Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра обчислювальної техніки

Практична робота № 3

з дисципліни «Шаблони проектування програмного забезпечення»

Тема: «Застосування структурних шаблонів»

Виконав

студент групи 1КІ-21б Квятковський А. А.

Перевірила

Войцеховська О. В.

Вінниця — 2025

Варіант — 19 Написати програму-симулятор створення армії для комп’ютерної гри (мечник, лучник, кірасир) з використанням структурного шаблона Bridge. Фабрики – це абстракції. Класи фабрик реалізувати за допомогою породжувального шаблона Factory Method. Реалізації використати з попередньої практичної роботи.

WarriorFactory: Інтерфейс для фабрик воїнів. Визначає метод CreateWarrior(string name), який створює об’єкт типу Warrior.

IArmor: Інтерфейс для броні. Визначає властивості Defense (захист) і Weight (вага), а також метод Equip() для екіпірування броні.

LightArmor: Реалізує IArmor. Має захист 5 і вагу 10. Метод Equip() виводить повідомлення про екіпірування легкої броні із зазначенням захисту та ваги.

MediumArmor: Реалізує IArmor. Має захист 10 і вагу 20. Метод Equip() виводить повідомлення про екіпірування середньої броні із зазначенням захисту та ваги.

HeavyArmor: Реалізує IArmor. Має захист 15 і вагу 30. Метод Equip() виводить повідомлення про екіпірування важкої броні із зазначенням захисту та ваги.

Warrior: Абстрактний клас, що представляє воїна. Містить захищені поля \_name (ім’я) та \_armor (броня типу IArmor). Конструктор ініціалізує ім’я та броню. Визначає абстрактний метод GetInfo(), який повертає інформацію про воїна.

Swordsman: Наслідується від Warrior. Представляє мечника. Конструктор передає ім’я та броню базовому класу. Метод GetInfo() викликає Equip() броні та повертає рядок із описом мечника, його захисту та впливу ваги броні на швидкість.

Archer: Наслідується від Warrior. Представляє лучника. Конструктор передає ім’я та броню базовому класу. Метод GetInfo() викликає Equip() броні та повертає рядок із описом лучника, його захисту та впливу ваги броні на точність.

Cuirassier: Наслідується від Warrior. Представляє кірасира. Конструктор передає ім’я та броню базовому класу. Метод GetInfo() викликає Equip() броні та повертає рядок із описом кірасира, його захисту та впливу ваги броні на маневреність.

SwordsmanFactory: Реалізує IWarriorFactory. Метод CreateWarrior(string name) створює та повертає екземпляр Swordsman із середньою бронею (MediumArmor).

ArcherFactory: Реалізує IWarriorFactory. Метод CreateWarrior(string name) створює та повертає екземпляр Archer із важкою бронею (HeavyArmor).

CuirassierFactory: Реалізує IWarriorFactory. Містить поле \_armor типу IArmor, яке задається через конструктор. Метод CreateWarrior(string name) створює та повертає екземпляр Cuirassier із заданою бронею.

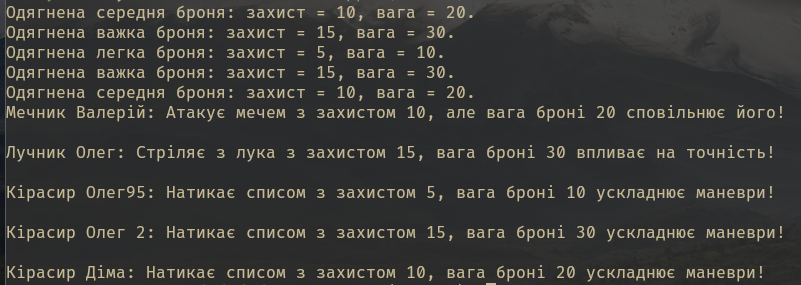


Рисунок 1 — Результат роботи

Висновок: Отже, під час лабораторної роботи було досліджено роботу з структурними шаблонами проектування, таки як: фабричний метод, міст, декоратор і проксі.

Лістинг програми

using System.Text;

Console.OutputEncoding = Encoding.UTF8;

// Створення фабрик

var swordsmanFactory = new SwordsmanFactory();

var archerFactory = new ArcherFactory();

var cuirassierFactory = new CuirassierFactory(new LightArmor());

// Створення воїнів

var swordsman = swordsmanFactory.CreateWarrior("Валерій").GetInfo();

var archer = archerFactory.CreateWarrior("Олег").GetInfo();

var cuirassier1 = cuirassierFactory.CreateWarrior("Олег95").GetInfo();

cuirassierFactory = new CuirassierFactory(new HeavyArmor());

var cuirassier2 = cuirassierFactory.CreateWarrior("Олег 2").GetInfo();

cuirassierFactory = new CuirassierFactory(new MediumArmor());

var cuirassier3 = cuirassierFactory.CreateWarrior("Діма").GetInfo();

Console.WriteLine(swordsman);

Console.WriteLine();

Console.WriteLine(archer);

Console.WriteLine();

Console.WriteLine(cuirassier1);

Console.WriteLine();

Console.WriteLine(cuirassier2);

Console.WriteLine();

Console.WriteLine(cuirassier3);

return 0;

public interface IArmor

{

int Defense { get; }

int Weight { get; }

void Equip();

}

public class LightArmor : IArmor

{

public int Defense { get; } = 5;

public int Weight { get; } = 10;

public void Equip()

{

Console.WriteLine($"Одягнена легка броня: захист = {Defense}, вага = {Weight}.");

}

}

public class HeavyArmor : IArmor

{

public int Defense { get; } = 15;

public int Weight { get; } = 30;

public void Equip()

{

Console.WriteLine($"Одягнена важка броня: захист = {Defense}, вага = {Weight}.");

}

}

public class MediumArmor : IArmor

{

public int Defense { get; } = 10;

public int Weight { get; } = 20;

public void Equip()

{

Console.WriteLine($"Одягнена середня броня: захист = {Defense}, вага = {Weight}.");

}

}

public abstract class Warrior

{

protected readonly IArmor \_armor;

protected readonly string \_name;

protected Warrior(string name, IArmor armor)

{

\_name = name;

\_armor = armor;

}

public abstract string GetInfo();

}

public class Swordsman : Warrior

{

public Swordsman(string name, IArmor armor) : base(name, armor) { }

public override string GetInfo()

{

\_armor.Equip();

return $"Мечник {\_name}: Атакує мечем з захистом {\_armor.Defense}, але вага броні {\_armor.Weight} сповільнює його!";

}

}

public class Archer : Warrior

{

public Archer(string name, IArmor armor) : base(name, armor) { }

public override string GetInfo()

{

\_armor.Equip();

return $"Лучник {\_name}: Стріляє з лука з захистом {\_armor.Defense}, вага броні {\_armor.Weight} впливає на точність!";

}

}

public class Cuirassier : Warrior

{

public Cuirassier(string name, IArmor armor) : base(name, armor) { }

public override string GetInfo()

{

\_armor.Equip();

return $"Кірасир {\_name}: Натикає списом з захистом {\_armor.Defense}, вага броні {\_armor.Weight} ускладнює маневри!";

}

}

public interface IWarriorFactory

{

Warrior CreateWarrior(string name);

}

public class SwordsmanFactory : IWarriorFactory

{

public Warrior CreateWarrior(string name) => new Swordsman(name, new MediumArmor());

}

public class ArcherFactory : IWarriorFactory

{

public Warrior CreateWarrior(string name) => new Archer(name, new HeavyArmor());

}

public class CuirassierFactory : IWarriorFactory

{

private readonly IArmor \_armor;

public CuirassierFactory(IArmor armor)=> \_armor = armor;

public Warrior CreateWarrior(string name) => new Cuirassier(name, \_armor);

}