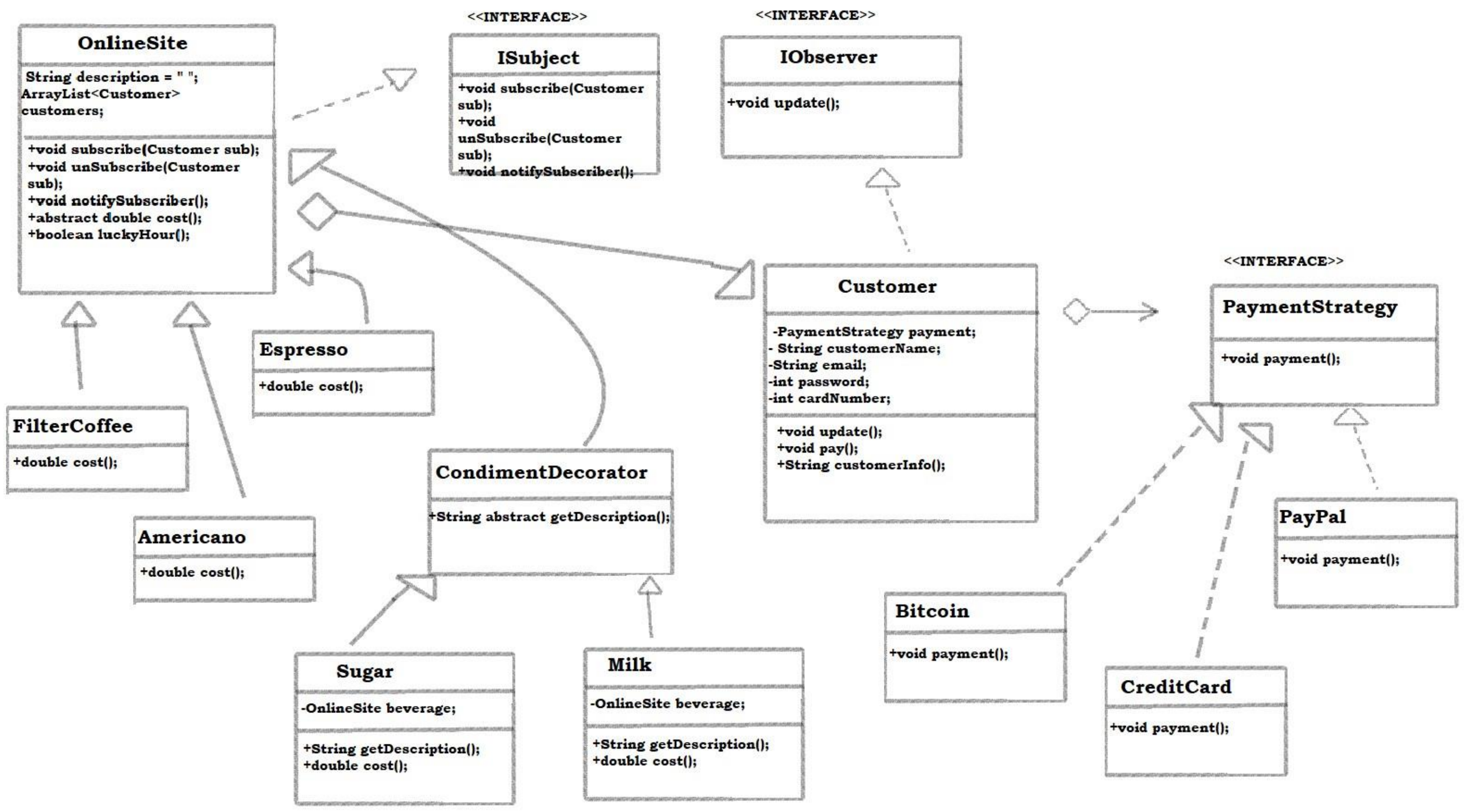
In this situation, I implement codes with 3 design patterns:

Strategy: The Strategy pattern allows us to dynamically swop out algorithms (i.e. application logic) at runtime. In my scenario, I want the to customer can changes the payment strategy on runtime.

Decorator: **It’s** a structural design pattern that lets you attach new behaviors to objects by placing these objects inside special wrapper objects that contain the behaviors. I use this pattern to calculate the total cost and get the name of the drink whatever the customer wants.

Observer: **Observer** is a behavioral design pattern that lets you define a subscription mechanism to notify multiple objects about any events that happen to the object they’re observing. In this scenario, I am using this pattern to give notifications to subscribers about lucky hours.

UML Class Diagram


OnlineSite
String description = " ";
ArrayList<Customer> customers;
+void subscribe(Customer sub);
+void unSubscribe(Customer sub);
+void notifySubscriber();
+abstract double cost();
+boolean luckyHour();
<<INTERFACE>>
ISubject
+void subscribe(Customer sub);
+void unSubscribe(Customer sub);
+void notifySubscriber();
<<INTERFACE>>
IObserver
+void update();
Customer
-PaymentStrategy payment;
- String customerName;
-String email;
-int password;
-int cardNumber;
+void update();
+void pay();
+String customerInfo();
<<INTERFACE>>
PaymentStrategy
+void payment();
Espresso
+double cost();
FilterCoffee
+double cost();
Americano
+double cost();
CondimentDecorator
+String abstract getDescription();
Sugar
-OnlineSite beverage;
+String getDescription();
+double cost();
Milk
-OnlineSite beverage;
+String getDescription();
+double cost();
PayPal
+void payment();
Bitcoin
+void payment();
CreditCard
+void payment();