МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем



**Лабораторна робота №2**

з дисципліни “Об’єктно-орієнтовне програмування”

***тема:*** «Реалізація основних принципів ООП мовою C#»

|  |  |
| --- | --- |
|  | Виконав Вернік Данило  студент II курсу  групи КП-22 |

Київ 2023

Мета роботи:

Ознайомитися з такими можливостями мови програмування C# як абстрактні класи, інтерфейси, делегати. Вивчити механізми оброблення подій у C#, а також можливості, які мають методи-розширення.

Постановка задачі

Для ієрархії класів, побудованої в лабораторній роботі №1, реалізувати:

1. Множину інтерфейсів. При чому один з класів повинен реалізовувати щонайменше 2 інтерфейси. Також продемонструвати реалізацію explicit implementation інтерфейса, обгрунтувати її використання.
2. Абстрактний клас. Забезпечити його наслідування. Наявність в цьому класі абстрактних методів - обов’язкова.
3. Механізм «делегат – подія – обробник події».
4. Перетворити код, який забезпечує роботу з подіями та обробниками подій, на код, що використовує (\*):
   1. анонімні методи;
   2. lambda-вирази;
   3. типи Action та Func (кожен з них).

(\*) - допускається реалізація коду однієї події різними способами, необов’язково різних подій.

1. Механізм створення та оброблення власних помилок:
   1. створити новий клас виключної ситуації;
   2. створити новий клас аргументів для передачі їх до обробника виключної ситуації;
   3. забезпечити ініціювання створеної виключної ситуації та продемонструвати, як працює обробник даної помилки;
   4. реалізувати різні сценарії оброблення помилки.
2. Метод-розширення будь-якого класу.

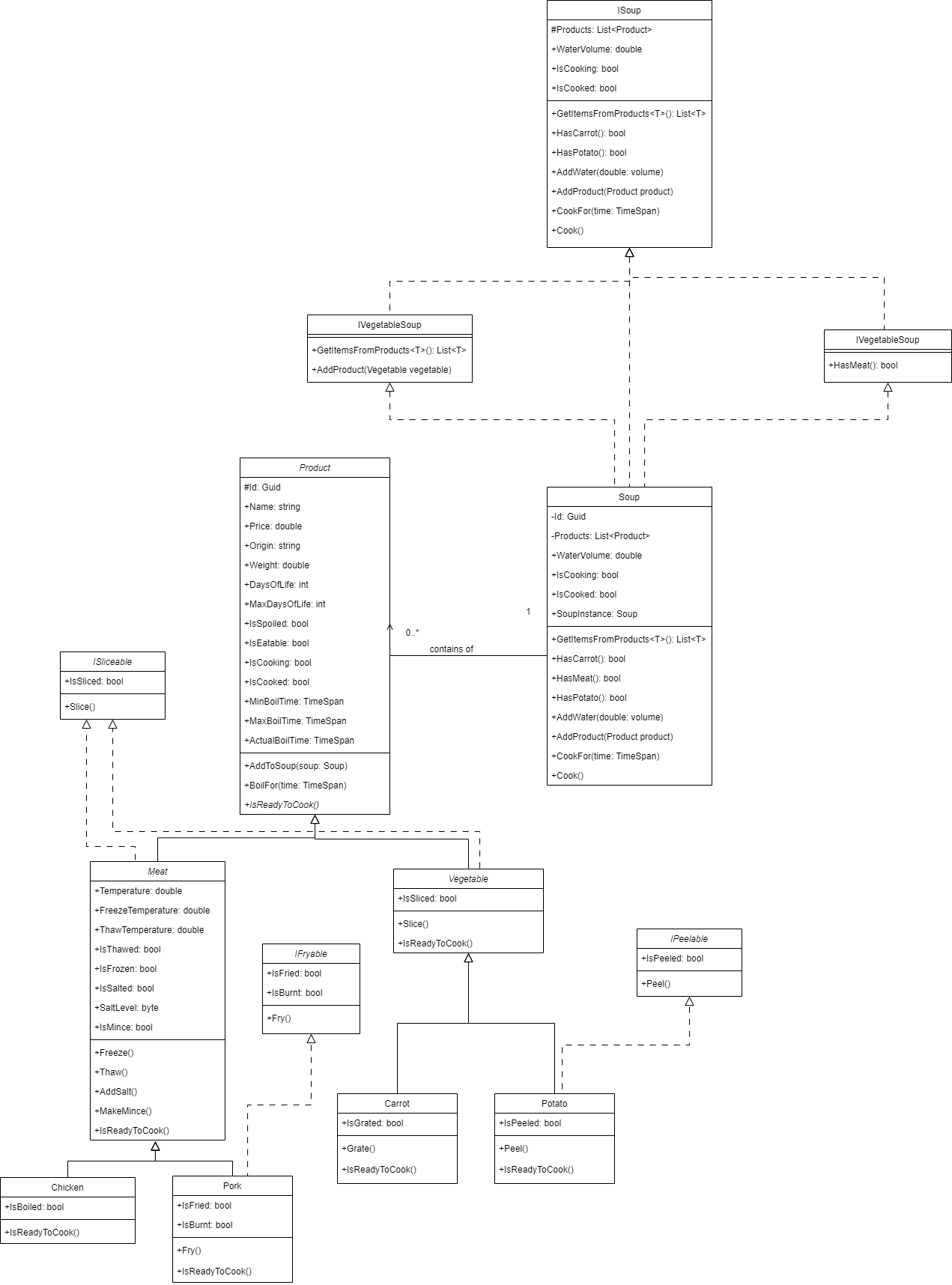
Протокол має містити: титульний аркуш, постановку задачі, UML діаграму класів, фрагменти коду, які демонструють виконання поставлених задач, висновки.

**Всього –** **10 балів**

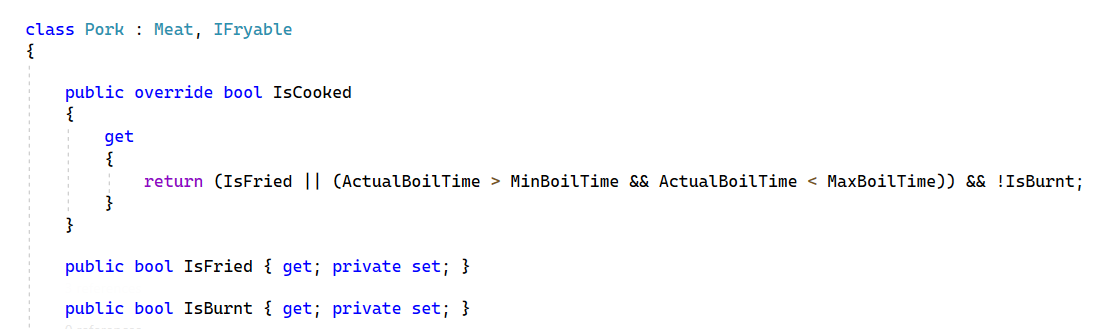
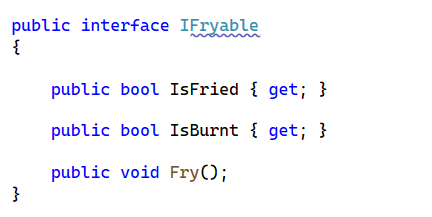
**Хід роботи:**

Почав виконання лабораторної роботи з обрання сфери з реального світу для створення ієрархії класів разом з інтерфейсом – «Продукти та суп».

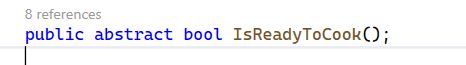
Розробив діаграми класів у сервісі app.diagrams.net для даної області:

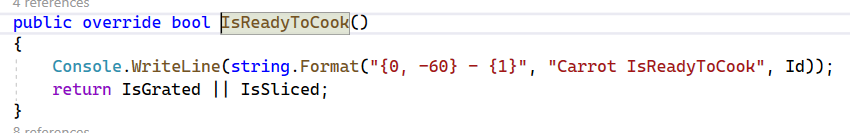


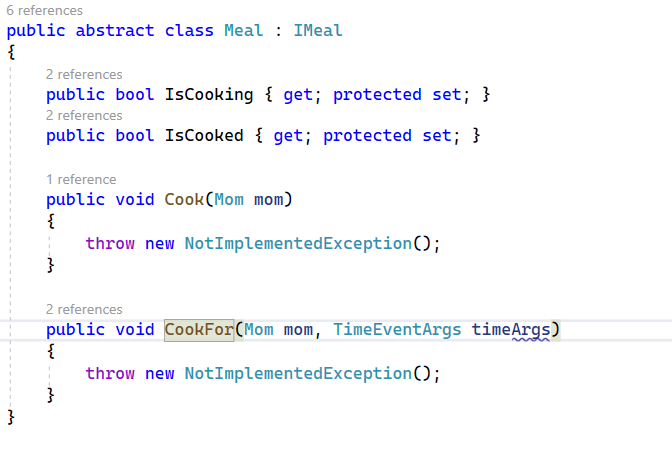
1) В UML діаграмі можна побачити використання інтерфейсів, що мають зв’язок, позначений пунктирною лінією.



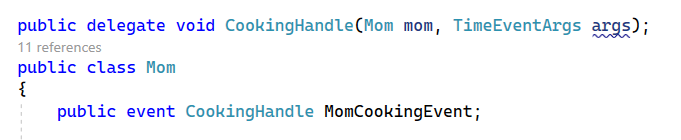
2) Є абстрактні класи з наявними абстрактними методами:

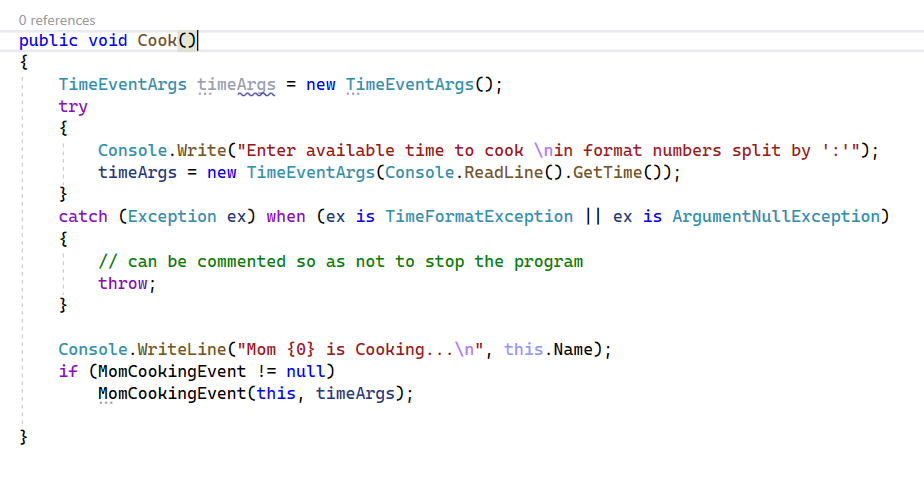




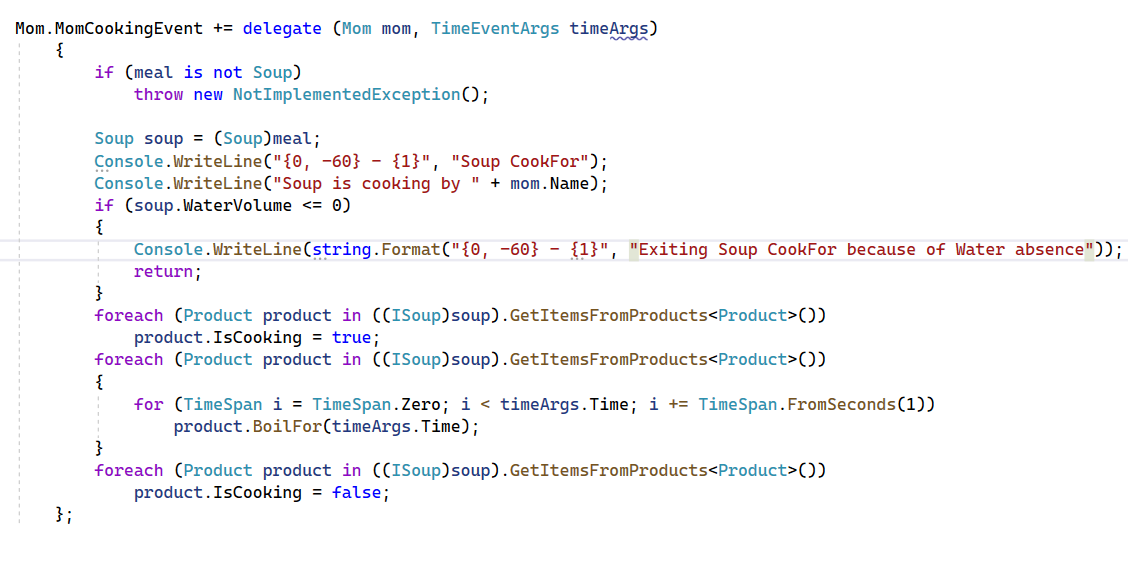
****

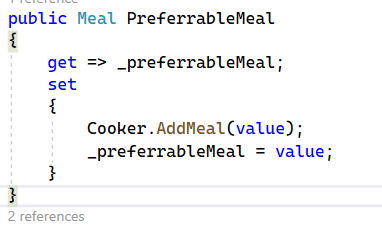
3) Механізм «делегат – подія – обробник події» зроблено таким чином:

****

****

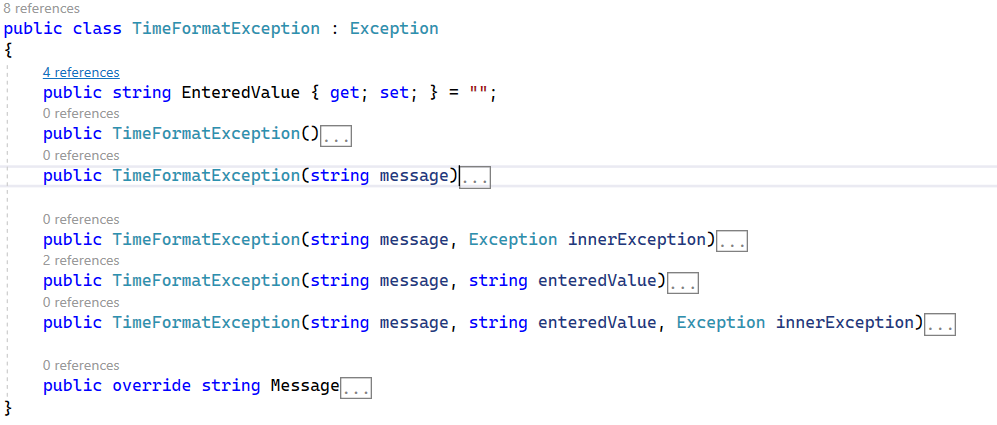
4) анонімні методи:



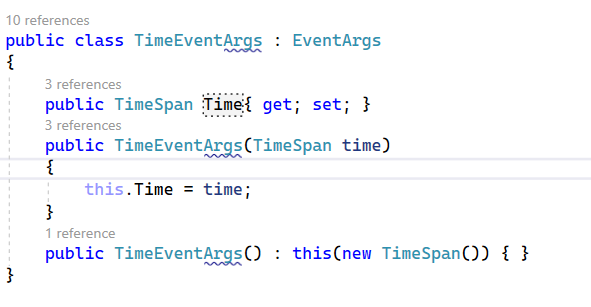
Лямбда-вираз: 

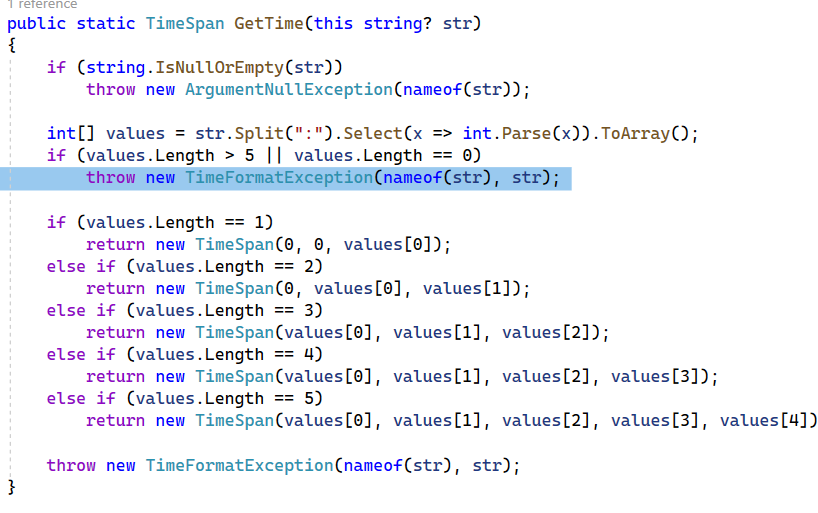
Func: 

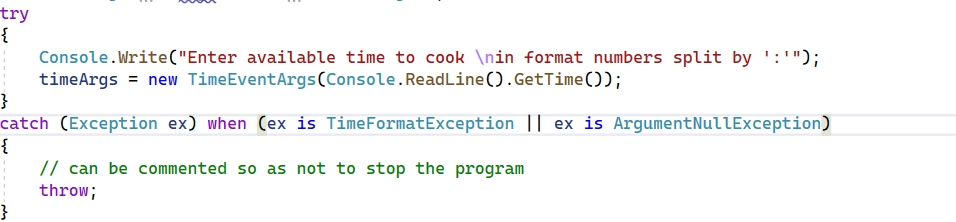
5) новий клас виключної ситуації;



новий клас аргументів для передачі їх до обробника виключної ситуації;



ініціювання створеної виключної ситуації та продемонструвати, як працює обробник даної помилки

реалізувати різні сценарії оброблення помилки

6) Метод-розширення будь-якого класу. 

**Висновки:**

У даній лабораторній роботі я ознайомився з такими можливостями мови програмування C# як абстрактні класи, інтерфейси, делегати. Вивчив механізми оброблення подій у C#, а також можливості, які мають методи-розширення

1. Множину інтерфейсів. При чому один з класів повинен реалізовувати щонайменше 2 інтерфейси. Також продемонструвати реалізацію explicit implementation інтерфейса, обгрунтувати її використання.
2. Абстрактний клас. Забезпечити його наслідування. Наявність в цьому класі абстрактних методів - обов’язкова.
3. Механізм «делегат – подія – обробник події».
4. Перетворити код, який забезпечує роботу з подіями та обробниками подій, на код, що використовує (\*):
   1. анонімні методи;
   2. lambda-вирази;
   3. типи Action та Func (кожен з них).

(\*) - допускається реалізація коду однієї події різними способами, необов’язково різних подій.

1. Механізм створення та оброблення власних помилок:
   1. створити новий клас виключної ситуації;
   2. створити новий клас аргументів для передачі їх до обробника виключної ситуації;
   3. забезпечити ініціювання створеної виключної ситуації та продемонструвати, як працює обробник даної помилки;
   4. реалізувати різні сценарії оброблення помилки.
2. Метод-розширення будь-якого класу.