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 «Об’єктно-орієнтоване програмування»   
на тему

**Шаблони проєктування в ООП. Комп’ютерна гра “Glory to Ukraine”**

Виконав студент

ІІ курсу групи КП-12   
Якубишин Анатолій Cергійович

Керівник роботи

доцент, к.т.н. Заболотня Т.М. Оцінка

(дата, підпис)

КИЇВ 2023

**ЗМІСТ**

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ 3](#_Toc135652687)

[ВСТУП 4](#_Toc135652688)

[1. ОПИС СТРУКТУРНО-ЛОГІЧНОЇ СХЕМИ ПРОГРАМИ 6](#_Toc135652689)

# **ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ**

**ПЗ** – (Програмне забезпечення) сукупність програм і програмних документів, необхідних для експлуатації цих програм.

**С#** – Об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET.

**MVVM** – Model-View-ViewModel — [шаблон проєктування](https://uk.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%B8_%D0%BF%D1%80%D0%BE%D1%94%D0%BA%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F), що застосовується під час проєктування архітектури [застосунків (додатків)](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA" \o "Застосунок). Публічно вперше був представлений Джоном Госсманом (John Gossman) у 2005 році як модифікація шаблону Presentation Model. MVVM орієнтований на такі сучасні платформи розробки, як [Windows Presentation Foundation](https://uk.wikipedia.org/wiki/WPF) та [Silverlight](https://uk.wikipedia.org/wiki/Silverlight" \o "Silverlight) від компанії [Microsoft](https://uk.wikipedia.org/wiki/Microsoft).

**WPF** – **Windows Presentation Foundation** (WPF, кодова назва — **Avalon**) — графічна (презентаційна) підсистема (аналог WinForms), яка починаючи з [.NET Framework 3.0](https://uk.wikipedia.org/wiki/.NET_Framework_3.0) в складі цієї платформи. Має пряме відношення до [XAML](https://uk.wikipedia.org/wiki/XAML). WPF разом з .NET Framework 3.0 вбудована в [Windows Vista](https://uk.wikipedia.org/wiki/Windows_Vista), а також доступна для установки в [Windows XP](https://uk.wikipedia.org/wiki/Windows_XP) Service Pack 2 і [Windows Server 2003](https://uk.wikipedia.org/wiki/Windows_Server_2003).

**ППO – Протиповітряна оборона.**

# **ВСТУП**

Дана курсова робота присвячена розробці програмного забезпечення комп’ютерної гри “Glory to Ukraine” за допомогою шаблонів проєктування. Багатато людей і не тільки дітей люблять грати в комп’ютерні ігри. Окрім розважальної мети, розроблене ПЗ має привернути увагу людей до важкої роботи сил ППО України та дасть змогу приміряти на себе роль захисника у розважальному вигляді. Взагалі ігри розвивають нестандартне мислення, спритність, швидкість реакції. 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 проектування.

*Об’єктом* дослідження є процес гри в комп’ютерну гру “Glory to Ukraine”.

*Метою роботи* є розроблення програмного забезпечення комп’ютерної гри “Glory to Ukraine” 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 cкладається з трьох основних типів модулів: моделі, види, та вид-моделі на базі шаблону MVVM.

Реалізовані шаблони проєктування: MVVM, Шаблонний метод, Стратегія, Фабричний метод, Команда, Легковаговик, Замісник, Спостерігач та міні шаблон – Одинак.

До функціональних можливостей програми належать: надання інформації щодо правил гри, зміна мови гри, надання характеристики щодо ракет ППО та ворожих літальних апаратів (літаки та дрони), запуск рівня та тренування.

Для функціювання розробленої програми (після встановлення з інсталятора, а не сам безпосередньо проект) потрібно 60 Мб вільного дискового простору.

Розроблене програмне забезпечення може бути використане людьми, що люблять грати в шутери та хочуть приємно провести час.

Пояснювальна записка складається зі вступу **ДОПИСАТИ!!!**

1. **ОПИС СТРУКТУРНО-ЛОГІЧНОЇ СХЕМИ ПРОГРАМИ**

**1.1. Модульна організація програми**

*Рис. 1.1.1. Модульна організація програми*

У програмі можна виділити декілька основних логічних модулів, що мають певну самостійність і обмінюються один з одним даними. Це модулі: Модель, Вид та Вид-Модель. Це є різновидом класичної багатошарової архітектури ПЗ, а саме його підвидом – MVVM.

**Моделі** – являють собою фундаментальні дані, що необхідні для роботи застосунку.

**Вигляд** — це графічний інтерфейс, тобто вікно, кнопки тощо. Cкладається з xaml розмітки та класу вікна у WPF.

**Вид-модель** 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. Фактично ViewModel призначена для того, щоб

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

Серед **Видів,**  з якими користувач може взаємодіяти, є головне вікно, інтсрукція, режим вибору рівня, вікно поля бою.

Для кожного **Виду** відповідно існує свій **Вид-Модель.**

## 1.2. Функціональні характеристики

Розроблене ПЗ дозволяє користувачу зручно керувати процесом гри за допомогою графічного інтерфейсу. Як тільки користувач заходить в гру , відкривається головне вікно, на якому можна змінити мову, при чому вибір користувача запам’ятається і при повторному запуску програми мова залишиться збереженою. Як тільки користувач встановлює гру у нього не пройдено жодного рівня і йому доступне тренування та перший рівень, також у користувача є можливість детально ознайомитися з графічної інструкцією. Під час проходження рівняння або тренування, користувач може натиснути на паузу (клавіша ESC) та з з’явленого вікна може зробити наступні дії: вийти з гри взагалі, вийти в меню, перезапустити рівень або продовжити гру. Також хочу зазначити, для того щоб пройти рівень потрібно знищити всі цілі, але при цьому якщо хоча б одна ціль була пропущена рівень не перезапускаєтьcя. Це було зроблено з метою, щоб людина мала змогу подивитися на порядок повітряних цілей та потренуватися збивати наступні цілі, але якщо людина не хоче витрачати свій час, знаючи, що вона вже програла, у неї можливість перезапустити рівень через меню.

Під час проходження рівняння користувачу відображаються дані про кількість ракет кожного типу, що наявні загалом, та кількість ракет, що залишилась. Також кількість ворожих цілей протягом рівня, та кількість збитих ворогів.

1. **ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ЗА ДОПОМОГОЮ ШАБЛОНІВ ПРОЕКТУВАННЯ** 
   1. **Обґрунтування вибору та опис шаблонів проектування для програмної реалізації гри “Glory to Ukraine”**
2. **MVVM**

*Визначення:*

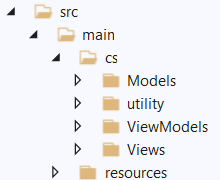
Model-View-ViewModel — [шаблон проєктування](https://uk.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%B8_%D0%BF%D1%80%D0%BE%D1%94%D0%BA%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F), що застосовується під час проєктування архітектури [застосунків (додатків)](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA" \o "Застосунок). Публічно вперше був представлений Джоном Госсманом (John Gossman) у 2005 році як модифікація шаблону Presentation Model. MVVM орієнтований на такі сучасні платформи розробки, як [Windows Presentation Foundation](https://uk.wikipedia.org/wiki/WPF) та [Silverlight](https://uk.wikipedia.org/wiki/Silverlight" \o "Silverlight) від компанії [Microsoft](https://uk.wikipedia.org/wiki/Microsoft).

*Обґрунтування використання шаблону:*

Оскільки цей шаблон є підвидом багатошарової архітектури для WPF, то до нього можна застосувати властивості цієї архітектури.

Основні випадки коли застосовується багатошарова архітектура:

* Є хорошим вибором для невеликих простих програм та вебсайтів
* як відправна точка, для ситуацій з дуже жорстким бюджетом і часовими обмеженнями.
* коли архітектор все ще аналізує бізнес-потреби і вимоги та не впевнений, який архітектурний стиль буде найкращим.

Випадок розроблення курсової роботи як раз підлягає цим критеріям.

*Рис. 2.1.1. Cтруктура MVVM*

Учасники шаблону:

* Види:
  + MainWindow.xaml і MainWindow.xaml.cs
  + ChooseLevel.xaml і ChooseLevel.xaml.cs
  + Guid.xaml і Guide.xaml.cs
  + Field.xaml і Field.xaml.cs

За допомогою прив’язки властивостей та прив’язки до команд відповідної Вид-Моделі взаємодіє з Вид-Моделлю.

* Вид-Моделі:
  + BaseViewModel – клас батько для всіх Вид-Моделей, реалізовує інтерфейс InotifyPropertyChanged
  + MainWindowViewModel
  + GuideViewModel
  + FieldViewModel

Також класи Вид-Моделей містять екземпляри класів команд,

Детальніше буде в описі шаблона Команда.

* Моделі:
  + Класи, що наслідують абстрактний клас AbstactDrone (Дрони)
  + Патрони, що наслідують абстрактний клас AbstractPatron
  + Літаки, що наслідують абстрактний клас AbstractPlain
  + GameStateSingltone, що відповідає за поточний стан ігрового поля
  + А також неосновні допоміжні класи такі як індекатор здоров’я ворога тощо.

1. **Шаблонний метод**

*Визначення:*

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.

*Обґрунтування використання шаблону:*

Шаблон влучно підходить для реалізації логіки поведінки ворогів та патронів. За допомогою цього шаблону можна задати кістяк поведінки для вище сказаних літальних апаратів, це дає змогу легко масштабувати ПЗ, додаючи нових ворогів та патрони, оскільки на верхнім рівнях ієрархії наслідування вказана основна структура поведінки і достатньо реалізувати абстрактні методи, що будуть відрізняти класи один від одного. Також це дозволяє уникнути дублювання коду, оскільки в шаблонних методах можна вказити не тільки перелік абстрактних методів, а ще й певні спільні дії для всіх класів наслідників. Також цей шаблон у поєданні з командою та стратегією використовується для перезапуску рівня.



*Рис. 2.1.2. Діаграма класів, які входять до шаблону Шаблонний метод*

*Учасники шаблону*:

Використання цього шаблону було неодноразовим. Два шаблонних метода в класі AbstractEnemy. Та один шаблонний метод в класі AbstractPatron.

Розгляну використання шаблону на прикладі ієрархії з коренем AbstractEnemy, бо реалізація з AbstractPatron є cхожою і навіть простішою.

* **AbstractEnemy** – абстрактний клас найвищого рівня, в якому містяться два шаблонних методи Fly() та end(Image bah, Image img, Grid grid)

Fly() містить в собі три абстрактних методи: SetSpeed(), SetDisplayProperty(), ExecuteBodyMethod().

Метод end містить в собі абстрактний метод reduceEnemy() та finish()

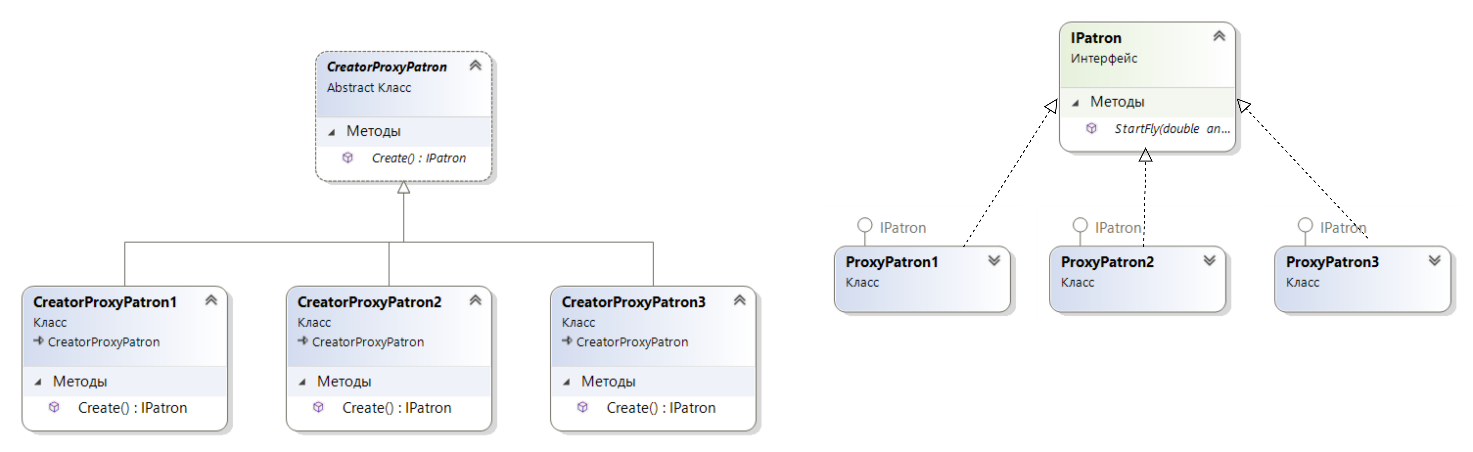
* **AbstractDron –** реалізує методи ExecuteBodyMethod() та finish() класу AbstractEnemy відповідно до логіки дрона.
* **AbstractPlain –** реалізує методи ExecuteBodyMethod() та finish() класу AbstractEnemy відповідно до логіки літака.
* **Dron1 –** наслідник **AbstractDron,** реалізує SetSpeed(), SetDisplayProperty(), reduceEnemy()
* **Plain1 –** наслідник **AbstractPlain**, реалізує SetSpeed(), SetDisplayProperty(), reduceEnemy()
* **Plain2 –** наслідник **AbstractPlain**, реалізує SetSpeed(), SetDisplayProperty(), reduceEnemy()

1. **Фабричний метод**

*Визначення:*

**Фабричний метод** — це породжувальний патерн проектування, який визначає загальний інтерфейс для створення об’єктів у суперкласі, дозволяючи підкласам змінювати тип створюваних об’єктів.

*Обґрунтування використання шаблону:*

Під час розроблення ПЗ виникає потреба змінювати тип патрона на відповідні клавіші або правою кнопкою миші. Фабричний метод дозволяє уникнути використання умовних операторів. Також цей шаблон використовується для генерації випадковим чином противників у режимі тренування. Покажу використання шаблону на прикладі запуску патрона у відповідності до вибраного типу патрона. Але зазначу, що цей щаблон поєднаний з шаблоном Замісник, про який буде детально розказано пізніше.

*Рис. 2.1.3. Діаграма класів, які входять до шаблону Фабричний метод*

*Учасники шаблону*:

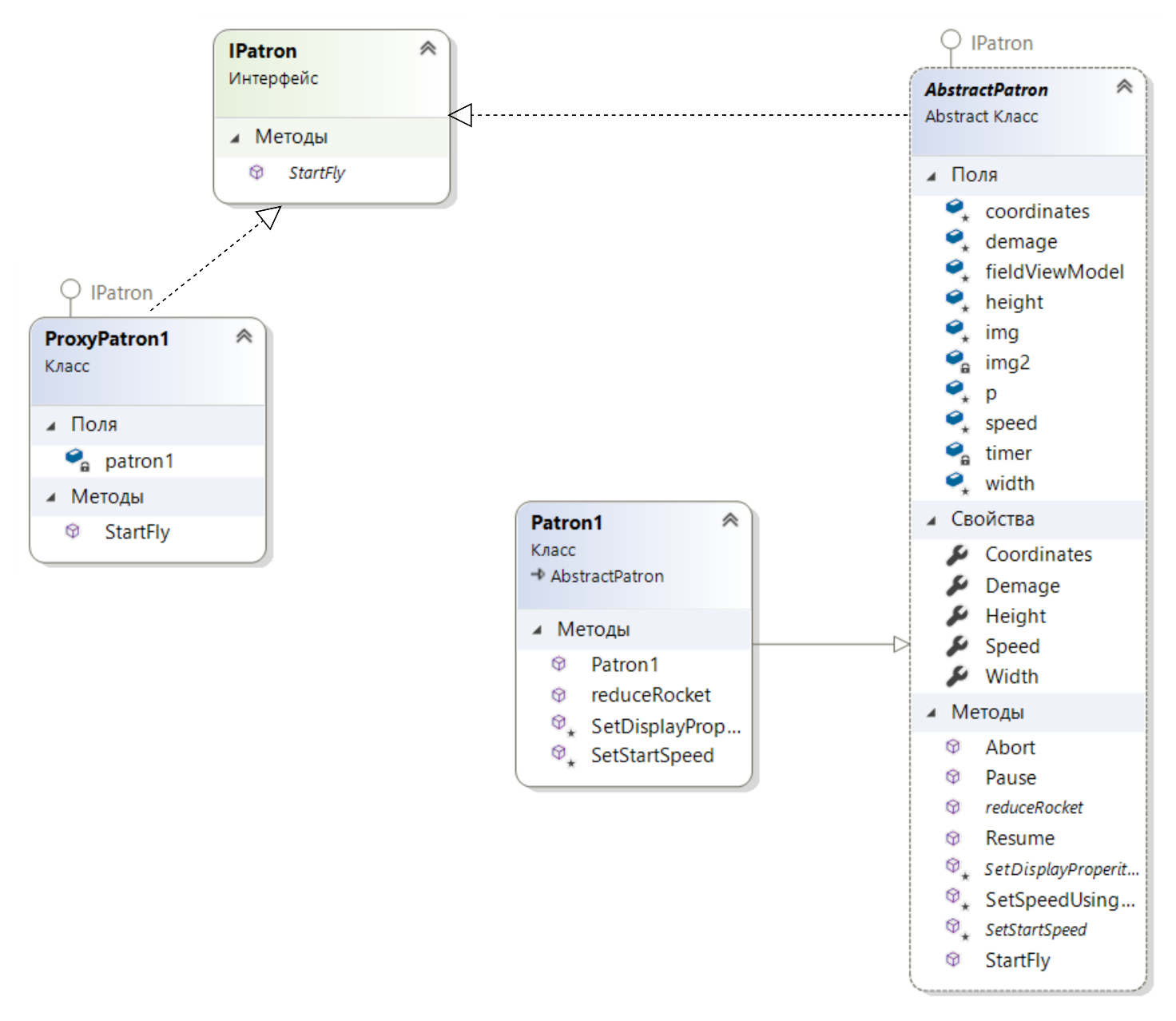
* **IPatron –** інтерфейс для патронів
* **ProxyPatron1, ProxyPatron2, ProxyPatron3 –** класи, що реалізують IPatron
* **CreatorProxyPatron (Abstract Creator)**
* **Сoncrete creators: CreatorProxyPatron1, CreatorProxyPatron2, CreatorProxyPatron3**

1. **Замісник**

*Визначення:*

**Замісник** — це структурний патерн проектування, що дає змогу підставляти замість реальних об’єктів спеціальні об’єкти-замінники. Ці об’єкти перехоплюють виклики до оригінального об’єкта, дозволяючи зробити щось до чи після передачі виклику оригіналові.

*Обґрунтування використання шаблону:*

****За правилами моєї гри ракетна установка потребує перезарядки в 0,7 секунд та в усіх рівнях кількість ракет обмежена. Тому розроблена система має перехоплювати виклик перед пуском снаряда, у разі, якщо перезарядка не пройшла ігнорувати пуск, у разі, якщо тип такого патрона закінчився виводити повідомлення “Змініть тип снаряду”. Цей шаблон в моїй реалізації тісно пов’язаний з Фабричним Методом, тому для кожного типу патрона свій Замісник.

*Рис. 2.1.4. Діаграма класів. Приклад заміщення патрону 1-го типу*

*Учасники шаблону*:

* **IPatron** **–** інтерфейс для патронів та їх замісників
* **AbstractPatron –** батьківський клас для патронів
* **Patron1, Patron2, Patron3 –** справжні класи
* **ProxyPatron1, ProxyPatron2, ProxyPatron3 –** класи замісників

1. **Cтратегія**

*Визначення:*

**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.

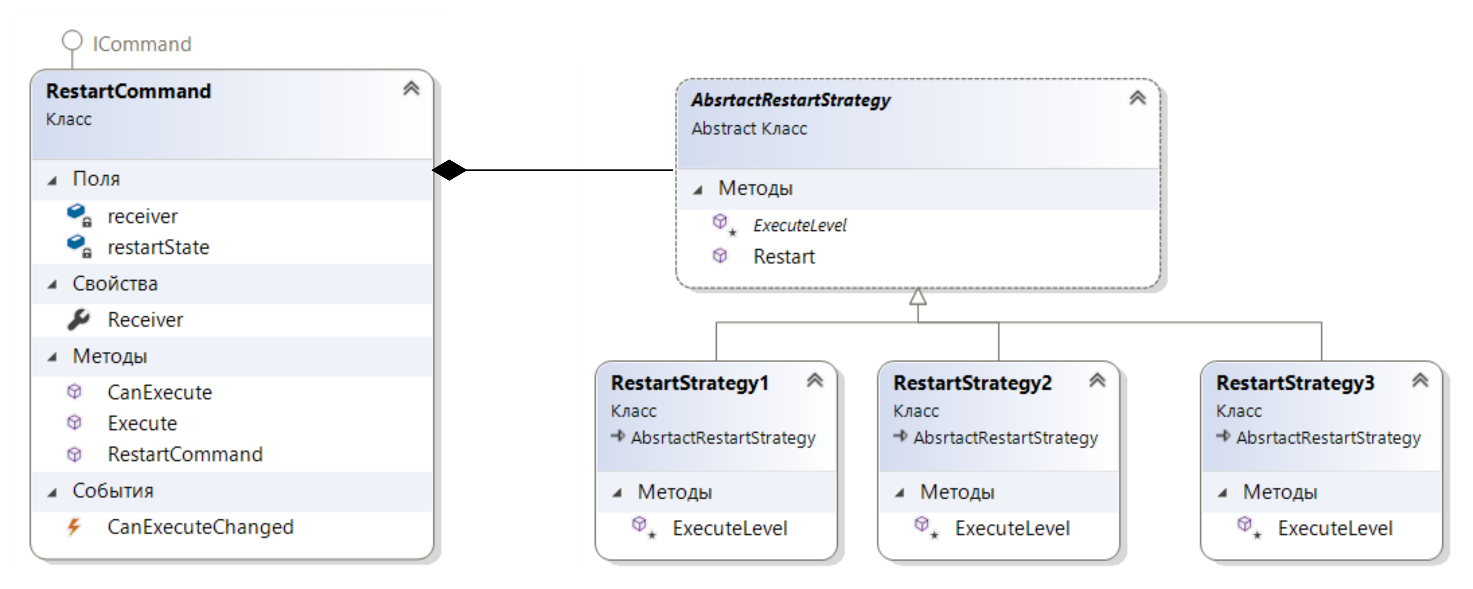
*Обґрунтування використання шаблону:*

Цей шаблон став мені у нагоді при реалізації двомовності гри (українські та англійські мові). В залежності від вибраної мови, потрібно змінювати текст виведених повідомлень під час гри та текст в головному меню та у вікні вибору рівня. Другий випадок використання – це задання літакам та дронам напрямку польоту (зліва-направо чи справа-наліво), бо від цього залежить відображення ворога. Оскільки в мене декілька Видів і відповідно Вид-Моделей, то для кожної вид моделі реалізована своя стратегія для мови. І третій випадок – перезапуск рівня, бо в залежності від поточного рівня треба перезапускати відповідний рівень і є сумішшю шаблону стратегія та шаблонного методу.



*Рис. 2.1.5.1 Діаграма класів. Приклад шаблону Стратегія (зміна мови)*

*Рис. 2.1.5.2 Діаграма класів. Приклад шаблону Стратегія (зміна напрямку польоту ворога)*

**

*Рис. 2.1.5.3 Діаграма класів. Приклад шаблону Стратегія + Шаблонний метод (перезапуск рівня)*

*Учасники шаблону*:

* **Клієнтський клас**, що агрегує інтерфейс чи абстрактний клас
* Інтерфейс чи абстрактний клас – **абстакція реалізаторів**
* **Конкретні реалізатори**

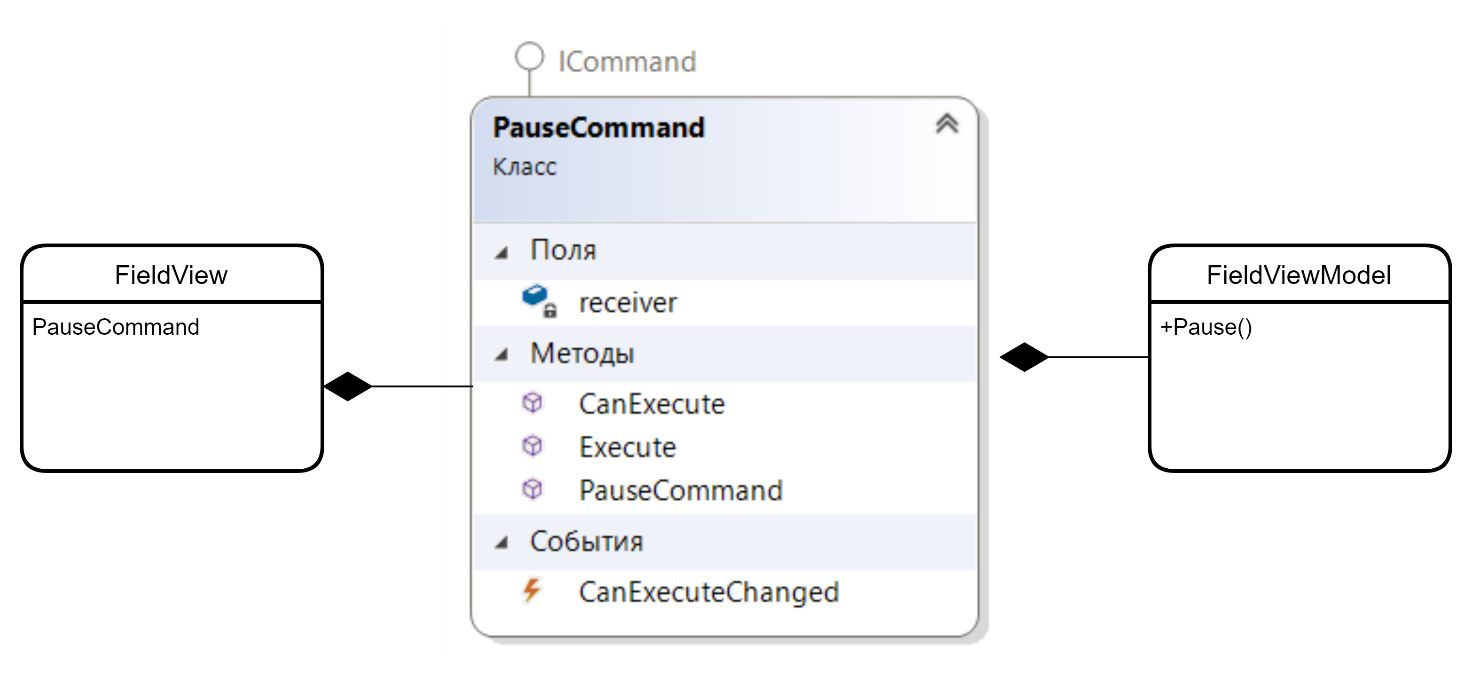
1. **Команда**

*Визначення:*

**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.

*Обґрунтування використання шаблону:*

Класи команди у мене найпоширені класи в проєкті. Оскільки я використовую основним шаблоном **MVVM**, то для прив’язки до Вид-Моделі з виду на якусь дії необхідно використовувати команди. Також різні кнопки на інтерфейсі, натиснення клавіш, можуть запускати одну і ту ж команду і тому передача команд як аргументів – це дуже зручно, бо достатньо змінити лише клас команди, а не переписувати всі дії для кожного графічного елемента.

****Розгляну приклад встановлення гри на паузу, бо всі інші ситуації мають схожу логіку і тому розглядати всі команди немає потреби.

*Рис. 2.1.6 Діаграма класів для шаблону Команда*

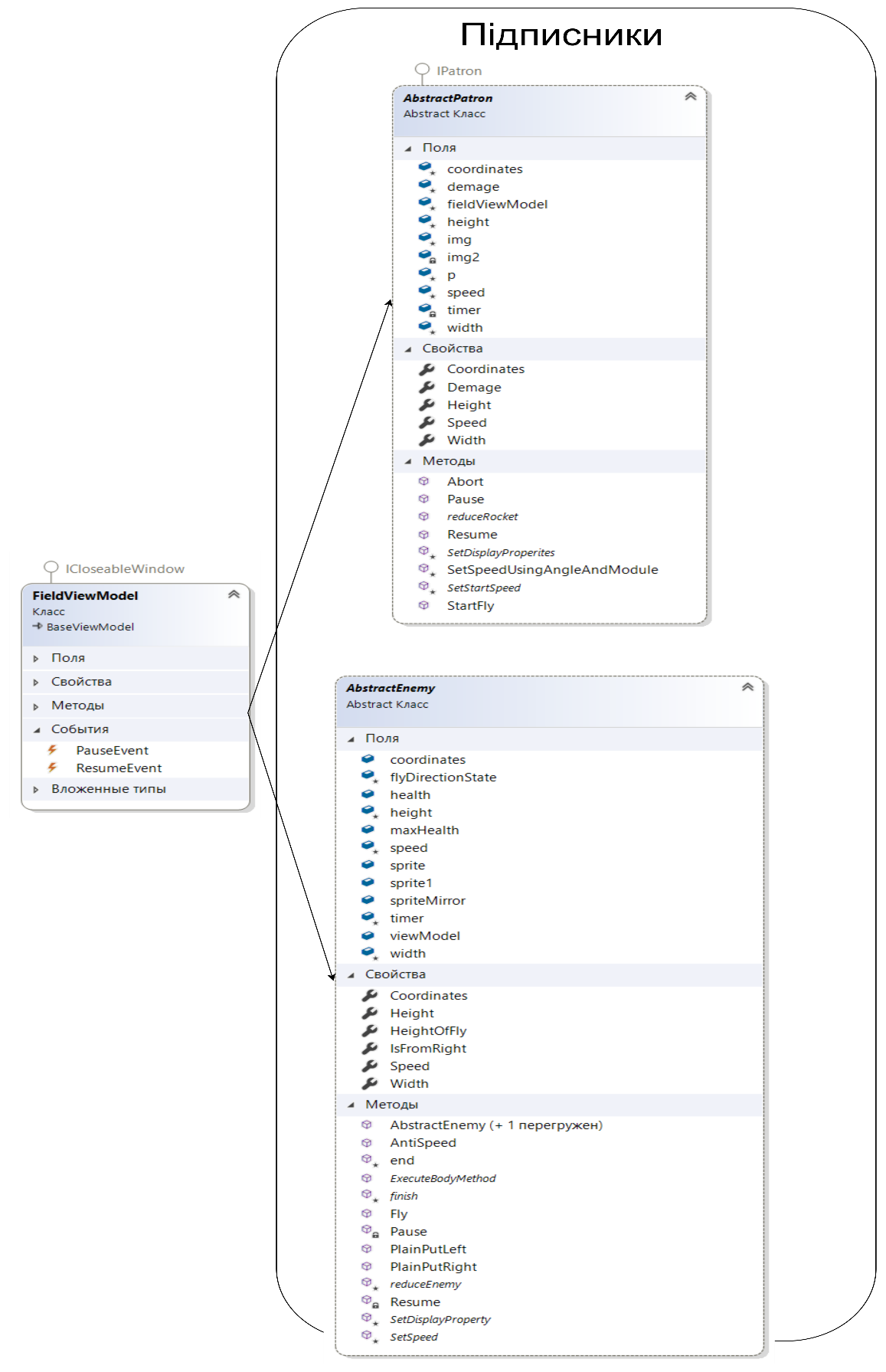
*Учасники шаблону*:

* **FieldView** – відіграє роль **Invoker**. При натиснення клавіші Escape відбувається виклик команди. Це реалізовано через сторонній модуль Microsoft.Xaml.Behaviours.WPF який робить прив’язку до команд при натисканні клавіші.
* **PauseCommand** – **Команда**, реалізує службовий інтерфейс ICommand для коректної роботи прив’язки
* **FieldViewModel -** відіграє роль **Receiver.**

**7) Спостерігач**

**Спостерігач** — це поведінковий патерн проектування, який створює механізм підписки, що дає змогу одним об’єктам стежити й реагувати на події, які відбуваються в інших об’єктах.

*Обґрунтування використання шаблону:*

Шаблон використовується для реалізації Паузи. FieldViewModel – тобто модель виду поля бою містить подію “поставити гру на паузу”. І на неї підписуються всі ігрові об’єкти, припиняючи свій рух. Ну і відповідно якщо є ****подія піти на паузу, то є і подія відновити гру.

*Рис. 2.1.7 Діаграма класів для шаблону Спостерігач*

*Учасники шаблону*:

* **Ініціатор події** (**FieldViewModel**), що запускає подію і змушує підписників реагувати на неї.
* **Підписники** (**AbstractPatron** та **AbstactEnemy**), що реагують на подію, в моємо випадку – це припинення руху та відновлення руху відповідно.

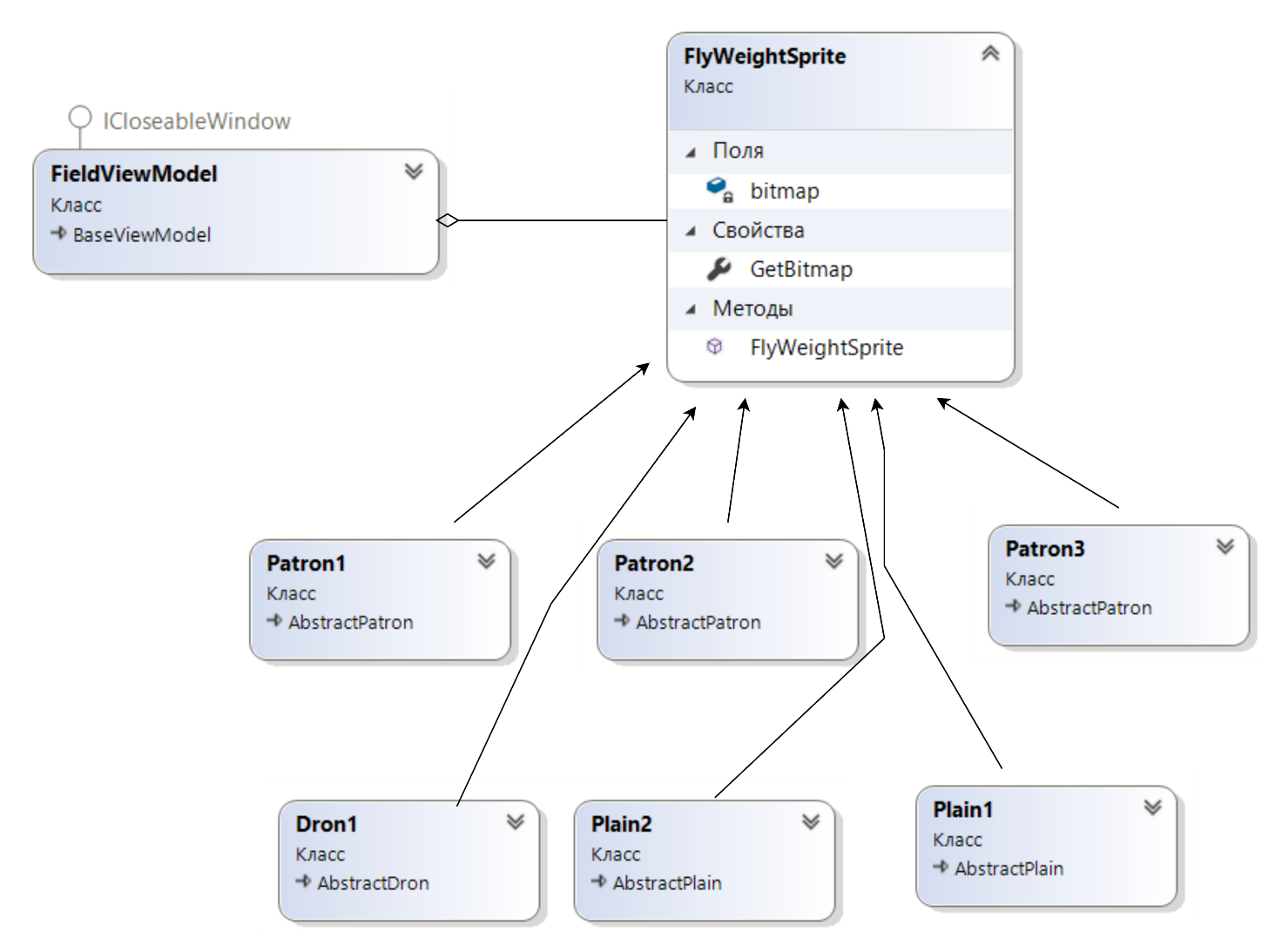
**8) Легковаговик**

*Визначення:*

**Легковаговик** — це структурний патерн проектування, що дає змогу вмістити більшу кількість об’єктів у відведеній оперативній пам’яті. Легковаговик заощаджує пам’ять, розподіляючи спільний стан об’єктів між собою, замість зберігання однакових даних у кожному об’єкті.

*Обґрунтування використання шаблону:*

В ігрових об’єктах, що наявні в грі можна виділити унікальний та повторючий стан. Унікальним буде стан координат ракети, а повторюваним буде спрайт, тобто графічне зображення ракети або ворога. Оскільки кількість таких спрайтів досить незначна, то це буде повторюваний стан. Щоб не завантажувати кожен раз в оперативну пам’ять спрайт відповідної ракети коже раз при створенні ракети, достатньо наперепед завантажити всі можливі спрайти, а потім використовувати лише посилення на них у ракетах та снарядах. Такий підхід значно зекономить ресурси комп’ютера.



*Рис. 2.1.8 Діаграма класів для шаблону Легковаговик*

*Учасники шаблону*:

* **FieldViewModel –** відіграє роль контейнера для об’єктів Легковаговиків, тобто забезпечує доступ до повторюваного стану спрайтів
* **FlyWeightSprite –** це своєрідна обгортка для Bitmap, була створена з метою інкапсулювання картинки спрайта ззовні.
* **Ігрові об’єкти** **–** використовують підвантажений спрайт у FlyWeightSprite.

**9) Одинак (міні-шаблон)**

*Визначення:*

**Одинак** — шаблон проєктування, відноситься до класу твірних шаблонів. Гарантує, що клас матиме тільки один екземпляр, і забезпечує глобальну точку доступу до цього екземпляра.

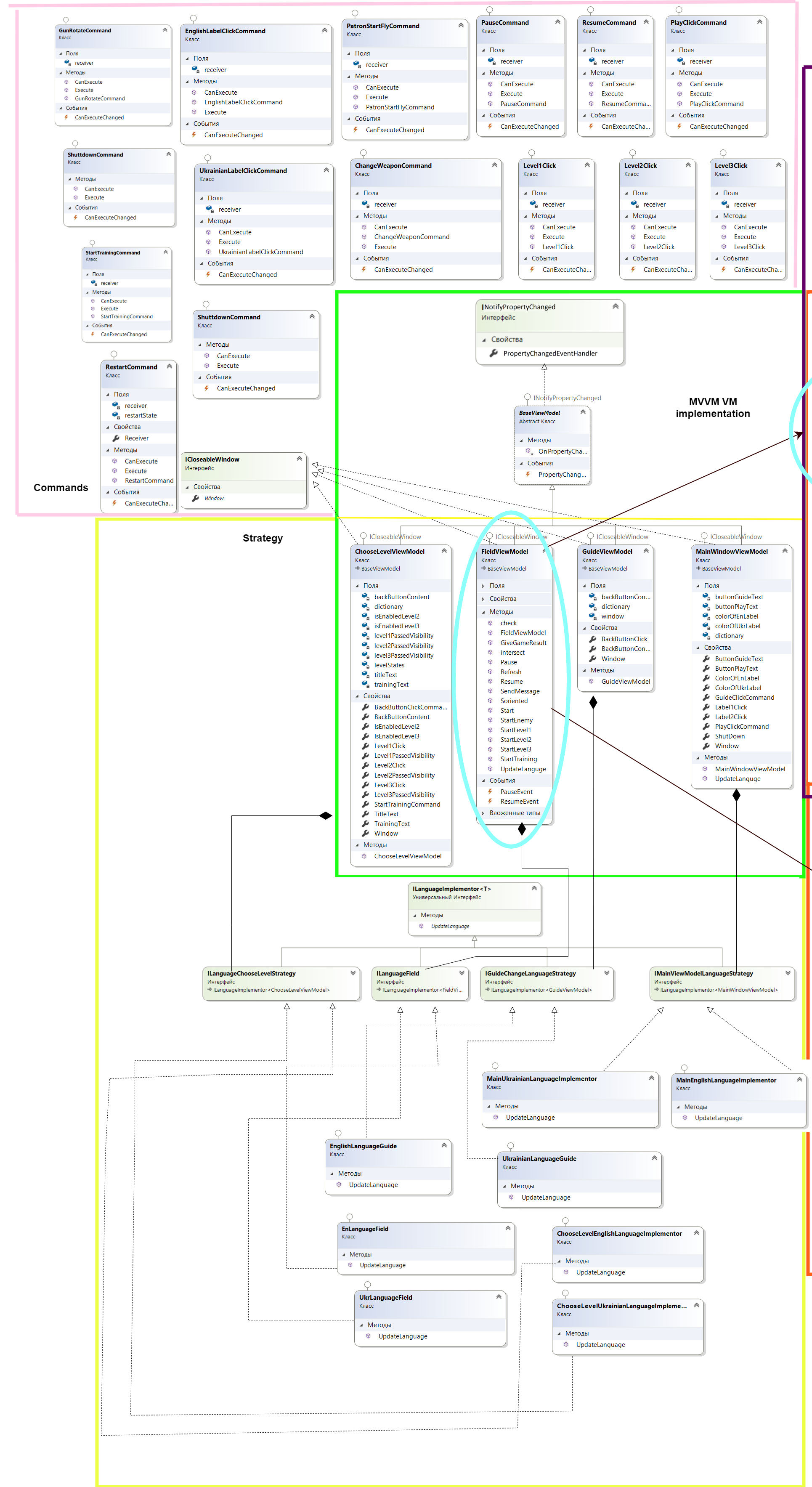
*Обґрунтування використання шаблону:*

В моїй грі потрібно забезпечити глобальний доступ до об’єкта стану гри. В якому міститься така важлива інформація як кількість патронів, що залишилися, стан перезаряду систему ППО

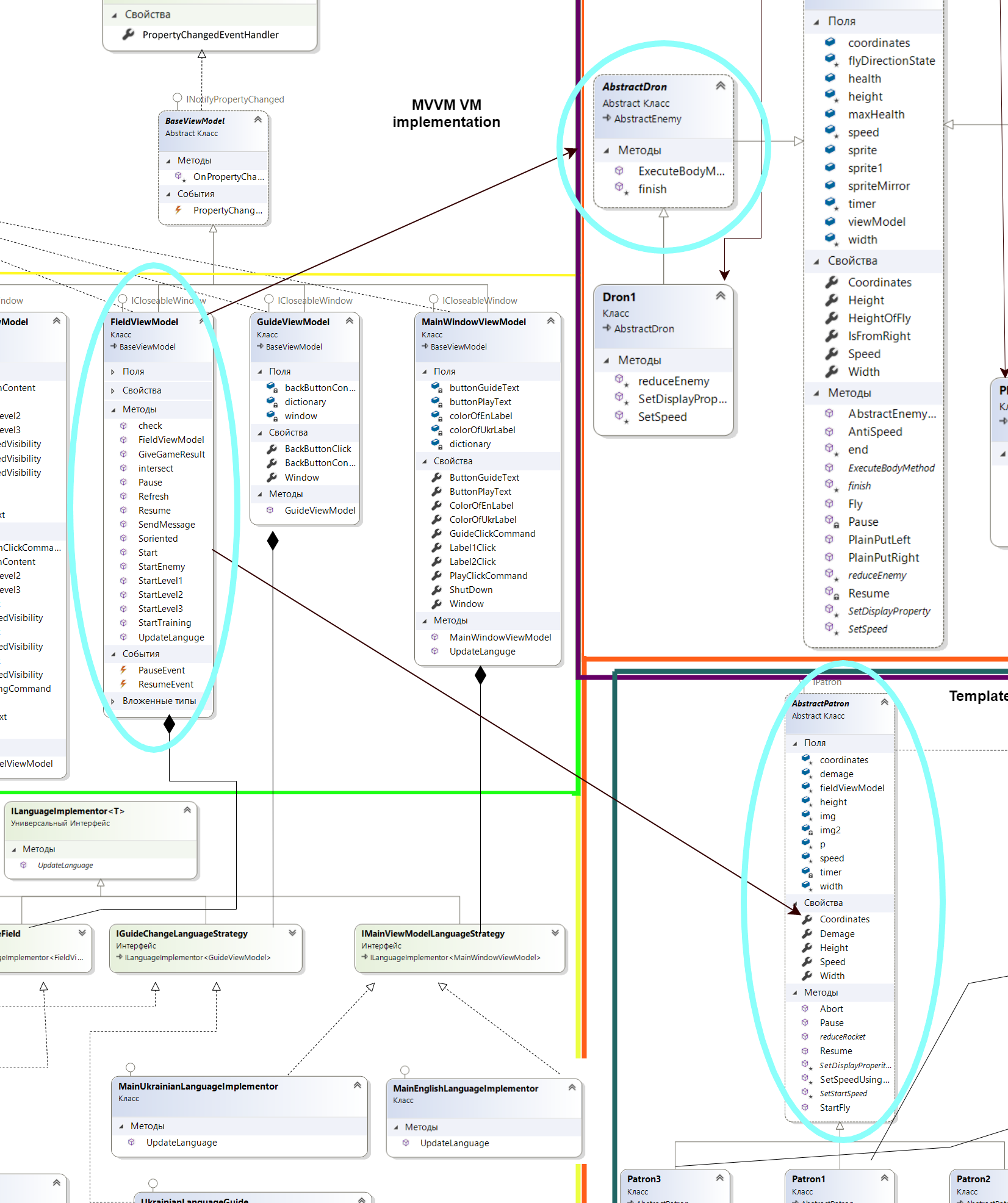
*Учасники шаблону*:

**GameStateSingleton**

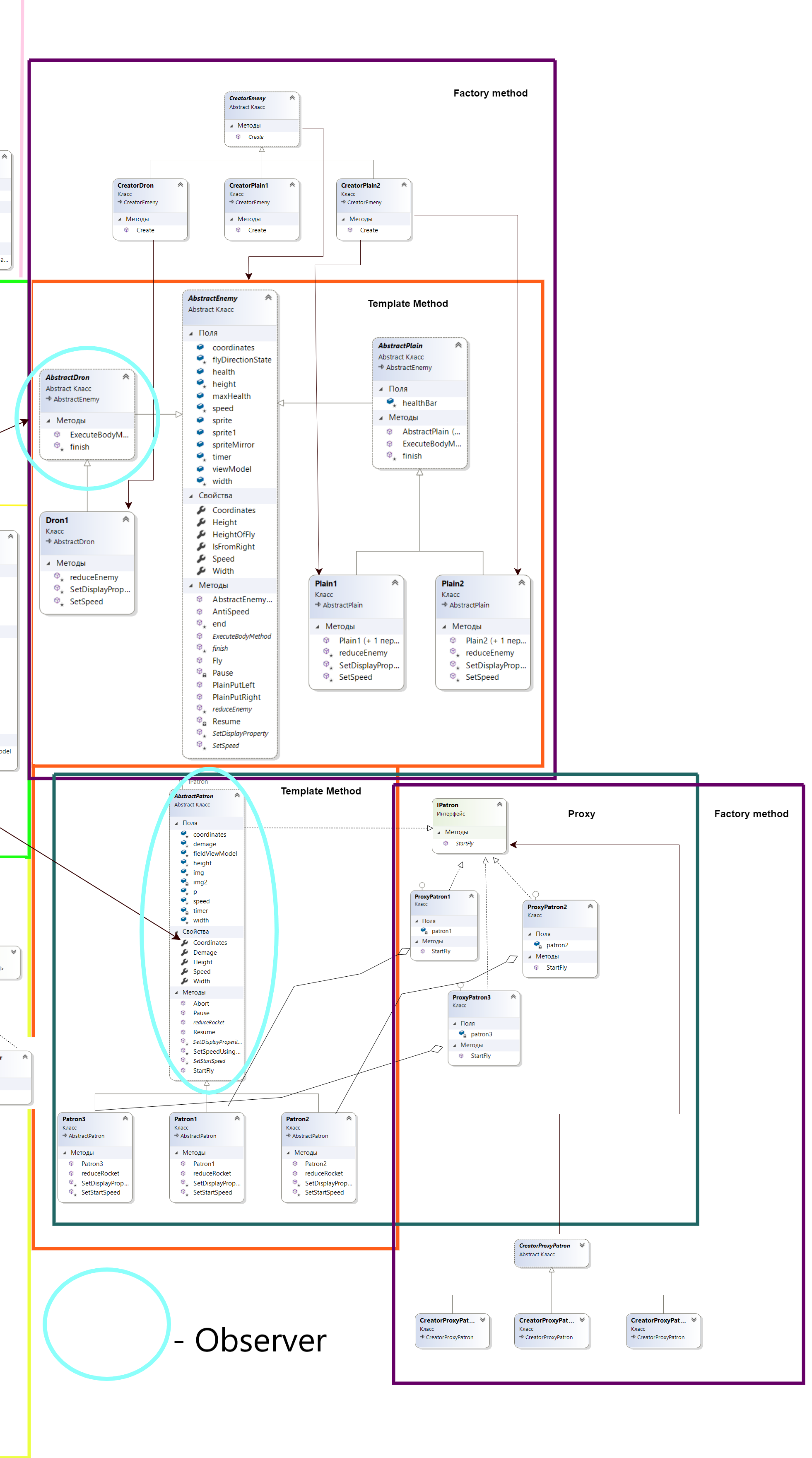
Шаблон складається лише з одного класу, тому це не можна назвати повноцінним шаблоном.

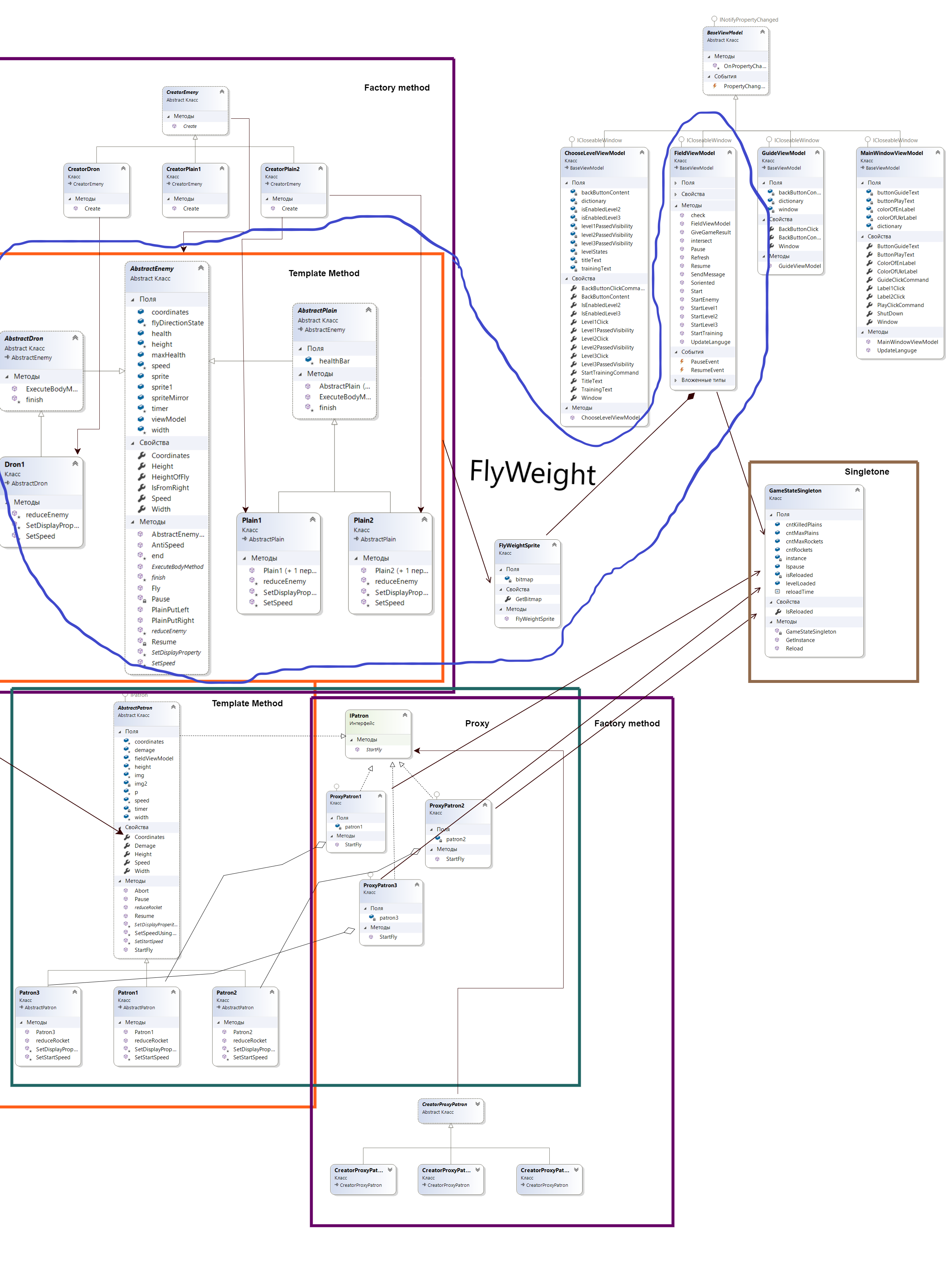
* 1.  **Діаграма класів**

*Рис. 2.2, аркуш 1. Діаграма класів гри “Glory to Ukraine”*



*Рис. 2.2, аркуш 2. Діаграма класів, фрагмент з Спостерігачем*

*Рис. 2.2, аркуш 3. Діаграма класів*



*Рис. 2.2, аркуш 4. Діаграма класів.*