Міністерство освіти і науки України
Національний авіаційний університет
Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра інженерії програмного забезпечення

Пояснювальна записка до курсової роботи
з дисципліни «Об'єктно-орієнтоване програмування»

Виконав: студент групи ПІ-221Б
Кондратюк Андрій Олександрович
Прийняв:
Дишлевий Олексій Петрович

Київ 2022

Зміст

Опис завдання

Завданням курсової роботи є проектування та реалізація на мові C# програмного забезпечення накопичення та обробки даних домену відповідно заданого варіанту

1. В якості типу застосування обрати консольне застосування з командним рядком, як виняток, Windows Forms чи WPF.

2. Спроектувати та реалізувати систему класів, в основу якої покладено логічну структуру даних, наведену у варіанті, для накопичення та обробки даних домену відповідно варіанту курсової роботи.

Варіант 9: “Футбол: планування проведення футбольних матчів”

Функціональні вимоги до програмного забезпечення

1. Управління гравцем

1.1 Можливість додати гравця

1.2 Можливість видалити гравця

1.3 Можливість зміни даних про гравця

1.3.1 Можливість змінити ім’я

1.3.2 Можливість змінити прізвище

1.3.3 Можливість змінити дату народження

1.3.4 Можливість змінити статус гравця

1.3.5 Можливість зміни статусу здоров’я

1.3.6 Можливість зміни зарплатні

1.4 Можливість перегляду інформації про гравця

1.5 Можливість перегляду всього списку гравців

2. Управління іграми

2.1 Можливість додати гру

2.2 Можливість видалити гру

2.3 Можливість зміни дані про гру

2.3.1 Додати гравця

2.3.2 Видалити гравця

2.3.3 Змінити дату проведення гри

2.3.4 Змінити місце проведення гри

2.3.5 Зміна кількості глядачів, присутніх на грі

2.3.6 Введення результату гри, якщо вона була проведена

2.4 Можливість перегляду інформації про гру

2.5 Можливість перегляду всього списку ігор

2.5.1 Можливість сортування ігор за датою проведення 2.5.2 Можливість сортування ігор за результатом на 4 частини: «Виграні», «Програні», «Нічия», «Ще не проведені»

3. Управління футбольними стадіонами

3.1 Можливість додати стадіон

3.2 Можливість видалити стадіон

3.3 Можливість зміни даних стадіону

3.3.1 Зміна кількості місць

3.3.2 Зміна ціни за місце

3.4 Можливість перегляду інформації про стадіон

3.4.1 Кількість місць

3.4.2 Ціна за місце

3.4.3 Дата проведення ігор та команди, які будуть грати

4. Пошук

4.1 Пошук гравця за ім’ям чи прізвищем

4.2 Пошук гри за датою проведення та назвою команди-противника

4.3 Пошук стадіону за назвою

Відповідно до завдання, з застосуванням програмного компоненту Windows Forms було розроблено програмне забезпечення накопичення та обробки даних про гравців, футбольні матчі, футбольні команди та стадіони.

Опис шарів та загальна архітектура проекту

В проекті реалізовано типову трьохшарову архітектуру, де представлений **шар зберігання і доступу до даних** Data Access Layer (в проекті DAL), **шар бізнес-логіки окремих компонентів** Business Logic Level (в проекті — BLL) та **шар представлення** Presentation Level (у проекті — PL) – графічний інтерфейс, реалізований за допомогою Windows Forms.

Шар доступу до даних

Шар доступу до даних — найнижчий рівень програмного застосунку, який безпосередньо здійснює управління та менеджмент даних, що необхідно зберегти або завантажити для подальшої роботи. У роботі він представлений окремим проектом під назвою **DAL**, що має класи сутностей, з якими працюватиме програма - клас гравця **Player**, клас команди **Team**, клас стадіону **Stadium** та клас матчу **Game**, а також статичні класи для полегшення доступу та збереження цих сутностей **FileOperations** і **Lists**.

Клас **Player** включає в себе символьні значення на позначення ім'я та прізвища гравця, значення типу дати-часу на позначення дати народження, символьні статус та стан здоров'я, число з плаваючою комою на позначення зарплати, символьне позначення позиції, на якій грає гравець та методи **ToString()** для коректного відображення у меню, **Display()** для відображення інформації про гравця та **GetObjectData()** - для отримання даних для серіалізації.

Клас **Team** включає в себе символьне значення на позначення назви команди, гравця, який має бути воротарем, список гравців на позиції захисників, півзахисників та форвардів. Крім цього, клас включає методи доступу та відображення інформації з класу Player і метод почленного копіювання **Copy()**.

Клас **Stadium** має в собі значення символьного типу для позначення назви стадіону, цілочисельне значення для кількості місць та ціна за місце типу double. Наявні методи попередньо згаданого класу.

Клас **Game** включає в себе домашню та гостьову команди, кількість глядачів, кількість забитих кожною командою голів, статус гри, дату її проведення, стадіон та методи попередніх класів.

Поля класів реалізовані як публічні властивості для спрощення їх сприйняття та зміни.

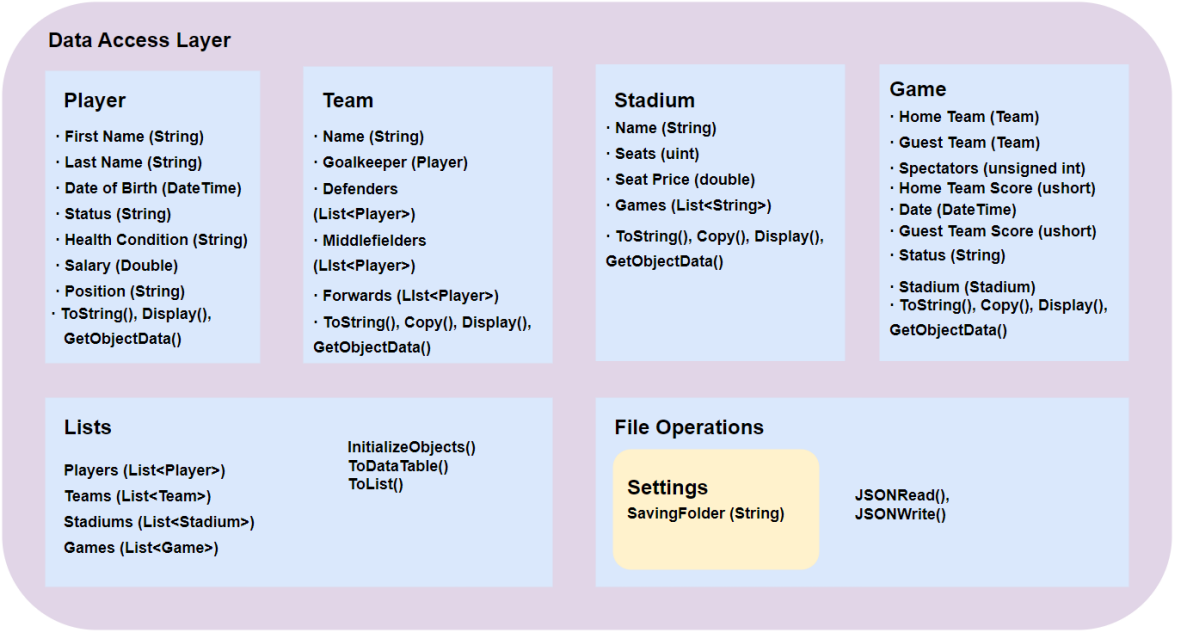
Також в проекті наявні статичні класи **Lists** та **FileOperations**.

Призначення класу **Lists** - зберігати списки з об'єктів, з якими працює програма і "передавати" їх для роботи вище у шари бізнес-логіки та представлення.

Метод **InitializeObjects()** - важлива функція, яка читає списки з файлів JSON і завантажує їх у пам'ять програми. Об'єкти десеріалізуються у списки в Lists і використовуються для роботи.

Клас **FileOperations** включає в себе методи для серіалізації списків з Lists в JSON-файли а також статичний підклас **Settings**, у якого в полі SavingFolder зберігається шлях до директорії, куди вищезгаданні файли будуть зберігатися.

SavingFolder - поле, яке завжди звертається до одного й того ж розташування всередині проекта, щоб з TXT-файлу прочитати шлях до папки, з якої можна прочитати чи в яку можна записати файли. Таким чином програма не “забуває” цю директорію після її закриття.



Шар бізнес-логіки

Всі дії з класами в DAL, окрім відображення інформації чи серіалізації, здійснюються через шар бізнес-логіки. Для кожного класу сутності та Lists тут реалізовані безпосередні дії. Всі класи тут - статичні.

Клас **GameLogic** має методи для додавання, пошуку по командам та даті й заміни гри.

Клас **PlayerLogic** - додавання, пошук та заміна гравця у списку.

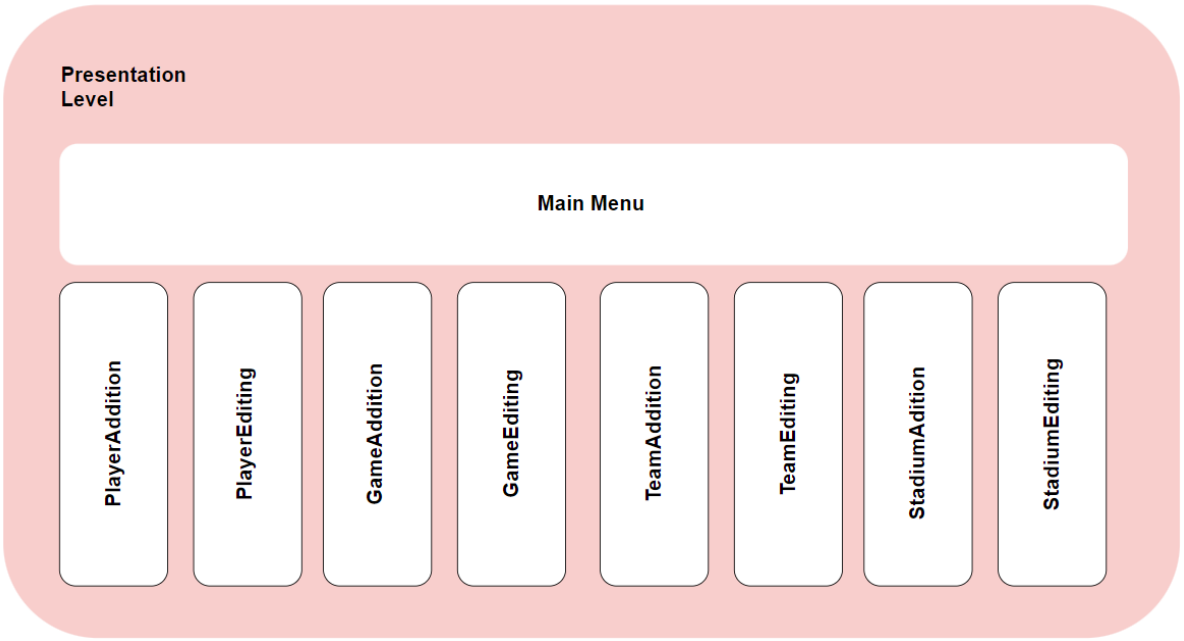
Клас **TeamLogic** - додавання, пошук та заміна команди у списку.

Клас **StadiumLogic** - додавання, пошук та заміна стадіону у списку.

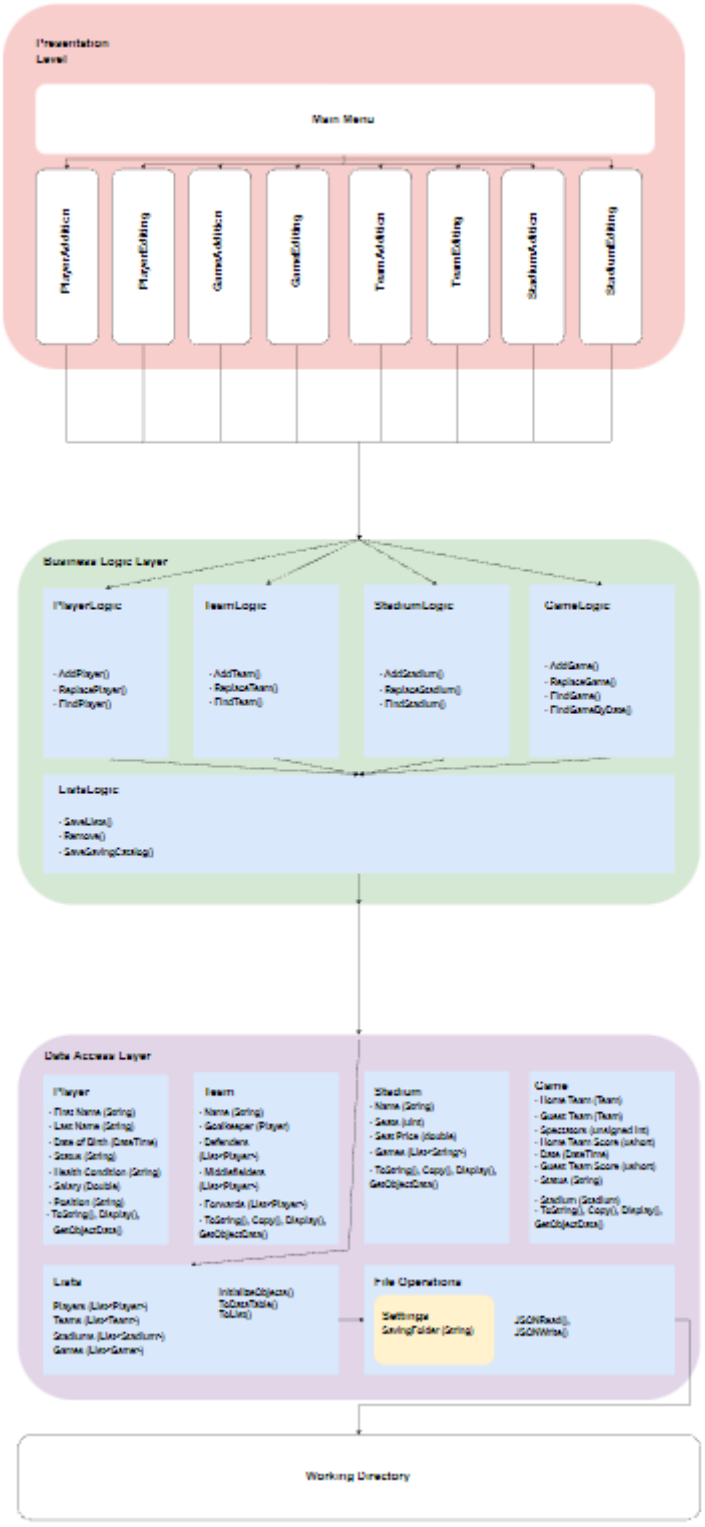
Клас **ListsLogic** має методи для видалення об’єкту зі списку, встановлення каталогу для зберігання та збереження списків у файли через встановлення.

Шар представлення

Шар представлення реалізовано за допомогою програмного забезпечення Windows Forms у вигляді графічного інтерфейсу. Це об’ємний за розміром і кількістю коду шар, оскільки тут здійснюється не лише демонстрація та зчитування введених даних, а ще їх перевірка та обробка виключень.



Діаграма класів і зв’язки



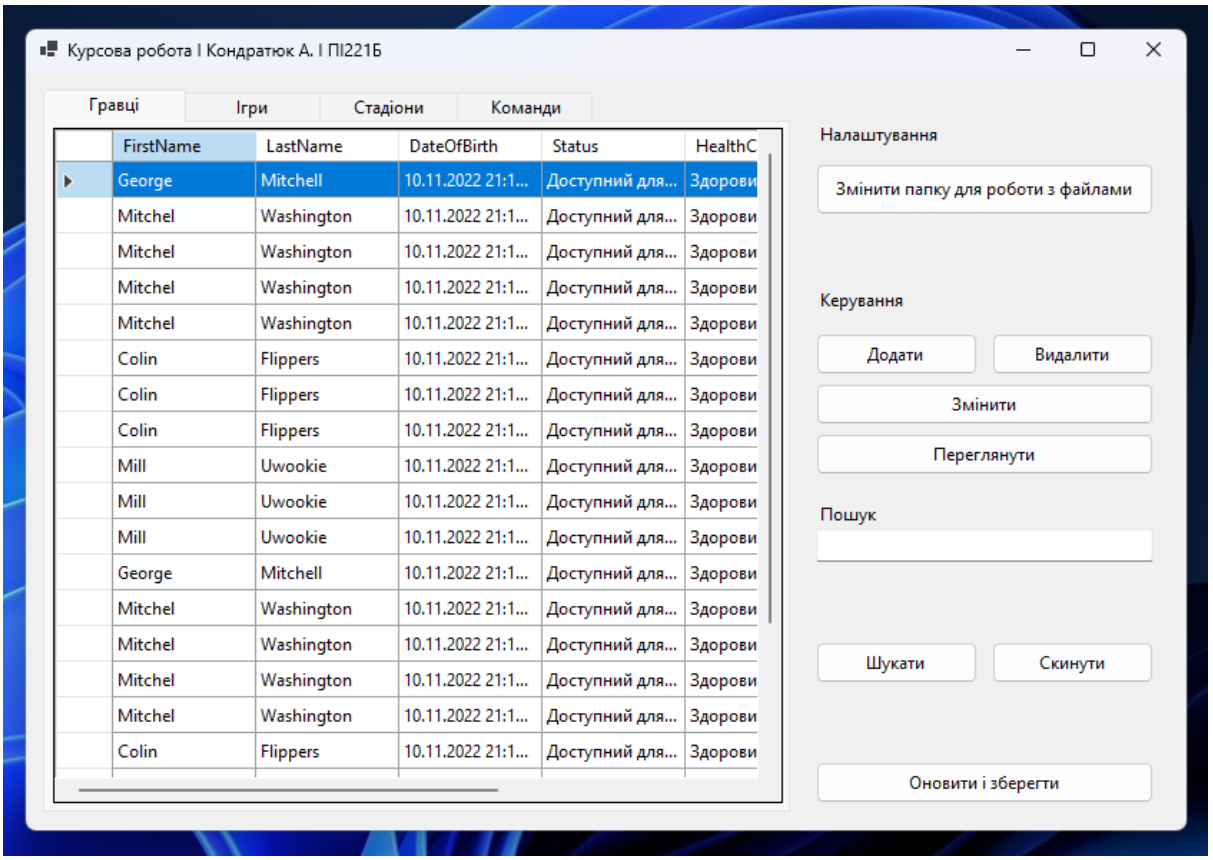
Опис роботи програми

Команди вводяться через елементи керування в головному меню. Ці компоненти відкривають спеціалізовані меню для кожного компоненту, де виконується введення або зміна даних. Ці меню звертаються до шару-бізнес логіки, через методи якого здійснюється управління елементами. Шар бізнес-логіки зв'язаний з шаром доступу до даних, який вносить зміни у початкові списки.

За потреби змінені списки записуються у файли за допомогою FileOperator, який серіалізує і зберігає їх в робочій директорії.

При запуску програми відкривається головне меню (MainMenu), яке завантажує дані з DAL (Lists), які в свою чергу завантажують їх з файлів JSON, або створюють нові пусті списки, у випадку якщо таких файлів немає чи виникають помилки при їх зчитуванні.

Головне меню представляє собою таблицю з боковим меню та вкладками вгорі, які перемикають клас, з яким працюватиме користувач. Всього їх чотири - гравці, ігри, стадіони та команди.



Керування здійснюється через бокові кнопки “Додати”, “Видалити”, “Змінити”, “Переглянути”.

Також вгорі можна вибрати чи змінити папку, в яку зберігатимуться JSON-файли. Програма запам’ятовує це розташування і автоматично їх відкриває, коли запускається.

Нижче розташоване пошукове поле та кнопки “Шукати” та “Скинути”. В залежності від вкладки, згідно завдання, пошук здійснюється за різними параметрами. У випадку гравців це ім’я та прізвище, ігор - назви граючих команд чи дата (з’являється, якщо натиснути на вкладку ігор), стадіону - назва.

В самому низу розташована кнопка “Оновити і зберегти”. Ця кнопка оновлює таблиці і записує їх в файл JSON.

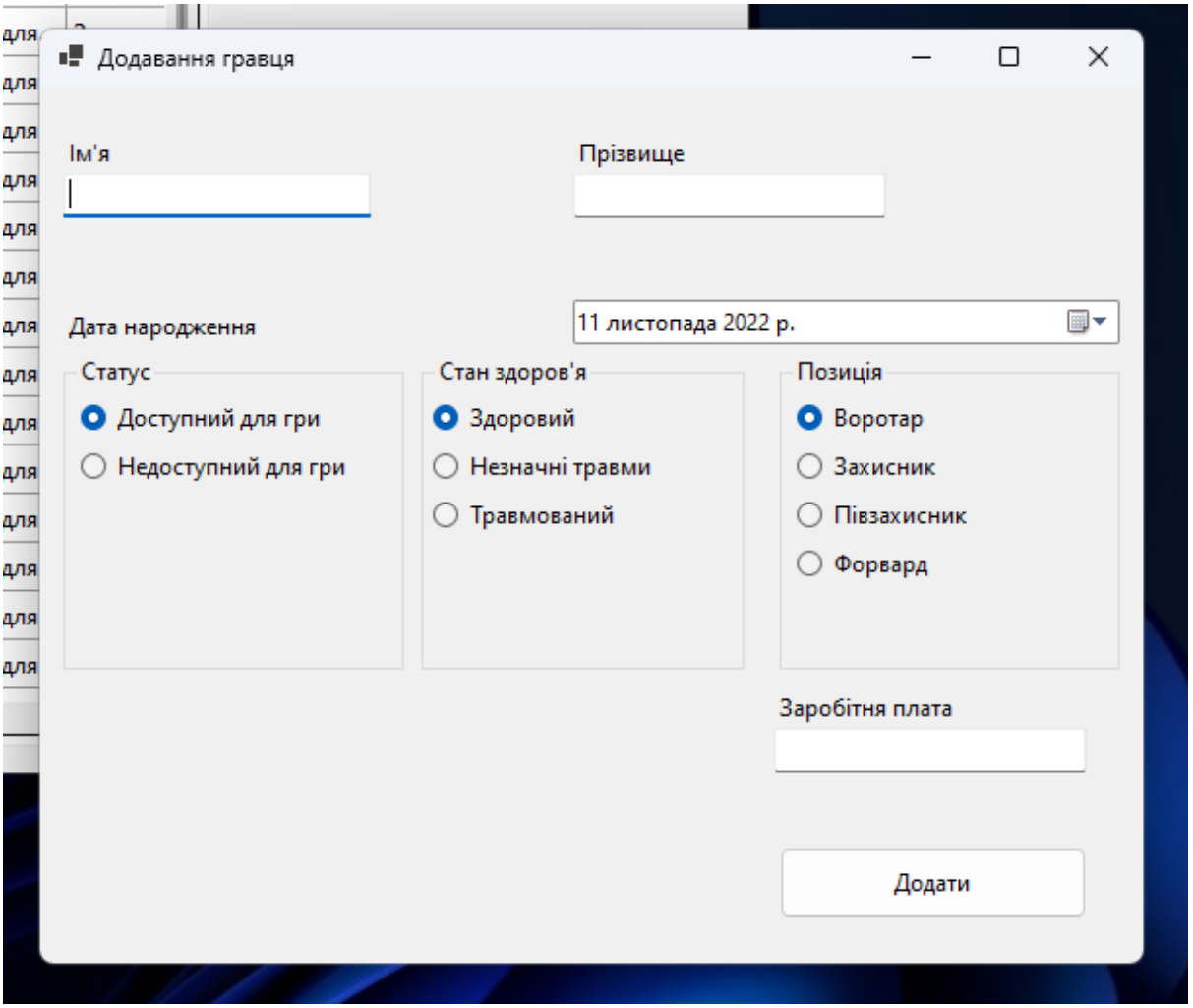
В залежності від відкритої вкладки, кнопки “Додати”, “Видалити”, “Змінити” та “Переглянути” відповідно додають, видаляють, змінюють чи переглядають різні сутності.

Кнопка “Видалити” - видаляє об’єкт зі списку, кнопка “Переглянути” - відображає невелике вікно і викликає метод **Display()** класу. “Додати” викликає вікно створення нового об’єкта певної сутності і додає її у список. “Змінити” - викликає майже ідентичне вікно, але за замовчуванням встановлює значення з виділеного у таблиці об’єкту, який потім замінює на новостворений.

Додавання, заміна, пошук гравця

Вікно додавання гравця пропонує користувачу ввести ім’я, прізвище та зарплату гравця, вибрати його дату народження через DateTimeSelector, а статус - через одну з радіо-кнопок.

Коли користувач натискає на кнопку “Додати”, програма перевіряє кожне введене поле й сигналізує про помилку, якщо вона була допущена. Потім, вона створює об’єкт певного класу (в цьому випадку - гравця) й викликає метод BLL.PlayerLogic.AddPlayer(), щоб додати його у таблицю.

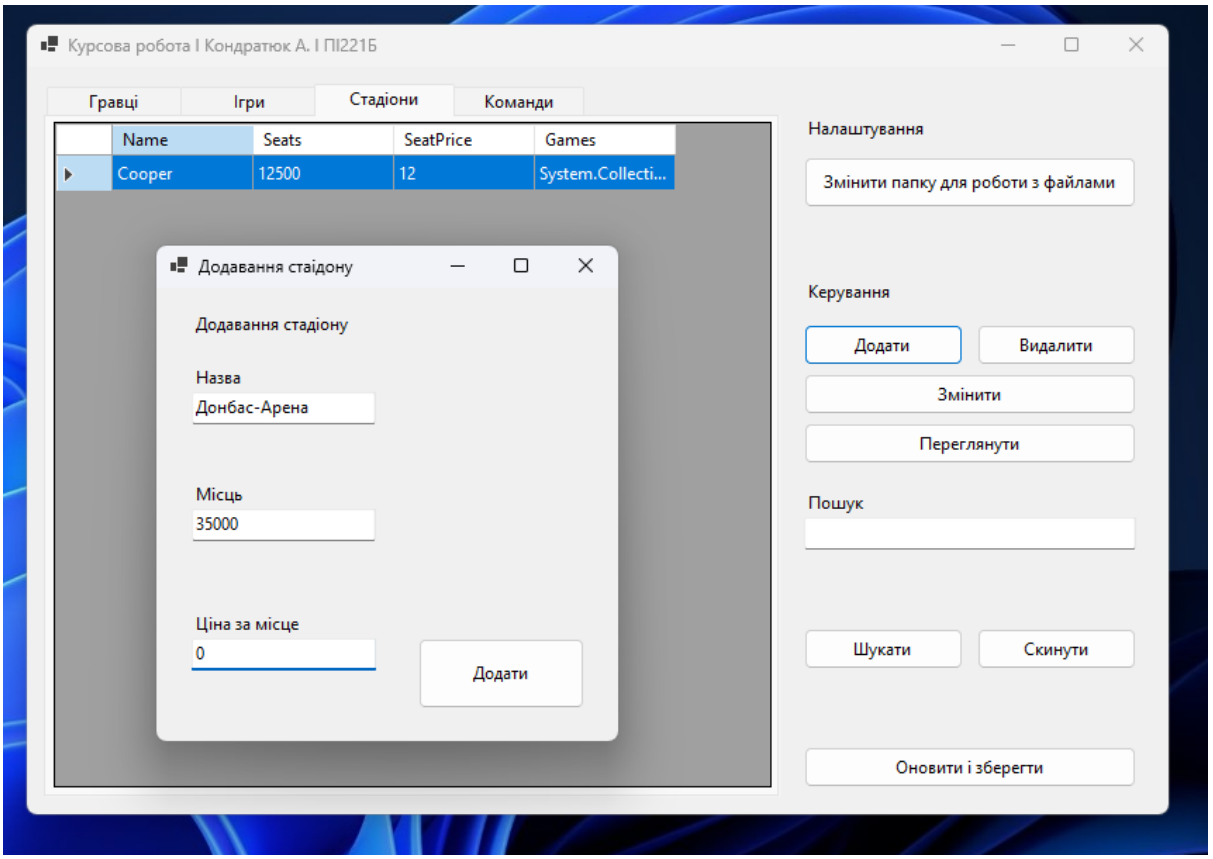


У випадку, коли об’єкт не додається, а змінюється, вікну передається об’єкт з таблиці, з якого воно завантажує значення й поміщає їх як вибрані у полях введення. Після того, як користувач зробив всі необхідні зміни, відбувається майже ідентична процедура перевірки та створення об’єкту. Після створення, тим не менше, об’єкт не додається у список через AddPlayer(), а замінює обраний об’єкт зі списку, використовуючи метод ReplacePlayer() з шару BLL.

Пошук здійснюється в головному меню. В поле вводиться ім’я або прізвище гравця і після натискання на “Пошук” програма робить джерелом даних для таблиці новостворений список гравців, в якому зберігаються лише ті, які відповідають пошуковому запиту. Щоб повернутися до загальної таблиці, потрібно натиснути “Скинути”.

Заміна, пошук, додавання стадіону

Стадіон - один з найпростіших об’єктів, тому його створення та зміна здійснюються у вікні з лише трьома полями. З програмної точки зору процедура майже ідентична до додавання гравця (перевірка правильності введення кожного поля, спроба створити об’єкт і додавання\заміна старого).

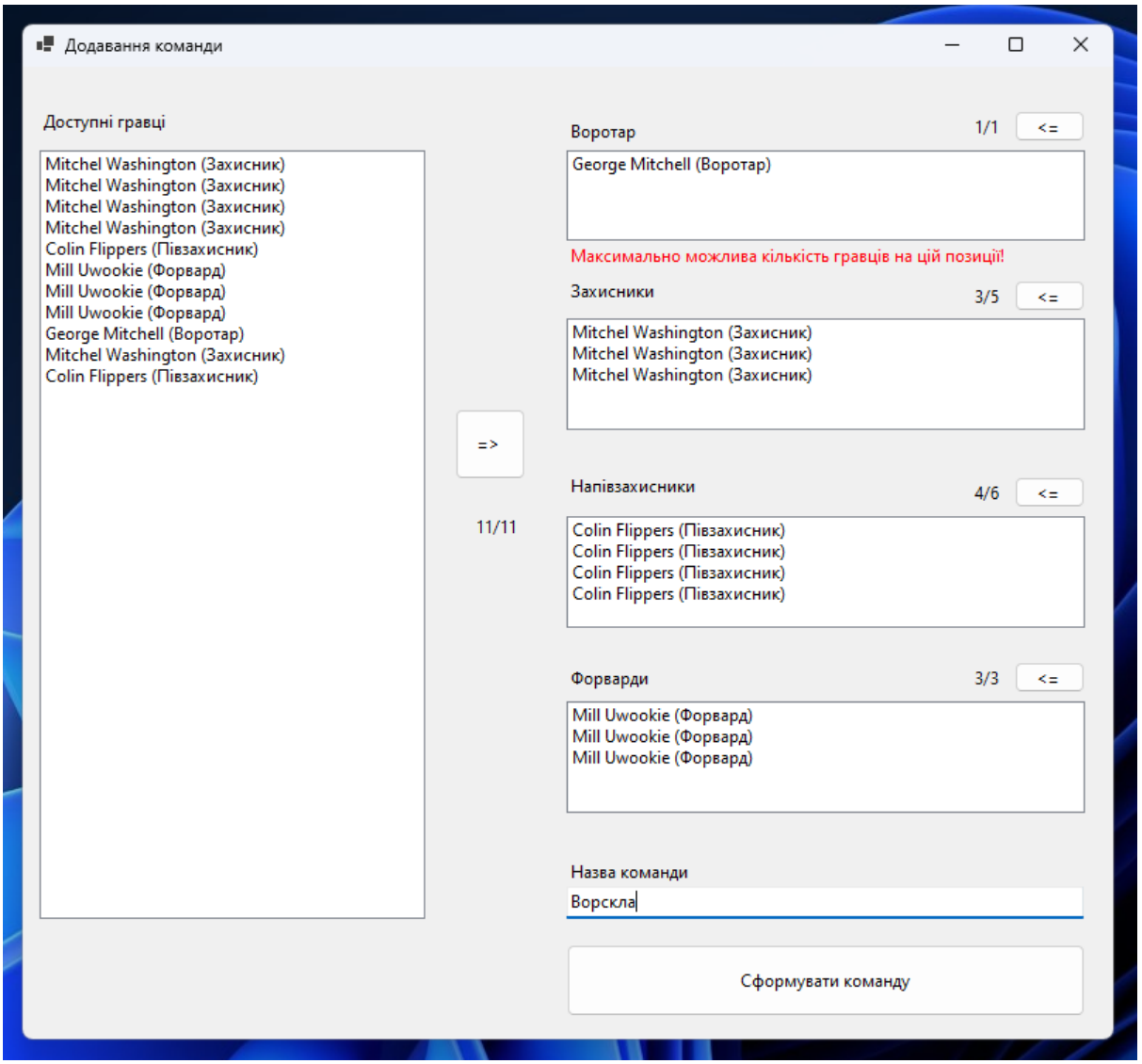


Пошук стадіону здійснюється за його назвою. Механізм пошуку такий самий, як і для пошуку гравця.

Пошук, заміна, додавання команди

Вікно додавання команди має вигляд чотирьох таблиць з кнопками зі стрілками. Справа розташовані доступні для додавання гравці - вони завантажуються зі списку DAL.Lists.Players. Посередині знаходиться кнопка зі стрілкою, що вказує направо і лічильник

гравців у команді знизу. Зправа - три таблиці на кожну позицію у команді. Біля кожної такої таблиці є лічильник гравців на цій позиції і кнопка повернення гравця до лівого списку доступних.



Коли користувач натискає на кнопку посередині, програма аналізує його позицію (вказана в дужках після імені) і переносить його до таблиці з відповідної позиції.

Кожна таблиця має свій ліміт гравців на цій позиції. Програма не дає додати декілька воротарів, більше ніж 5 захисників, 6 півзахисників чи 3 форварди. Але і створити команду з 15 гравців теж не можна - програма перевіряє, щоб гравців було рівно 11.

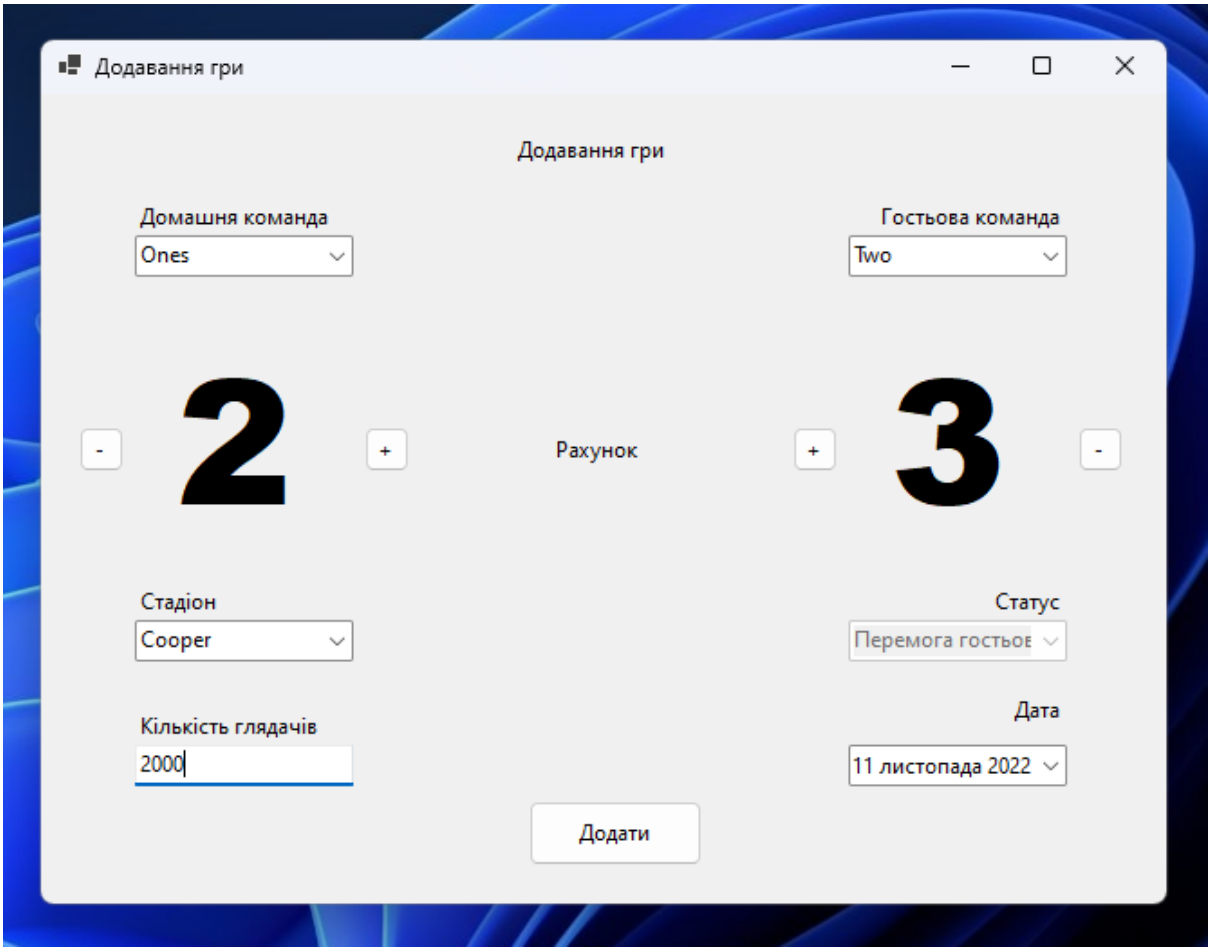
Знизу розташоване поле для введення назви команди та кнопка додавання.

Механізм редагування команди точно такий, як і при редагуванні гравця чи стадіону.

Пошук здійснюється за назвою команди з ідентичним до вищезгаданих класів процесом.

Заміна, додавання, пошук гри

Додавання гри - вікно, де розташовані списки для вибору домашньої та гостьової команди (DAL.Lists.Teams), стадіону (зі списку DAL.Lists.Stadiums) та поле для введення кількості глядачів.



Найбільше впадає в око рахунок і інструменти його редагування. За замовчуванням рахунок встановлений як 0 : 0, і це єдиний рахунок, за якого можна обрати статус гри з-поміж “Ще не проведено” та “Нічия”. При зміні рахунку за допомогою кнопок збоку від цифр

статус встановлюється автоматично (перемога певної команди, нічия).

Зміна виконується за тією ж процедурою, що й зміна інформації про інші об’єкти згаданих вище класів.

Пошук здійснюється за назвою однієї з граючих команд.

Список використаних джерел

1. Microsoft Learn: Build skills that open doors in your career. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/ (дата звернення: 10.11.2022).

2. METANIT.COM - Сайт о программировании. *METANIT.COM - Сайт о программировании*. URL: https://metanit.com/ (дата звернення: 10.11.2022).

3. Flowchart Maker & Online Diagram Software. *Flowchart Maker & Online Diagram Software*. URL: https://app.diagrams.net/ (дата звернення: 10.11.2022).

4. What is Three-Tier Architecture. *IBM - Deutschland | IBM*. URL: https://www.ibm.com/cloud/learn/three-tier-architecture (дата звернення: 10.11.2022).

5. C# Tutorial (C Sharp). *W3Schools Online Web Tutorials*. URL: https://www.w3schools.com/CS/index.php (дата звернення: 10.11.2022).

Лістинг коду проекту

github.com/levndays/Term-Paper-11.22-kondratiuk