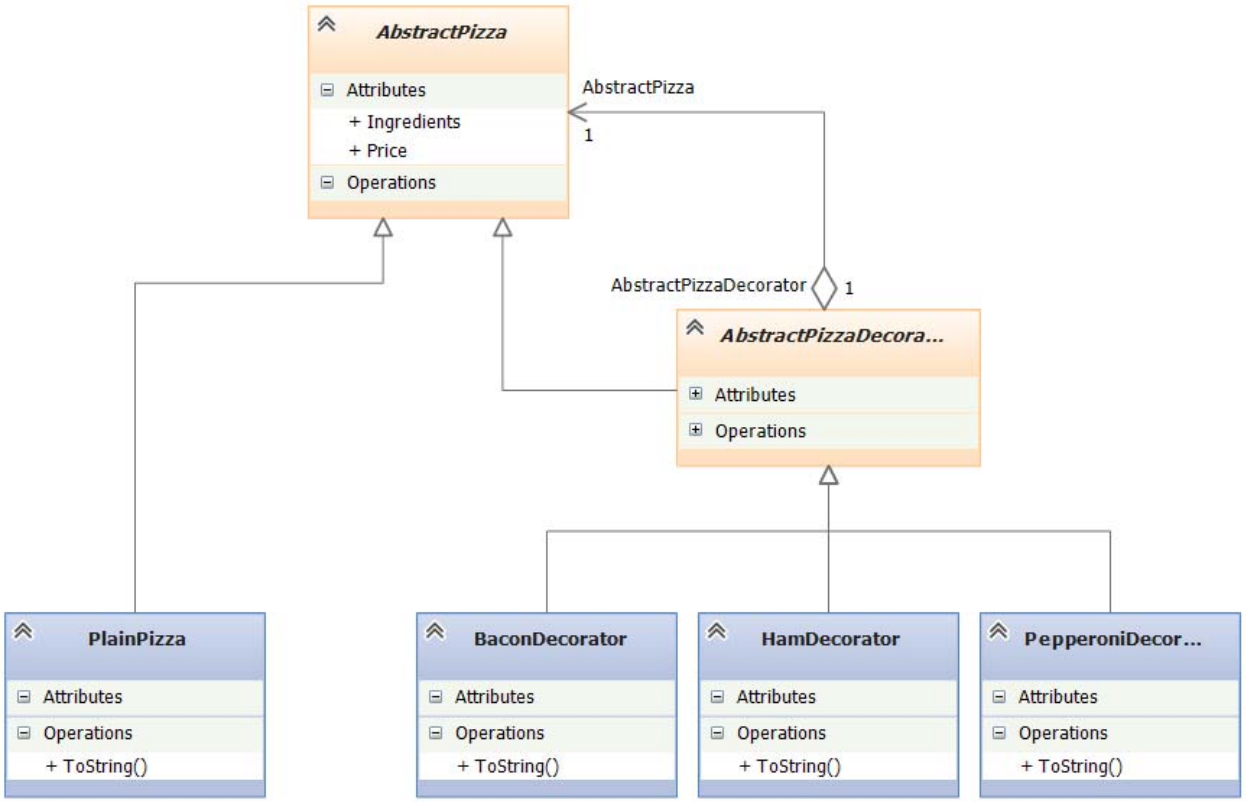
PizzaShopDesignPattern

Idea: At arbejde med følgende desing pattern: Decorator, Factory, Observer og Adapter..
Background: C# Note OBJECT-ORIENTED PROGRAMMING III – ADVANCED s.337-362,
Design Patterns in c# https://www.dofactory.com/net/design-patterns

Decorator

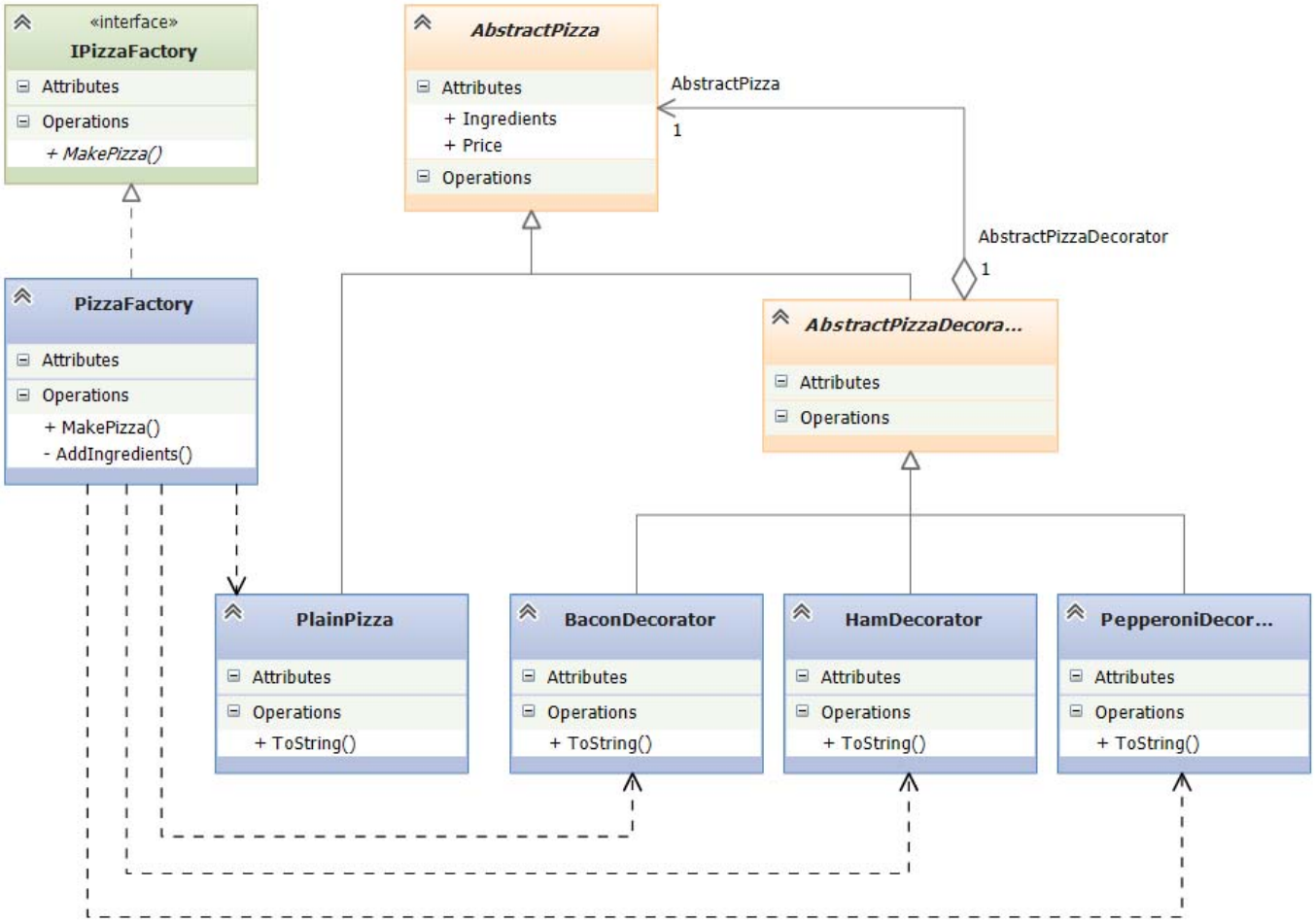
I denne opgave skal der implementeres en PizzaShop vha Decorator design pattern.
Der skal laves et simpelt Consol project, hvor følgende klasser implementeres:



Test i main.

Factory

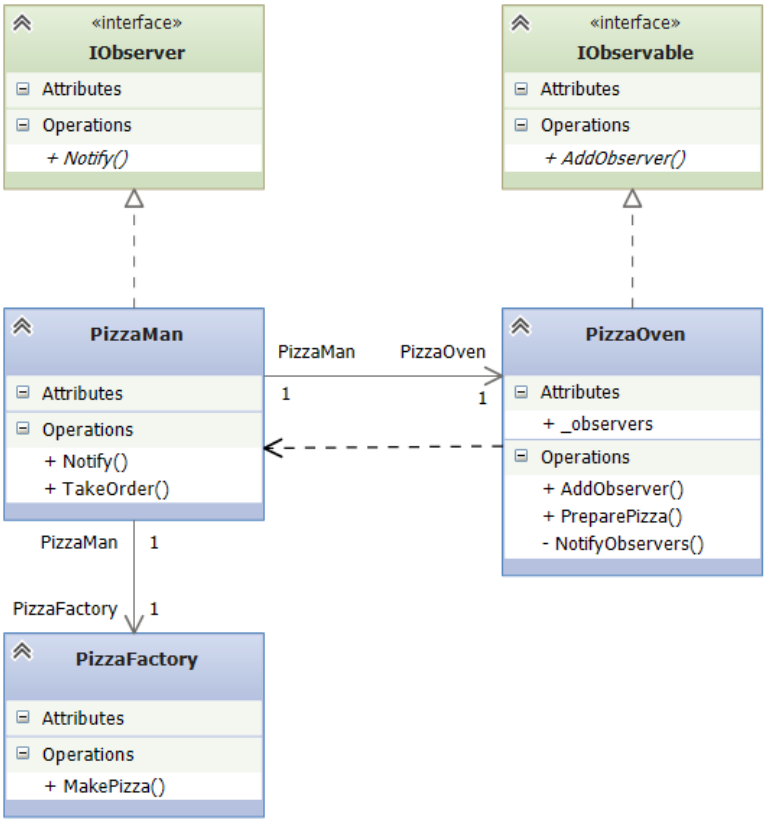
Udvid PizzaShoppen så der implementeres et Factory Design Pattern til at skabe Pizza objekter. MakePizza( ) kan tage et array af string som argument (en liste af ingredientser der skal tilføjes pizzaen (hvis man ikke ønsker en Margarita :) fx Bacon, Ham, Pepperoni ...). MakePizza( ) kan i en foreach-løkke kalde en privat hjælpemetode AddIngredients( ), der kan indeholde en switch-sætning til at afgøre hvilke "dekorationer" Margaritaen skal have.



Test i main

Observer

Udvid PizzaShoppen med Observer Design Pattern, fx ved at tilføje en PizzaMan og en PizzaOven ala følgende:



TakeOrder skal kalde MakePizza( ) på PizzaFactory og PreparePizza( )på PizzaOven.
PreparePizza kan have følgende udseende:

```
public void preparePizza(AbstractPizza p)
{

Task.Factory.StartNew(() =>
{
int preparetime = random.Next(5, 15);
System.Threading.Thread.Sleep(preparetime*1000);
NotifyObservers(p);
});
}
```

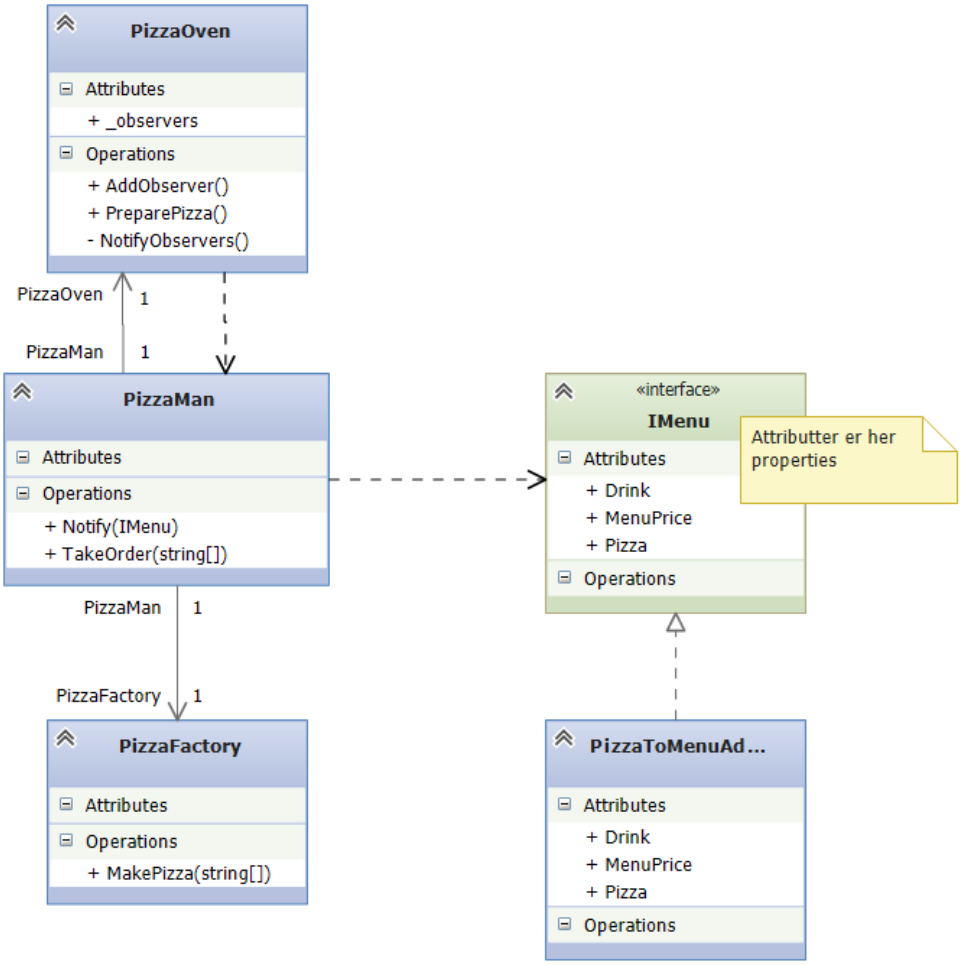
og NotifyObservers følgende udseeende:

```
private void NotifyObservers(AbstractPizza pizza)
{
foreach (var observer in _observers)
{
observer.Notify(pizza);
}
}
```

Test i main

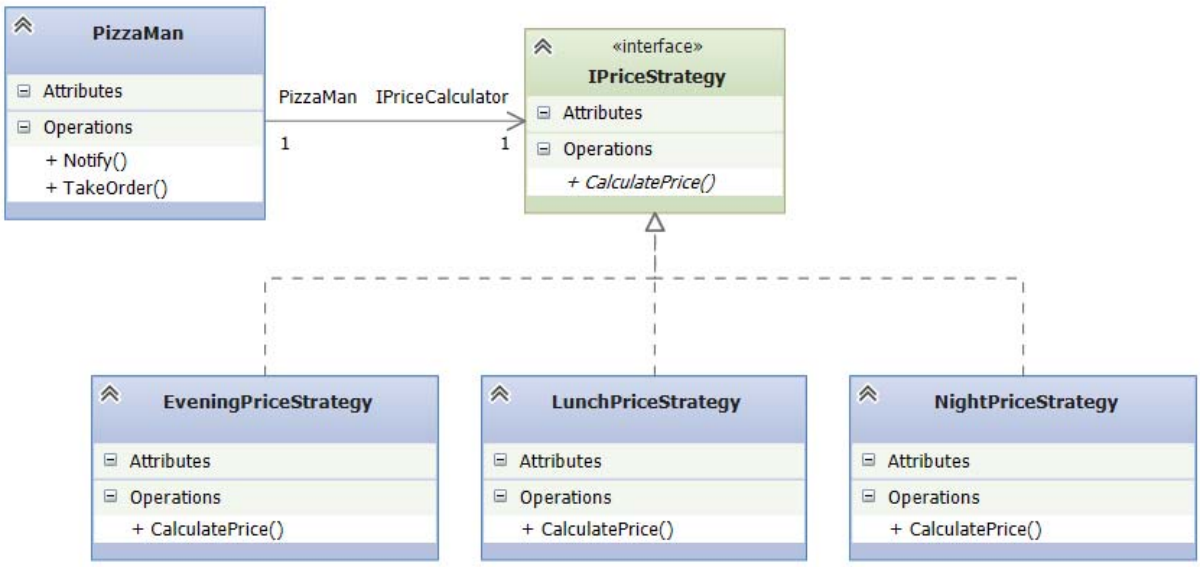
Adapter

Udvid PizzaShoppen med Adapter Design Pattern så der kan indføres en Pizza Menu med drik (fx øl el. vand) ala følgende:



Strategy

Udvid PizzaShoppen med Strategy Design Pattern, det kunne fx. være med forskellige priser for Frokost (-20%) og Nat (+10%), men der kunne være mange regler, alle implementeret i forskellige implementeringer af IPriceStrategy.



I første omgang er Client vores PizzaMan der benytter Strategy pattern når han skal finde prisen evt. med noget logik ala:

public enum TimeOfDay { Lunch, Evening, Night };

....

```
switch (DayTime)
{
case TimeOfDay.Evening: price = EveningPriceStrategy.CalculatePrice(pizza); break;
case TimeOfDay.Lunch: price = LunchPriceStrategy.CalculatePrice(pizza); break;
case TimeOfDay.Night: price = NightPriceStrategy.CalculatePrice(pizza); break;
}
```

Husk:

Test test test ... og meget gerne også en unittest :)

Har I tid til overs lav en APP med lister så der kan vælges Pizzaer, Ingredienser, Menuer m. Drikke mm

God fornøjelse
Henrik