Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної

техніки

Кафедра інформатики та програмної

інженерії

Звіт

з лабораторної роботи № 3 з дисципліни
«Основи розробки програмного забезпечення на платформі Microsoft.NET»

« Шаблони проектування. Породжуючі

шаблони »

Варіант 3

Виконав студент ІП-13 Григоренко Родіон Ярославович
(шифр, прізвище, ім'я, по батькові)

Перевірила Ліщук Катерина Ігорівна
(прізвище, ім'я, по батькові)

Київ 202 3

Лабораторна робота 2

Шаблони проектування. Породжуючі шаблони

Мета: - ознайомитися з основними шаблонами проектування, навчитися застосовувати їх при проектуванні і розробці ПЗ.

Постановка задачі:
При виконанні комп‘ютерного практикуму необхідно виконати наступні дії:

1) Вивчити породжуючі патерни. Знати загальну характеристику та призначення кожного з них, особливості реалізації кожного з породжуючих патернів та випадки їх застосування.

2) Реалізувати задачу згідно варіанту, запропонованого нижче у вигляді консольного застосування на мові C#. Розробити інтерфейси та класи з застосування одного або декількох патернів. Повністю реалізувати методи, пов‘язані з реалізацією обраного патерну.

3) Повністю описати архітектуру проекту (призначення методів та класів), особливості реалізації обраного патерну. Для кожного патерну необхідно вказати основні класи та їх призначення,

4) Навести UML-діаграму класів

5) Звіт повинен містити:

a. обґрунтування обраного патерну (чому саме він);

b. опис архітектури проекту (призначення методів та класів);

c. UML-діаграму класів

d. особливості реалізації обраного патерну

e. текст програмного коду

f. скріншоти результатів виконання.

Варіант 3

Необхідно реалізувати задачу «Гра - Цивілізації». У грі беруть участь кілька цивілізацій. У кожній цивілізації представлені індивідууми декількох видів: воїн, робочий, аристократ і ін. Кожна раса володіє фінансовим запасом і деяким набором територій з обов'язково розміщеними на них об'єктами типів: ліси, поля, житло і заводи. Продемонструвати створення різних рас, команд індивідуумів, ініціалізацію територій і зміну складу об'єктів гри.

1. Архітектура Проекту.

Виконуючи третю лабораторну роботу, було створено консольне застосування на мові C#, котре реалізує гру з завдання. Було створено 32 класи та 7 інтерфейсів, а саме:

Program.cs - власне клас програми з виконуваною функцією Main

Civilization.cs - клас Цивілізація

Land.cs - клас, який відображає землю, провінцію, тощо

Unit - клас, який відображає певну територіальну одиницю: Ліс, Поле, Житло, Фабрика

Forest -.клас, який відображає ліс

Field - клас, який відображає поле

Dwelling - клас, який відображає житло

Factory - клас, який відображає фабрику

Individual - клас, який відображає індивідуума

Warrior - клас, який відображає Воїна

Worker - клас, який відображає Робочого

Aristocrat - клас, який відображає Аристократа

Atlantis- клас, який відображає Цивілізацію Атлантида

AtlantisWarrior- клас, який відображає Воїна Атлантиди

AtlantisWorker - клас, який відображає Робочого Атлантиди

AtlantisAristocrat - клас, який відображає Аристократа Атлантиди

Hyperborea- клас, який відображає Цивілізацію Гіперборея

HyperboreanWarrior- клас, який відображає Воїна Гіпербореї

HyperboreanWorker - клас, який відображає Робочого Гіпербореї

HyperboreanAristocrat - клас, який відображає Аристократа Гіпербореї

Rome- клас, який відображає Цивілізацію Рим

RomanWarrior- клас, який відображає Воїна Риму

RomanWorker - клас, який відображає Робочого Риму

RomanAristocrat - клас, який відображає Аристократа Риму

Sparta- клас, який відображає Цивілізацію Спарта

SpartanWarrior- клас, який відображає Воїна Спарти

SpartanWorker - клас, який відображає Робочого Спарти

SpartanAristocrat - клас, який відображає Аристократа Спарти

AtlanticFactory - клас, який відображає фабрику для створення об’єктів, що відносяться до Цивілізації Атлантида

HyperboreanFactory - клас, який відображає фабрику для створення об’єктів, що відносяться до Цивілізації Гіперборея

RomanFactory - клас, який відображає фабрику для створення об’єктів, що відносяться до Цивілізації Рим

SpartanFactory - клас, який відображає фабрику для створення об’єктів, що відносяться до Цивілізації Спарта

ICivilizationFactory - інтерфейс, що забезпечує функціонал для фабрик різних Цивілізацій

IAttackable - інтерфейс, що забезпечує можливість класам успадкованим від нього бути атакованими

IIndividual - інтерфейс, що забезпечує функціонал для індивідуумів бути покращеними

IWarrior- інтерфейс, що забезпечує функціонал для Воїнів різних Цивілізацій

IWorker- інтерфейс, що забезпечує функціонал для Робочих різних Цивілізацій

IAristocrat- інтерфейс, що забезпечує функціонал для Аристократів різних Цивілізацій

IUsable - інтерфейс, що забезпечує функціонал для територіальних одиниць: Ліс, Поле, Житло, Фабрика

При виконанні лабораторної роботи була проаналізована задача і виходячи з цього було прийняте рішення використати кілька паттернів проектування, а саме: Абстракта Фабрика, Одинак та Компоновщик

Абстрактна Фабрика:

Через те що, завдання перебачає існування багатьох Цивілізацій і кожна Цивілізація має пов’язані класи як то різні індивідууми: Воїн, Робочий, Аристократ та різні землі і територіальні одиниці, було прийнято рішення використати паттер проектування Абстрактна Фабрика аби полегшити і спростити створення різних об’єктів пов’язаних з певною цивілізацією і зробити програму більш гнучкою.

Одинак:

Через те, що може існувати лише одна цивілізація певного виду, наприклад є клас Rome і може бути лише один екземпляр цього класу, тому було використано паттерн проектування Одинак, аби уніфікувати екземпляр цього класу і зробити доступним у всій області проекту.

Компоновщик:

Через те, що кожна Цивілізація має землі/провінції, а кожна земля/провінція має територіальні одиниці: Ліси, Поля, Житло, Фабрики, було використано паттерн проектування Компоновщик.

3.Код класів:

Civilization.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab3DotNet
{
    public class Civilization
    {
```

```
        private List<Individual> _individuals = new List<Individual>();
        private List<Land> _lands = new List<Land>();

        public List<Land> Lands => _lands;
        public List<Individual> Individuals => _individuals;
        public void AddIndividual(Individual individual)
        {
            _individuals.Add(individual);
        }

        public void AddLand(Land land)
        {
            _lands.Add(land);
        }

    }
}
```

Land.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab3DotNet
{
```

```
    public class Land : IAttackable
    {
        public string Name;
        private List<Unit> _units = new List<Unit>();

        public List<Unit> Units => _units;
        public Land(string name)
        {
            Name = name;
        }
        public void AddUnit(Unit unit)
        {
            _units.Add(unit);
        }
        public void ReceiveAttack()
        {
            Console.WriteLine($"{Name} is under attack");
        }
    }
}
```

Unit.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;

namespace Lab3DotNet
{
    public interface IUsable
    {
        void Use();
    }
    public class Unit
    {
        public int Size;
        public int Value;

        public Unit(int size, int value)
        {
            Size = size;
            Value = value;
        }

        public virtual void Use() { }
    }

    public class Forest : Unit, IUsable
    {
        public Forest(int size, int value) : base(size,value)
        {

        }
```

```
        public override void Use()
        {
            Console.WriteLine("The forest is being used. Wood is being
produced.");
        }
    }

    public class Field : Unit, IUsable
    {
        public Field(int size, int value) : base(size, value)
        {

        }
        public override void Use()
        {
            Console.WriteLine("The field is being used. Food is being produced.");
        }
    }

    public class Dwelling : Unit, IUsable
    {
        public Dwelling(int size, int value) : base(size, value)
        {

        }
        public override void Use()
        {
```

```
            Console.WriteLine("The dwelling is being used. Individuals are having
rest.");
        }
    }

    public class Factory : Unit, IUsable
    {
        public Factory(int size, int value) : base(size, value)
        {

        }
        public override void Use()
        {
            Console.WriteLine("The factory is being used. Things are being
produced.");
        }
    }
}
```

Individual.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Lab3DotNet
{
    public interface IAttackable
    {
        void ReceiveAttack();
    }
    public interface IIndividual
    {
        void Upgrade();
    }
    public class Individual : IAttackable
    {
        public string Name;
        public int Health;
        public int Power;
        public int Authority;
        public int Performance;

        public Individual(string name, int health, int power, int authority, int
performance)
        {
            Name = name;
            Health = health;
            Power = power;
            Authority = authority;
            Performance = performance;
        }
        public void ReceiveAttack()
```

```
        {
            Console.WriteLine($"{Name} is under attack");
        }
    }

    public interface IWarrior
    {
        void Attack(IAttackable target);
        void Defend(IAttackable target);
        void Train();
    }

    public interface IWorker
    {
        void Work();
    }

    public interface IAristocrat
    {
        void Rule();
    }



    public class Warrior : Individual, IWarrior, IIndividual
    {
        public Warrior(string name, int health, int power, int authority, int
performance) : base(name, health, power, authority, performance)
```

```
    {

    }

    public void Upgrade()
    {
        Health += 3;
        Power += 8;
        Authority += 1;
        Performance += 1;
    }

    public void Attack(IAttackable target)
    {
        Console.WriteLine("\n");
        if (target is Individual)
        {
            Individual IndividualTarget = (Individual)target;
            Console.WriteLine($"{Name} is attacking
{IndividualTarget.Name}");
            SpecificAttack();

        }
        else if (target is Land)
        {
            Land LandTarget = (Land)target;
            Console.WriteLine($"{Name} is attacking {LandTarget.Name}");
            SpecificAttack();
```

```
            }
            target.ReceiveAttack();
            Console.WriteLine("\n");
        }
        public void Defend(IAttackable target)
        {
            Console.WriteLine("\n");
            if (target is Individual)
            {
                Individual IndividualTarget = (Individual)target;
                Console.WriteLine($"{Name} is defending
{IndividualTarget.Name}");
                SpecificDefend();
            }
            else if (target is Land)
            {
                Land LandTarget = (Land)target;
                Console.WriteLine($"{Name} is defending {LandTarget.Name}");
                SpecificDefend();
            }
            Console.WriteLine("\n");
        }
        public void Train()
        {
            Console.WriteLine("\n");
            Console.WriteLine($"{Name} is raining");
            Console.WriteLine("\n");
        }
```

```
    public virtual void SpecificAttack() { }
    public virtual void SpecificDefend() { }
}

public class Worker : Individual, IWorker, IIndividual
{
    public Worker(string name, int health, int power, int authority, int performance) : base(name, health, power, authority, performance)
    {

    }
    public void Upgrade()
    {
        Health += 2;
        Power += 2;
        Authority += 2;
        Performance += 8;
    }
    public void Work()
    {
        Console.WriteLine("\n");
        Console.WriteLine($"{Name} is Working");
        Console.WriteLine("\n");
    }

}

public class Aristocrat : Individual, IAristocrat, IIndividual
```

```
    {
        public Aristocrat(string name, int health, int power, int authority, int performance) : base(name, health, power, authority, performance)
        {


        }
        public void Upgrade()
        {
            Health += 3;
            Power += 4;
            Authority += 8;
            Performance += 1;
        }
        public void Rule()
        {
            Console.WriteLine("\n");
            Console.WriteLine($"{Name} is Ruling");
            Console.WriteLine("\n");
        }
    }
}
```

Atlantis.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;

namespace Lab3DotNet
{
    public class Atlantis : Civilization
    {
        private static Atlantis instance;

        public static Atlantis Instance
        {
            get
            {
                if (instance == null)
                {
                    instance = new Atlantis();
                }
                return instance;
            }
        }

    }

    public class AtlanticWarrior : Warrior
    {

        public AtlanticWarrior(string name, int health, int power, int authority, int performance) : base(name, health, power, authority, performance)
```

```
        {

        }

        public override void SpecificAttack()
        {
            Console.WriteLine($"{Name}: Atlantic attack!");
        }

        public override void SpecificDefend()
        {
            Console.WriteLine($"{Name}: Atlantic defence!");
        }
    }

    public class AtlanticWorker : Worker
    {
        public AtlanticWorker(string name, int health, int power, int authority, int performance) : base(name, health, power, authority, performance)
        {

        }
    }

    public class AtlanticAristocrat : Aristocrat
    {
        public AtlanticAristocrat(string name, int health, int power, int authority, int performance) : base(name, health, power, authority, performance)
        {
```

```
        }
    }
}
```

Hyperborea.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab3DotNet
{
    class Hyperborea : Civilization
    {
        private static Hyperborea instance;

        public static Hyperborea Instance
        {
            get
            {
                if (instance == null)
                {
                    instance = new Hyperborea();
                }
```

```
            return instance;
        }
    }
}

public class HyperboreanWarrior : Warrior
{
    public HyperboreanWarrior(string name, int health, int power, int
authority, int performance) : base(name, health, power, authority, performance)
    {

    }
    public override void SpecificAttack()
    {
        Console.WriteLine($"{Name}: Massacre!");
    }

    public override void SpecificDefend()
    {
        Console.WriteLine($"{Name}: Massacre!");
    }
}

public class HyperboreanWorker : Worker
{
    public HyperboreanWorker(string name, int health, int power, int
authority, int performance) : base(name, health, power, authority, performance)
    {
```

```
        }
    }

    public class HyperboreanAristocrat : Aristocrat
    {
        public HyperboreanAristocrat(string name, int health, int power, int
authority, int performance) : base(name, health, power, authority, performance)
        {

        }
    }
}
```

Rome.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab3DotNet
{
    class Rome : Civilization
    {
        private static Rome instance;
```

```
        public static Rome Instance
        {
            get
            {
                if (instance == null)
                {
                    instance = new Rome();
                }
                return instance;
            }
        }
    }

    public class RomanWarrior : Warrior
    {
        public RomanWarrior(string name, int health, int power, int authority, int performance) : base(name, health, power, authority, performance)
        {

        }
        public override void SpecificAttack()
        {
            Console.WriteLine($"{Name}: Phalanx!");
        }

        public override void SpecificDefend()
        {
            Console.WriteLine($"{Name}: Phalanx!");
```

```
        }
    }

    public class RomanWorker : Worker
    {
        public RomanWorker(string name, int health, int power, int authority, int performance) : base(name, health, power, authority, performance)
        {

        }
    }

    public class RomanAristocrat : Aristocrat
    {
        public RomanAristocrat(string name, int health, int power, int authority, int performance) : base(name, health, power, authority, performance)
        {

        }
    }
}
```

Sparta.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;

namespace Lab3DotNet
{
    class Sparta : Civilization
    {
        private static Sparta instance;

        public static Sparta Instance
        {
            get
            {
                if (instance == null)
                {
                    instance = new Sparta();
                }
                return instance;
            }
        }
    }

    public class SpartanWarrior : Warrior
    {
        public SpartanWarrior(string name, int health, int power, int authority, int performance) : base(name, health, power, authority, performance)
        {
```

```
        }
        public override void SpecificAttack()
        {
            Console.WriteLine($"{Name}: Phalanx!");
        }

        public override void SpecificDefend()
        {
            Console.WriteLine($"{Name}: Phalanx!");
        }
    }

    public class SpartanWorker : Worker
    {
        public SpartanWorker(string name, int health, int power, int authority, int performance) : base(name, health, power, authority, performance)
        {

        }
    }

    public class SpartanAristocrat : Aristocrat
    {
        public SpartanAristocrat(string name, int health, int power, int authority, int performance) : base(name, health, power, authority, performance)
        {

        }
    }
```

```
}
```

AbstractFactory.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab3DotNet
{
    public interface ICivilizationFactory
    {
        IWarrior CreateWarrior(string name);
        IWorker CreateWorker(string name);
        IAristocrat CreateAristocrat(string name);
    }

    public class AtlanticFactory : ICivilizationFactory
    {
        public IWarrior CreateWarrior(string name)
        {
            return new AtlanticWarrior(name,10,10,2,2);
        }

        public IWorker CreateWorker(string name)
```

```
    {
        return new AtlanticWorker(name, 7, 2, 2, 10);
    }

    public IAristocrat CreateAristocrat(string name)
    {
        return new AtlanticAristocrat(name, 7, 5, 10, 2);
    }
}

public class HyperboreanFactory : ICivilizationFactory
{
    public IWarrior CreateWarrior(string name)
    {
        return new HyperboreanWarrior(name, 15, 20, 2, 2);
    }

    public IWorker CreateWorker(string name)
    {
        return new HyperboreanWorker(name, 7, 5, 2, 10);
    }

    public IAristocrat CreateAristocrat(string name)
    {
        return new HyperboreanAristocrat(name, 10, 10, 15, 3);
    }
}
```

```
public class RomanFactory : ICivilizationFactory
{
    public IWarrior CreateWarrior(string name)
    {
        return new RomanWarrior(name, 15, 15, 5, 3);
    }

    public IWorker CreateWorker(string name)
    {
        return new RomanWorker(name, 7, 4, 3, 10);
    }

    public IAristocrat CreateAristocrat(string name)
    {
        return new RomanAristocrat(name, 10, 8, 20, 3);
    }
}

public class SpartanFactory : ICivilizationFactory
{
    public IWarrior CreateWarrior(string name)
    {
        return new SpartanWarrior(name, 15, 20, 8, 3);
    }

    public IWorker CreateWorker(string name)
    {
        return new SpartanWorker(name, 10, 8, 3, 10);
```

```
        }

        public IAristocrat CreateAristocrat(string name)
        {
            return new SpartanAristocrat(name, 10, 8, 10, 3);
        }
    }

}
```

Program.cs

```
using System;

namespace Lab3DotNet
{
    class Program
    {
        static void Main(string[] args)
        {
            HyperboreanFactory HyperboreanFactory = new HyperboreanFactory();
            RomanFactory RomanFactory = new RomanFactory();
            SpartanFactory SpartanFactory = new SpartanFactory();


Hyperborea.Instance.AddIndividual((Individual)HyperboreanFactory.CreateWa
rrior("Erik"));
```

```
Hyperborea.Instance.AddIndividual((Individual)HyperboreanFactory.CreateWarrior("Bjørn"));

Hyperborea.Instance.AddIndividual((Individual)HyperboreanFactory.CreateWarrior("Arne"));

Hyperborea.Instance.AddIndividual((Individual)HyperboreanFactory.CreateWorker("Halfdan"));

Hyperborea.Instance.AddIndividual((Individual)HyperboreanFactory.CreateWorker("Njal"));

Hyperborea.Instance.AddIndividual((Individual)HyperboreanFactory.CreateWorker("Sten"));

Hyperborea.Instance.AddIndividual((Individual)HyperboreanFactory.CreateAristocrat("Harald"));

Hyperborea.Instance.AddIndividual((Individual)HyperboreanFactory.CreateAristocrat("Ødger"));

Hyperborea.Instance.AddIndividual((Individual)HyperboreanFactory.CreateAristocrat("Ulf"));


        Hyperborea.Instance.AddLand(new Land("Aebeltoft"));
        Hyperborea.Instance.AddLand(new Land("Ravndal"));
        Hyperborea.Instance.AddLand(new Land("Sandvik"));


        Hyperborea.Instance.Lands[0].AddUnit(new Forest(10,10));
        Hyperborea.Instance.Lands[0].AddUnit(new Field(20, 5));
        Hyperborea.Instance.Lands[0].AddUnit(new Dwelling(5, 10));
```

```
        Hyperborea.Instance.Lands[1].AddUnit(new Forest(30, 50));
        Hyperborea.Instance.Lands[1].AddUnit(new Field(50, 30));
        Hyperborea.Instance.Lands[1].AddUnit(new Dwelling(10, 20));
        Hyperborea.Instance.Lands[1].AddUnit(new Factory(5, 100));

        Hyperborea.Instance.Lands[2].AddUnit(new Forest(10, 15));
        Hyperborea.Instance.Lands[2].AddUnit(new Forest(20, 25));
        Hyperborea.Instance.Lands[2].AddUnit(new Field(20, 15));


Rome.Instance.AddIndividual((Individual)RomanFactory.CreateWarrior("Luciu
s"));

Rome.Instance.AddIndividual((Individual)RomanFactory.CreateWarrior("Sever
us"));

Rome.Instance.AddIndividual((Individual)RomanFactory.CreateWarrior("Marc
us"));

Rome.Instance.AddIndividual((Individual)RomanFactory.CreateWorker("Tiberi
us"));

Rome.Instance.AddIndividual((Individual)RomanFactory.CreateWorker("Quint
us"));

Rome.Instance.AddIndividual((Individual)RomanFactory.CreateWorker("Publi
us"));

Rome.Instance.AddIndividual((Individual)RomanFactory.CreateAristocrat("Alb
us"));

Rome.Instance.AddIndividual((Individual)RomanFactory.CreateAristocrat("Att
icus"));
```

```
Rome.Instance.AddIndividual((Individual)RomanFactory.CreateAristocrat("Caesar"));

        Rome.Instance.AddLand(new Land("Gallia"));
        Rome.Instance.AddLand(new Land("Italia"));
        Rome.Instance.AddLand(new Land("Britannia"));

        Rome.Instance.Lands[0].AddUnit(new Forest(10, 10));
        Rome.Instance.Lands[0].AddUnit(new Field(30, 10));
        Rome.Instance.Lands[0].AddUnit(new Dwelling(20, 40));

        Rome.Instance.Lands[1].AddUnit(new Forest(30, 50));
        Rome.Instance.Lands[1].AddUnit(new Field(50, 30));
        Rome.Instance.Lands[1].AddUnit(new Dwelling(10, 20));
        Rome.Instance.Lands[1].AddUnit(new Factory(5, 100));

        Rome.Instance.Lands[2].AddUnit(new Forest(15, 20));
        Rome.Instance.Lands[2].AddUnit(new Forest(20, 30));
        Rome.Instance.Lands[2].AddUnit(new Field(30, 50));


Sparta.Instance.AddIndividual((Individual)SpartanFactory.CreateWarrior("Achilles"));

Sparta.Instance.AddIndividual((Individual)SpartanFactory.CreateWarrior("Ares"));

Sparta.Instance.AddIndividual((Individual)SpartanFactory.CreateWarrior("Hercules"));
```

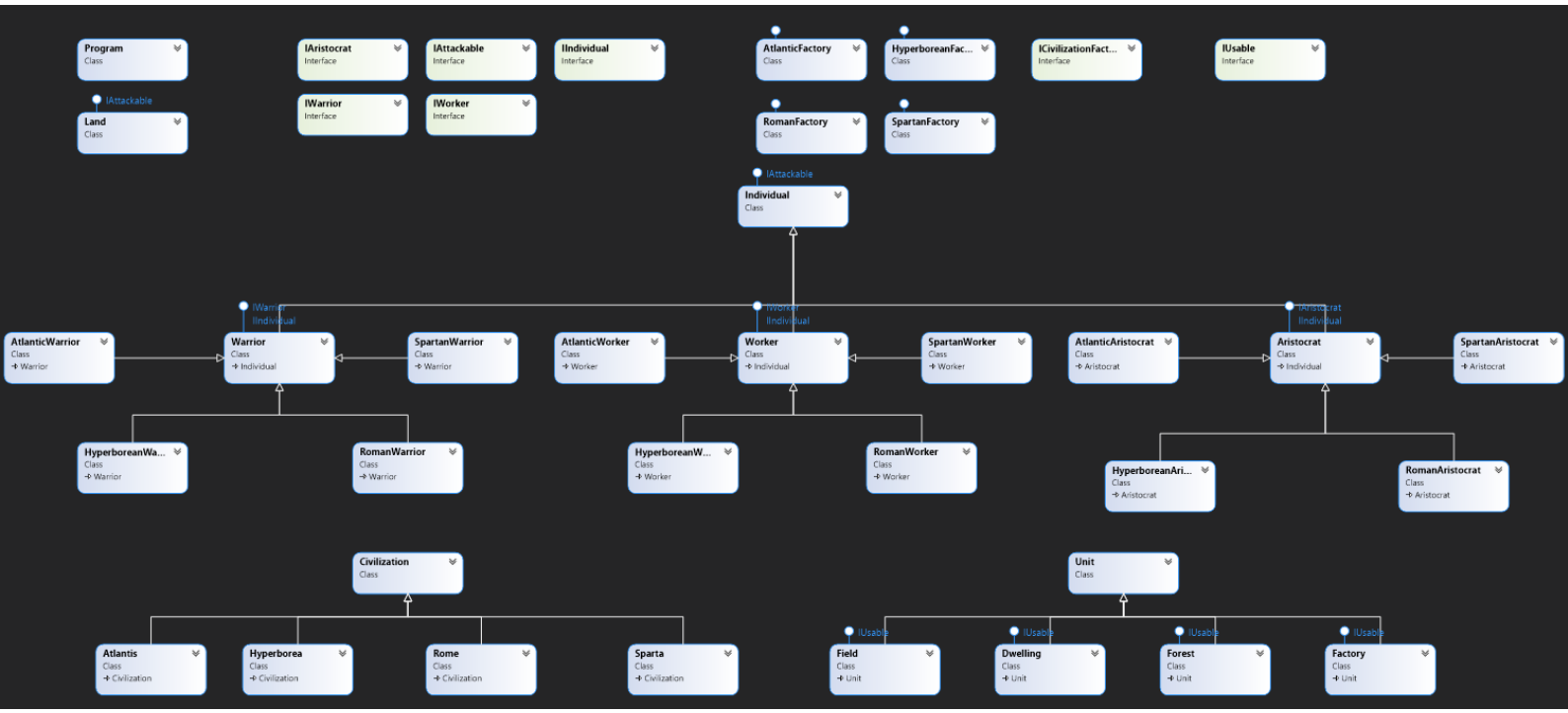
```
Sparta.Instance.AddIndividual((Individual)SpartanFactory.CreateWorker("Orpheus"));

Sparta.Instance.AddIndividual((Individual)SpartanFactory.CreateWorker("Hector"));

Sparta.Instance.AddIndividual((Individual)SpartanFactory.CreateWorker("Alpheus"));

Sparta.Instance.AddIndividual((Individual)SpartanFactory.CreateAristocrat("Aragorn"));

Sparta.Instance.AddIndividual((Individual)SpartanFactory.CreateAristocrat("Perseus"));

Sparta.Instance.AddIndividual((Individual)SpartanFactory.CreateAristocrat("Leonidas"));

        Sparta.Instance.AddLand(new Land("Arcadia"));
        Sparta.Instance.AddLand(new Land("Laconia"));
        Sparta.Instance.AddLand(new Land("Achaea"));

        Sparta.Instance.Lands[0].AddUnit(new Forest(30, 50));
        Sparta.Instance.Lands[0].AddUnit(new Field(50, 50));
        Sparta.Instance.Lands[0].AddUnit(new Dwelling(15, 40));

        Sparta.Instance.Lands[1].AddUnit(new Forest(30, 50));
        Sparta.Instance.Lands[1].AddUnit(new Field(50, 30));
        Sparta.Instance.Lands[1].AddUnit(new Dwelling(10, 20));
        Sparta.Instance.Lands[1].AddUnit(new Factory(5, 100));
```

```
        Sparta.Instance.Lands[2].AddUnit(new Forest(10, 20));
        Sparta.Instance.Lands[2].AddUnit(new Dwelling(20, 50));
        Sparta.Instance.Lands[2].AddUnit(new Field(30, 50));


((Warrior)Hyperborea.Instance.Individuals[0]).Attack(Rome.Instance.Individua
ls[3]);

((Warrior)Hyperborea.Instance.Individuals[1]).Attack(Rome.Instance.Individua
ls[5]);

((Warrior)Hyperborea.Instance.Individuals[2]).Attack(Rome.Instance.Individua
ls[7]);

((Warrior)Hyperborea.Instance.Individuals[0]).Attack(Rome.Instance.Individua
ls[6]);

((Warrior)Hyperborea.Instance.Individuals[1]).Attack(Rome.Instance.Individua
ls[4]);


((Warrior)Rome.Instance.Individuals[0]).Defend(Rome.Instance.Individuals[3])
;

((Warrior)Rome.Instance.Individuals[1]).Defend(Rome.Instance.Individuals[5])
;

((Warrior)Rome.Instance.Individuals[2]).Defend(Rome.Instance.Individuals[7])
;

((Warrior)Rome.Instance.Individuals[0]).Defend(Rome.Instance.Individuals[6])
;
```

```
((Warrior)Rome.Instance.Individuals[1]).Defend(Rome.Instance.Individuals[4])
;


((Warrior)Rome.Instance.Individuals[0]).Attack(Hyperborea.Instance.Individua
ls[0]);

((Warrior)Rome.Instance.Individuals[1]).Attack(Hyperborea.Instance.Individua
ls[1]);

((Warrior)Rome.Instance.Individuals[2]).Attack(Hyperborea.Instance.Individua
ls[2]);


((Warrior)Sparta.Instance.Individuals[0]).Defend(Rome.Instance.Individuals[0]
);

((Warrior)Sparta.Instance.Individuals[1]).Defend(Rome.Instance.Individuals[1]
);

((Warrior)Sparta.Instance.Individuals[2]).Defend(Rome.Instance.Individuals[2]
);


((Warrior)Sparta.Instance.Individuals[0]).Attack(Hyperborea.Instance.Lands[0]
);

((Warrior)Sparta.Instance.Individuals[1]).Attack(Hyperborea.Instance.Lands[1]
);

((Warrior)Sparta.Instance.Individuals[2]).Attack(Hyperborea.Instance.Lands[2]
);
```

```
((Warrior)Hyperborea.Instance.Individuals[0]).Defend(Hyperborea.Instance.La
nds[0]);

((Warrior)Hyperborea.Instance.Individuals[1]).Defend(Hyperborea.Instance.La
nds[1]);

((Warrior)Hyperborea.Instance.Individuals[2]).Defend(Hyperborea.Instance.La
nds[2]);


            Rome.Instance.Lands[0].Units[0].Use();
            Rome.Instance.Lands[0].Units[1].Use();
            Rome.Instance.Lands[2].Units[2].Use();

            Hyperborea.Instance.Lands[1].Units[0].Use();
            Hyperborea.Instance.Lands[1].Units[1].Use();
            Hyperborea.Instance.Lands[1].Units[2].Use();
            Hyperborea.Instance.Lands[1].Units[3].Use();

            Sparta.Instance.Lands[0].Units[0].Use();
            Sparta.Instance.Lands[1].Units[1].Use();
            Sparta.Instance.Lands[2].Units[2].Use();
        }
    }
}
```

4.UML-діаграма класів



6.Результати виконання

В Program.cs були творені різні індивідууми, землі та територіальні одиниціта додані до відповідних цивілізацій, після чого демонструється функціонал цих об’єктів.

Воїни Гіпербореї атакують індивідуумів Риму:

```
((Warrior)Hyperborea.Instance.Individuals[0]).Attack(Rome.Instance.Individuals[3]);
((Warrior)Hyperborea.Instance.Individuals[1]).Attack(Rome.Instance.Individuals[5]);
((Warrior)Hyperborea.Instance.Individuals[2]).Attack(Rome.Instance.Individuals[7]);
((Warrior)Hyperborea.Instance.Individuals[0]).Attack(Rome.Instance.Individuals[6]);
((Warrior)Hyperborea.Instance.Individuals[1]).Attack(Rome.Instance.Individuals[4]);
```

```
Erik is attacking Tiberius
Erik: Massacre!
Tiberius is under attack



Bjorn is attacking Publius
Bjorn: Massacre!
Publius is under attack



Arne is attacking Atticus
Arne: Massacre!
Atticus is under attack



Erik is attacking Albus
Erik: Massacre!
Albus is under attack



Bjorn is attacking Quintus
Bjorn: Massacre!
Quintus is under attack
```

Воїни Риму захищають індивідуумів Риму:

```
((Warrior)Rome.Instance.Individuals[0]).Defend(Rome.Instance.Individuals[3]);
((Warrior)Rome.Instance.Individuals[1]).Defend(Rome.Instance.Individuals[5]);
((Warrior)Rome.Instance.Individuals[2]).Defend(Rome.Instance.Individuals[7]);
((Warrior)Rome.Instance.Individuals[0]).Defend(Rome.Instance.Individuals[6]);
((Warrior)Rome.Instance.Individuals[1]).Defend(Rome.Instance.Individuals[4]);
```

```
Lucius is defending Tiberius
Lucius: Phalanx!



Severus is defending Publius
Severus: Phalanx!



Marcus is defending Atticus
Marcus: Phalanx!



Lucius is defending Albus
Lucius: Phalanx!



Severus is defending Quintus
Severus: Phalanx!
```

Воїни Риму атакують індивідуумів Гіперборей

```
((Warrior)Rome.Instance.Individuals[0]).Attack(Hyperborea.Instance.Individuals[0]);
((Warrior)Rome.Instance.Individuals[1]).Attack(Hyperborea.Instance.Individuals[1]);
((Warrior)Rome.Instance.Individuals[2]).Attack(Hyperborea.Instance.Individuals[2]);
```

```
Lucius is attacking Erik
Lucius: Phalanx!
Erik is under attack



Severus is attacking Bjorn
Severus: Phalanx!
Bjorn is under attack



Marcus is attacking Arne
Marcus: Phalanx!
Arne is under attack
```

Воїни Спарти захищають індивідуумів Риму

```
((Warrior)Sparta.Instance.Individuals[0]).Defend(Rome.Instance.Individuals[0]);
((Warrior)Sparta.Instance.Individuals[1]).Defend(Rome.Instance.Individuals[1]);
((Warrior)Sparta.Instance.Individuals[2]).Defend(Rome.Instance.Individuals[2]);
```

```
Achilles is defending Lucius
Achilles: Phalanx!



Ares is defending Severus
Ares: Phalanx!



Hercules is defending Marcus
Hercules: Phalanx!
```

Воїни Спарти атакують землі Гіпербореї

```
((Warrior)Sparta.Instance.Individuals[0]).Attack(Hyperborea.Instance.Lands[0]);
((Warrior)Sparta.Instance.Individuals[1]).Attack(Hyperborea.Instance.Lands[1]);
((Warrior)Sparta.Instance.Individuals[2]).Attack(Hyperborea.Instance.Lands[2]);
```

```
Achilles is attacking Aebeltoft
Achilles: Phalanx!
Aebeltoft is under attack



Ares is attacking Ravndal
Ares: Phalanx!
Ravndal is under attack



Hercules is attacking Sandvik
Hercules: Phalanx!
Sandvik is under attack
```

Воїни Гіпербореї захищають землі Гіпербореї

```
((Warrior)Hyperborea.Instance.Individuals[0]).Defend(Hyperborea.Instance.Lands[0]);
((Warrior)Hyperborea.Instance.Individuals[1]).Defend(Hyperborea.Instance.Lands[1]);
((Warrior)Hyperborea.Instance.Individuals[2]).Defend(Hyperborea.Instance.Lands[2]);
```

```
Erik is defending Aebeltoft
Erik: Massacre!



Bjorn is defending Ravndal
Bjorn: Massacre!



Arne is defending Sandvik
Arne: Massacre!
```

Використання територіальних одиниць різними Цивілізаціями

```
Rome.Instance.Lands[0].Units[0].Use();
Rome.Instance.Lands[0].Units[1].Use();
Rome.Instance.Lands[2].Units[2].Use();

Hyperborea.Instance.Lands[1].Units[0].Use();
Hyperborea.Instance.Lands[1].Units[1].Use();
Hyperborea.Instance.Lands[1].Units[2].Use();
Hyperborea.Instance.Lands[1].Units[3].Use();

Sparta.Instance.Lands[0].Units[0].Use();
Sparta.Instance.Lands[1].Units[1].Use();
Sparta.Instance.Lands[2].Units[2].Use();
```

```
The forest is being used. Wood is being produced.
The field is being used. Food is being produced.
The field is being used. Food is being produced.
The forest is being used. Wood is being produced.
The field is being used. Food is being produced.
The dwelling is being used. Individuals are having rest.
The factory is being used. Things are being produced.
The forest is being used. Wood is being produced.
The field is being used. Food is being produced.
The field is being used. Food is being produced.
```

Висновок

Виконуючи дану лабораторну роботу, я покращив свої навички створення програмного забезпечення на мові C#, створив проект, використавши різні паттерни проектування, вивчив усі паттерни проектування, зрозумів як їх застосовувати в реальних проектах, та закріпив теоретичні знання на практиці.