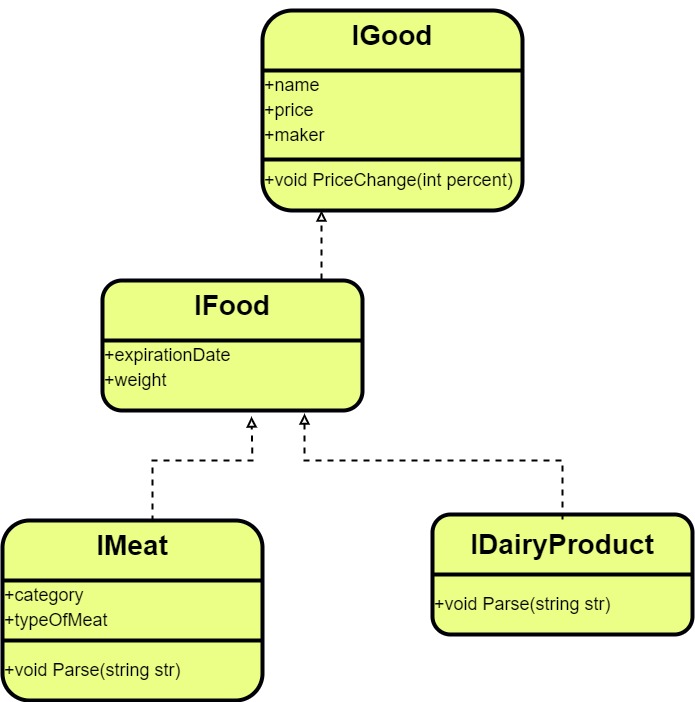
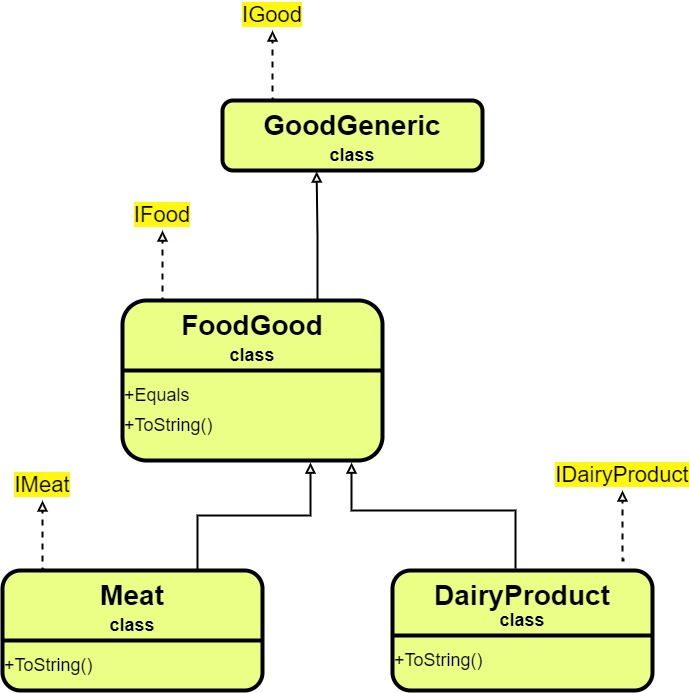
Доброго дня!   
 Роздуми проектування  
 Бокала Святослава,  
 студента 3-курсу механіко-математичного факультету,

Львівського Національного Університету ім. Івана Франка.  
  
Отже, почнемо.  


На цій картинці Ви можете бачити, як будуть будуватися інтерфейси для класів продуктів. Ієрархія Інтерфейсів починається з IGood, це загальний інтерфейс для всіх продуктів, починаючи від їжі закінчуючи всім чим ми хочемо. Для нього властиво мати такі властивості. По-перше, «назва» – “name”, кожний продукт має назву за яким його впізнають. По-друге, «ціна» ­– “price“, як і відомо все має свою ціну. По-третє, «виробник» – “maker”,також за кожним продуктом стоїть свій виробник. Також в цьому інтерфейсом робимо метод void типу, а саме ChangePrice який буде повертати зміну ціни на задану кількість процентів, які буду приходити у виді параметру.

Далі від нього вибудовуємо надбудови, а саме інтерфейси конкретних типів продуктів, типу «їжа», «фільми», «ліки» і т.д. . У мене лише їжа, тому я зробив інтерфейс, IFood, лише для неї. Він імплементує інтерфейс IGood, і також має такі властивості, як «термін придатності» -“expirationDate”, а також   
«вага» - “weight”.

Вже від нього вибудовуємо більш конкретні інтерфейси для продуктів харчування, такі як IMeat і IDairyProduct. Для інтерфейса м’ясо, визначаємо ще такі властивості, як «категорія м’яса» - “category”, а також   
«тип м’яса» - “typeOfMeat”. Для обох класів зробимо метод void типу Parse, який буде парсити з стрічки в наш продукт, де стрічка приходить як параметр.

Отже, з інтерфейсами розібралися. Тепер перейдемо до реалізації класів.

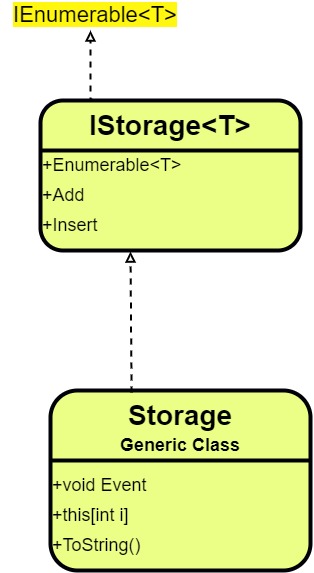
Чесно скажу, довго думав над цим і мені все ж здається, що GoodGeneric і FoodGood потрібно робити абстрактними класами, для того щоб проект був більш гнучкий і можливий до змін.

Отже, GoodGeneric, це загальний, батьківський клас для будь-якого продукту, також який імплементує інтерфейс IGood.

FoodGood, клас який узагальнює будь-яку харчову продукцію, наслідуваний від GoodGeneric, а також імплементує інтерфейс IFood. Цей клас має такі медоти, як Equals і ToString().

Meat, клас який наслідуться від FoodGood, а також імплементує інтерфейс IMeat, в собі має метод ToString().

Те саме і для DairyProduct, тільки використовуємо інший інтерфейс, а саме IDairyProduct.

Ніби все про класи розказав, тепер давайте перейдемо до самого Storage.

Отже ось так в нас виглядає Storage. По-перше утворюємо інтерфейс IStorage<T>, який імплементує інтерфейс IEnumerable<T>. В ньому є властивість IEnumerable<T> для того аби бути спроможним підставити будь-яку колекцію чи то список, словник, множина і т.д.. Також має методи за допомогою яких ми додаємо до колекції, а також вилучаємо, відповідно їх назва «Add» та «Insert».

Тепер перейдімо до самого класу Storage. Він імплементує наш інтерфейс IStorage<T>, а також має додаткові параматре, а саме подія, яка буде виконуватись в методі «Add» і при зчитувані з файлу, якщо наш продукт не задовольняє певні вимоги, як наприклад з продуктами харчування, коли термін придатності вийшов. Також коли виконується подія будем записувати це в який файл, в нашому випадку у текстовий, а загалом можна в будь-куди інше, як наприклад в базу даних SQL. Звісно надамо можливість добиратись до елемента колекцію по індексу. І також надамо метод ToString().