Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**КУРСОВА РОБОТА**

з дисципліни «Об’єктно-орієнтоване програмування»

на тему

**Шаблони проектування в ООП. Комп’ютерна гра: Садівник**

Виконала студентка

ІІ курсу групи КП-51

Граділь Анастасія Валеріївна

залікова книжка КП-5207

Керівник роботи

доц., к.т.н. Заболотня Т.М.

Оцінка

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, підпис)

КИЇВ 2017

**ЗМІСТ**

**ВСТУП**.........................................................................................................................3

1. **ОПИС СТРУКТУРНО-ЛОГІЧНОЇ СХЕМИ ПРОГРАМИ**……............4
   1. Основний принцип роботи програми...................................................4
2. **ПРОГРАМНА РЕАЛІЗАЦІЯ ГРИ ЗА ДОПОМОГОЮ ШАБЛОНІВ ПРОЕКТУВАННЯ**..........................................................................................5
   1. Опис реалізованих класів.......................................................................5
   2. Опис реалізованих шаблонів................................................................12
   3. Діаграма класів…..........................................................................……15
   4. Опис роботи програми……………………..........................................17

**ВИСНОВКИ**..............................................................................................................22

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**.....................................................23

**ВСТУП**

Дана курсова робота присвячена розробці комп’ютерної гри за допомогою використання шаблонів проектування. Дана тематика вибрана для курсової роботи тому, що результати абстрагування об’єктів у грі дозволяють застосувати всі вивчені методи та принципи об’єктно-орієнтованого програмування для створення програмного забезпечення (ПЗ), зокрема шаблони проектування.

*Об’єктом* курсової роботи є сама гра.

*Метою* роботи є розроблення ПЗ для гри з використанням шаблонів проектування.

Для досягнення визначеної мети необхідно виконати такі *завдання*:

* абстрагувати об’єкти предметної галузі;
* розробити структурну організацію ПЗ за допомогою використання основних принципів ООП та шаблонів проектування;
* визначити та описати функціональні характеристики програми;
* обґрунтувати вибір шаблонів проектування;
* розробити графічний інтерфейс користувача;
* виконати реалізацію ПЗ відповідно до технічного завдання (ТЗ);
* виконати тестування розробленої програми;
* оформити документацію з курсової роботи.

Розроблене ПЗ складається з таких логічних частин: інтерфейсу користувача, модуля логіки взаємодії користувача з полем. Використані шаблони проектування: **Декоратор, Компонувальник, Абстрактна фабрика, Прототип, Фасад, Команда, Стан та Адаптер.**

Розроблене ПЗ може бути використане з розважальною метою.

Пояснювальна записка складається зі вступу, двох розділів, загальних висновків та списку використаних джерел (3 найменування). Робота містить 15 рисунків. Загальний обсяг роботи – 20 друковані сторінки.

1. **ОПИС СТРУКТУРНО-ЛОГІЧНОЇ СХЕМИ ПРОГРАМИ**
   1. **Основний принцип роботи програми**

Модульна організація програми

Логіка Гри

Посередник між інтерфейсам та логікою гри

Модуль Интерфейсу

1. **ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ЗА ДОПОМОГОЮ ШАБЛОНІВ ПРОЕКТУВАННЯ**
   1. **Опис реалізованих класів**
2. Program

Завдання:

Клас реалізовує коритувацький інтерфейс.

Методи:

* Main() – відповідає за головне меню гри. Переводить користувача у відповідну частину гри.
* newGame (RenderWindow app) – відповідає за виведення ігрового поля та підтримку кнопок.
* shopGame(RenderWindow app)– відображає магазин, де користувач може обрати нове дерево до саду
* OnClose(object sender, EventArgs e) – закриває вікно гри

1. Gardener

Завдання:

Є своєрідним посередником між логікою гри та інтерфейсом. Є Одинаком. Реалізовує шаблон фасад.

Методи:

* public static Gardener gardener() - Повертає єдиний екземпляр садівника, якщо він є і створює, якщо немає.
* public void killWarms() – Збирає всі фрукти саду і віддає команді, Яка вбиває шкідників.
* public void refreshGarden() – Оновлює стан саду для кожного кадру.
* public void display(RenderWindow app) – Виводить всі елементи саду на екран.
* public int getFruits() – повертає кількість фруктів, зібрану в саду за час гри.
* public GardenComponent bought(int id) - повертає товар з магазину, по ID товару.
* public void AddNewElementToGarden(GardenComponent elem) – додає новий елемент до саду.
* public int getFallenFruit() – повертає кількість фруктів, що гравець не встиг зібрати

1. Garden

Завдання:

Наслідує GardenComponent. Містить список усіх дерев та декорацій.

Методи:

* public override void add(GardenComponent t) – додає новий компонент до саду
* public override void remove(GardenComponent t) – видаляє компонент з саду
* private void checkState() – перевіряє чи заповнений сад. Перемикає стан
* public override double getWeight() – повертає загальну вагу всіх плодів саду.
* public override int getFruits() – відповідає за збір стиглих фруктів
* public override void display(RenderWindow app) – виводить всі елементи саду на екран.
* public override respond refresh() – оновлює стан саду після кожного кадру
* public override void getFruitList(List<Fruit> l) – повертає список усіх фруктів саду.
* public int getFallenFruits() – повертає кількість плодів, що не були зібрані

1. DisWarm

Завдання:

Команда, що змушує всіх виконувачів(Фруктів) звільнитись від шкідників.

Методи:

* public void killWarm() – для кожного виконувача виконує очищення від шкідників.

1. GardenState

Завдання:

Абстрактний клас. Описує стан саду. Впливає на поведінку при додаванні нових компонентів до саду.

Методи:

* public abstract void addComponent(List<GardenComponent> l, GardenComponent component) – прототип метода додавання нового компонента до саду.

1. NotFullGardenState

Завдання:

Наслідується від GardenState. Описує переповнений стан саду. Не дає додавати нових елементів в сад, бо він переповнений.

Методи:

* public void addComponent(List<GardenComponent> l, GardenComponent component) – метод, що не дає виконати додавання нового компонента до саду, бо він переповнений

1. FullGardenState

Завдання:

Наслідується від GardenState. Описує незаповнений стан саду. Додає нові елементи в сад.

Методи:

* public void addComponent(List<GardenComponent> l, GardenComponent component) – метод, що додає нові компоненти до саду.

1. GardenComponent

Завдання:

Абстрактний клас. Описує компонент саду. Частина шаблону Компонувальник.

Методи:

* public abstract void add(GardenComponent component) – додає новий компонент.
* public abstract void remove(GardenComponent component) – видаляє компонент.
* public abstract double getWeight() – повертає вагу фруктів компонента.
* public abstract int getFruits() – повертає кількість зібраних фруктів
* public abstract void display(RenderWindow app) – виводить компонент на екран.
* public abstract respond refresh() – оновлює стан компонента поля кожного кадру.
* public abstract void getFruitList(List<Fruit> l) – додає до списку всі фрукти даного компонента.
* public virtual object Clone() – забезпечує копіювання єлемента.

1. Tree

Завдання:

Абстракний клас, що компонентом саду та описує інтерфейс для конкретного дерева.

Методи:

* private bool checkTime() – перевіряє чи час додавати нові плоди
* public override void add(GardenComponent t) – додає плоди
* public override void remove(GardenComponent t) – видаляє плоди
* public override double getWeight() – повертає загальну вагу плодів
* public abstract Fruit newFruit() – прототип абстрактної фабрики. Щоб дерево створювало плоди свого сорту
* private void initTime() – скидує час.
* public override object Clone() – забезпечує копіювання дерева
* public override int getFruits() – кількість фруктів, зібраних на цьому дереві
* public override void display(RenderWindow app) – виводить дерево на екран.
* protected SFML.System.Vector2f getRandPos() – повертає рандомну позицію для нового фрукта
* public override respond refresh() – оновлює стан дерева після кожного кадру
* public override void getFruitList(List<Fruit> l) – додає до списку всі червиві фрукти з дерева.

1. AppleTree, PearTree, PlumTree, CherryTree

Завдання:

Класи, що описують дерева свого виду.

Методи:

* public override Fruit newFruit() – повертає фрукт того ж виду, що і дерево

1. Fruit

Завдання:

Абстрактний клас, що описує інтерфейс для всіх фруктів.

Методи:

* public virtual float ripen() – робить фрукт більш стиглим
* private void initTime() – скидує час
* private bool checkTime() – перевіряє чи час фрукту ставати червивим
* public virtual Fruit disWarm() – прибирає шкідників з фрукта
* public override void add(GardenComponent t) – нічого не відбувається
* public override void remove(GardenComponent t) – нічого не відбувається
* public override double getWeight() – повертає вагу фрукта
* public virtual bool isWarmly() – каже чи червивий фрукт
* public override int getFruits() – повертає 1, якщо фрукт можна зривати, -1, якщо його можна зривати, але він червивий, 0, якщо не стиглий
* public override void display(RenderWindow app) – виводить фрукт на екран
* public override respond refresh() – оновлює стан фрукту посля кожного кадру
* public override void getFruitList(List<Fruit> l) – додає себе до списку, якщо фрукт червивий.

1. Apple, Plum, Pear, Cherry

Завдання:

Клас, що описує поведінку конкретного фрукта.

1. DecorFruit

Завдання:

Є декоратором Фруктів

1. WarmlyFruit

Завдання:

Є декоратором, що змінює поведінку фрукта, роблячи його червивим

1. Shop

Завдання:

Клас, що містить всі дерева, що є прототипами для дерев саду.

Методи:

* public static GardenComponent bought(string tree) - повертає копію товару, по назві.
  1. **Опис реалізованих шаблонів**
* Декоратор

Призначений для динамічного підключення додаткових станів та поведінки до об’єкта та є гнучкою альтернативою практиці створення підкласів з метою розширення функціональності.

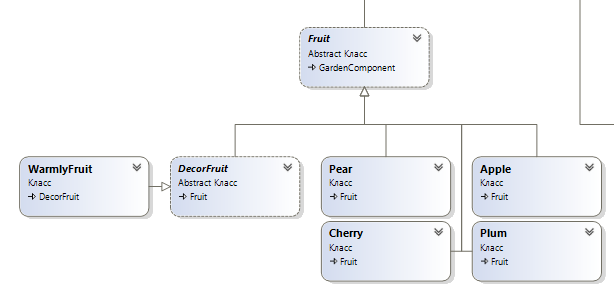


Рисунок 1 Структура шаблону у даній програмі

*Обґрунтування використання даного шаблону.* Даний шаблон в курсовій роботі використовується для динамічної зміни поведінки фруктів.

* Компонувальник

Призначений для створення деревоподібної структури для подання ієрархії об’єктів, де кожний об’єкт можна розглядати незалежно або як набір вкладених об’єктів через один інтерфейс. Таким чином, з’являється можливість уніфіковано, однаково поводитися з кожним об’єктом.

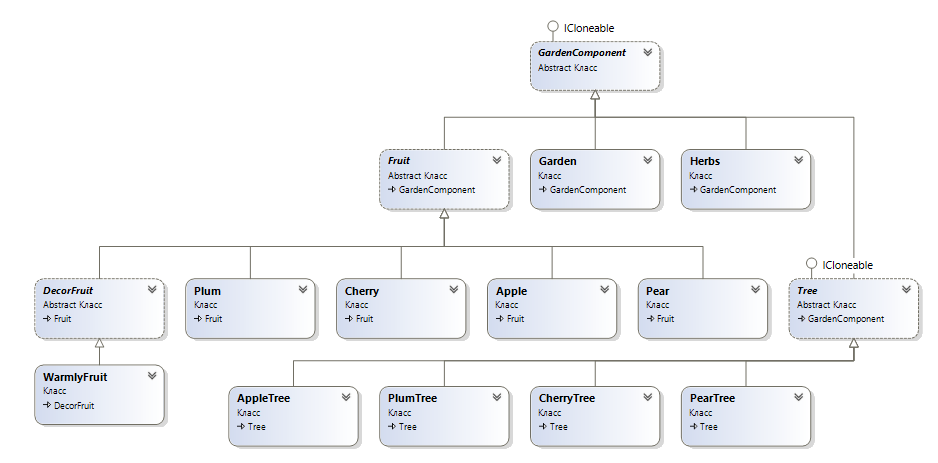


Рисунок 2 Структура Компонувальника у даній програмі

*Обґрунтування використання даного шаблону.* Даний шаблон в курсовій роботі використовується так як:

–у наявності є оригінальна структура, що складається з об’єктів та композицій об’єктів;

–необхідно забезпечити ігнорування клієнтом істотних відмінностей між окремим об’єктом та складеним об’єктом;

–необхідно реалізувати уніфіковану обробку всіх об’єктів.

* Стан

Дозволяє Саду варіювати свою поведінку залежно від внутрішнього стану.

(Залежно від того, наскільки заповнений сад)

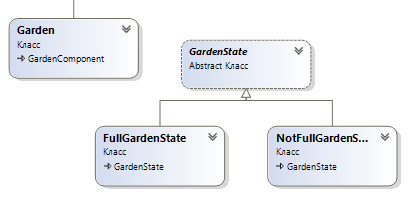


Рисунок 3 Диаграма класів шаблону Стан

*Обґрунтування використання даного шаблону.* Даний шаблон в курсовій роботі використовується так як поведінка об'єкта залежить від його стану і повинна змінюватися під час виконання програми.

* Фабричний метод

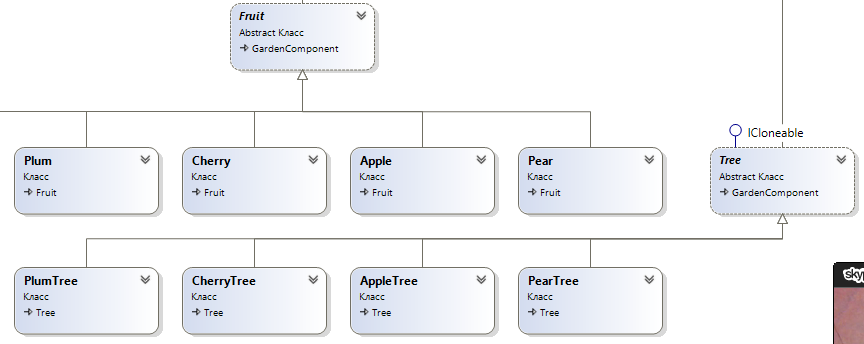
Визначає інтерфейс для створення об’єкта, але рішення про те, який саме об’єкт створювати, залишає за підкласами.

Рисунок 4 Діаграма шаболону Фабричний метод

*Обґрунтування використання даного шаблону:* необхідно,щоб дерева давали плоди тільки свого виду

* Прототип

Задає види створюваних об’єктів за допомогою екземпляра-прототипу і створює нові об’єкти шляхом копіювання цього прототипу.

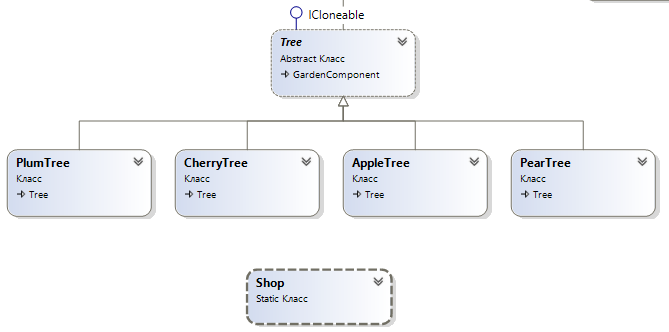


Рисунок 5 Диаграма Шаблона Прототип

*Обґрунтування використання даного шаблону.* Даний шаблон в курсовій роботі використовується так як необхідно було створювати товари, що купуються в магазині.

* Фасад

Структурує об’єкти, надаючи до них всіх доступ через єдиний шлюз (Садівника).

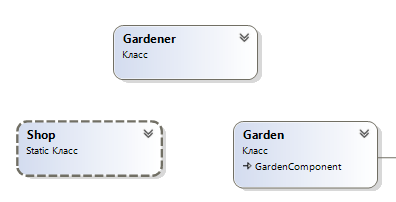


Рисунок 6 Диаграма шаблона Фасад

*Обґрунтування використання даного шаблону.* Даний шаблон в курсовій роботі використовується для зручності взаємодії інтерфейсу користувача з логікою програми.

* Команда

Даний шаблон інкапсулює алгоритм видалення шкідників в єдину сутність



Рисунок 7 Диаграма шаблону Команда

*Обґрунтування використання даного шаблону.* Даний шаблон в курсовій роботі використовується інкапсуляції алгоритму.

* Адаптер

«Адаптер» забезпечує спільну роботу класів з несумісними інтерфейсами шляхом створення спільного об’єкта, через який вони можуть взаємодіяти.

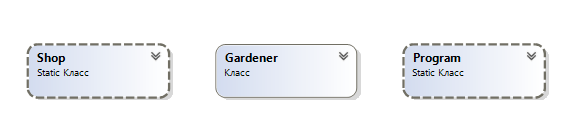


Рисунок 8 Диаграма шаблона Адаптер

*Обґрунтування використання даного шаблону.* Для роботи з магазином. Тільки садівник знає як правильно замовити товар з магазину. З інтерфейсу надходить інформація, зручна для користувача.

* 1. **Діаграма класів**



Рисунок 9 Загальна структура класів

* 1. **Опис роботи програми**

Одразу після запуску гри відкривається головне меню програми.

Натиснувши на кнопку поряд з вільним місцем, гравець пересуває її на це вільне місце. При цьому кількість шагів збільшується на 1.



Рисунок 10 Головне меню

Коли гравець натискає на кнопку «Нова гра», відкривається сад.



Рисунок 11 Початок гри

Тут користувач може зайти до магазину та придбати товар для саду 

Рисунок 12 Магазин

Коли він обирає конкретний товар, він повертається до саду та має змогу розмістити елемент, де захоче.

Дерева ростуть і приносять плоди. Іноді плоди стають червивими. Користувач має змогу зібрати врожай або вбити шкідників.



Рисунок 13 Сад з деревами, посадка нових дерев



Рисунок 14 Плоди на деревах



Рисунок 15 Червиві плоди

У інформаційній частині екрану відображається кількість фруктів, що користувач встиг зібрати, та не встиг.



Рисунок 16 Статистика Гри

**ВИСНОВКИ**

Метою даної курсової роботи було розроблення гри «Садівник» з використанням шаблонів проектування. Підставою для розроблення стало завдання на виконання курсової роботи з дисципліни «Об’єктно-орієнтоване програмування» студентами ІІ курсу кафедри ПЗКС НТУУ «КПІ».

Для досягнення поставленої мети у повному обсязі виконано завдання, визначені у аркуші завдання на курсову роботу; розроблено графічні матеріали; реалізовано всі вимоги до програмного продукту, наведені у технічному завданні; створено відповідну документацію.

Програму створено на основі використання шаблонів проектування: зокрема, до структури ПЗ входить реалізація семи шаблонів, які належать до різних груп.

Для розроблення програмного забезпечення використані мова програмування C# та технологія розробки графічного інтерфейсу SFML

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Гамма, Э. Приемы объектно-ориентированого проектирования. Паттерны проектирования [Текст] / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб: Питер, 2009. – 366 с.
2. Э. Фримен, Э. Фримен, К. Сьерра, Б. Бейтс. Паттерны проектирования. — СПб.: Питер, 2011. — 656 с.: ил.
3. Design Patterns:Elements of Reusable Object-Oriented Software/Erich Gamma, Richard Helm,Ralph Johnson, John M.Vlissides//Шаблони проектування-2012.- С.30-69