Во второй контрольной мы рассмотрим более привычную для маглов предметную область. Вы когда-нибудь были в ресторанах быстрого питания? Обычно там все работает следующим образом: вы делаете заказ, выбирая из меню блюда. При-этом блюда часто объединены в так называемые комбо-наборы, которые уже включают определенные позиции и имеют фиксированную цену. После совершения заказа ему присваивается некий номер, а сам заказ передается на кухню, где специально обученные граждане готовят блюда и кладут их на ваш поднос. Когда ваш заказ выполняют полностью, на специальном табло загорается его номер и вы, счастливый (ая) и молодой (ая) спешите приступить к трапезе.

**Задание 1 – 3 балла**

Для начала давайте разметим предметную область. Добавьте в проект иерархию классов, реализующих логику указанных типов блюд. При этом каждое блюдо должно обладать свойствами:

* Name (string)
* Cost (decimal)
* IsReady (bool)

Каждое свойство должно содержать оба аксессора – get и set. Однако в свойстве IsReady разрешите обращаться к аксессору set только из самого класса и его наследников.

Добавьте в комбо-блюдо свойство для хранения списка обычных блюд.

Добавьте класс Order, который должен содержать в приватном поле список блюд (любого типа). Добавьте в класс строковое свойство OrderNumber, а также конструктор, который принимает значение номера заказа и выставляет его соответствующему свойству.

**Задание 2 – 1 балл**

Добавьте в класс простого блюда метод MarkAsReady, который должен выставлять признак готовности блюда в true. Переопределите свойство IsReady в классе комбо-блюда: сделайте его автоматически вычисляемым – возвращающим true, если все блюда в комбо готовы.

**Задание 3 – 2 балла**

В постановке задачи мы с вами упомянули важный подход, подразумевающий разделение логики нашего класса Order на 2 части: для взаимодействия с заказчиком и для работы на кухне. Например, клиент может должен уметь добавлять блюдо в заказ, смотреть информацию по заказу (например, в виде чека), а также получать обратную связь по готовности заказа. В свою очередь, на кухне должны уметь видеть составляющие заказ простые блюда, а также менять статус каждого из этих блюд. Добавьте в вашу программу 2 интерфейса, описывающих способ работы с заказом: IClientOrder и IKitchenOrder. IClientOrder должен содержать следующие элементы:

* Метод AddMeal (принимает блюдо и не возвращает ничего)
* Метод GetReceiptInfo (не принимает ничего и возвращает список строк)
* Событие OnOrderReady (описывает сигнатуру метода, принимающего 1 строковый аргумент и не возвращающего ничего)

В свою очередь IKitchenOrder должен содержать следующие элементы:

* Метод MakeMealReady (принимает строку – название блюда и не возвращает ничего)
* Метод GetSimpleMeals (не принимает ничего, возвращает массив или список ПРОСТЫХ БЛЮД)

Укажите, что ваш класс Order реализует оба интерфейса.

**Задание 4 – 2 балла**

Реализуйте метод GetSimpleMeals. Его основная идея в том, что он возвращает простые блюда, которые готовят на кухне. Таким образом, если в заказе лежит комбо-блюдо, его необходимо разложить на массив простых блюд.

Реализуйте метод AddMeal, который принимает любое блюдо и добавляет его в приватный список.

**Задание 5 – 1 балл**

Реализуйте метод GetReceiptInfo, который возвращает список строк, используемых для формирования позиций фискального чека. Каждая строка должна содержать название и цену блюда, например: «Биг-мак – 150р», «Комбо “Счастливый завтрак” – 350р». В этом методе раскладывать комбо на составляющие не нужно)

**Задание 6 – 2 балла**

Добавьте событие OnOrderReady. Добавьте метод MakeMealReady, который принимает название простого блюда, ищет первое такое неготовое блюдо в заказе и помечает его как готовое. В конце проверяет блюда в заказе, если все они готовы – вызывает срабатывание события OnOrderReady.

**Задание 7 – 1 балл**

Создаете в методе Main заказ как объект типа IClientOrder (например, IClientOrder order = new Order(“A1”)). Добавьте в него как минимум 1 простое и 1 комбо-блюдо. Добавьте обработчик события готовности (вывод сообщения в консоль). Используйте оператор as, чтобы перевести ваш заказ в тип IKitchenOrder. Выполните метод MakeMealReady необходимое количество раз, чтобы в консоли появилось сообщение о готовности заказа.

.

In the second classroom assignment, we will look at a more familiar subject area for Muggles. Have you ever been to fast food restaurants? Everything there usually works like this: you make an order by choosing dishes from the menu. At the same time, several dishes are often combined into so-called combo sets, which already include a certain set of items and have a fixed price. After making an order, a certain number is assigned to it, and the order itself is transferred to the kitchen, where specially trained citizens prepare dishes and put them on your tray. When your order is completed, its number lights up on a special tableau and you, happy and young, hurry to start your meal.

**Task 1 – 3 points**

First, let us settle the subject area. Add a hierarchy of classes to the project that implement the logics of the specified types of dishes. Make sure each dish has the following set of properties:

• Name (string)

• Const (decimal)

• IsReady (bool)

Each property must contain both get and set accessors. However, for the **IsReady** property, allow an access to the **set** assessor only from the class itself and its descendants.

Add a property to the combo dish to store a list of simple dishes.

Add the Order class, which should contain a list of dishes (of both types) in the private field. Add a string property OrderNumber to the class, as well as a constructor that accepts the value of the order number and sets it to the corresponding property.

**Task 2 – 1 point**

Add method MarkAsReady to the simple dish class setting the ready dish attribute to true. Override the IsReady property in the combo dish class: make It automatically calculated - returning true if all the dishes in the combo are ready.

**Task 3 – 2 points**

In the problem statement, we mentioned an important approach that implies dividing the logic of our Order class into 2 parts: for interacting with the customer and for all the kitchen issues. For example, a customer may need to add a dish to an order, view information about the order (for example, in the form of a receipt), and receive feedback when the order gets ready. In its turn, the kitchen should be able to see the simple dishes that make up the order, as well as change the status of each of these dishes. Add 2 interfaces to your program that describe how to work with the order: IClientOrder and IKitchenOrder. IClientOrder must contain the following elements:

• AddMeal method (accepts a dish and returns nothing)

• GetReceiptInfo method (accepts nothing and returns a list of strings)

• OnOrderReady event (describes the signature of a method that accepts 1 string argument and returns nothing)

In its turn, IKitchenOrder must contain the following elements:

• MakeMealReady method (accepts a string – the name of the dish and returns nothing)

• GetSimpleMeals method (accepts nothing, returns an array or a list of SIMPLE DISHES)

Specify that your Order class implements both interfaces.

**Task 4 – 2 points**

Implement the GetSimpleMeals method. Its main idea is that it returns simple dishes that are prepared in the kitchen. Thus, if the order contains a combo dish, it must be decomposed into an array of simple dishes.

Implement the AddMeal method, which accepts any dish and adds it to the private list.

**Task 5 – 1 point**

Implement the GetReceiptInfo method, which returns a list of strings used to generate fiscal receipt positions. Each line should contain the name and price of the dish, for example: "Big Mac – 150R", "Happy Breakfast combo – 350R". In this method, you do not need to decompose the combo into components)

**Task 6 – 2 points**

Add the OnOrderReady event. Add the MakeMealReady method, which takes the name of a simple dish, searches for the first such non-ready dish in the order, and marks it as ready. At the end, it checks the dishes in the order, if they are all ready – triggers the OnOrderReady event.

**Task 7 – 1 point**

Create an order in the Main method as an object of the IClientOrder type (for example, IClientOrder order = new Order(“A1")). Add at least 1 simple and 1 combo dish to it. Add an OnOrderReady event handler (output the message to the console). Use the **as** operator to convert your order to the IKitchenOrder type. Run the MakeMealReady method as many times as necessary to display a readiness-indicating message in the console.