Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра обчислювальної техніки

Практична робота № 4

з дисципліни «Шаблони проектування програмного забезпечення»

Тема: «Практичне використання шаблонів поведінки»

Виконав

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

Перевірила

Войцеховська О. В.

Вінниця — 2025

Варіант — 19 Розробити магазин продажу юнітів для продуктів з програми-симулятора створення армії для комп’ютерної гри (з практичної роботи No 3). Реалізувати три алгоритми формування ціни на товар (звичайна ціна, акційна ціна та суперціна). Задачу розв’язати за допомогою шаблона Strategy. Реалізувати сповіщення VIP ігроків про поточні акції з використанням шаблона Observer.

Warrior: Абстрактний клас, що представляє воїна. Містить властивості Name (ім’я) і BasePrice (базова ціна). Конструктор ініціалізує ім’я та базову ціну. Визначає абстрактну властивість Type, яка вказує тип воїна. Метод ToString() повертає рядок із назвою, типом воїна та базовою ціною.

Swordsman: Наслідується від Warrior. Представляє мечника. Конструктор передає ім’я та базову ціну базовому класу. Властивість Type повертає "Мечник".

Archer: Наслідується від Warrior. Представляє лучника. Конструктор передає ім’я та базову ціну базовому класу. Властивість Type повертає "Лучник".

Cuirassier: Наслідується від Warrior. Представляє кірасира. Конструктор передає ім’я та базову ціну базовому класу. Властивість Type повертає "Кірасир".

IPricingStrategy: Інтерфейс для стратегій ціноутворення. Визначає метод CalculatePrice(Warrior warrior), який повертає ціну воїна з урахуванням стратегії.

IVipObserver: Інтерфейс для спостерігачів (VIP-гравців). Визначає метод Update(string message), який обробляє сповіщення від магазину.

RegularPriceStrategy: Реалізує IPricingStrategy. Метод CalculatePrice повертає базову ціну воїна без знижок.

SalePriceStrategy: Реалізує IPricingStrategy. Метод CalculatePrice застосовує знижки: 5% для мечника, 10% для лучника, 15% для кірасира.

SuperSalePriceStrategy: Реалізує IPricingStrategy. Метод CalculatePrice застосовує більші знижки: 15% для мечника, 20% для лучника, 25% для кірасира.

VipPlayer: Реалізує IVipObserver. Містить поле \_name (ім’я гравця). Конструктор ініціалізує ім’я. Метод Update виводить сповіщення для гравця з його ім’ям та отриманим повідомленням.

ArmyStore: Клас, що представляє магазин армії. Містить:

Поле \_inventory типу List<Warrior> для зберігання воїнів.

Поле \_observers типу List<IVipObserver> для зберігання підписаних VIP-гравців.

Поле \_pricingStrategy типу IPricingStrategy для поточної стратегії ціноутворення.

Конструктор ініціалізує стратегію ціноутворення.

Метод SetPricingStrategy змінює стратегію ціноутворення.

Метод AddWarrior додає воїна до інвентарю.

Метод CalculateTotalPrice обчислює загальну ціну всіх воїнів у інвентарі, використовуючи поточну стратегію.

Метод DisplayInventory виводить інформацію про всіх воїнів у інвентарі та загальну ціну.

Метод Subscribe додає спостерігача (VIP-гравця) до списку.

Метод Unsubscribe видаляє спостерігача зі списку.

Метод NotifyObservers сповіщає всіх спостерігачів, викликаючи їхній метод Update із заданим повідомленням.

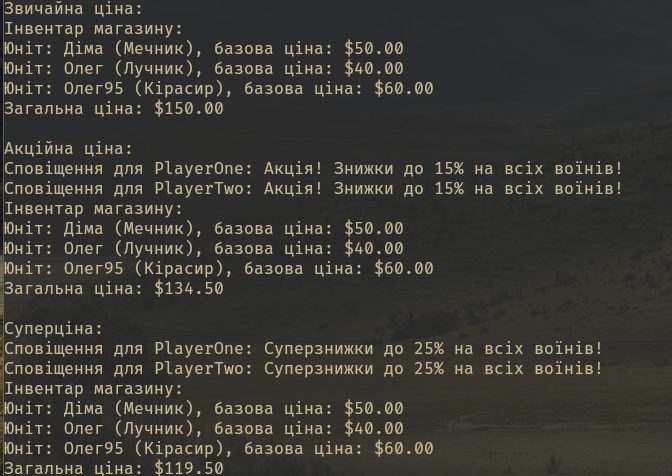


Рисунок 1 — Результат роботи

Висновок: Отже, під час лабораторної роботи було досліджено роботу з поведінковими шаблонами проектування, таким як: стратегія.

Лістинг програми

using System.Text;

Console.OutputEncoding = Encoding.UTF8;

var swordsman = new Swordsman("Діма", 50.0);

var archer = new Archer("Олег", 40.0);

var cuirassier = new Cuirassier("Олег95", 60.0);

IPricingStrategy regularPrice = new RegularPriceStrategy();

IPricingStrategy salePrice = new SalePriceStrategy();

IPricingStrategy superSalePrice = new SuperSalePriceStrategy();

ArmyStore store = new ArmyStore(regularPrice);

store.AddWarrior(swordsman);

store.AddWarrior(archer);

store.AddWarrior(cuirassier);

var vipPlayer1 = new VipPlayer("PlayerOne");

var vipPlayer2 = new VipPlayer("PlayerTwo");

store.Subscribe(vipPlayer1);

store.Subscribe(vipPlayer2);

Console.WriteLine("Звичайна ціна:");

store.SetPricingStrategy(regularPrice);

store.DisplayInventory();

Console.WriteLine("\nАкційна ціна:");

store.SetPricingStrategy(salePrice);

store.NotifyObservers("Акція! Знижки до 15% на всіх воїнів!");

store.DisplayInventory();

Console.WriteLine("\nСуперціна:");

store.SetPricingStrategy(superSalePrice);

store.NotifyObservers("Суперзнижки до 25% на всіх воїнів!");

store.DisplayInventory();

return 0;

public abstract class Warrior

{

public string Name { get; }

public double BasePrice { get; }

public Warrior(string name, double basePrice)

{

Name = name;

BasePrice = basePrice;

}

public abstract string Type { get; }

public override string ToString()

{

return $"Юніт: {Name} ({Type}), базова ціна: {BasePrice:C}";

}

}

public class Swordsman : Warrior

{

public Swordsman(string name, double basePrice) : base(name, basePrice) { }

public override string Type => "Мечник";

}

public class Archer : Warrior

{

public Archer(string name, double basePrice) : base(name, basePrice) { }

public override string Type => "Лучник";

}

public class Cuirassier : Warrior

{

public Cuirassier(string name, double basePrice) : base(name, basePrice) { }

public override string Type => "Кірасир";

}

public interface IVipObserver

{

void Update(string message);

}

public class VipPlayer : IVipObserver

{

private readonly string \_name;

public VipPlayer(string name)

{

\_name = name;

}

public void Update(string message)=> Console.WriteLine($"Сповіщення для {\_name}: {message}");

}

public interface IPricingStrategy

{

double CalculatePrice(Warrior warrior);

}

public class RegularPriceStrategy : IPricingStrategy

{

public double CalculatePrice(Warrior warrior) => warrior.BasePrice;

}

public class SalePriceStrategy : IPricingStrategy

{

public double CalculatePrice(Warrior warrior) => warrior.BasePrice \* (1 - warrior switch

{

Swordsman => 0.05,

Archer => 0.10,

Cuirassier => 0.15,

\_ => 0.0

});

}

public class SuperSalePriceStrategy : IPricingStrategy

{

public double CalculatePrice(Warrior warrior) => warrior.BasePrice \* (1 - warrior switch

{

Swordsman => 0.15,

Archer => 0.20,

Cuirassier => 0.25,

\_ => 0.0

});

}

public class ArmyStore

{

private readonly List<Warrior> \_inventory = new List<Warrior>();

private readonly List<IVipObserver> \_observers = new List<IVipObserver>();

private IPricingStrategy \_pricingStrategy;

public ArmyStore(IPricingStrategy pricingStrategy)=> \_pricingStrategy = pricingStrategy;

public void SetPricingStrategy(IPricingStrategy pricingStrategy) => \_pricingStrategy = pricingStrategy;

public void AddWarrior(Warrior warrior)=> \_inventory.Add(warrior);

public double CalculateTotalPrice()

{

double total = 0;

foreach (var warrior in \_inventory)

{

total += \_pricingStrategy.CalculatePrice(warrior);

}

return total;

}

public void DisplayInventory()

{

Console.WriteLine("Інвентар магазину:");

foreach (var warrior in \_inventory)

{

Console.WriteLine(warrior);

}

Console.WriteLine($"Загальна ціна: {CalculateTotalPrice():C}");

}

public void Subscribe(IVipObserver observer)

{

\_observers.Add(observer);

}

public void Unsubscribe(IVipObserver observer)

{

\_observers.Remove(observer);

}

public void NotifyObservers(string message)

{

foreach (var observer in \_observers)

{

observer.Update(message);

}

}

}