МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

КУРСОВА РОБОТА

з дисципліни «Програмування»

на тему

Шаблони проєктування в ООП. Програмне забезпечення комп’ютерної гри «Шашки»

Виконав студент
ІІ курсу групи КП-03
Мишко Роман
Андрійович
залікова книжка КП-0309

Керівник роботи
доцент, к.т.н. Заболотня Т.М.

Оцінка

_____
(дата, підпис)

КИЇВ 2022

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

ПЗ – (Програмне забезпечення) сукупність програм системи оброблення інформації та програмних документів, необхідних для експлуатації цих програм. Це набір інструкцій, які розповідають комп'ютеру, як працювати.

XML – (від англ. Extensible Markup Language**)** стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками.

JSON – (від англ. JavaScript Object Notatio) текстовий формат обміну даними між комп'ютерами. Цей формат використовується переважно для передавання структурованої інформації через мережу (завдяки процесу, що називають серіалізацією).

C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET.

UML – (від англ. Unified Modeling Language) уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML був створений для визначення, візуалізації, проєктування й документування в основному програмних систем.

MVVM – (від англ. Model-View-ViewModel) шаблон проєктування, що застосовується під час проєктування архітектури застосунків (додатків). MVVM орієнтований на такі сучасні платформи розробки, як Windows Presentation Foundation та Silverlight від компанії Microsoft.

ВСТУП

Дана курсова робота присвячена розробці програмного забезпечення комп’ютерної гри «Шашки» за допомогою використання шаблонів проєктування. Комп’ютерні ігри є невід’ємною частиною дозвілля більшості людей у всьому світі. Настільки гра “Шашки” є одна з найбільш популярних інтелектуальних настільних ігор. Тому розроблене програмне забезпечення є дійсно актуальним, оскільки є тільки електронною копією популярної настільної гри. Дана тематика обрана для виконання курсової роботи тому, що результати абстрагування об’єктів у цій предметній галузі дозволяють застосувати вивчені принципи та методи об’єктно-орієнтованого програмування для створення програмного забезпечення, зокрема шаблони проєктування.

Об’єктом дослідження є процес гри в комп’ютерну симуляцію «Шашки».

Метою роботи є розроблення програмного забезпечення комп’ютерної гри «Шашки» з використанням шаблонів проєктування.

Для досягнення визначеної мети необхідно виконати такі *завдання*:

- абстрагувати об’єкти предметної галузі;
- розробити структурну організацію ПЗ за допомогою застосування основних принципів ООП та шаблонів проєктування;
- визначити та описати функціональні характеристики програми;
- обґрунтувати вибір шаблонів проєктування, використаних для побудови програми;
- розробити користувацький графічний інтерфейс;
- виконати реалізацію програмного забезпечення відповідно до вимог технічного завдання;
- виконати тестування розробленої програми;
- оформити документацію з курсової роботи.

Розроблене ПЗ комп’ютерної гри «Шашки» складається з таких модулів: модуль графічних інтерфейсів, модуль взаємодії користувача з графічними інтерфейсами, модуль основної бізнес-логіки, модуль сервісів, таких як: логування, репозиторій, стилі та команди.

Реалізовані шаблони проєктування: Команда, Стратегія, Міст, Шаблонний метод, MVVM.

До функціональних можливостей програми належать: перевірки коректності вхідних даних, експортування даних турнірної таблиці, механіка гри та робота штучного інтелекту.

Для функціонування розробленої програми необхідно забезпечити наявність на комп’ютері 25 Мб вільного дискового простору та встановленого .Net Framework 5.0.

Розроблене програмне забезпечення може бути використане людьми, які є шанувальниками даної настільної гри з метою змагання між собою в турнірній таблиці.

Пояснювальна записка складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел. Робота містить 19 рисунки. Загальний обсяг роботи – 30 друкованих сторінок, з них 24 сторінка основного тексту та 1 сторінка списку використаних джерел.

1. ОПИС СТРУКТУРНО-ЛОГІЧНОЇ СХЕМИ ПРОГРАМИ

1.1. Модульна організація програми

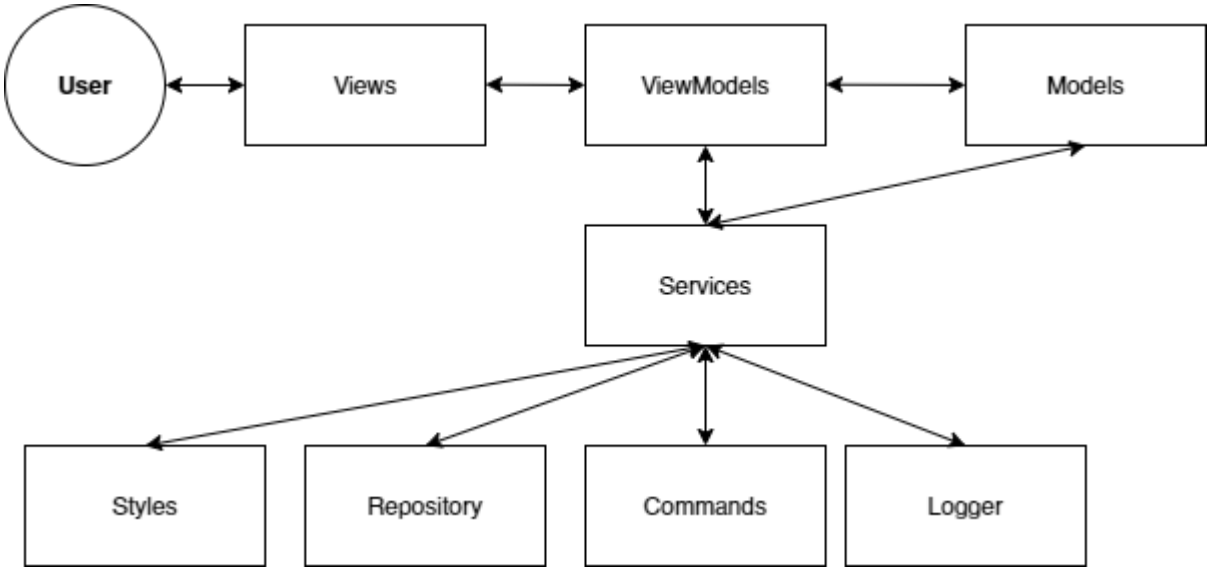

Рис. 1.1.1. Модульна організація програми

Модуль "Models" є модулем логіки програмного забезпечення і реалізує головні функції додатку, містить посилання на важливі елементи та інші модулі, здійснює передачу даних та запитів. Даний модуль містить об'єкти.

Модуль "ViewModels" здійснює передачу даних та запитів, є модулем взаємодії користувача з графічним інтерфесом та прехідною ланкою до основної бізнес-логіки в "Models".

Модуль "Views" дозволяє взаємодіяти користувачу з програмою за допомогою графічних інтерфейсів.

Модуль "Services" складається з наступних підмодулів:

1) **Модуль "Styles"** забезпечує реалізацію стилів Base, Green, Pacific.
2) **Модуль "Repository"** реалізує доступ до json та xml файлів з метою читання та редагування даних, експортування даних турнірної таблиці.
3) **Модуль "Commands"** містить в собі групу класів, які реалізуть інтерфейс ICommand для подальшого застосування в "ViewModels".
4) **Модуль "Logger"** забезпечує логування помилок та повідомлень в файлову систему комп'ютера, а також в консоль.

1.2. Функціональні характеристики

При запуску програми створюється об’єкт головного вікна, що містить 4 кнопки: Play, Stats, Styles, Exit.

При натисканні на кнопку Play, створиться об’єкт вікна InputForm, де потрібно користувачеві ввести свій нікнейм.

При натисканні на кнопку OK, створиться новий об’єкт вікна, а саме GameWithAIWindow, що при ініціалізації створить дошку та заповнить її шашками. На вікні є вивід загального часу, кількість очок та шашок гравців, а також таблиця і попередніми ходами та їхнім часом. На вікні також присутня кнопка повернення до головного меню. Після виграшу одного з гравців (користувача чи штучного інтелекту) буде показано повідомлення, час гри зупиниться і дані збережуться в файл, для подальшого виводу в турнірній таблиці.

При натисканні на кнопку Stats, створиться новий об’єкт вікна StatisticsWindow, яке міститиме турнірну таблицю та 3 кнопки (Json, Xml, MainMenu). У даному вікні користувач має можливість експортувати дані турнірної таблиці в форматі JSON та XML.

При натисканні на кнопку Styles, дизайн вікон головного меню і статистики буде змінюватися відповідно до реалізованих стилів в модулі “Styles”. Наразі реалізовані такі стилі: Base, Green, Pacific.

2. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ЗА ДОПОМОГОЮ ШАБЛОНІВ ПРОЄКТУВАННЯ

2.1 Обґрунтування вибору та опис шаблонів проєктування для програмного забезпечення комп'ютерної гри «Танцювальна студія»

1. «MVVM»

MVVM використовується для відокремлення моделі та її відображення. Необхідністю цього є надання можливості змінювати їх незалежно одну від одної.

Шаблон MVVM ділиться на три частини:

a) Модель (Model) являє собою фундаментальні дані, що необхідні для роботи застосунку.

b) Вигляд (View) — це графічний інтерфейс, тобто вікно, кнопки тощо.

c) Модель вигляду з одного боку є абстракцією Вигляду, а з іншого надає обгортку даних з Моделі, які мають зв'язуватись. Тобто вона містить Модель, яка перетворена до Вигляду, а також містить у собі команди, якими може скористатися Вигляд для впливу на Модель. ViewModel призначена для того, щоб

1. Здійснювати зв'язок між моделлю та вікном
2. Відслідковувати зміни в даних, що зроблені користувачем
3. Відпрацьовувати логіку роботи View (механізм команд)

Обґрунтування використання шаблону:

У грі "Шашки" повинні бути реалізовані модулі графічних інтефейсів, що в даному шаблоні буде представлятися за домопогою Views та бізнес-логіки, яка буде представлятися Modules. Оскільки необхідно поєднати два модулі, графічних інтерфейсів та логіки, MVVM надає можливість реалізувати дану задачу за допомогою ViewModels.

Учасники шаблону:

- **Views** - містить усі графічні інтерфейси, що реалізовані через xaml: MainWindow, InputForm, GameWithAIWindow, StatisticsWindow,

WindowBase, App.

- **Models** - містить класи, що описують бізнес-логіку програми: User, Piece, Move, CheckerAI, CheckerBoard, BoardRwCol, GameWithAIModel, HistoryMove, IGameWithAIModel, PreloaderModel, StatisticsModel, Extentions.
- **ViewModels** - складовий компонент, що містить класи, які пов’язують графічні інтерфейси та бізнес-логіку за допомогою команд. Класи даного модуля: ViewModelBase, PreloaderViewModel, InputFormViewModel, GameWithAIViewModel, StatisticsViewModel.

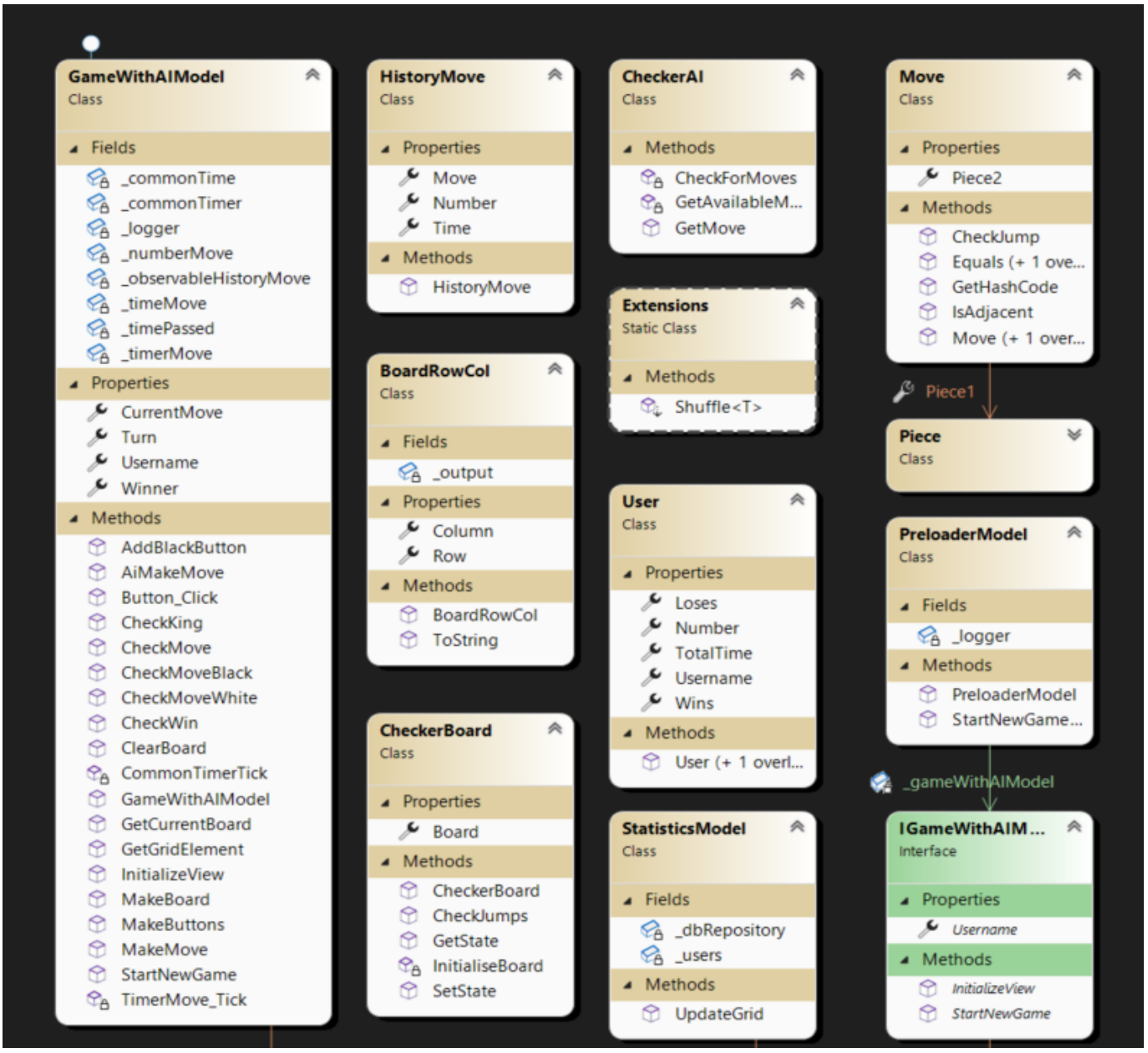

Рис. 2.1.1, аркуш 1. UML діаграма класів, які входять до шаблону «MVVM»

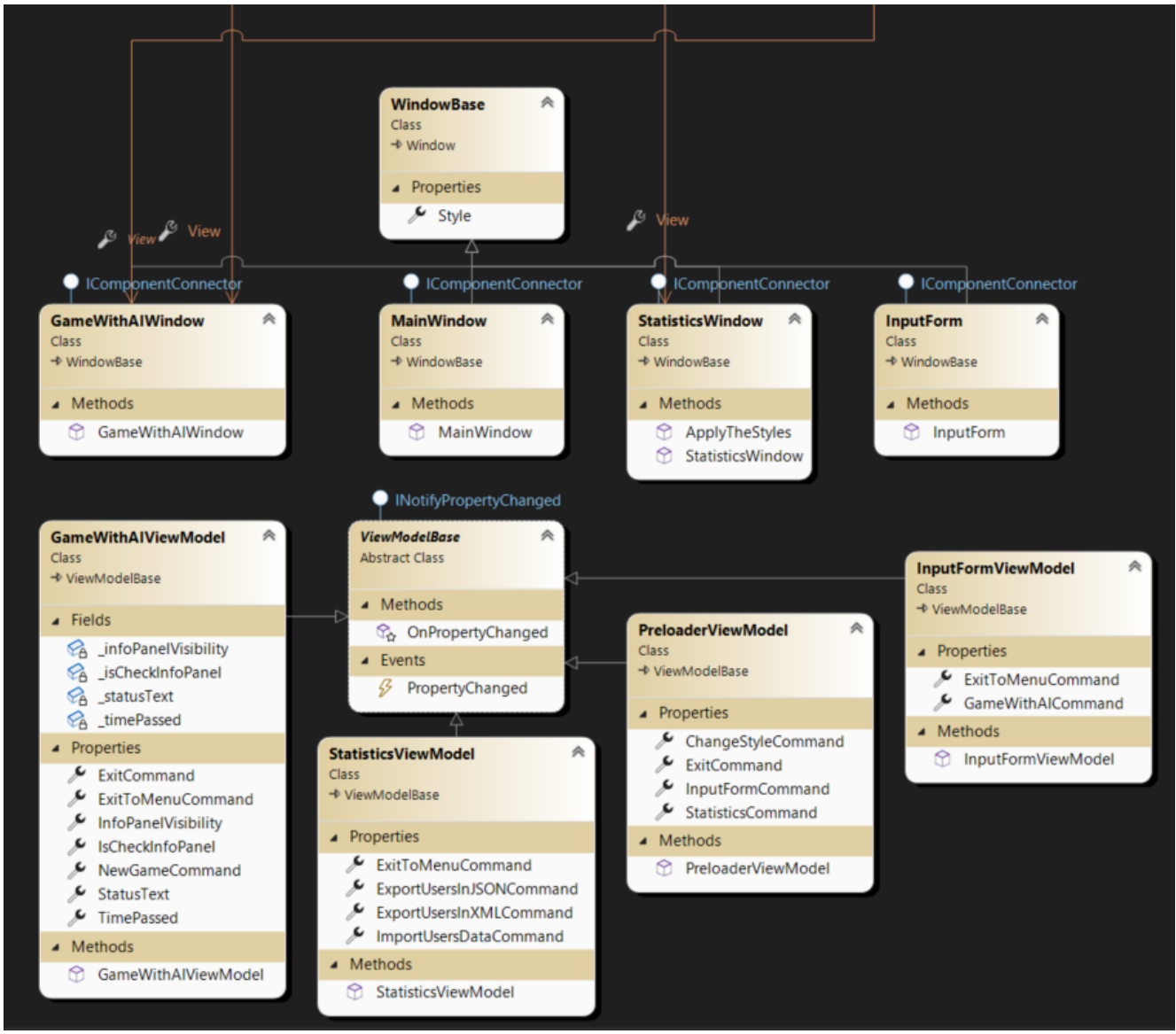

Рис. 2.1.1, аркуш 2. UML діаграма класів, які входять до шаблону «MVVM»

2. «Команда»

Це поведінковий патерн проектування, який перетворює запити на об’єкти, дозволяючи передавати їх як аргументи під час виклику методів, ставити запити в чергу, логувати їх, а також підтримувати скасування операцій. Шаблон «Команда» прибирає пряму залежність між об’єктами, що викликають операції, та об’єктами, які їх безпосередньо виконують. UML діаграма класів представлена на рис. 2.1.1.

Обґрунтування використання шаблону:

Шаблон тісно пов’язаний з попереднім шаблоном “MVVM”, оскільки команди що надходять від користувача, повинні бути прив’язані до певного класу ViewModel, який повинен містити властивість типу ICommand. Таким чином, шаблон створює певну структуру програми, у вигляді користувацького інтерфейсу та шарів бізнес логіки. Класи об’єднуються під загальним інтерфейсом, оскільки мають єдиний метод запуску: Execute. Одні й ті самі відправники можуть працювати з різними командами, не знаючи про їхню реалізацію.

Учасники шаблону:

- **ICommand** – описує інтерфейс, спільний для всіх конкретних команд. Всередині описано метод, який запускає конкретні команди - Execute, метод, який визначає, чи можемо виконати команду - CanExecute та хендлер подій - CanExecuteChanged.
- **ConcreteCommands** – конкретні команди реалізують різни запити, дотримуючись загального інтерфейсу: ChangeStyleCommand, ExitCommand, ExitToMenuCommand, ExportUsersInJSONCommand, ExportUsersInXMLCommand, GameWithAICommand, ImportUsersDataCommand, InputFormCommand, NewGameCommand, StatisticsCommand

- **Client** – модуль “ViewModels” створює об’єкти конкретних команд. Класи даного модуля розміщуються в DataContext класів модуля “Views”.
- **Invoker** - користувач, що натискає на кнопку.
- **Receiver -** модуль “Modules” містить бізнес-логіку програми.

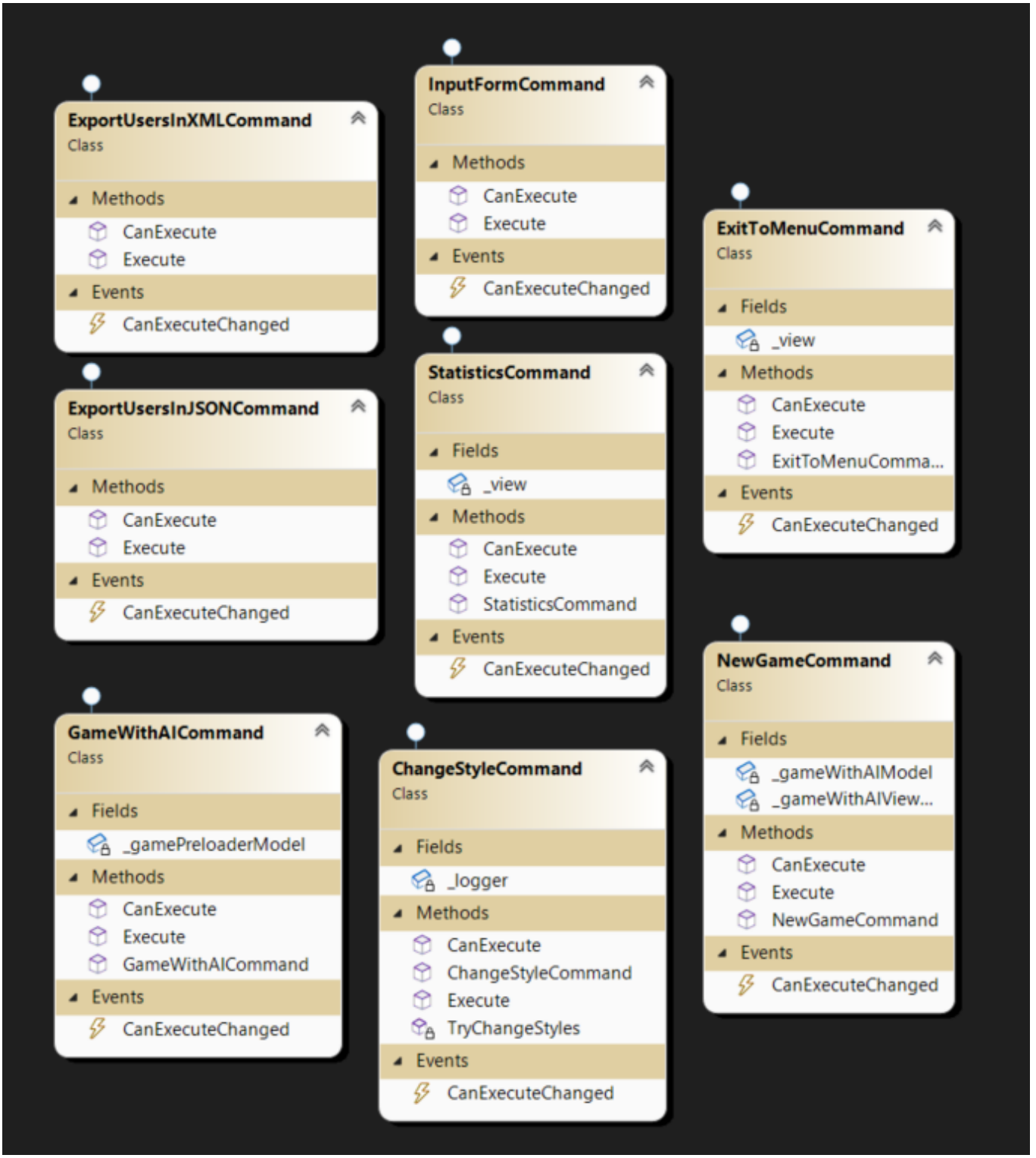

Рис. 2.1.2., аркуш 1. UML діаграма класів, які входять до шаблону «Команда»

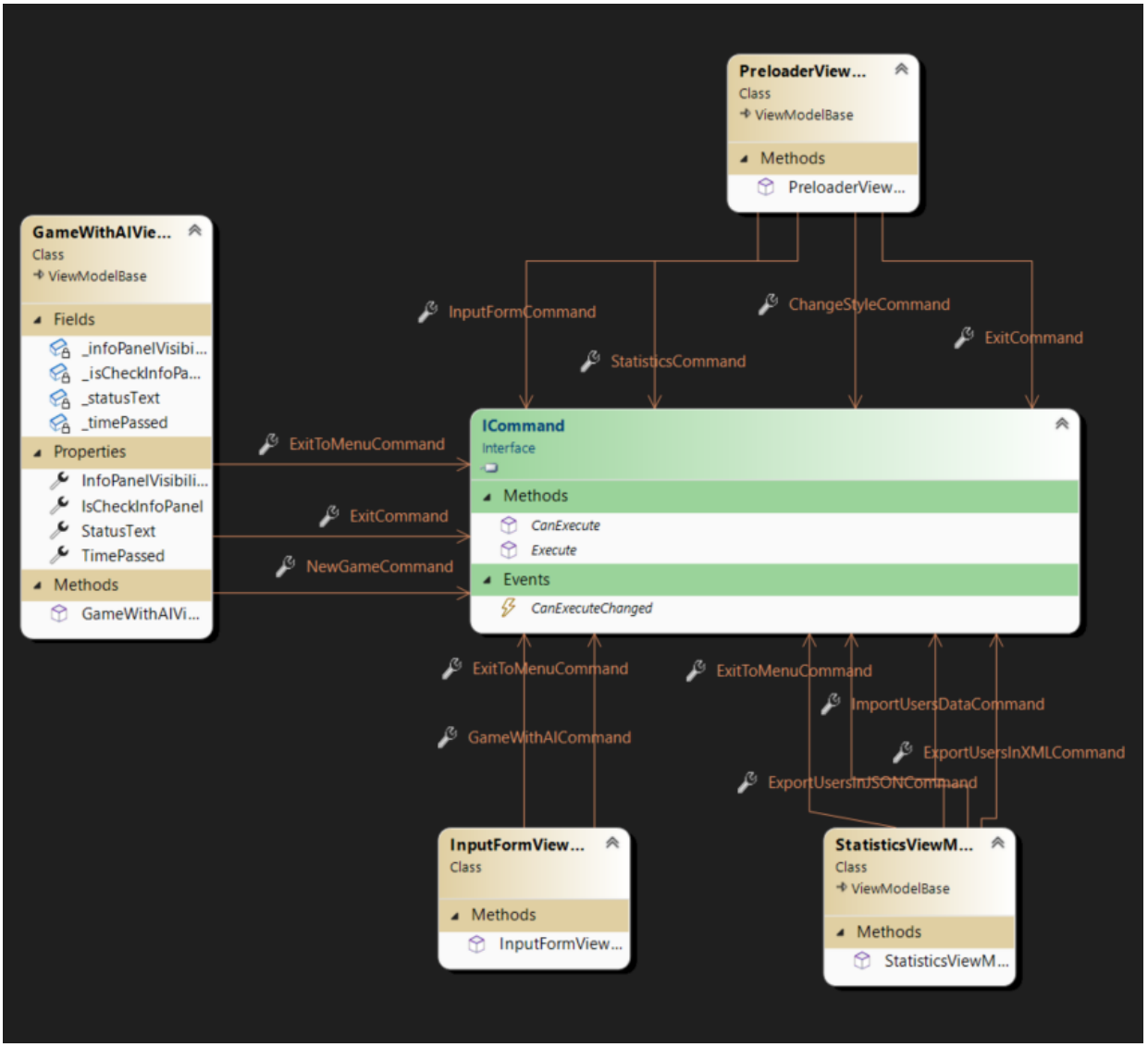

Рис. 2.1.2, аркуш 2. UML діаграма класів, які входять до шаблону «Команда»

3. «Стратегія»

Стратегія — це поведінковий патерн проектування, який визначає сімейство схожих алгоритмів і розміщує кожен з них у власному класі. Після цього алгоритми можна заміняти один на інший прямо під час виконання програми.

Введення інтерфейсу дозволяє класам-клієнтам нічого не знати про класи, що реалізують цей інтерфейс і інкапсулюють в собі конкретні алгоритми. UML діаграма класів представлена на рис. 2.1.2.

Обґрунтування використання шаблону:

Оскільки модуль логування має реалізовувати різні види запису повідомлень, а саме вивід в консоль та запис в файловий документ,, то доцільно було використати поведінковий шаблон проєктування «Стратегія», який дозволяє змінювати один алгоритм виконання на інший прямо під час роботи програми.

Учасники шаблону:

- **Client** - створює об’єкт конкретної стратегії та передає його до конструктора Context.
- **Context -** клас Logger зберігає посилання на об’єкт конкретної стратегії, працюючи з ним через загальний інтерфейс стратегій. Під час виконання програми об’єкт класу Contex отримує виклики від клієнта й делегує їх об’єкту конкретної стратегії.
- **Strategy** - визначає інтерфейс ILogType, спільний для всіх варіацій алгоритму. Context використовує цей інтерфейс для виклику алгоритму.
- **ConcreteStrategies** - FileLog, AuditLog реалізують різні варіації алгоритму.

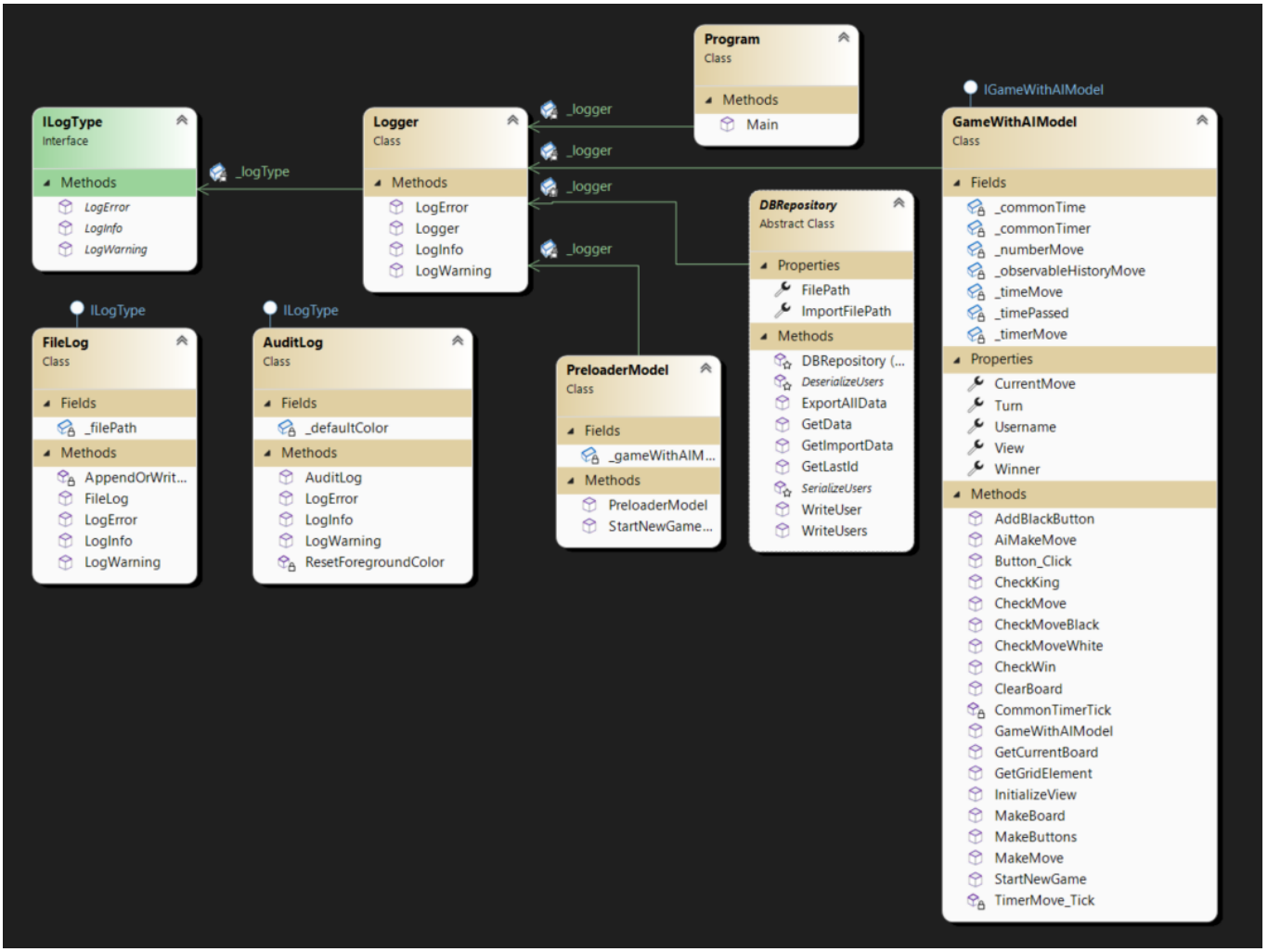

Рис. 2.1.3. UML діаграма класів, які входять до шаблону «Стратегія»

4. «Шаблонний метод»

Це поведінковий патерн проектування, який визначає кістяк алгоритму, перекладаючи відповідальність за деякі його кроки на підкласи. Патерн дозволяє підкласам перевизначати кроки алгоритму, не змінюючи його загальної структури. UML діаграма класів представлена на рис. 2.1.4.

Обґрунтування використання шаблону:

У грі присутня рейтингова таблиця гравців, з якої користувачі мають змогу вивантажити дані у різних форматах, таких як JSON та XML. Алгоритм експортування даних відрізняється лише частиною, де відбувається серіалізація чи десеріалізація. Саме тому, було вирішено реалізувати даний шаблон проектування.

Учасники шаблону:

- **Client** - створює об’єкт конкретного класу шаблонного метода та викликає функцію експортування.
- **AbstractClass** - визначає кроки алгоритму й містить шаблонний метод, що складається з викликів цих кроків. Кроки можуть бути як абстрактними, так і містити реалізацію за замовчуванням. DBRepository надає спільний алгоритм експортування даних: відбувається читання даних з файлу (БД) в об’єкти User, для подальшої сереалізації та створення файлу у вибраній користувачем директиві .
- **ConcreteClasses -** перевизначають деякі або всі кроки алгоритму. Конкретні класи не перевизначають сам шаблонний метод. Xml та Json наслідуються від абстрактного класу DBRepository та перевизначають Serialize та Deserialize етапи в алгоритмі.

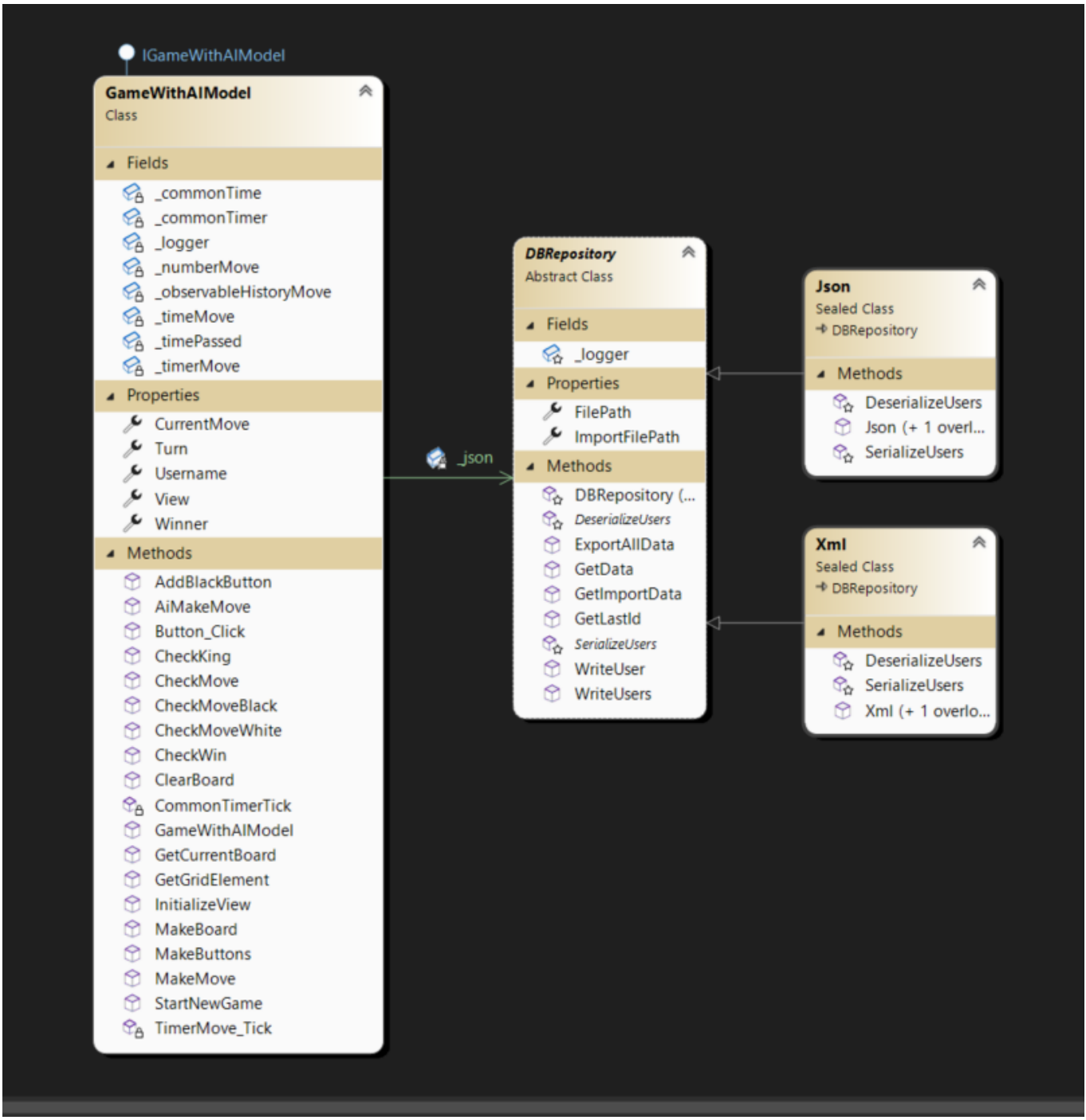

Рис. 2.1.4. UML діаграма класів, які входять до шаблону «Шаблонний метод»

5. «Міст»

Це структурний патерн проектування, який розділяє один або кілька класів на дві окремі ієрархії — абстракцію та реалізацію, дозволяючи змінювати код в одній гілці класів, незалежно від іншої. UML діаграма класів представлена на рис. 2.1.5.

Обґрунтування використання шаблону:

У грі присутня функція зміну стилів вікон. При додаванні нових вікон та нових стилів, кількість комбінацій зростатиме в геометричній прогресії. Шаблон проектування “Міст” виділяє усі стилі та вікна в окремі ієрархії. Вікна лише зберігають необхідний об’єкт стилю всередині своїх класів. Таким чином, ми позбуваємось розростання дерева класів, тому шаблон було використано доцільно.

Учасники шаблону:

- **Client** - працює тільки з об’єктами абстракції. Не рахуючи початкового зв’язування абстракції з однією із реалізацій, клієнтський код не має прямого доступу до об’єктів реалізації.
- **Abstraction -** WindowBase містить керуючу логіку. Код абстракції делегує реальну роботу пов’язаному об’єктові реалізації.
- **Implementation -** IStyle описує загальний інтерфейс для всіх реалізацій. Всі методи, які тут описані, будуть доступні з класу абстракції та його підкласів. Інтерфейси абстракції та реалізації можуть або збігатися, або бути абсолютно різними. Проте, зазвичай в реалізації живуть базові операції, на яких будуються складні операції абстракції.
- **RefinedAbstractions** - GameWithAIWindow, MainWindow, StatisticsWindow, InputForm містять різні варіації керуючої логіки. Як і батьківский клас, працює з реалізаціями тільки через загальний інтерфейс реалізацій.

- **ConcreteImplementations** - BasePaletteStyle, GreenPaletteStyle, PasaficPaletteStyle містять платформо-залежний код.

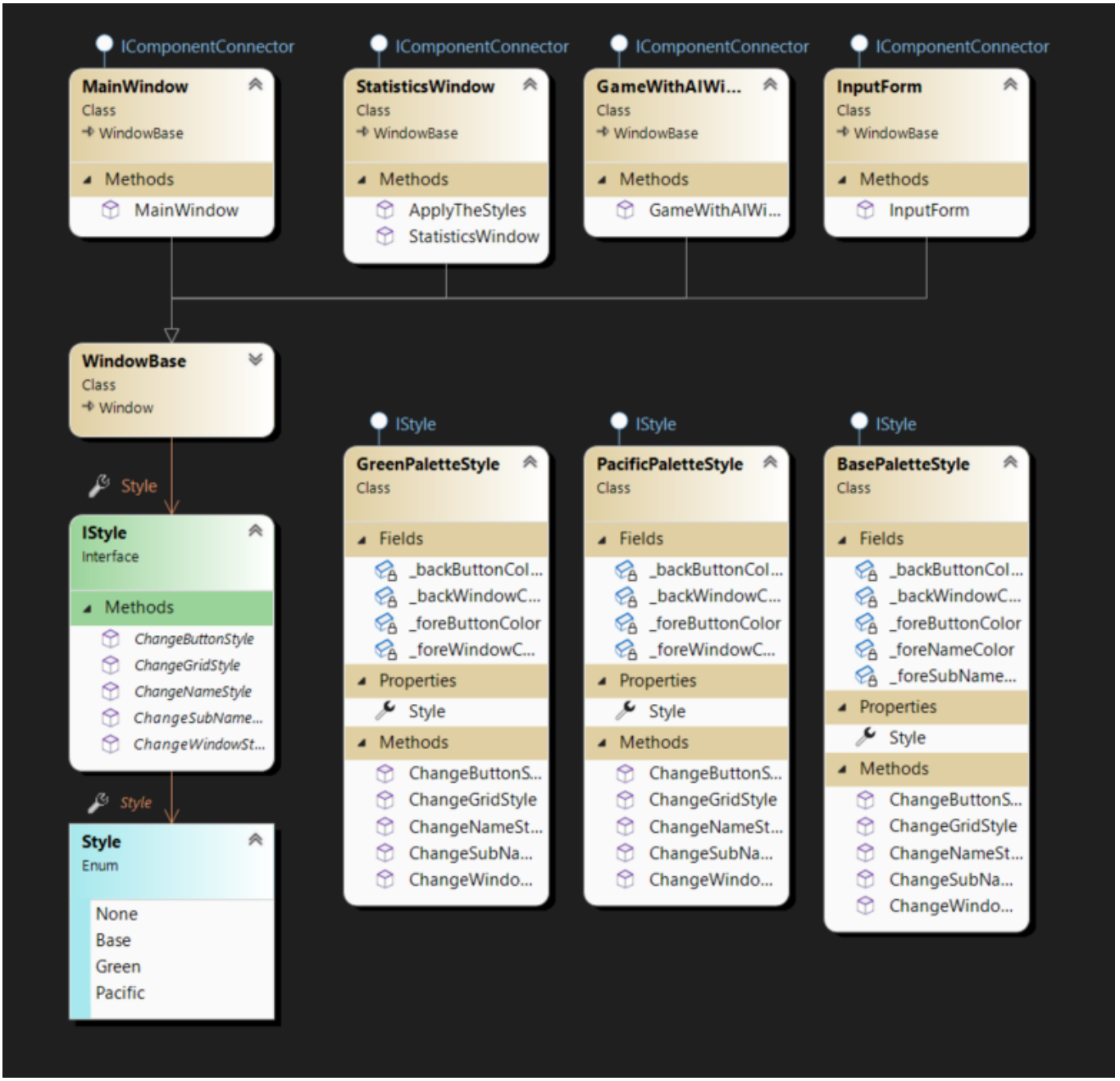

Рис. 2.1.5. UML діаграма класів, які входять до шаблону «Міст»

2.2 Діаграма класів

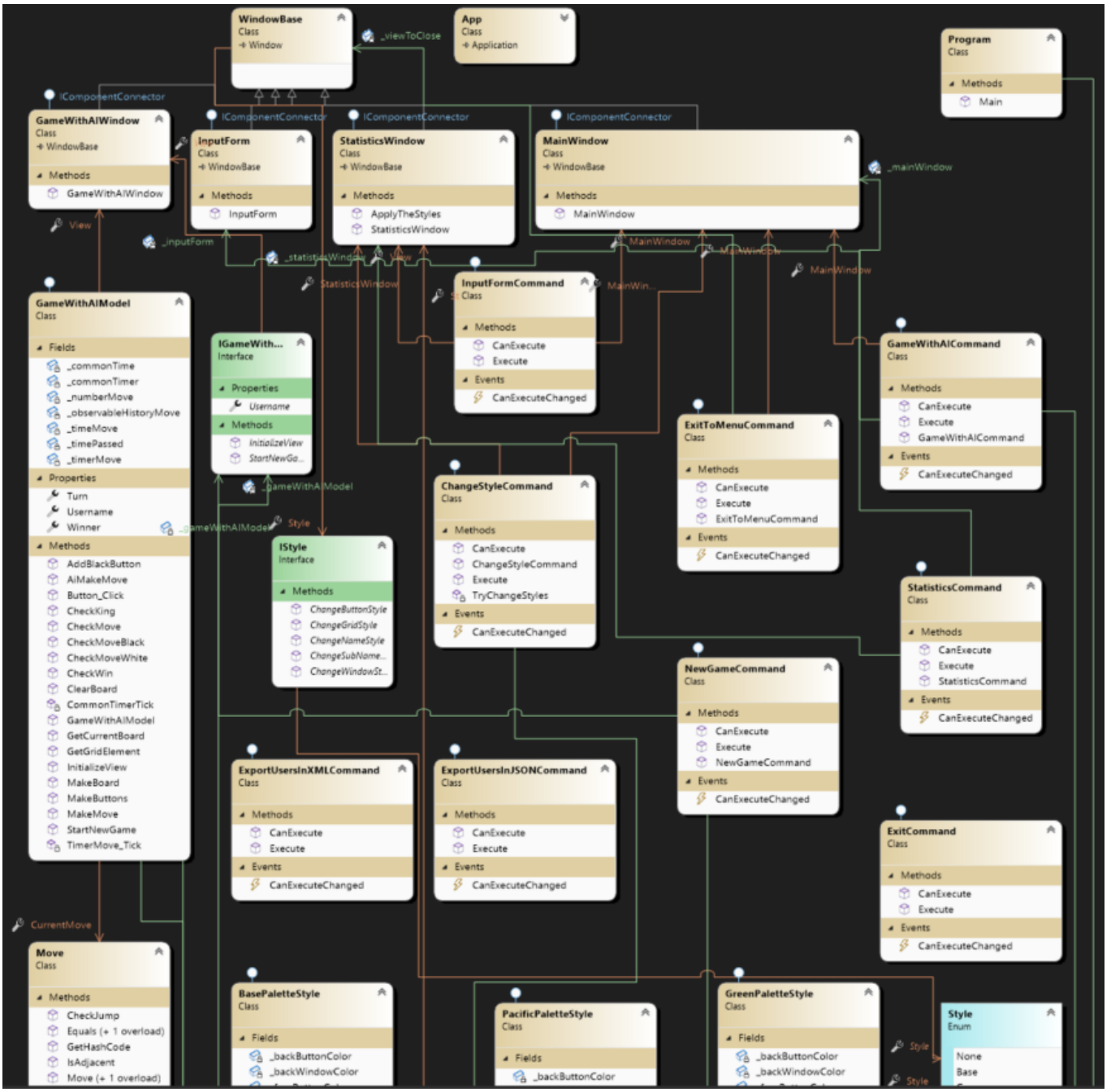

Рис. 2.2, аркуш 1.UML діаграма класів комп'ютерної гри «Шашки»

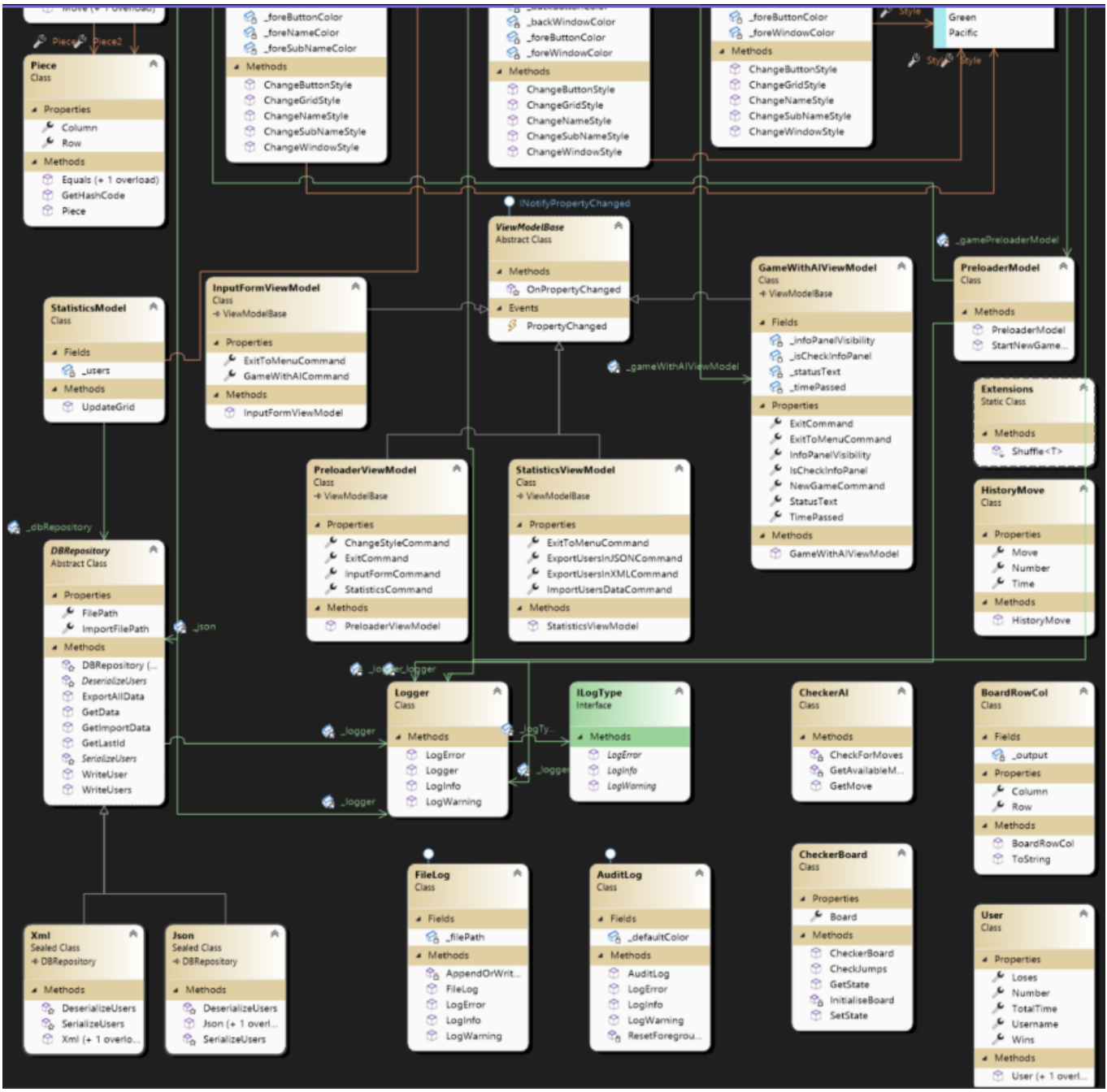

Рис. 2.2, аркуш 2.UML діаграма класів комп'ютерної гри «Шашки»

2.3 Опис результатів роботи програми

Для демонстрування роботи розробленої комп’ютерної гри «Шашки» було розроблено графічний інтерфейс користувача:

При запуску програми користувач попадає в загальне меню, де має наступні 4 опції: розпочати гру “Play”, переглянути статистику (рейтингову таблицю) “Stats”, змінити стиль вікон (меню та статистики) “Styles” та вийти зі гри “Exit”.

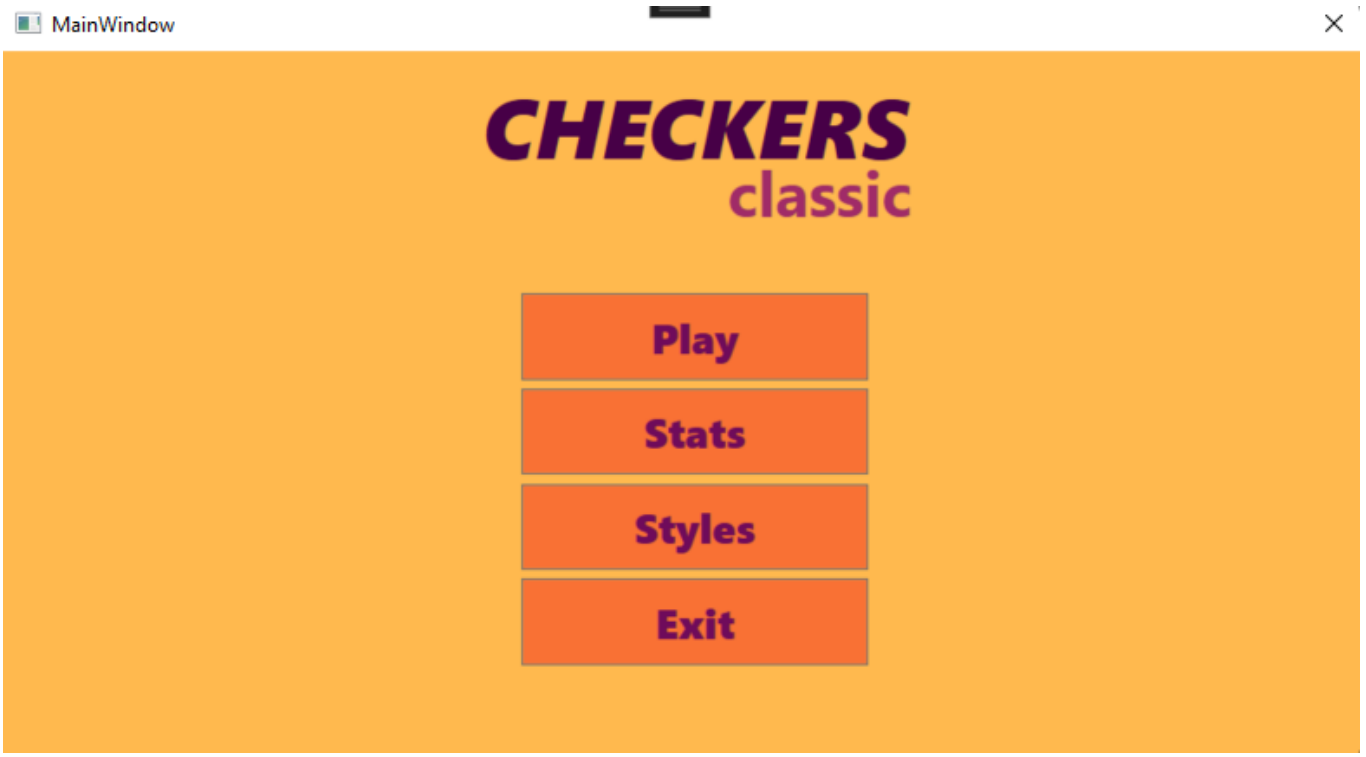

Рис. 2.3.1 Ілюстрація головного вікна при запуску програми

При натисканні на копку “Play” користувачу буде необхідно вказати свій нікнейм перед початком гри, або ж користувач матиме змогу вернутися в загальне меню.

Рис. 2.3.2 Ілюстрація форми вводу нікнейма гравця

Після натискання на кнопку “OK” гравець попадає у вікно “Game with AI”, де він розпочинає гру проти простого штучного інтелекту. На графічному інтерфейсі розміщується дошка із фігурами, загальним часом гри, кількість набраних очок гравцями, кількісь шашок, що залишилися в гравців, кнопка “Main menu” та таблиця із історією ходів та часом ходів.

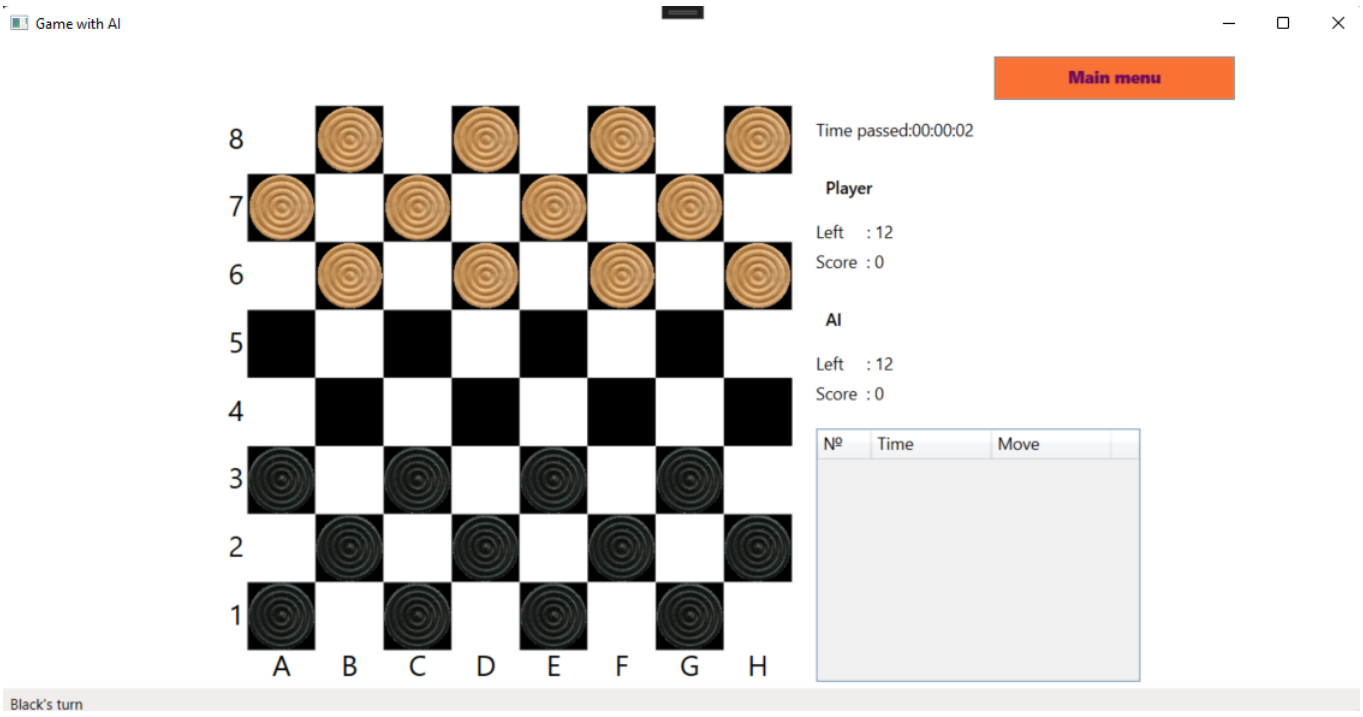

Рис. 2.3.3 Ілюстрація вікна “Game with AI”

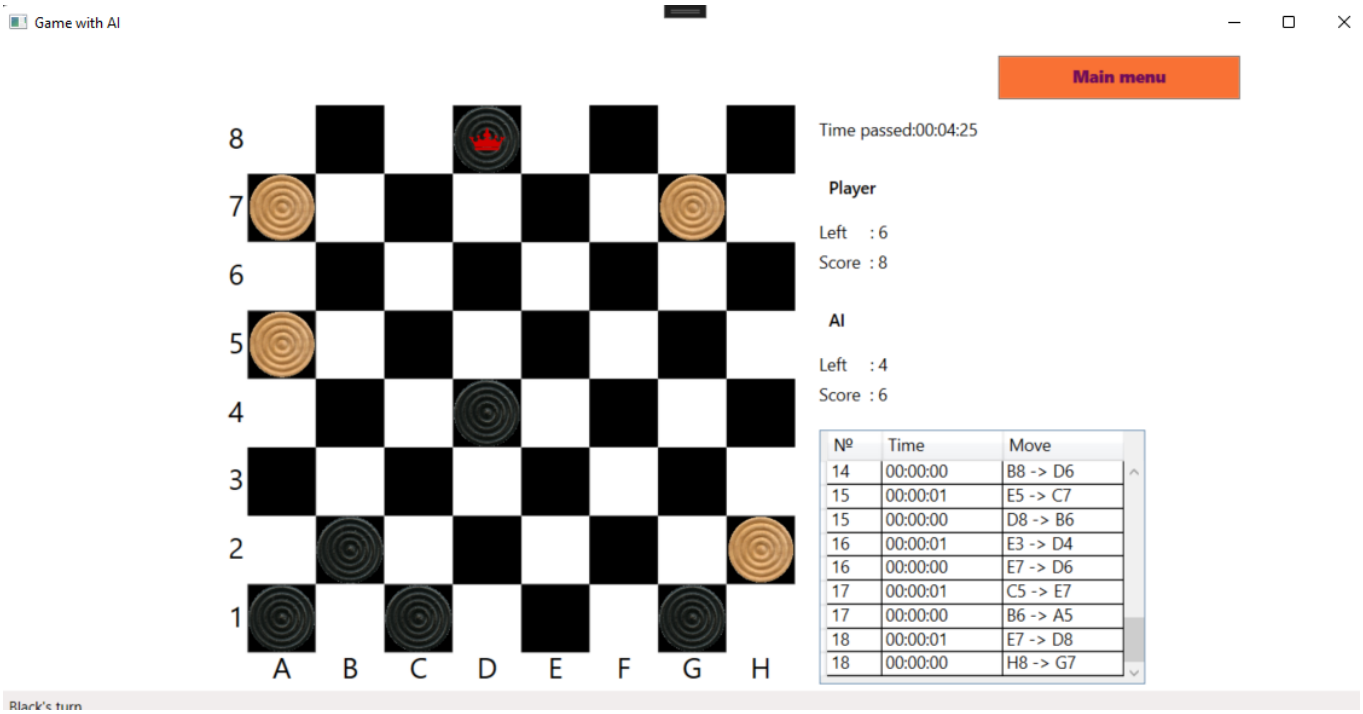

Рис. 2.3.4 Ілюстрація вікна "Game with AI" під час матчу, із зміною позицій шашок, кількості набраних очок, кількості шашок, що залишилися в гравців та заповненою таблицею з історією ходів

Після того, як в одного з гравців не залишається ні одної шашки, гра завершується, загальний час зупиняється, показується повідомлення про перемогу та дані про гравця записуються в турнірну таблицю "Stats".

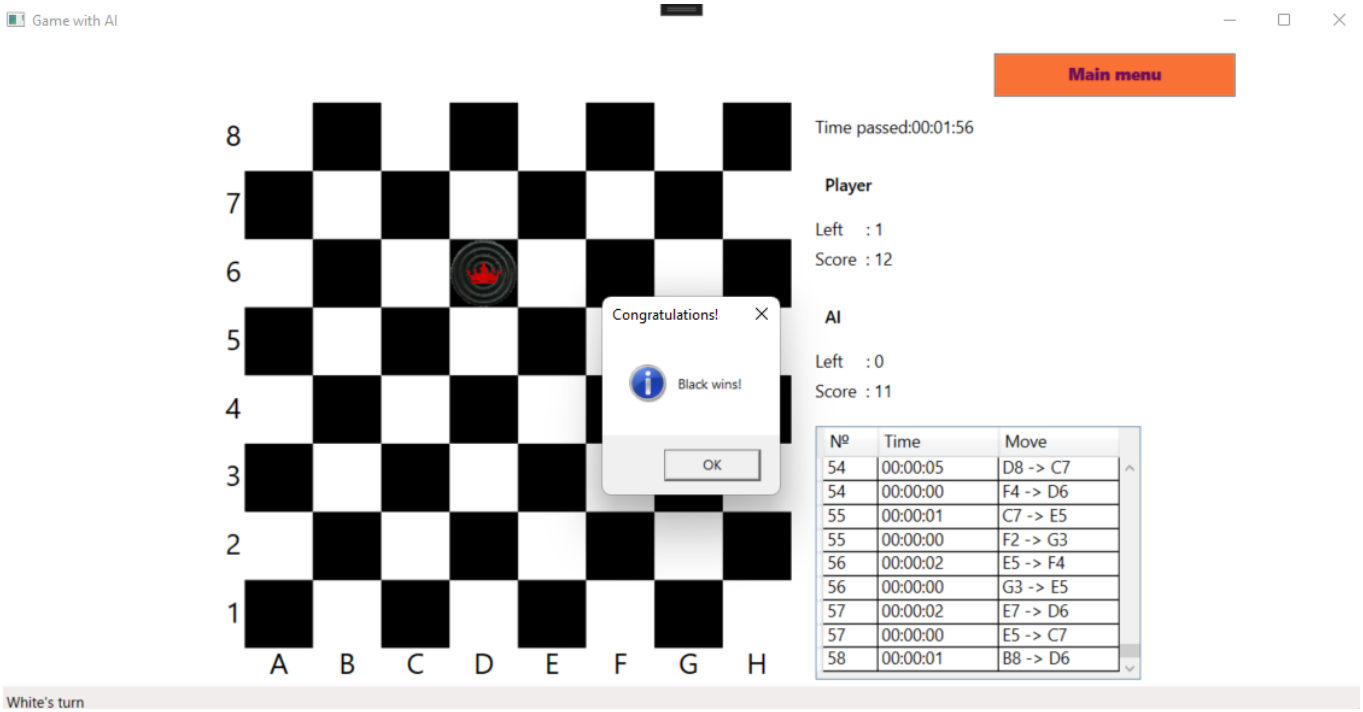

Рис. 2.3.5 Ілюстрація завершення гри перемогою гравця чорних шашок

При натисканні на кнопку “Stats” користувач переходить у вікно “Statistics”, де показано турнірну таблицю з результатами всіх гравців: виводиться id гравця, його нікнейм, кількість перемог, кількість програшів та загальний час продених ігор у форматі hh:mm:ss.

На графічному інтерфейсі розміщуються 3 кнопки: “Export in JSON”, “Export in XML” та “Main menu”. Користувач має можливість вивантажити дані турнірної таблиці у форматі JSON та XML.

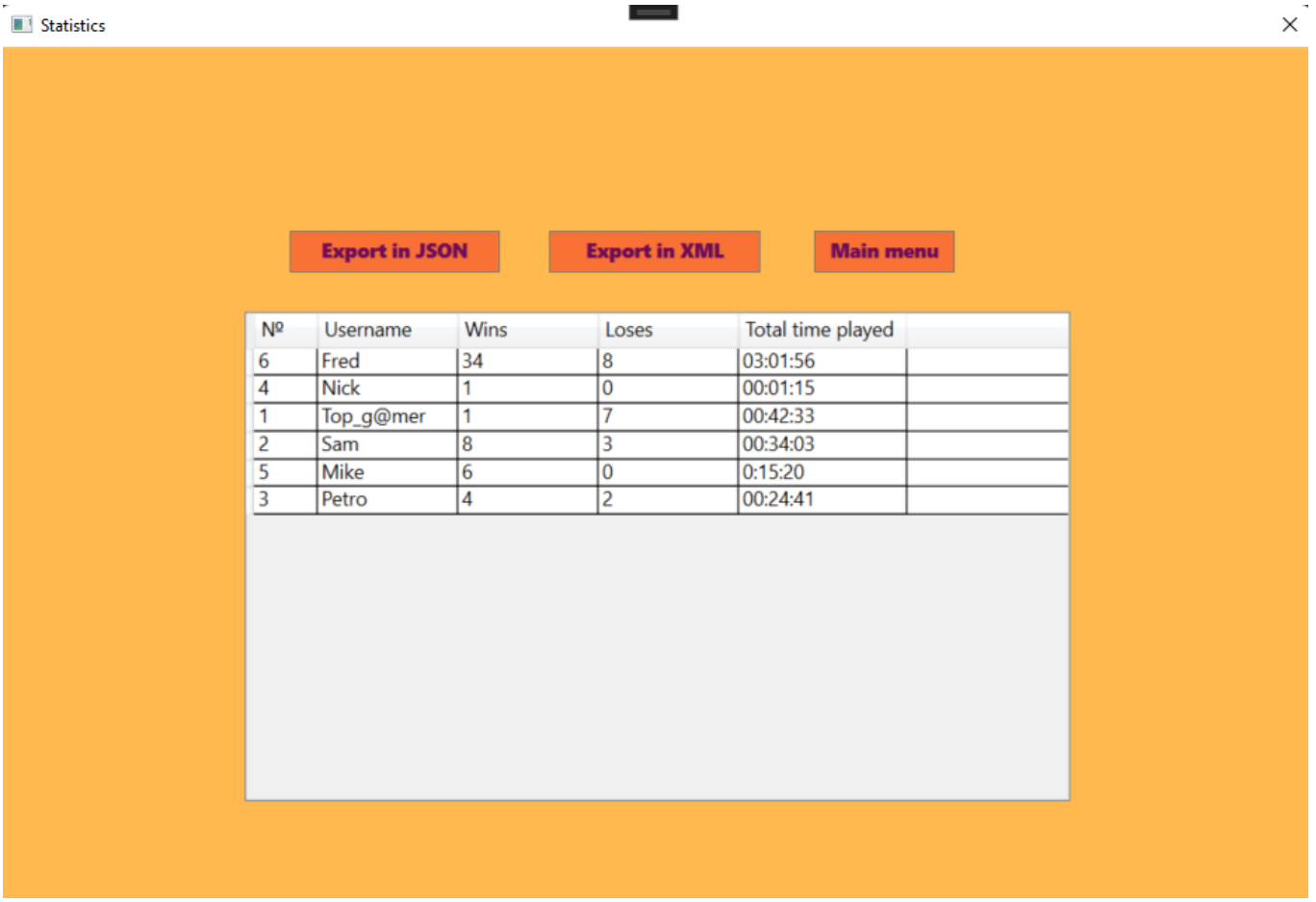

Рис. 2.3.6 Ілюстрація вікна “Statistics”

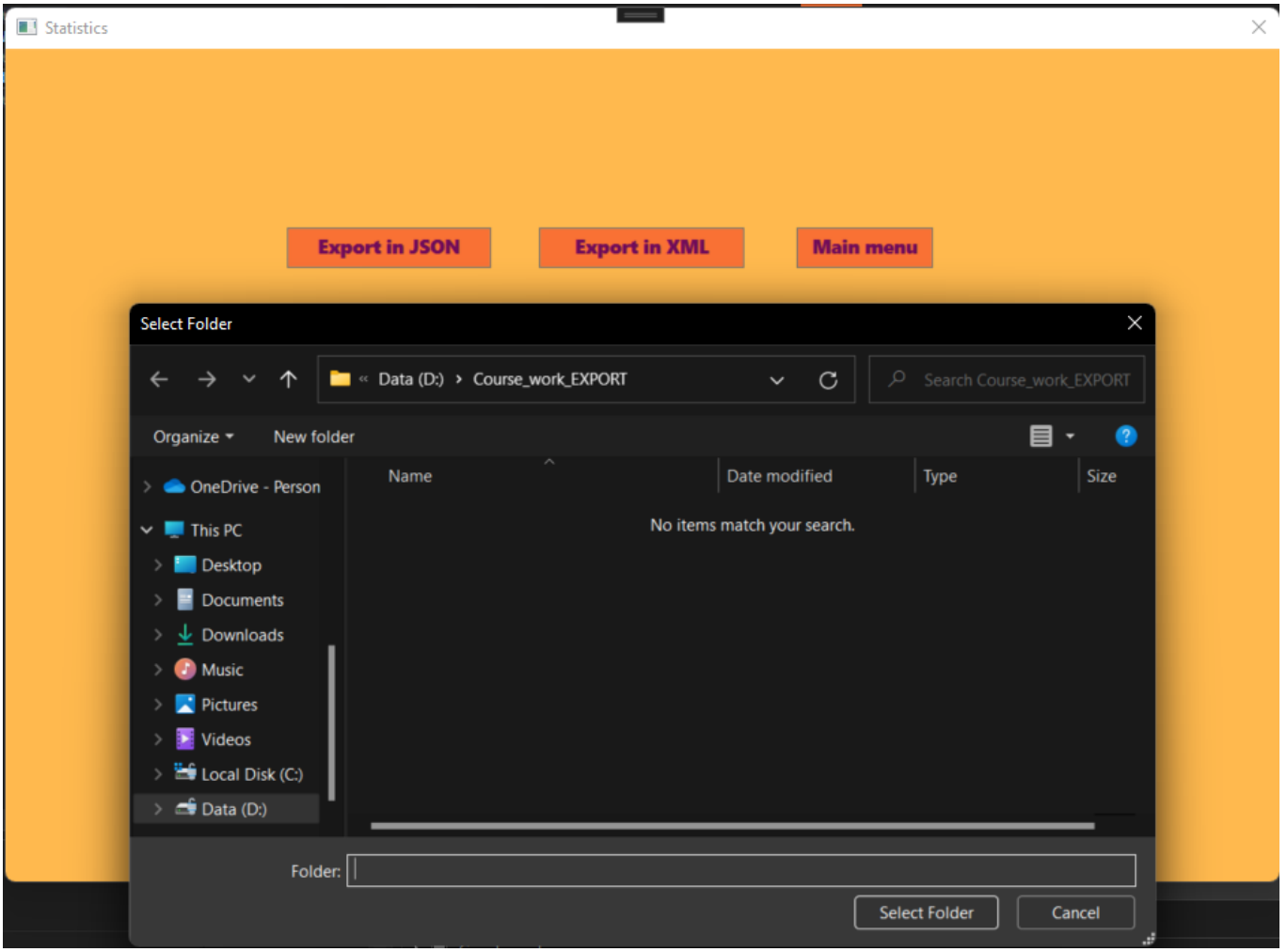

Рис. 2.3.7 Ілюстрація вікна вибору папки для експорту даних з турнірної таблиці

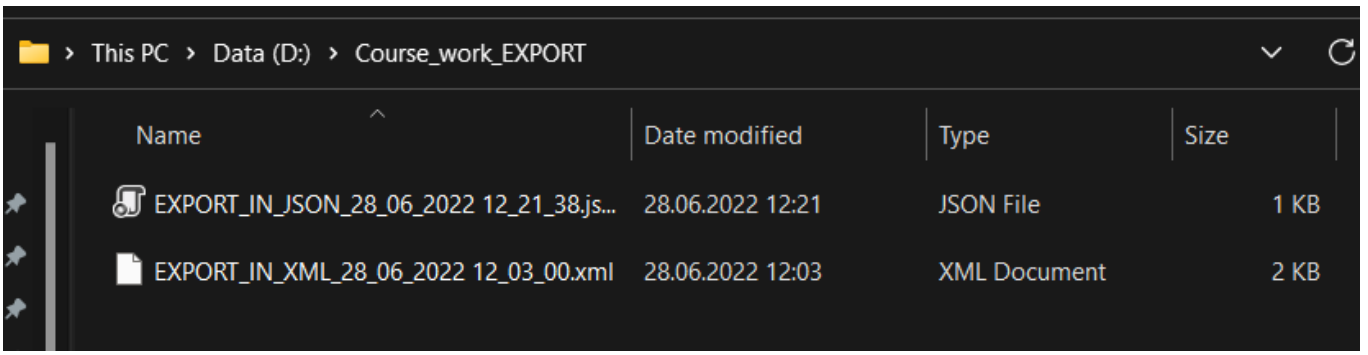

Рис. 2.3.8 Ілюстрація експортованих файлів з даними у форматі JSON та XML

При натисканні на кнопку “Styles” змінюється оформлення вікон “MainWindow” та “Stats”. Загалом є 3 імплементовані стилі: BasePaletteStyle, GreenPaletteStyle та PacificPaletteStyle.

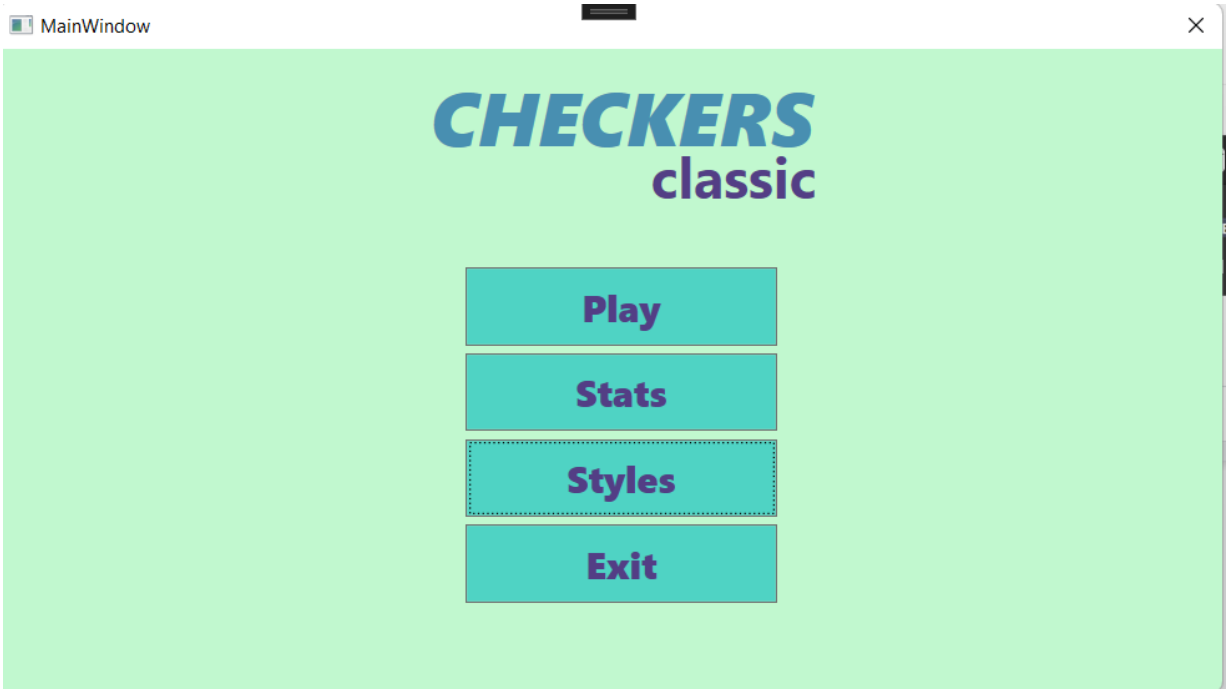

Рис. 2.3.9 Ілюстрація GreenPaletteStyle застосованого на вікно MainWindow

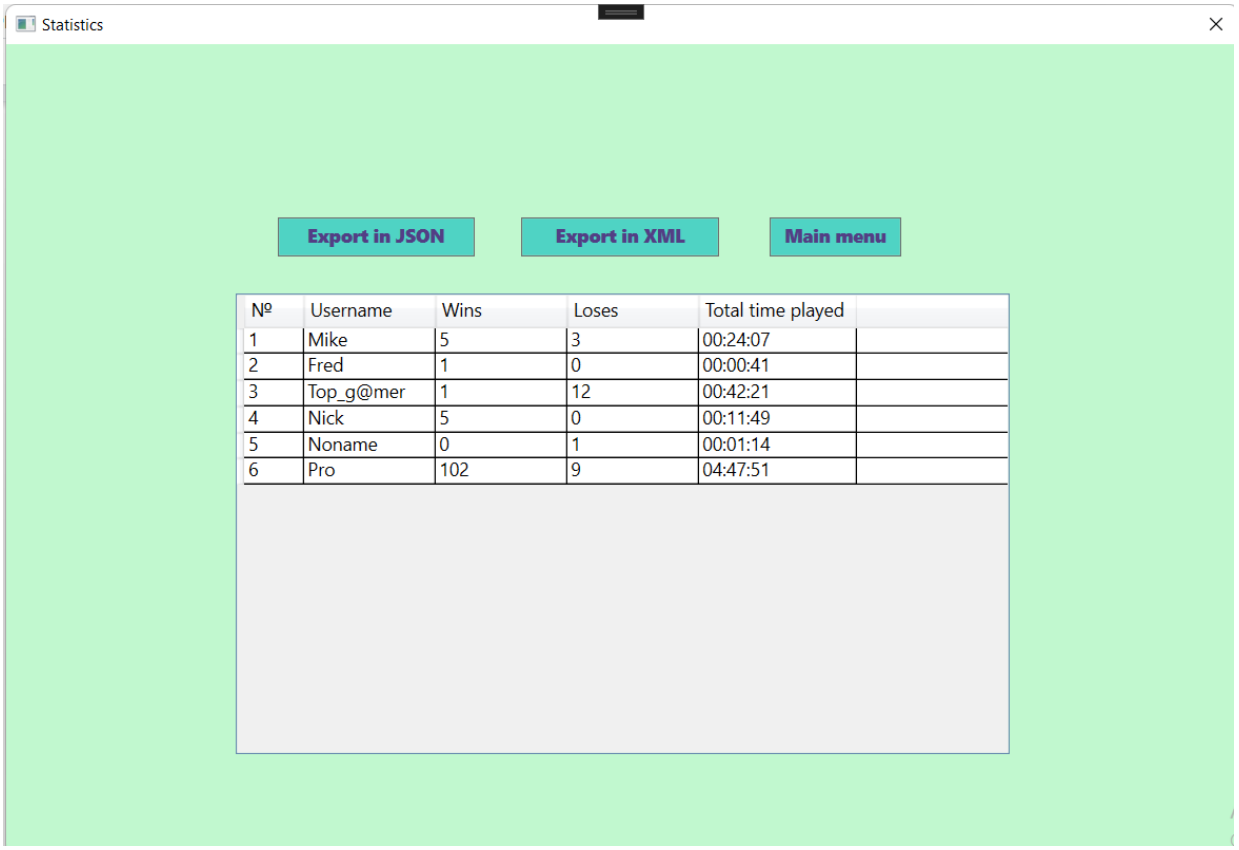

Рис. 2.3.10 Ілюстрація GreenPaletteStyle застосованого на вікно StatisticsWindow

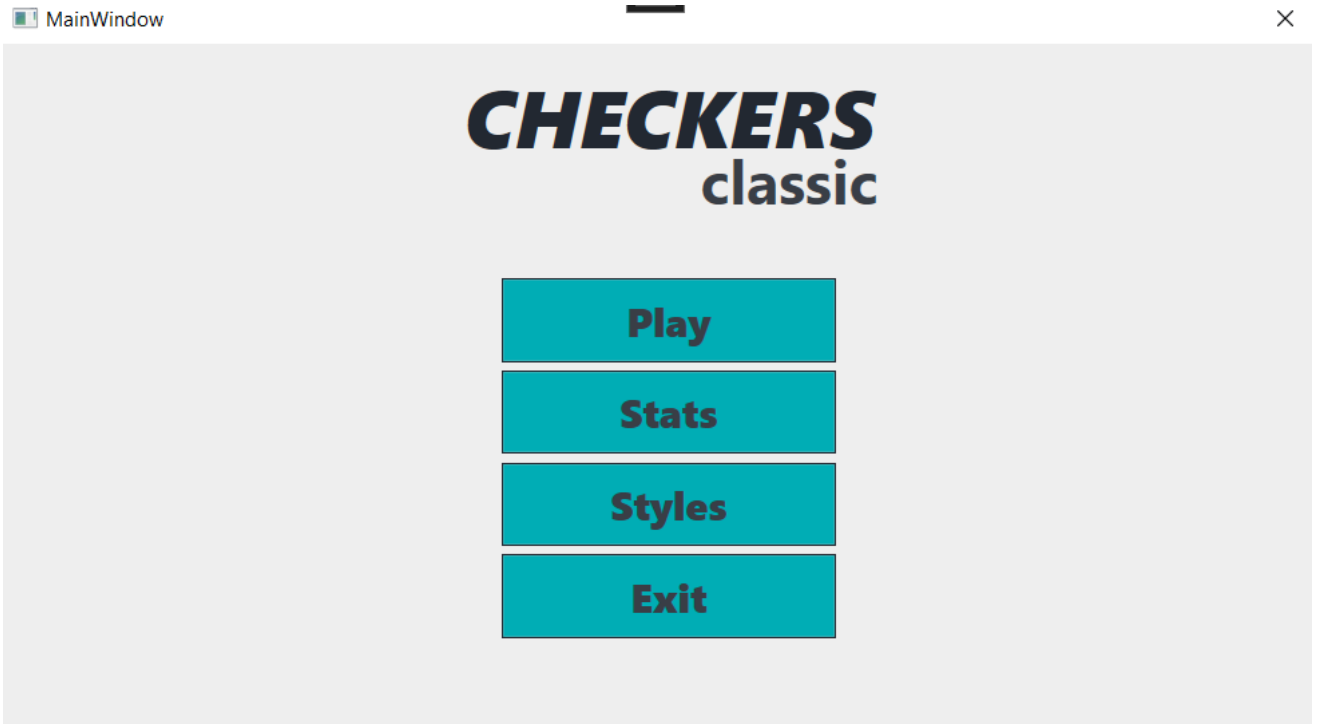

Рис. 2.3.11 Ілюстрація PacificPaletteStyle застосованого на вікно MainWindow

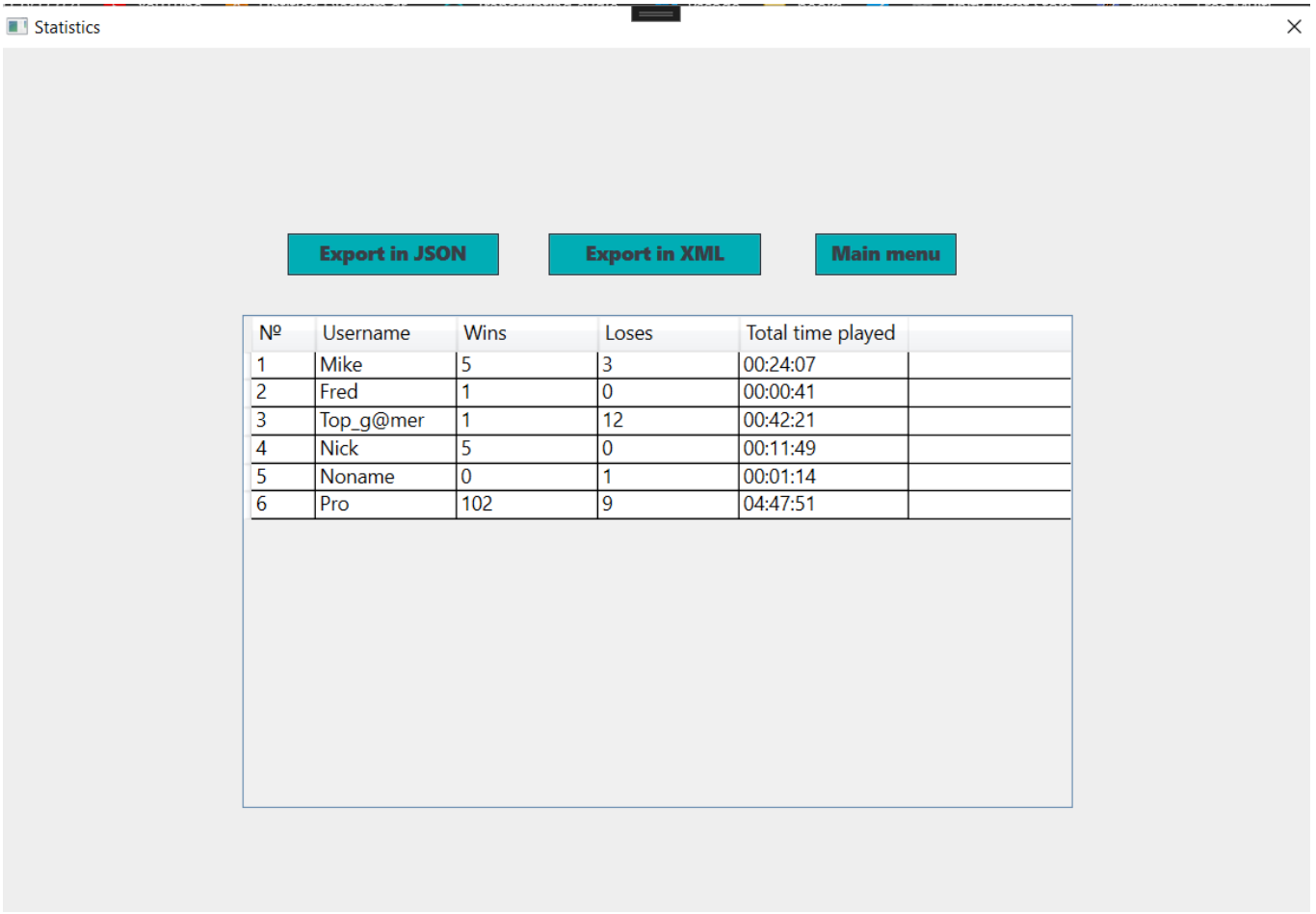

Рис. 2.3.12 Ілюстрація PacificPaletteStyle застосованого на вікно StatisticsWindow

ВИСНОВКИ

Метою даної курсової роботи було розроблення програмного забезпечення комп’ютерної гри «Шашки» з використанням шаблонів проєктування. Підставою для розроблення стало завдання на виконання курсової роботи з дисципліни «Програмування» студентами ІІ курсу кафедри ПЗКС НТУУ «КПІ ім. І.Сікорського».

Для досягнення поставленої мети у повному обсязі виконано завдання, визначені у аркуші завдання на курсову роботу; розроблено графічні матеріали; реалізовано всі вимоги до програмного продукту, програмного та апаратного забезпечення, наведені у технічному завданні; створено відповідну документацію.

Розроблене програмне забезпечення представляє собою комп’ютерну гру «Шашки», у якій користувач має змогу грати проти штучного інтелекту. Гра містить вікно із рейтинговою таблицею з усіма гравцями, та їх даними. Користувачі мають змогу вивантажити дані з рейтингової таблиці в різних форматах. Користувчі мають змогу змінювати кольорову палітру вікон головного меню та статистики. Для демонстрування роботи комп’ютерної гри було створено користувацький графічний інтерфейс.

Програму створено на основі використання шаблонів проєктування: зокрема до структури розробленого програмного забезпечення входить реалізація п’яти шаблонів: “MVVM”, “Міст”, “Стратегія”, “Команда” та “Шаблонний метод”.

Для розроблення програмного забезпечення використано мову програмування C# та платформу .Net.

Подальшим дослідженням даної тематики є розширення функціоналу даної гри, а саме, розроблення мережевої версії, для гри з користувачами по всьому світу.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Software Design Patterns [Електронний ресурс]: [Веб-сайт]. - Режим доступу: https://www.geeksforgeeks.org/software-design-patterns
2. Model–view–viewmodel [Електронний ресурс] : [Веб-сайт]. - Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel
3. Шаблони проєктування [Електронний ресурс]: [Веб-сайт].

   -Режим доступу: https://refactoring.guru/uk/design-patterns.

4. Patterns - WPF Apps With The Model-View-ViewModel Design Pattern [Електронний ресурс] : [Веб-сайт]. - Режим доступу до ресурсу: https://docs.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern.