МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **КНІТ**
Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи № 3
З дисципліни: *“Моделювання та аналіз програмного забезпечення”*
На тему: *“Твірні шаблони”*

Лектор:
доц. каф. ПЗ
Сердюк П.В.

Виконав:
ст. гр. ПЗ-22
Чаус Олег

Прийняв:
викл. каф. ПЗ
Микуляк А. В.

« ____ » _____ 2023 р.

∑= ____ _____.

Львів – 2023

Тема роботи: Твірні шаблони.

Мета роботи: Здобути навички використання твірних шаблонів проектування при моделюванні програмних систем.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Твірні шаблони (англ. Creational patterns) — це шаблони проєктування, що абстрагують процес побудови об'єктів. Вони допоможуть зробити систему незалежною від способу створення, композиції та представлення її об'єктів. Шаблон, який породжує класи використовує успадкування, щоб варіювати створюваний клас, а шаблон, що створює об'єкти,делегує інстанціювання іншому об'єктові. Ці шаблони важливі, коли система більше залежить від композиції об'єктів, ніж від успадкування класів. Таким чином, замість прямого кодування фіксованого набору поведінок, визначається невеликий набір фундаментальних поведінок, за допомогою композиції яких можна отримувати складніші. Тобто для створення об'єктів з конкретною поведінкою потрібно щось більше, ніж просте інстанціювання екземпляру класу. Шаблони, що породжують, інкапсулюють знання про конкретні класи, які застосовуються у системі та приховують деталі того, як ці класи створюються і стикуються між собою. Єдина інформація про об'єкти, що відома системі — їхні інтерфейси.

ЗАВДАННЯ

Розробити твірні шаблони проектування відповідно до прецедентів обраного варіанту ігрової логіки. Вибрати один з прецедентів, для якого найбільш доцільно застосувати твірний шаблон (не всі прецеденти цього потребуватимуть).

Обов’язкові шаблони для реалізації:

- Singleton
- Абстрактна фабрика

Реалізувати один із шаблонів на вибір:

- Будівельник
- Прототип

Представити діаграму класів ігрового проекту (із відображенням на ній використаних шаблонів).

UML-ДІАГРАМА КЛАСІВ

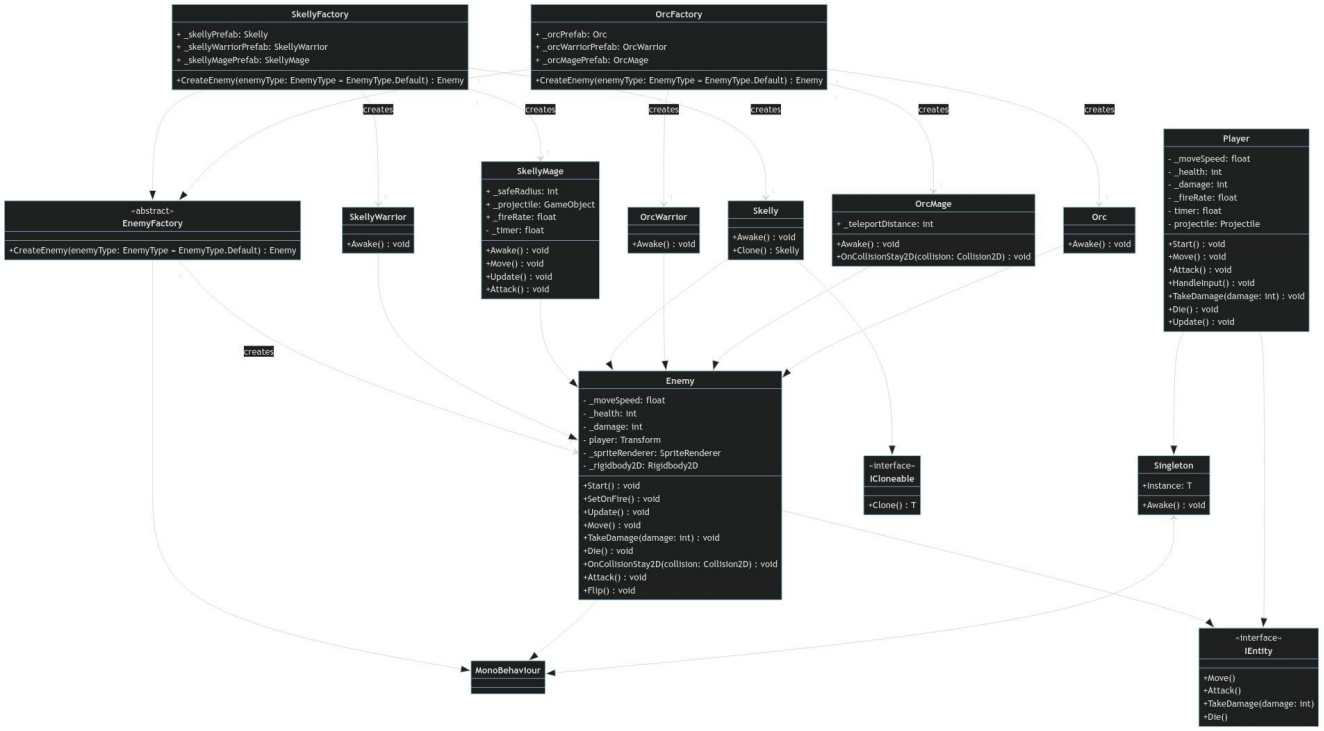

Рис. 1. Загальна UML-діаграма класів.

Було реалізовано три твірних патерни – абстрактна фабрика, одинак та прототип.

- Одинак

Одинак - це породжувальний патерн проектування, який гарантує, що клас має лише один екземпляр та надає глобальну точку доступу до нього.

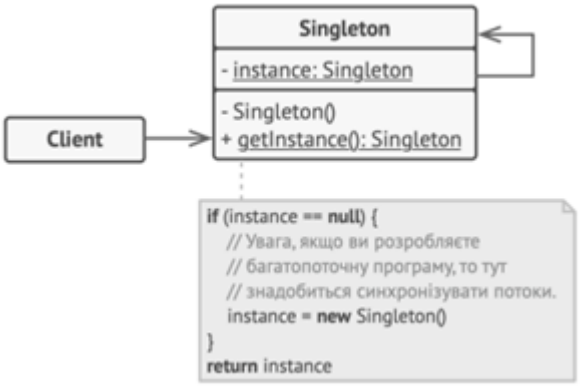

Рис. 2. Класичний випадок використання Singleton

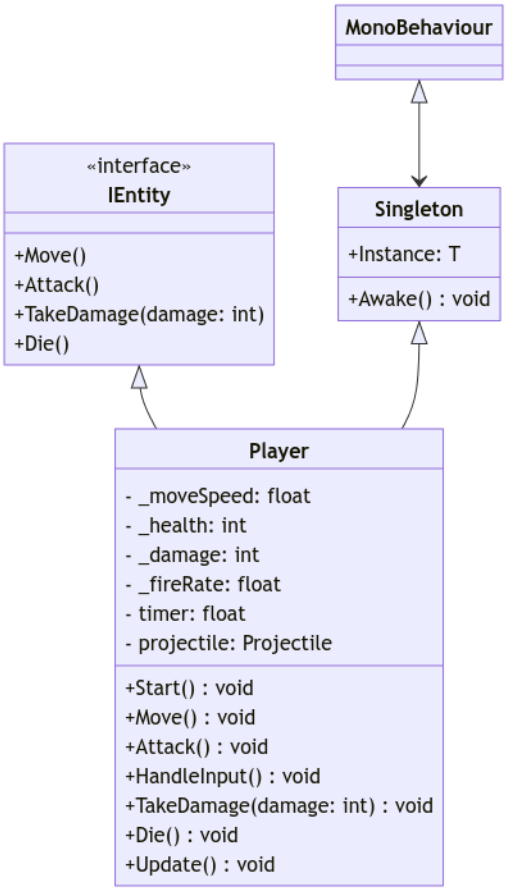

Рис. 3. Використання патерну Singleton.

```
public abstract class Singleton<T> : MonoBehaviour
{
   public static T Instance { get; private set; }
   protected virtual void Awake()
   {
       if (Instance == null)
       {
           Instance = GetComponent<T>();
       }
       else
       {
           Destroy(gameObject);
       }
   }
}
```

- Абстрактна фабрика

Абстрактна фабрика - це твірний патерн проектування, що дає змогу створювати сімейства пов’язаних об’єктів, не прив’язуючись до конкретних класів створюваних об’єктів.

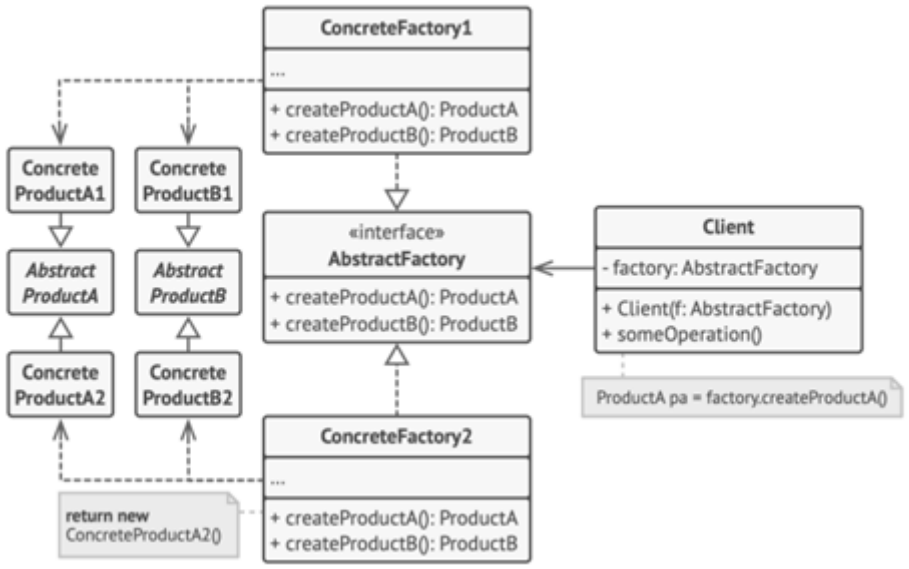

Рис. 4. Класичний випадок використання Abstract Factory.

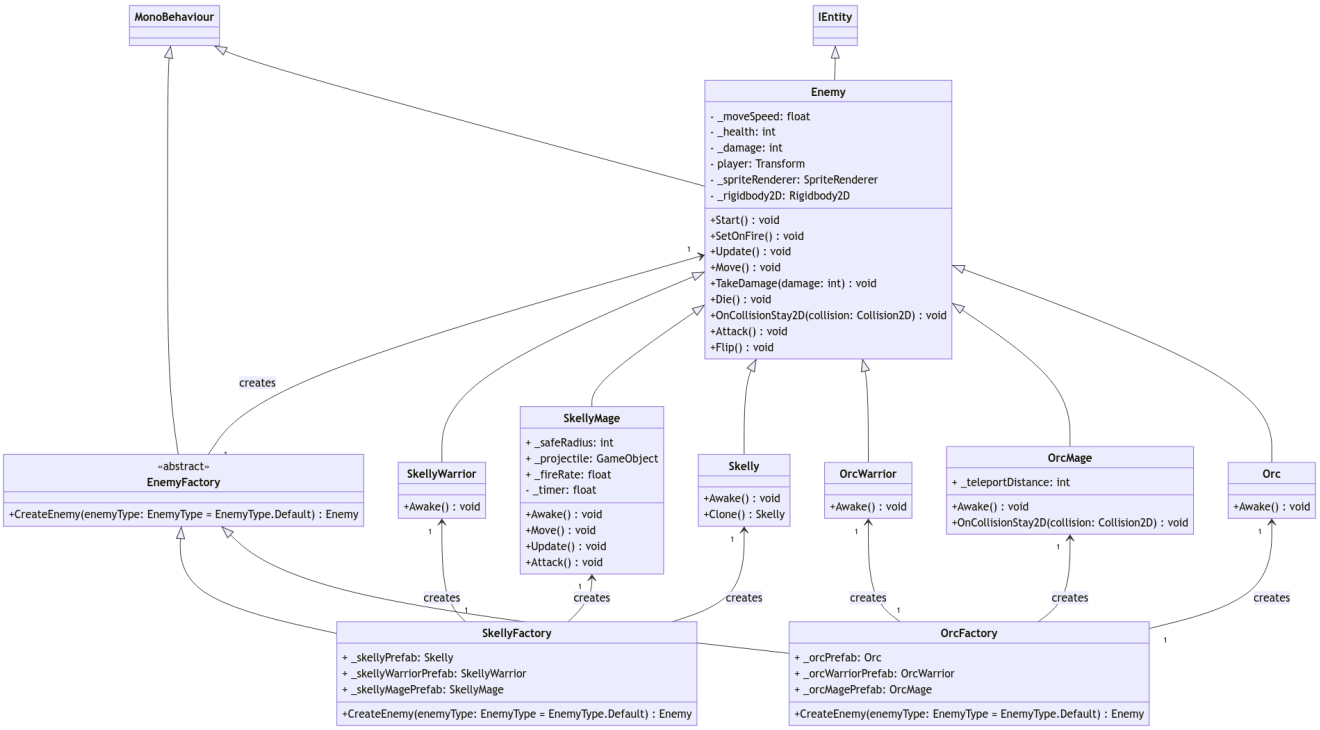

Рис. 5. Використання шаблону Abstract Factory.

```
public abstract class EnemyFactory : MonoBehaviour
{
   public abstract Enemy CreateEnemy(EnemyType enemyType = EnemyType.Default);
}
public class OrcFactory : EnemyFactory
{
   [SerializeField] private Orc _orcPrefab;
   [SerializeField] private OrcWarrior _orcWarriorPrefab;
```

```
    [SerializeField] private OrcMage _orcMagePrefab;
    public override Enemy CreateEnemy(EnemyType enemyType = EnemyType.Default)
    {
        Enemy component = null;
        if (enemyType == EnemyType.Default)
        {
            GameObject orc = Instantiate(_orcPrefab.gameObject);
            Orc orcComponent = orc.GetComponent<Orc>();
            component = orcComponent;
        }
        if (enemyType == EnemyType.Warrior)
        {
            GameObject orcWarrior = Instantiate(_orcWarriorPrefab.gameObject);
            OrcWarrior orcWarriorComponent =
orcWarrior.GetComponent<OrcWarrior>();

            component = orcWarriorComponent;
        }
        if (enemyType == EnemyType.Mage)
        {
            GameObject orcMage = Instantiate(_orcMagePrefab.gameObject);
            OrcMage orcMageComponent = orcMage.GetComponent<OrcMage>();

            component = orcMageComponent;
        }
        return component;
    }
}
public class SkellyFactory : EnemyFactory
{
    [SerializeField] private Skelly _skellyPrefab;
    [SerializeField] private SkellyWarrior _skellyWarriorPrefab;
    [SerializeField] private SkellyMage _skellyMagePrefab;
    public override Enemy CreateEnemy(EnemyType enemyType = EnemyType.Default)
    {
        Enemy component = null;
        if (enemyType == EnemyType.Default)
        {
            GameObject skelly = Instantiate(_skellyPrefab.gameObject);
            Skelly skellyComponent = skelly.GetComponent<Skelly>();

            component = skellyComponent;
        }
        else if (enemyType == EnemyType.Warrior)
        {
            GameObject skellyWarrior =
```

```
Instantiate(_skellyWarriorPrefab.gameObject);
            SkellyWarrior skellyWarriorComponent =
skellyWarrior.GetComponent<SkellyWarrior>();

            component = skellyWarriorComponent;
        }
        else if (enemyType == EnemyType.Mage)
        {
            GameObject skellyMage = Instantiate(_skellyMagePrefab.gameObject);
            SkellyMage skellyMageComponent =
skellyMage.GetComponent<SkellyMage>();

            component = skellyMageComponent;
        }

        return component;
    }
}
```

- Прототип

Прототип - це твірний патерн проектування, що дає змогу копіювати об’єкти, не вдаючись у подробиці їхньої реалізації.

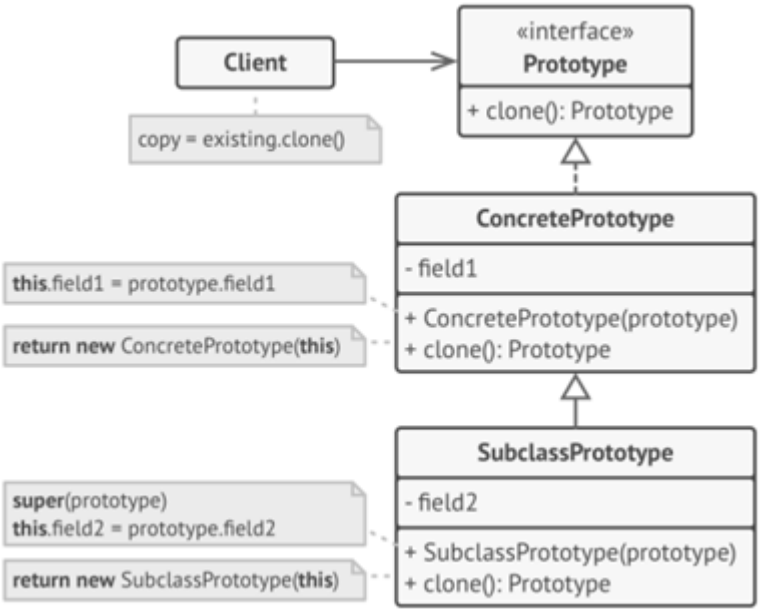

Рис. 6. Приклад використання патерну Prototype.

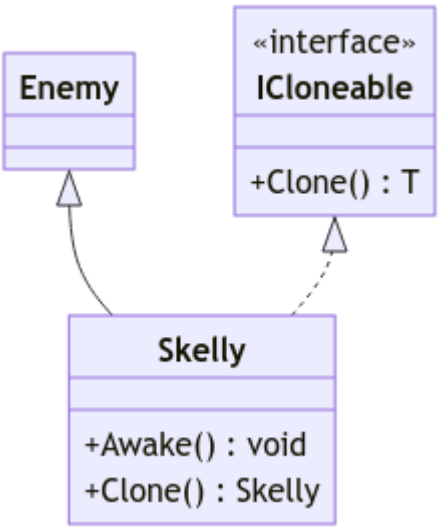

Рис. 7. Використання Прототипу.

```
public interface ICloneable<T>
{
   T Clone();
}
public class Skelly : Enemy, ICloneable<Skelly>
{
   private void Awake()
   {
       _moveSpeed = 4;
       _health = 15;
       _damage = 3;
   }

   public Skelly Clone()
   {
       return Instantiate(this);
   }
}
```

ВИСНОВКИ

Здобув навички використання твірних шаблонів проектування при моделюванні програмних систем.