ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КПІ імені Ігоря Сікорського”

Кафедра програмного забезпечення комп’ютерних систем

**ЛАБОРАТОРНА РОБОТА №2**

з дисципліни

Програмування

C# .Net. Реалізація основних

принципів ООП мовою C#

Студента 2 курсу ФПМ

гр. КП-23 Верьовочкіна М. С.

Київ – 2024

**Завдання на лабораторну роботу №2.**

**«C# .Net. Розширені можливості реалізації ООП у мові C#. Події.»**

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).

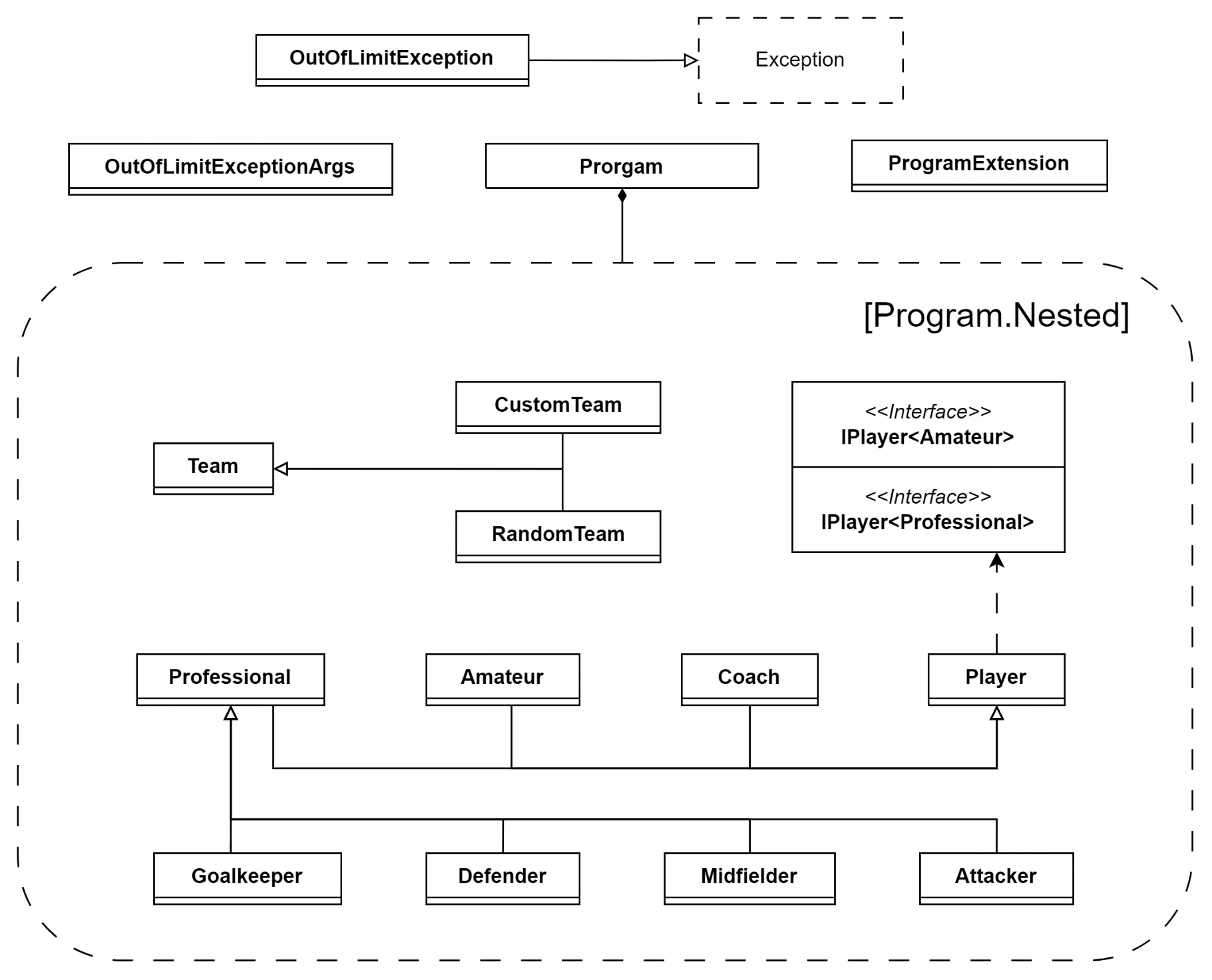
(\*) - допускається реалізація коду однієї події різними способами, необов’язково різних подій.

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.

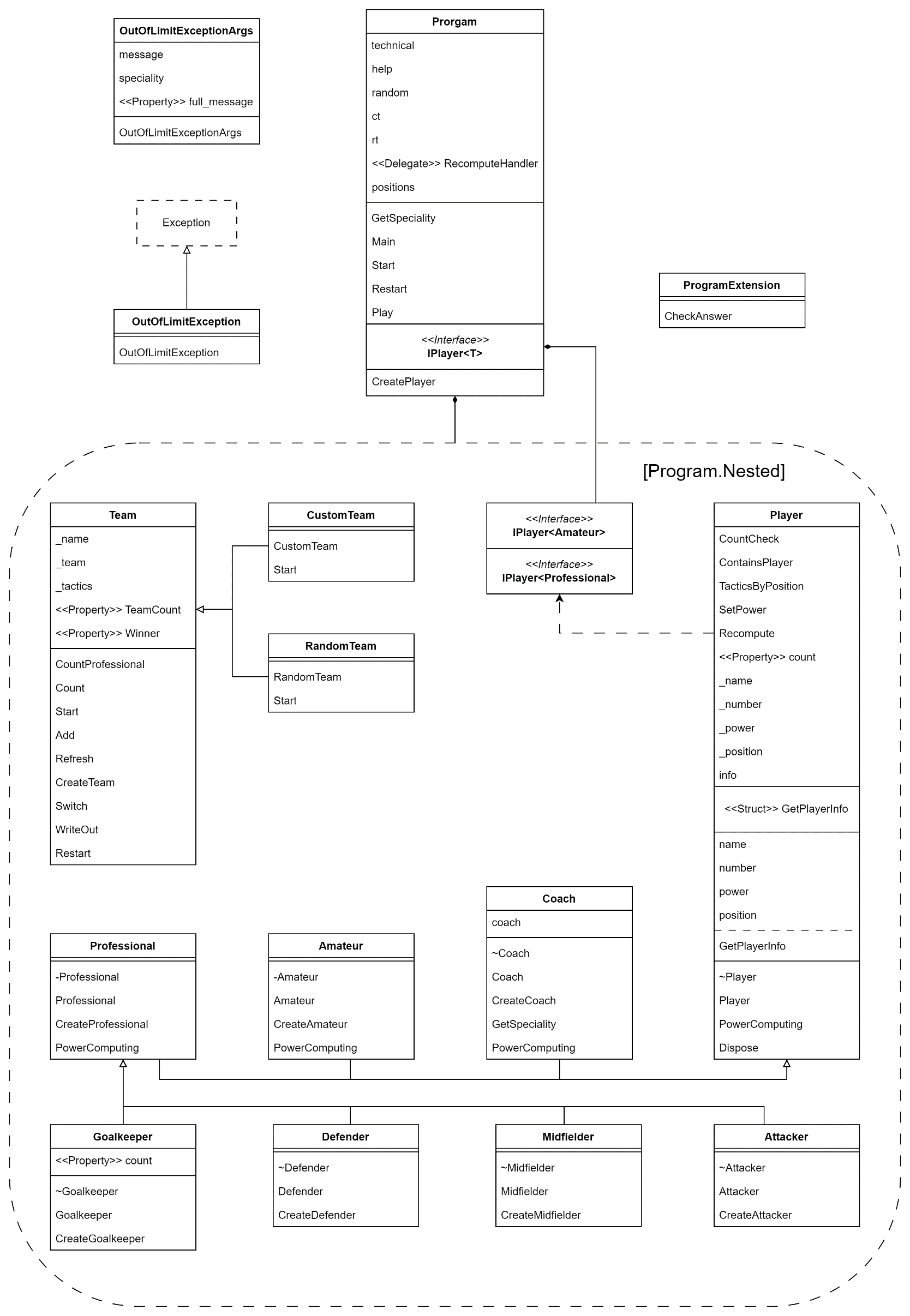
Протокол має містити: титульний аркуш, постановку задачі, UML діаграму класів, фрагменти коду, які демонструють виконання поставлених задач, висновки.

**Всього –** **10 балів**

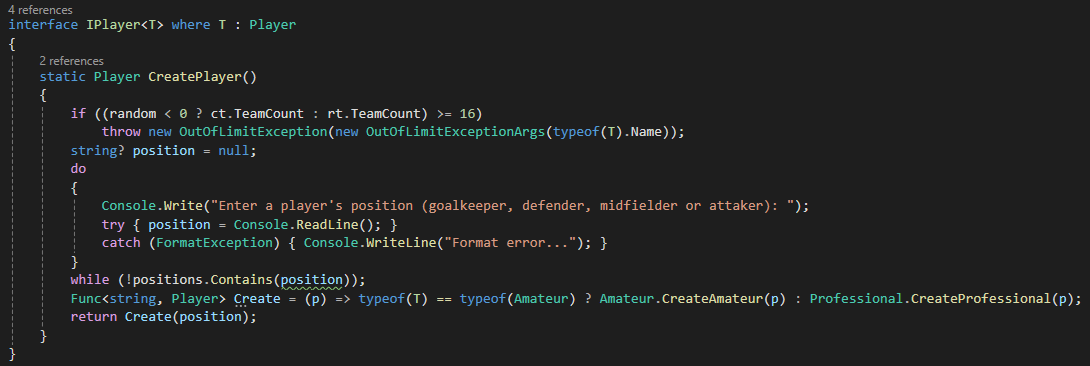
UML-діаграма розробленої програми (спрощена версія; File 1: «Minimized UML.png»):



UML-діаграма розробленої програми (повна версія; File 2: «Full UML.png»):



1. Було створено інтерфейс IPlayer типу generic:



Його реалізує клас Player у двох різнотипових варіантах: IPlayer<Amateur> і IPlayer<Professional>:



Явний виклик інтерфейсу здійснює функція Main(). Необхідність явного виклику очевидна за збереження цілісності типів даних при створенні об’єктів відповідних класів:

A screen shot of a computer code

Description automatically generated

1. Створено абстрактний клас Team та його нащадків CustomTeam і RandomTeam:

A screenshot of a computer

Description automatically generated

Батьківський клас має абстрактний метод Start(bool), який реалізують його нащадки:

A computer screen shot of a program code

Description automatically generated

A computer screen shot of a program code

Description automatically generatedA computer screen shot of a program code

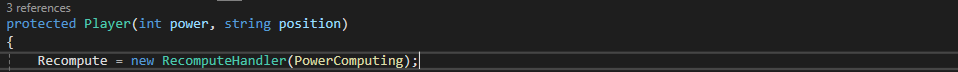
Description automatically generated

1. Створено та використано механізм «делегат – подія – обробник події»:



A screen shot of a computer

Description automatically generated

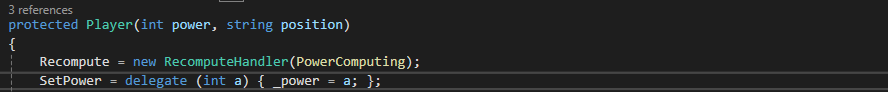


A computer screen shot of a computer

Description automatically generated



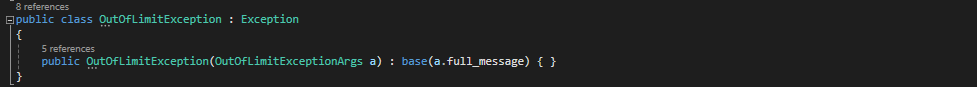
1. Для розширення функціональності програми та кращого забезечення цілісності полів та методів класів з їхніми модифікаторами доступу було створено ще 4 подібні механізми (оголошення наведені у другому скріншоті пункту 3) із застосуванням анонімних методів, лямбда-виразів і типів Action, Func:



A computer code on a black background

Description automatically generated

1. Розроблено механізм оброблення власних помилок, що запобігає та попереджає про перевищення ліміту кількості гравців певного типу чи загальної кількості:



Також створено окремий клас аргументів для обробника виключної ситуації:

A computer screen shot of text

Description automatically generated

У коді реалізовані різні сценарії ініціювання та обробки цієї помилки. У якості прикладу продемонстрована спроба створення другого голкіпера при виконанні програми:

A computer screen shot of a black screen

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

A computer screen shot of a code

Description automatically generated

A computer screen with text

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Додано метод-розширення основного класу Program, який запитує підтвердження на визначене запитання:

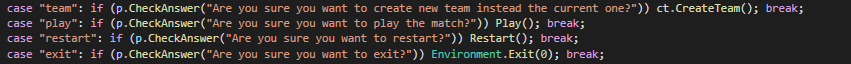
A computer screen with colorful text

Description automatically generated

Він викликається головною функцією:

A black screen with white text

Description automatically generated



**Висновок:** в ході виконання лабораторної роботи було продемонстровано можливість розширення програми за допомогою таких елементів мови C#, як абстрактні класи, інтерфейси, делегати, методи-розширення, механізми оброблення подій та власних помилок.

**Список файлів та посилань:**

* 1. **File 1:** «Minimized UML.png»

***Посилання:*** <https://github.com/VerovochkinMaksym/programming-23-24/blob/8e81c8481e30d0b709c6e33648791da8b9598fd5/sem_1_lab_2/Minimized%20UML.png>

* 1. **File 2:** «Full UML.png»

***Посилання:*** <https://github.com/VerovochkinMaksym/programming-23-24/blob/8e81c8481e30d0b709c6e33648791da8b9598fd5/sem_1_lab_2/Full%20UML.png>

* 1. Повний розроблений код міститься за посиланням: <https://github.com/VerovochkinMaksym/programming-23-24/blob/8e81c8481e30d0b709c6e33648791da8b9598fd5/sem_1_lab_2/Program.cs>