МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

ШАБЛОНИ ПРОЕКТУВАННЯ

**Методичні вказівки
до виконання лабораторних робіт
з дисципліни
«Об’єктно-орієнтоване програмування»**

для студентів напряму підготовки 6.050103 «Програмна інженерія»

Рекомендовано
Вченою радою ФПМ НТУУ «КПІ»
(Протокол № 6 від 26.01.2015 р.)

Київ
НТУУ «КПІ»
2015

Шаблони проектування: методичні вказівки до виконання лабораторних робіт з дисципліни «Об’єктно-орієнтоване програмування» для студентів напряму підготовки 6.050103 «Програмна інженерія» [Електронне видання] / Т.М.Заболотня. К.: НТУУ «КПІ», 2015. - 154с.

Навчально-методичне видання

ШАБЛОНИ ПРОЕКТУВАННЯ

Методичні вказівки до виконання лабораторних робіт
з дисципліни «Об’єктно-орієнтоване програмування»
для студентів напряму підготовки 6.050103 «Програмна інженерія»

Методичні вказівки розроблено для ознайомлення студентів з теоретичними відомостями та практичними прийомами побудови об’єктно-орієнтованого програмного забезпечення з використанням шаблонів проектування, а також вимогами до виконання лабораторних робіт, зокрема правилами їх оформлення.

Навчальне видання призначене для студентів, які навчаються за напрямом 6.050103 «Програмна інженерія» факультету прикладної математики НТУУ «КПІ».

Автор: *Заболотня Тетяна Миколаївна,* канд.техн.наук.

Відповідальний
за випуск *Сулема Євгенія Станиславівна,* канд.тех.наук, доцент.

Рецензент *Марченко Олександр Іванович,* канд. техн. наук, доцент.

ВСТУП

Відповідно до **Положення про організацію навчального процесу у вищих навчальних закладах**, затвердженого наказом Міністерства освіти України від 2 червня 1993 р. № 161, а також до **Положення про організацію навчального процесу в НТУУ "КПІ"** (п.5. Організаційні форми навчального процесу) **лабораторне заняття** є одним з основних видів навчальних занять, на якому студент під керівництвом викладача особисто проводить експерименти чи дослідження з метою практичного підтвердження окремих теоретичних положень навчальної дисципліни, набуває практичних навичок роботи з обчислювальною технікою, оволодіває методикою експериментальних досліджень у конкретній предметній галузі та обробки отриманих результатів [1, 2].

Перелік тем лабораторних занять визначається робочою навчальною програмою дисципліни. Заміна лабораторних занять іншими видами навчальних занять не дозволяється.

Лабораторне заняття включає проведення поточного контролю підготовленості студентів до виконання конкретної лабораторної роботи, виконання завдань теми заняття, оформлення індивідуального звіту про виконану роботу та його захист перед викладачем.

Виконання лабораторної роботи оцінюється викладачем. Підсумкові оцінки, отримані студентом за виконання лабораторних робіт, враховуються при виставленні семестрової підсумкової оцінки (рейтингу) з дисципліни. Наявність позитивних оцінок, одержаних студентом за всі лабораторні роботи, передбачені робочою навчальною програмою, є

необхідною умовою його допуску до семестрового контролю по даній дисципліні.

Дисципліна «**Об'єктно-орієнтоване програмування**» (далі – **ООП**) у межах підготовки студентів денної форми навчання за кваліфікаційним рівнем «бакалавр» за напрямом підготовки 6.050103 «Програмна інженерія» складається з трьох модулів ECTS: «Основи об'єктноорієнтованого програмування», який викладається в III-ому семестрі; а також «Шаблони проектування» та курсова робота з дисципліни у IV-ому семестрі. [3].

Дане навчальне видання містить методичні вказівки щодо виконання лабораторних робіт протягом IV семестру під час вивчення студентами кредитного модуля «Шаблони проектування».

Відповідно до робочої навчальної програми даного модуля [4] основною метою виконання лабораторних робіт з ООП у IV семестрі є закріплення теоретичних знань щодо основних характеристик шаблонів проектування; запам'ятовування поширених ситуацій, коли використання цих шаблонів є доцільним; набуття вмінь та навичок реалізації шаблонів під час створення програмного коду. Завдання, наведені у методичних вказівках, розраховані на 18 академічних годин аудиторних занять та охоплюють відомості по всім видам об'єктно-орієнтованих шаблонів проектування за класифікацією GoF [4, 5].

Кожна лабораторна робота включає формулювання мети та основних завдань; теоретичну частину з певної теми, яка базується на лекційному матеріалі; приклад виконання завдання; індивідуальні завдання, а також контрольні запитання та список рекомендованої літератури для самостійної роботи студентів.

Крім того, дане навчальне видання містить рекомендації щодо порядку виконання лабораторних робіт та обробки отриманих результатів, а також вимоги до оформлення звіту з виконаної роботи та правила його захисту перед викладачем.

Слід зазначити, що студенти 2 курсу (IV семестру навчання) вже мають досвід самостійної реалізації цілісної програмної розробки в межах лабораторних та курсової робіт з дисципліни «Основи програмування» на першому курсі, який вони можуть застосовувати під час виконання лабораторних робіт з ООП. Крім того, вони у повній мірі можуть використовувати знання щодо базових алгоритмів, структур даних, мов програмування, принципів створення інтерфейсу користувача, отриманих при вивченні таких дисциплін як «Основи програмної інженерії», «Алгоритми та структури даних», «Основи програмування» тощо.

В цілому, можна зазначити, що виконання лабораторних робіт з ООП сприяє розширенню і поглибленню теоретичних знань, здобутих студентами в процесі вивчення даної дисципліни, розвиває навички їх практичного використання та забезпечує підтримку самостійного розв’язання конкретних завдань.

1. ЗАГАЛЬНІ ВИМОГИ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

1.1. Порядок виконання лабораторних робіт

Лабораторні роботи, запропоновані у даному навчальному виданні, є однотипними, тому порядок їх виконання є спільним для всіх робіт.

Етап 1. Теоретична підготовка до виконання роботи.

Крок 1. Вивчення лекційного матеріалу щодо шаблонів проектування, які відповідають темі роботи, а також засвоєння положень, що перевіряються контрольними питаннями до кожної лабораторної роботи. За необхідности, звернення до додаткових джерел інформації, які наведено у списку рекомендованої літератури.

Крок 2. Аналіз контрольних прикладів побудови коду відповідно до заданих шаблонів проектування.

Крок 3. Ознайомлення з індивідуальним завданням та визначення шаблонів проектування, які доцільно застосувати для його розв’язання. На даному кроці допускається звернення за консультацією до викладача з метою уникнення написання програмного коду з реалізацією неправильного шаблону.

Етап 2. Створення програмного забезпечення для вирішення задач лабораторної роботи.

Крок 1. Розробка структурної організації класів у відповідності до обраних шаблонів проектування.

Крок 2. Програмна реалізація класів.

Крок 3. Відлагодження програми, отримання та аналіз результатів її роботи.

Етап 3. Оформлення звіту з результатами виконання роботи.

Етап 4. Захист лабораторної роботи.

Крок 1. Демонстрація викладачеві програмного коду з обгрунтуванням вибору шаблонів преоктування.

Крок 2. Демонстрація роботи програми.

Крок 3. Відповіді на запитання викладача.

У випадку, якщо студент використав неправильний для заданої лабораторної роботи шаблон проектування або некоректно його реалізував, до кінця поточного заняття він має можливість виправити помилку та спробувати повторно захистити роботу без зниження оцінки.

При відсутності звіту про виконання лабораторної роботи або відповідних знань студент не отримує бали за роботу. У такому разі він повинен виправити всі помилки, опанувати необхідний теоретичний матеріал та спробувати повторно захистити роботу на наступному занятті (із зняттям штрафних балів, визначених у РНП модуля [4]).

1.2. Вимоги до розроблюваного програмного забезпечення

1. Програмне забезпечення для розв’язання поставлених у лабораторній роботі задач **має базуватися на реалізації шаблонів проектування**:
   - кожна задача потребує використання *одного* шаблону проектування (хоча, у випадку, якщо студент вважає за доцільне розширити своє завдання та реалізувати декілька шаблонів проектування для виконання поставленої задачі, це є припустимим);
   - кожний варіант завдань містить задачі з різних шаблонів проектування, таким чином в межах однієї лабораторної роботи не може бути двох задач, для розв’язку яких необхідне застосування спільного шаблону;
   - у випадку, якщо студент реалізував у програмному коді шаблон проектування, який не входить до переліку шаблонів з тематики даної лабораторної роботи, він повинен обгрунтувати свій вибір.

Така ситуація є припустимою тільки за умови наведення вагомих аргументів.

2. Програмне забезпечення повинне мати **інтерфейс користувача** (консольний чи Windows.Forms значення не має), за допомогою якого можна проглядати на екрані результати роботи програми. Необов'язковим є створення складного інтерфейсу. Достатньо забезпечити виведення на екран тільки важливих значень полів об'єктів чи службових повідомлень, які б наочно демонстрували коректність роботи програми.
3. Мова програмування, яка використовується для написання програмного коду, може бути будь-якою, але обов'язково об'єктно-орієнтованою, адже метою кредитного модуля «Шаблони проектування» є опанування особливостей побудови програм на основі шаблонів проектування, що можна реалізувати за допомогою різних мов (C#, Java, PHP тощо).
4. Текст програми повинен легко сприйматись. Тому в коді необхідно виділяти коментарями основні логічні частини шаблонів, що реалізуються.

1.3. Правила оформлення звіту

Кожна лабораторна робота повинна супроводжуватися звітом про її виконання, який подається у друкованому вигдялі та містить:

а) титульний аркуш;

б) тему, мету роботи та постановку задач за варіантом;

в) коротке обгрунтування вибору шаблону проектування;

г) опис структури програми (UML-діаграму класів);

д) текст програми;

е) приклади результатів, які можуть бути отримані при використанні розробленої програми;

ж) висновки щодо роботи програми.

Оскільки кожний варіант завдань у лабораторних роботах складається з двох задач, пункти в)–ж) мають бути виконані для кожної задачі окремо.

Нижче наведено детальні рекомендації щодо оформлення звіту про результати виконання лабораторної роботи.

Звіт розпочинається з **титульного аркуша**, у якому зазначаються: офіційна назва навчального закладу та кафедри, на якій виконана робота; дисципліна, з якої виконується лабораторна робота; тема лабораторної роботи; ПІБ автора; ПІБ, посада, науковий ступінь, вчене звання викладача, який перевіряє роботу; відмітки про результати захисту звіту; місто та рік виконання роботи.

Приклад оформлення титульного аркуша наведено в Додатку А.

Обгрунтування вибору шаблону проектування для розв’язання тієї чи іншої задачі повинно базуватися, з одного боку, на загальних теоретичних відомостях про цей шаблон (зокрема, про його призначення, про стандартні випадки його застосування), з іншого боку – на результатах аналізу ситуації, описаної у кожній конкретній задачі.

Як галузевий стандарт для **опису структури програми** має використовуватись мова візуального моделювання програмних систем UML (Unified Modeling Language – універсальна мова моделювання) [6, 7].

Приклади результатів, які можуть бути отримані при використанні розробленої програми, повинні містити дані, що наочно демонструють коректність роботи програми, а також підтверджують переваги використання обраного шаблону у заданій ситуації.

Для ілюстрування можливостей інтерфейсу користувача допускається використання скріншотів розробленого програмного забезпечення.

У загальних **висновках** по лабораторній роботі необхідно підсумувати отримані результати щодо ефективності розв’язання поставлених задач за допомогою використання шаблонів проектування.

Весь звіт необхідно друкувати на одному боці аркуша білого паперу формату А4 (210×297 мм).

Основний текст звіту має бути набраний з дотриманням таких вимог:

- поля: ліве – 30 мм, верхнє і нижнє – 20 мм, праве – 10 мм;
- шрифт: Times New Roman, 14 пт;
- міжрядковий інтервал – 1.5 пт;
- відступ першого рядка – 1 см;
- вирівнювання: назви розділів і підрозділів – по центру, назви пунктів і підпунктів – з лівого боку, основний текст – по ширині.

Текст програм має бути набраний з дотриманням таких вимог (при необхідності дозволяється тексти програм розміщувати в альбомному форматі):

- шрифт: Courier New 8 пт;
- міжрядковий інтервал: 1.0 пт.

Всі *рисунки* повинні мати підрисунковий напис. Підрисунковий напис вирівнюється по центру і починається зі скорочення “Рис.”, потім ставиться пробіл та порядковий номер рисунку. Після номера рисунка ставиться крапка, пробіл та пишеться назва рисунка.

На всі рисунки, розміщені у звіті, має бути посилання в тексті звіту. Посилання на рисунок у тексті виконується за його номером, розташованим після скорочення “рис.”.

Нумерацію сторінок виконують арабськими цифрами. Першою сторінкою звіту з лабораторної роботи є оформлений за зразком титульний аркуш, який включають до загальної нумерації, але номер сторінки на ньому не проставляють. На всіх наступних сторінках обов’язково проставляють у правому нижньому куті номер сторінки без крапки в кінці використовуючи шрифт Times New Roman 10 пт.

Друкарські *помилки*, описки і графічні неточності, які виявилися в роздрукованій роботі, можна виправляти підчищенням або зафарбуванням коректором і нанесенням на тому ж місці виправленого тексту (фрагменту рисунка). Виправлення необхідно робити чорним кольором. Допускається наявність не більше двох виправлень на одній сторінці.

Звіт з лабораторних робіт виконується українською мовою.

Не дозволяється при оформленні звіту про результати лабораторних робіт використовувати професійний жаргон.

Дозволяється виконання звіту у рукописному вигляді.

Захист лабораторних робіт можливий тільки за умови наявності звіту.

2. ЗАВДАННЯ НА ЛАБОРАТОРНІ РОБОТИ

2.1. Лабораторна робота №1

Тема роботи: реалізація структурних шаблонів проектування.

Мета роботи: ознаойомлення з основними характеристиками шаблонів «Декоратор», «Компонувальник» та «Міст», запам’ятовування поширених ситуацій, коли використання цих шаблонів є доцільним, набуття вмінь та навичок реалізації шаблонів під час створення програмного коду.

Теоретичні відомості

Декоратор

Це структурний шаблон проектування, який призначений для ***динамічного підключення додаткових станів та поведінки*** до об’єкта та є гнучкою альтернативою практиці створення підкласів з метою розширення функціональності. Об’єкт при цьому “не знає”, що він є “декорованим”, тому шаблон доцільно використовувати для розширення програмних систем. Відомий також під назвою *Обгортка* (*Wrapper*).

Структурний шаблон «Декоратор» потрібно використовувати, коли:

–обов’язки та поведінка об’єктів повинні додаватися динамічно;

–конкретні реалізації повинні бути відокремлені від обов'язків і поведінки;

–розширення об’єктів шляхом створення підкласів недоцільне або неможливе;

–конкретні функції не повинні перебувати на верхніх рівнях ієрархії об'єктів;

–необхідно реалізувати можливості, які потрібні не всім об’єктам та не завжди і можуть бути видалені при необхідності;

–наявність великої кількості малих об’єктів, які мають схожу реалізацію, є припустимою.

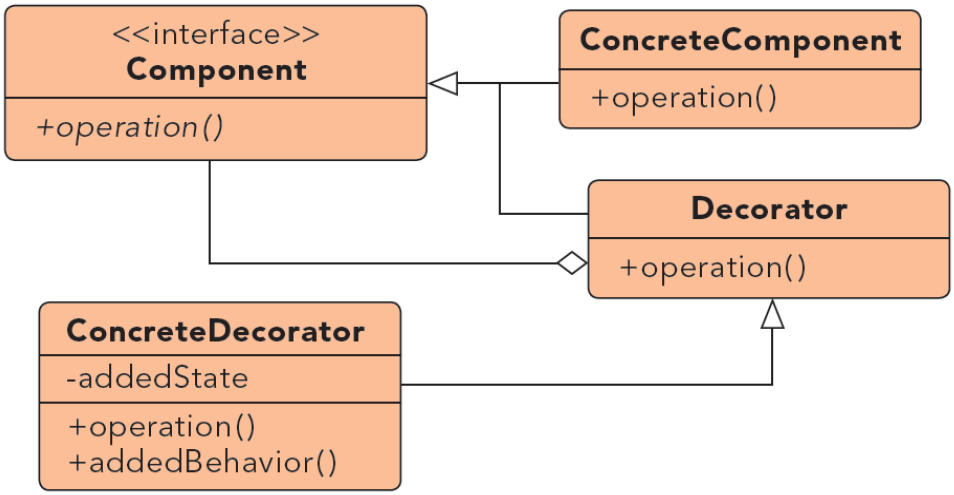

Рис. 1. Структурна схема шаблону «Декоратор»

Учасники шаблону:

–**Component** – клас, який задає інтерфейс для об’єктів, на які можуть бути динамічно покладені додаткові обов’язки, а також для майбутніх декораторів;

–**ConcreteComponent** - визначає об’єкт, до якого додаються нові стани та поведінка;

–**Decorator** – містить посилання на об’єкт **компонент** та успадковує реалізацію його інтерфейсу за замовчанням.

–**ConcreteDecorator** (ConcreteDecoratorA, ConcreteDecoratorB) - конкретні декоратори.

Контрольний приклад

Завдання

У зброярському магазині продаються гвинтівки. Кожна з них має свої характеристики: назва моделі, кількість набоїв тощо. На гвинтівку можна встановити додаткові пристрої: глушник, снайперський приціл і т.д. Створити структуру класів, яка б дозволяла описати функціональні характеристики вдосконаленої гвинтівки.

Розв’язок

```
namespace Decorator_example
{
    class Program
    {
        static void Main(string[] args)
        {
            //Створюємо екземпляр класу "Гвинтівка" та два "Декоратори"
            Weapon R1 = new Rifle("Colt 45m", 20, 80, 100);
            Decorator S1 = new SilencerDecorator();
            Decorator S2 = new SightDecorator();
            //Пов’язуємо декоратори та робимо виклики
            S1.SetGadgetOn(R1);
            S2.SetGadgetOn(S1);
            S2.Show_Info();
            S2.Make_Shot();
            //Очікування на відгук користувача
            Console.ReadKey();
        }
    }

    //Абстрактний клас "Зброя"
    abstract class Weapon
    {
        protected string Model;
        protected int Ammunition;
        protected byte Accuracy;
        protected byte Noisiness;

        public abstract void Show_Info();
        public abstract void Make_Shot();
    }

    //Гвинтівка
    class Rifle : Weapon
    {
        public Rifle(string M,  byte A, byte N, int Am)
        {
            this.Model = M;
            this.Accuracy = A;
            this.Noisiness = N;
            this.Ammunition = Am;
        }

        public override void Show_Info()
        {
            Console.WriteLine("This is rifle.\nModel - {0}\nAccuracy - {1}\nNoisness -
{2}\nAmmunition - {3}",
                this.Model, this.Accuracy, this.Noisiness, this.Ammunition);
        }
```

```
        public override void Make_Shot()
        {
            if (Ammunition == 0)
            {
                Console.WriteLine("Out of ammunition");
                return;
            }
            Ammunition--;
            Console.WriteLine("Shot is made. Ammunition left: {0}", Ammunition);
        }
    }

    //Декоратор
    abstract class Decorator:Weapon
    {
        protected Weapon Wp;

        public void SetGadgetOn(Weapon baseWeapon)
        {
            this.Wp = baseWeapon;
        }

        public override void Show_Info()
        {
            if (Wp != null) Wp.Show_Info();
        }

        public override void Make_Shot()
        {
            if (Wp != null) Wp.Make_Shot();
        }
    }

    //Конкретні декоратори:
    //Глушник
    class SilencerDecorator : Decorator
    {

        public override void Make_Shot()
        {
            Console.Write("Shot was silenced.");
            base.Make_Shot();
        }
        public override void Show_Info()
        {
            base.Show_Info();
            Console.WriteLine("Additional Gadget: Silencer");
        }
    }

    //Приціл
    class SightDecorator : Decorator
    {
        public override void Make_Shot()
        {
            Console.Write("Shot was corrected.");
            base.Make_Shot();
        }


        public override void Show_Info()
        {
            base.Show_Info();
            Console.WriteLine("Additional Gadget: Sight\n");
        }
    }
 }
/*Результат:
This is rifle.
Model - Colt 45m
Accuracy - 20
Noisness - 80
Ammunition - 100
```

```
Additional Gadget: Silencer
Additional Gadget: Sight

Shot was corrected.Shot was silenced.Shot is made. Ammunition left: 99
*/
```

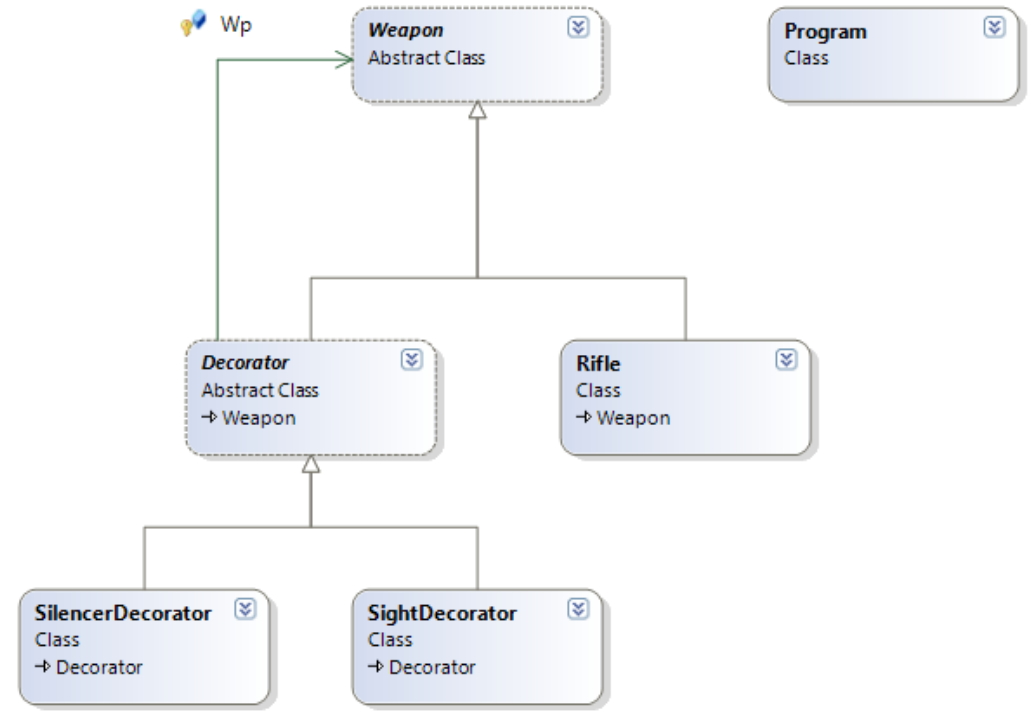

Рис. 2. Діаграма класів шаблону «Декоратор»

Компонувальник

Це структурний шаблон проектування, який 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.

Структурний шаблон «Компонувальник» потрібно використовувати, коли:

–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.

Але також можна розглядати і такі варіанти:

–шаблон «Декоратор», який містить операції типу Додати (Add), Видалити (Remove) та Знайти (Find)

–шаблон «Пристосуванець» для розділення компонентів за умови, що характеристика «місцезнаходження» може бути проігнорована і всі операції будуть починатися з кореневої вершини композиції об’єктів

–шаблон «Відвідувач» для локалізації операцій, які на даний момент розподілені між класами Composite та Component

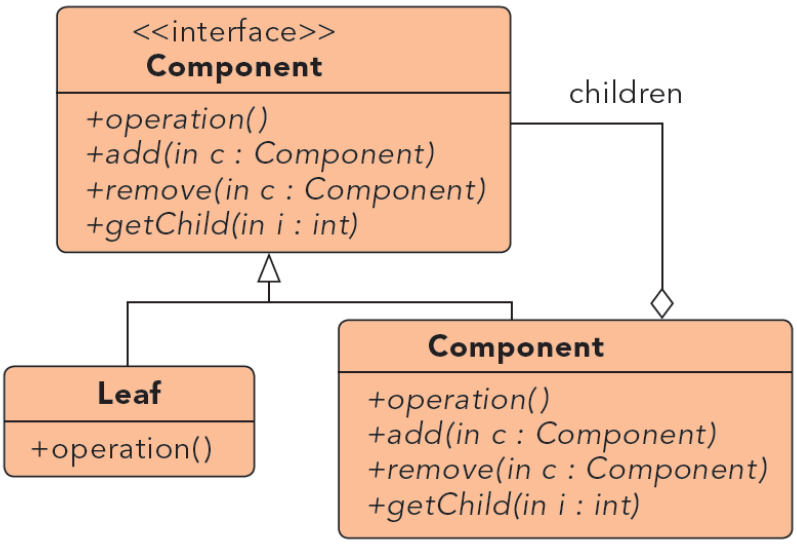

Рис. 3. Структурна схема шаблону «Компонувальник»

Учасники шаблону:

–**Component** – компонент, який задає інтерфейс для об’єктів, котрі компонуються. Також він надає відповідну реалізацію операцій за замовчанням, спільну для всіх класів; об’являє єдиний інтерфейс для

доступу до нащадків та керування ними; визначає інтерфейс для доступу до батьківського елемента компонента у рекурсивній структурі та при необхідності реалізує його;

–**Leaf** – компонент без реалізації контейнерних функцій. Визначає листові вершини та не має нащадків. Визначає поведінку примітивних об’єктів у композиції; входить до складу контейнерів;

–**Composite** – складений об’єкт. Визначає поведінку контейнерних об’єктів, у яких є нащадки. Зберігає ієрархію компонентів-нащадків; реалізує операцій, які відносяться до інтерфейсу управління нащадками.

Контрольний приклад

Завдання

Мікрорайон міста складається з вулиць, вулиці - з будинків, будинки – з квартир. Кожна квартира споживає електроенергію. Створити структуру класів, яка б дозволяла отримувати інформацію щодо споживання електроенергії по квартирі, будинку, вулиці та мікрорайону в цілому.

Розв’язок

```
namespace CompositeExample
{
    class MainApp
    {
        static void Main()
        {
            //district is a root
            Composite district = new Composite("KPI-1");
            //streets
            Composite street1 = new Composite("Metallistov street");
            district.Add(street1);
            Composite street2 = new Composite("Kovalsky lane");
            district.Add(street2);
            //buildings
            Composite build1 = new Composite("build#10");
            Composite build2 = new Composite("build#12");
            Composite build3 = new Composite("build#5");

            street1.Add(build1);
            street1.Add(build2);
            street2.Add(build3);
            Flat fl1 = new Flat("#1");
            Flat fl2 = new Flat("#1");
            Flat fl3 = new Flat("#2");
            Flat fl4 = new Flat("#1");
```

```
            build1.Add(fl1);
            build2.Add(fl2);
            build2.Add(fl3);
            build3.Add(fl4);

            // Display all info
            district.GetElectroInfo();
            district.Display(2);
            Console.ReadKey();
        }
    }

    abstract class ResidentalComponent
    {
        protected string name;
        protected int electro;

        // Constructor
        public ResidentalComponent(string name)
        {
            this.name = name;
        }

        public abstract void Add(ResidentalComponent c);
        public abstract void Remove(ResidentalComponent c);
        public abstract int GetElectroInfo();
        public abstract void Display(int depth);

    }

    /// <summary>
    /// The 'Composite' class
    /// </summary>
    class Composite : ResidentalComponent
    {
        private List<ResidentalComponent> _children = new List<ResidentalComponent>();

        // Constructor
        public Composite(string name) : base(name)
        {
            this.electro = 0;
        }

        public override void Add(ResidentalComponent component)
        {
            _children.Add(component);
        }

        public override void Remove(ResidentalComponent component)
        {
            _children.Remove(component);
        }

        public override void Display(int depth)
        {
            Console.WriteLine(new    String('-',    depth)    +    name    +    "    "    +
this.electro.ToString());
            // Recursively display child nodes
            foreach (ResidentalComponent component in _children)
              component.Display(depth + 2);
         }

        public override int GetElectroInfo()
        {
            this.electro = 0;
            foreach (ResidentalComponent component in _children)
                this.electro += component.GetElectroInfo();
            return this.electro;
        }
    }
```

```
class Flat : ResidentalComponent
    {

        public Flat(string name) : base(name)
        {
            Random rnd = new Random();
            this.electro = rnd.Next(150);
        }
        public override void Add(ResidentalComponent c)
        {
            Console.WriteLine("Impossible operation");
        }
        public override void Remove(ResidentalComponent c)
        {
            Console.WriteLine("Impossible operation");
        }
        public override void Display(int depth)
        {
            Console.WriteLine(new String('-', depth) + name + " " + electro.ToString());
        }
        public override int GetElectroInfo()
        {
            return this.electro;
        }
    }
}

/*Результат:
--KPI-1 32
----Metallistov street 24
------build#10 8
--------#1 8
------build#12 16
--------#1 8
--------#2 8
----Kovalsky lane 8
------build#5 8
--------#1 8
*/
```

Міст

Це структурний шаблон проектування, що дозволяє розділяти абстракцію і реалізацію таким чином, щоб вони могли змінюватися незалежно. Шаблон bridge (від англ. – «міст») використовує інкапсуляцію, агрегування та успадкування для того, щоб розділити відповідальність між класами.

Відомий також під назвою *Handle/Body (описувач/тіло)*.

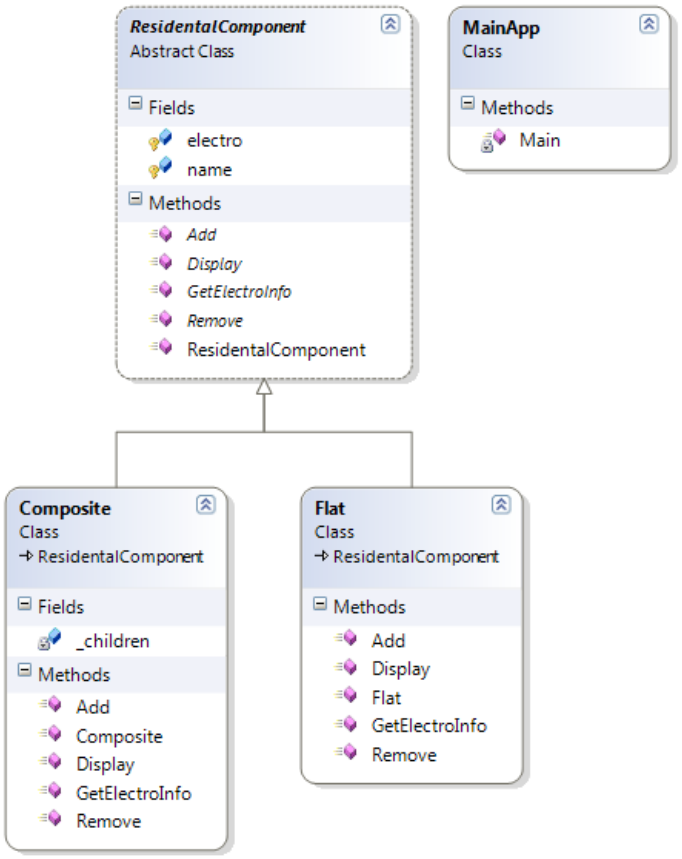

Рис. 4. Діаграма класів шаблону «Компонувальник»

Структурний шаблон «Міст» потрібно використовувати, коли:

– потрібно уникнути постійної прив'язки абстракції до реалізації / абстракція та реалізація не повинні бути зв’язаними під час компіляції;
– конкретну реалізацію необхідно вибирати під час виконання програми;
– абстракція та реалізація повинні бути незалежно розширюваними новими підкласами. У такому випадку «Міст» дозволяє комбінувати різні абстракції та реалізації та змінювати їх незалежно;
– зміни в реалізації абстракції не повинні позначатися на клієнтах, тобто клієнтський код не повинен перекомпілювати;
– деталі реалізації повинні бути прихованими від клієнта;
– кількість класів швидко зростає, що є ознакою того, що ієрархію потрібно розділити на 2 частини.

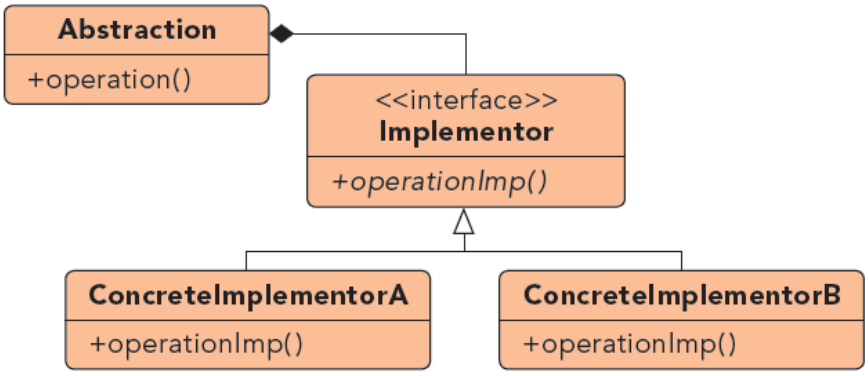

Рис. 5. Структурна схема шаблону «Міст»

Учасники шаблону:

– **Abstraction** визначає інтерфейс абстракції і агрегує (зберігає посилання) на об'єкт типу Implementor;
– **Refined Abstraction** (опціонально) - це більш специфічний підклас основної абстракції, який розширює інтерфейс, визначений нею;
– **Implementor** визначає інтерфейс для класів реалізації. Він не зобов'язаний точно відповідати інтерфейсу класу Abstraction, більше того обидва інтерфейси можуть бути абсолютно різні. Зазвичай інтерфейс класу Implementor надає тільки примітивні операції, а клас Abstraction визначає операції більш високого рівня, що базуються на цих примітивах;
– **ConcreteImplementor** - один з конкретних (платформо-залежних) виконавців.

Контрольний приклад

Завдання

Пасажири метрополітену можуть сплачувати за проїзд або жетонами, або картками, доступна сума на яких поповнюється через спеціальні пристрої. За допомогою шаблону проектування створити структуру класів, яка б дозволяла описати процес оплати за проїзд пасажирами обох типів.

Розв'язок

```
namespace BridgeExample
{
    class Program
    {
        static void Main(string[] args)
        {
            List<AbstractPassenger> passengers = new List<AbstractPassenger>{
                new AbstractPassenger(new CardPassenger()),
                new AbstractPassenger(new CardPassenger()),
                new AbstractPassenger(new TokenPassenger())
            };

            foreach (AbstractPassenger ap in passengers)
                if (ap.PayForTransit())
                    Console.WriteLine("Passenger has paid");
                else
                    Console.WriteLine("Passenger hasn't paid");
            Console.ReadKey();
        }
    }

    interface IPassenger
    {
        bool PayForTransit();
    }

    class AbstractPassenger
    {
        IPassenger pass;
        public AbstractPassenger(IPassenger p)
        {
            this.pass = p;
        }

        public bool PayForTransit()
        {
            return this.pass.PayForTransit();
        }
    }

    class TokenPassenger : IPassenger
    {
        List<float> tokens = new List<float> { 2, 2, 2 };
        public bool PayForTransit()
        {
            if (tokens.Count > 0)
            {
                tokens.Remove(0);
                return true;
            }
            return false;
        }
    }

    class CardPassenger : IPassenger
    {
        float cardsum = 100;
        public bool PayForTransit()
        {
            if (cardsum >= 2)
            {
                cardsum -= 2;
                return true;
            }
            return false;
        }
    }
}
```

```
/*Результат:
Passenger hasn't paid
Passenger hasn't paid
Passenger has paid
*/
```

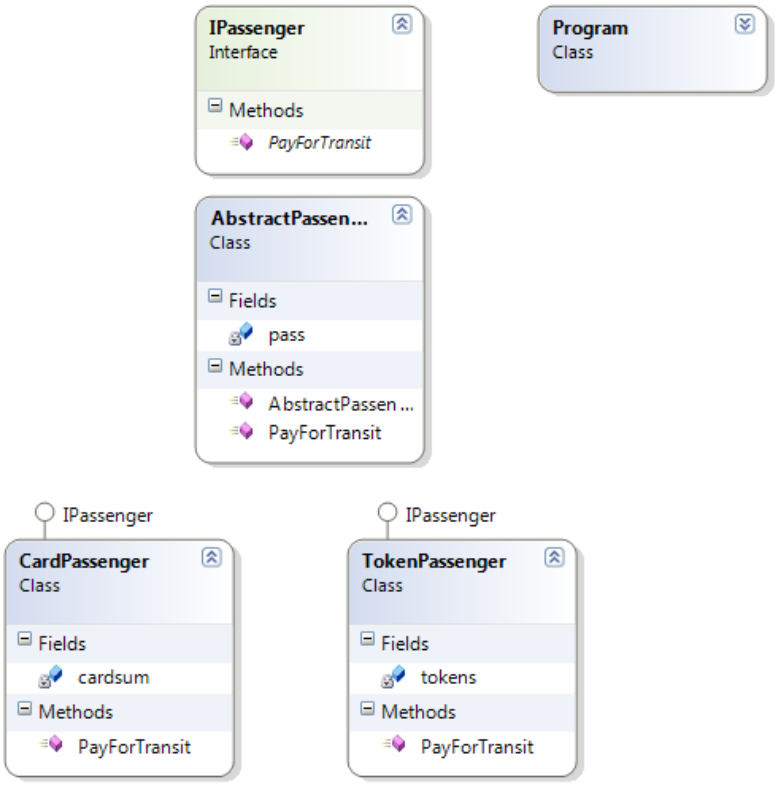

Рис. 6. Діаграма класів шаблону «Міст»

Індивідуальні завдання

Варіант 1

1. В офісі працюють люди різних професій, які займають відповідно різні посади: наприклад, директор, офіс-менеджер, системний адміністратор. Всі вони мають спільні характеристики: ПІБ, робоче місце (кімнату, стіл), заробітну плату, а також спільні функції: прийти на роботу/піти з роботи, отримати зарплатню, звільнитися тощо. Але крім того кожен робітник має свої власні професійні функції, а також інформацію, що його характеризує. Створити загальний базовий клас «Офісний працівник» і на його основі класи, що описують об’єкти

«Директор», «Офіс-менеджер», «Системний адміністратор»; продемонструвати їх роботу.

2. За допомогою шаблона проектування забезпечити виведення на екран меню програми у вигляді, який передбачений політикою безпеки для поточного користувача. Наприклад, не виводити (або робити недоступними) деякі пункти меню для користувачів більш низького рівня. Також передбачити наявність більшої кількості даних про користувачів вищого рівня. Наприклад, дані про банківську картку для оплати рахунків, дані про мобільний телефон тощо

Варіант 2

1. Магазин «Все для дому» реалізує господарські товари. Всі товари мають спільні характеристики: наприклад, назва, ціна, відділ, де вони продаються, термін придатності чи гарантійний термін, а також оригінальні характеристики, які залежать від виду товару. Одні товари продаються тільки вроздріб, а інші можуть бути продані й оптом (при цьому ціна товару стає на 10% меншою). Створити базовий клас «Товар» та похідні класи, що міститимуть додаткову інформацію про товари, які можуть бути проданими як вроздріб, так і оптом.

2. Розробити програмне забезпечення автоматичної зйомки фотоапаратом. В залежності від освітлення та часу виконувати підключення тих чи інших можливостей фотоапарату: спалаху, знищення ефекту червоних очей, автофокус на посмішці, знімання об’єкту, що рухається тощо.

Варіант 3

1. У штаті вищого навчального закладу є багато різних типів працівників, зокрема науково-педагогічні працівники, адміністративні працівники та методисти. Всі вони мають спільні характеристики: наприклад, ПІБ, корпус та кімната, де знаходиться робоче місце, факультет тощо. Але крім того всі вони мають і свої власні обов'язки: наприклад, читання лекцій, проведення іспитів, складання розкладу занять, моніторинг успішності навчального процесу тощо. Розробити базовий клас «Працівник вузу», а також похідні від нього класи, що реалізують характерні функції працівників та містять додаткову інформацію про них.

2. За допомогою відповідного шаблона проектування створити структуру класів, до якої будуть входити класи, які реалізуються у об'єкти «Студент» та «Група студентів». Обидва об'єкти повинні мати імена та метод «Показати інформацію». Крім того, забезпечити підтримку об'єктами метода «Скласти іспит».

Варіант 4

1. В аптеці продаються ліки різних виробників. Всі вони мають назву, групу ліків, до якої вони належать (антибіотики, протизапальні, шлункові тощо), ціну, термін зберігання. Однак, імпортні ліки повинні мати додатковий сертифікат про проходження препаратом лабораторного контролю в Україні. Крім того, у випадку замовлення покупцем імпортного препарату, фармацевт (якщо звернеться до бази даних ліків в аптеці) повинен побачити не тільки інформацію про наявність ліків в аптеці та їх ціну, а й дані про вітчизняні аналоги препарату, які як правило, є дешевшими. Розробити структуру класів, які можна використовувати для

комп’ютерної обробки даних про ліки (як вітчизняні, так й імпортні) в аптеці.

2. За допомогою шаблона проектування створити структуру класів, до якої будуть входити класи, які відповідають таким об’єктам як Лікар, Відділення та Поліклініка. Кожний такий об’єкт має своє ім’я. Поліклініка та відділення характеризуються кількістю лікарів та хворих, а також мають власних завідувачів. У програмі передбачити реалізацію спеціального методу – «Пройти медичний огляд» у одного, декількох або всіх лікарів.

Варіант 5

1. На виставці собак всі учасники мають такі спільні характеристики як зріст, вага, вік, порода, ім’я хазяїна. Однак, деякі породи собак можуть мати додаткові обов’язкові характеристики: наприклад, куповані хвіст та вушка, додаткові щеплення тощо. Крім того, різняться також і функції, природні для собак: одні є мисливськими, інші – бійцівськими. Отже, крім загальних команд, вони мають вміти виконувати і ті, що притаманні лише їх породі. Створити базовий клас «Собака» та декілька класів, що характеризують певні породи. Продемонструвати програмну реалізацію даних класів.

2. За допомогою шаблона проектування реалізувати механізм видання заробітної плати у навчальному закладі. Спочатку гроші надходять на рахунок самого закладу, потім на рахунки підрозділів, потім конкретному співробітнику. Відповідно кожний підрозділ має відомість, що містить список його співробітників та розмір їх заробітної плати.

Варіант 6

1. Програмний текстовий чат версії 1.0 підтримує виведення тексту повідомлення у віконці діалогу з іншим користувачем у звичайному вигляді (шрифт 14 кегль, Times New Roman, чорний). На основі вже існуючого движка чату реалізувати виведення тексту заданим кольором, типом шрифта та розміром. Також на початку повідомлення додати час його надходження.

2. За допомогою шаблона проектування реалізувати механізм збирання коштів на ремонт школи. Кожен класний керівник збирає кошти з батьків свого класу. Кожний завуч збирає гроші з керівників старших, середніх та молодших класів. Директор збирає гроші із завучів.

Варіант 7

1. Всі тварини сімейства котячих мають спільні риси: зріст, вагу, середовище проживання, вік, породу. Ті тварини, які мешкають у зоопарку, цирку, дома у людей, мають також ім’я. Кожна порода, окрім загальних дій, притаманних всім котячим (їсти, спати тощо), має свої власні дії (великі хижаки – рись, пума – полюють на здобич, домашні коти – граються із сонячним зайчиком тощо). Створити базовий клас «Тварина сімейства котячі», а також похідні від нього класи, які наслідують загальні риси та дії від нього, а також реалізують додаткові дії, притаманні конкретній тварині. Продемонструвати реалізацію структури класів.

2. За допомогою шаблона проектування реалізувати механізм доведення наказу директора до всіх співробітників компанії. Директор інформує про наказ всіх своїх заступників на нараді. Заступники директора, в свою чергу, передають інформацію всім начальникам відділів, які їх підпорядковуються. Якщо начальник якогось відділу

підпорядковується одразу декільком заступникам директора, тоді він отримає звістку про наказ декілька разів. Начальники відділів передають інформацію про наказ всім своїм підлеглим

Варіант 8

1. За допомогою використання шаблона Bridge створіть програму-калькулятор, яка працює в 2 режимах: звичайного калькулятора (забезпечує підтримку 4 основних математичних операцій: додавання, віднімання, множення, ділення) та інженерного калькулятора (забезпечує підтримку 4 основних математичних операцій, а також видобування квадратного кореня, обчислення залишку від ділення, піднесення числа до заданого ступеня).

2. За допомогою шаблона проектування реалізувати механізм продажу косметики через мережу дистриб'юторів (мережевий маркетинг). Кожна людина може бути або кінцевим продавцем косметики, або менеджером певної кількості підлеглих продавців, яким він збуває товар. Кожний менеджер продає косметику на певний відсоток дорожче, ніж він її отримав. Але перевищити кінцеву роздрібну вартість продукту, встановлену компанією, дистриб'ютор не має права.

Варіант 9

1. Організувати виконання команди «Голос!» (а також, можливо, інших команд) різними тваринами у цирку.

2. За допомогою шаблона проектування реалізувати модель мікрорайону. Наприклад, район повинен складатися з вулиць, вулиці містять будинки; будинки складаються з під'їздів та квартир. Забезпечити виведення на екран інформації про кожний тип об'єктів (кількість

населення вулиці, будинку тощо). Також забезпечити можливість додавання та видалення об’єктів всіх типів. Крім того, реалізувати метод «Переписати населення», який передбачає підсумовування населення мікрорайону шляхом обчислення населення вкладених в нього об’єктів.

Варіант 10

1. За допомогою шаблона реалізувати програму, яка демонструє складання тесту об’єктами класу «віртуальний студент». Відомо, що такий студент може бути декількох типів: відмінник, такий, що навчається добре, задовільно та незадовільно. Умовно встановимо, що відмінник завжди відповідає на всі поставлені запитання; студент, що навчається добре відповідає на 80% запитань; задовільно – на 60% запитань та незадовільно – на 40% запитань.

2. Домашні тварини (кішки та собаки) характеризуються віком та іменем. Вони живуть у квартирах, які знаходяться у під’їздах, що розташовані у будинках. ЗА допомогою шаблона проектування забезпечити можливість виведення списку тварин по квартирам, під’їздам, будинкам з вказівкою імені та віку. Крім цього розробити метод визначення середнього віку кішок та собак, які живуть на заданій території.

Варіант 11

1. Реалізувати різноманітні методи проведення віртуального медичного огляду лікарями як мінімум трьох спеціалізацій (стоматолог, хірург, ортопед і т.ін.). Забезпечити підтримку віртуального медичного огляду певною множиною лікарів.

2. У банку вакансій кожна позиція характеризується назвою, компанією, що її пропонує, категорією вакансії та регіоном. Забезпечити виведення на екран інформації про вакансії будь-якого типу групування. Також реалізувати виведення тільки кількості вакансій з різними типами групування (за компанією, за категорією, за регіоном тощо).

Варіант 12

1. Реалізувати виведення інформації про користувачів, зареєстрованих у програмі, у різних форматах в залежності від типа користувача. Наприклад, повна інформація (звичайний користувач), обмежена інформація (адміністратор системи), мінімум інформації (vip-користувач).

2. У вищому навчальному закладі студент є частиною навчальної групи. Група входить до складу потоку на кафедрі та курсу на факультеті в цілому. І студенти, і група, і факультет мають ім’я або назву. Також студент характеризується масивом оцінок, які він отримує протягом сесії. Потік студентів складається з масиву груп, курс – з масиву потоків, факультет – з масиву курсів. Організувати виведення оцінок студентів, згрупувавши їх по групам, потокам, курсам. Також реалізувати метод обчислення середнього балу для студента, групи тощо.

Варіант 13

1. Реалізувати виведення системних повідомлень програми (попередження про помилки, інформаційні вікна, діалогові вікна тощо) мовою, зрозумілою користувачу, який працює з даною системою. Мова є параметром профілю користувача та встановлюється для кожного

користувача окремо. Нехай українська мова буде мовою, що встановлюється за замовчанням.

2. За допомогою шаблона проектування створити структуру класів, до якої будуть входити класи, які відповідають особам, з яких сформовано список адресатів розсилки листа. Адресати можуть бути як окремими людьми, так і групами людей, об’єднаних за певним критерієм. Крім того, адресат може бути фізичною особою або цілою компанією. Забезпечити виконання методу «Відправити лист» будь-якому списку адресатів

Варіант 14

1. Реалізувати масив даних, в якому зберігається інформація про покупки в інтернаціональному магазині. Організувати при виконанні команди «Купити товар» видачу покупцеві чеку про оплату товару у форматі, притаманному регіону проживання покупця (формат чека впливає на форму відображення часу та дати покупки, валюти, найменування товару тощо).

2. Структура дитячого оздоровчого табору має такий вигляд: загін – палата – окрема дитина. Кожна палата та загін мають свою назву. Організувати метод, який віртуально демонструє виконання команд «підйом» та «відбій» по цілому табору.

Варіант 15

1. Розробити програму, яка реалізує декілька можливих рівнів захисту. Користувач може обирати ступінь захисту своєї програми: наприклад, вільний доступ; доступ за паролем; введення спеціального коду з 4 цифр тощо.

2. За допомогою шаблона проектування забезпечити виведення на екран власних ініціалів, які складаються з примітивних графічних об’єктів (ліній, дуг і т.ін.).

Варіант 16

1. Реалізувати декілька різноманітних способів збегірання текста до файлу. Перший спосіб – звичайний: просто зберігаємо текст без жодних змін. Другий спосіб – видаляємо всі зайві пробіли з тексту перед збереженням. Третій спосіб – застосовуємо кодування текста (або архівацію). В залежності від необхідності забезпечити збереження тексту до файлу доцільним способом.

2. За допомогою шаблона проектування реалізувати модель бібліотеки. Бібліотека містить відділи, відділи – стелажі з різної тематики (наприклад, література 19 ст., детективи тощо), стелажі складаються з полиць, які в свою чергу містять книжки. Кожна книга має свій унікальний код та шифр. Книг з однаковим кодом не може бути. А екземпляри однієї книжки можуть мати однаковий шифр. Забезпечити виведення на екран інформації про об’єкти будь-якого типу (книга, полиця тощо). Крім того, реалізувати виведення кількості найменувань (не екземплярів!) книг на полиці, стелажі, відділі, а також загальну кількість книг на полиці, стелажі, у відділі.

Варіант 17

1. Створити декілька видів клієнтських карток у мережі ресторанів. Карткою першого виду передбачається надання 5% знижки на вартість замовлення. Картка другого виду надає можливість отримати 10% знижки на вартість замовлення, а також 2 безалкогольні напої безкоштовно

(у разі замовлення не менше, ніж на суму 100 грн.). Картка третього виду передбачає можливість отримання 20% знижки на замовлення та пляшки вина у подарунок від ресторану (у разі замовлення, незалежно від суми замовлення). Продемонструвати, як працює метод «Обчислити суму замовлення», за допомогою створення відповідних абстракції та її реалізації. Передбачити випадок, коли клієнт ресторану не має картки взагалі.

2. За допомогою шаблона проектування реалізувати модель магазину з медіа-продукцією. Магазин має відділи (наприклад: ігри, кіно, музика тощо), відділи складаються зі стелажів (наприклад: Ігри. Стратегії, Ігри.Квести, Кіно.Детективи тощо), на яких розміщено саму продукцію. Кожний товар має декілька копій (екземплярів). Забезпечити виведення на екран інформації про товари (причому виводити дані тільки про сам товар, а не про кількість його екземплярів). Крім того, реалізувати виведення даних про кількість товару на стелажі, у відділі, та в магазині в цілому.

Варіант 18

1. Гра «Морський бій». Реалізувати шаблон проектування, який дозволяє розміщувати на гральному полі одно-, дво- та трипалубні кораблі. Також в залежності від типу корабля забезпечити визначення його стану (цілий, поранений, вбитий).

2. За допомогою шаблона проектування забезпечити виведення на екран власних ініціалів, які складаються з примітивних графічних об’єктів (ліній, дуг і т.ін.).

Варіант 19

1. Розробити програмне забезпечення автоматичної зйомки фотоапаратом. В залежності від освітлення та часу виконувати підключення тих чи інших можливостей фотоапарату: спалаху, знищення ефекту червоних очей, автофокус на посмішці, знімання об’єкту, що рухається тощо.

2. За допомогою відповідного шаблона проектування створити структуру класів, до якої будуть входити класи, які реалізуються у об’єкти «Студент» та «Група студентів». Обидва об’єкти повинні мати імена та метод «Показати інформацію». Крім того, забезпечити підтримку об’єктами метода «Скласти іспит».

Варіант 20

1. В офісі працюють люди різних професій, які займають відповідно різні посади: наприклад, директор, офіс-менеджер, системний адміністратор. Всі вони мають спільні характеристики: ПІБ, робоче місце (кімнату, стіл), заробітну плату, а також спільні функції: прийти на роботу/піти з роботи, отримати зарплатню, звільнитися тощо. Але крім того кожен робітник має свої власні професійні функції, а також інформацію, що його характеризує. Створити загальний базовий клас «Офісний працівник» і на його основі класи, що описують об’єкти «Директор», «Офіс-менеджер», «Системний адміністратор»; продемонструвати їх роботу.

2. Реалізувати різноманітні методи проведення віртуального медичного огляду лікарями як мінімум трьох спеціалізацій (стоматолог, хірург, ортопед і т.ін.). Забезпечити підтримку віртуального медичного огляду певною множиною лікарів.

Варіант 21

1. За допомогою шаблона проектування створити структуру класів, до якої будуть входити класи, які відповідають таким об’єктам як Лікар, Відділення та Поліклініка. Кожний такий об’єкт має своє ім’я. Поліклініка та відділення характеризуються кількістю лікарів та хворих, а також мають власних завідувачів. У програмі передбачити реалізацію спеціального методу – «Пройти медичний огляд» у одного, декількох або всіх лікарів.

2. Організувати виконання команди «Голос!» (а також, можливо, інших команд) різними тваринами у цирку.

Варіант 22

1. За допомогою шаблона проектування реалізувати модель мікрорайону. Наприклад, район повинен складатися з вулиць, вулиці містять будинки; будинки складаються з під’їздів та квартир. Забезпечити виведення на екран інформації про кожний тип об’єктів (кількість населення вулиці, будинку тощо). Також забезпечити можливість додавання та видалення об’єктів всіх типів. Крім того, реалізувати метод «Переписати населення», який передбачає підсумовування населення мікрорайону шляхом обчислення населення вкладених в нього об’єктів.

2. Реалізувати виведення інформації про користувачів, зареєстрованих у програмі, у різних форматах в залежності від типа користувача. Наприклад, повна інформація (звичайний користувач), обмежена інформація (адміністратор системи), мінімум інформації (vip-користувач).

Варіант 23

1. У банку вакансій кожна позиція характеризується назвою, компанією, що її пропонує, категорією вакансії та регіоном. Забезпечити виведення на екран інформації про вакансії будь-якого типу групування. Також реалізувати виведення тільки кількості вакансій з різними типами групування (за компанією, за категорією, за регіоном тощо).

2. У вищому навчальному закладі студент є частиною навчальної групи. Група входить до складу потоку на кафедрі та курсу на факультеті в цілому. І студенти, і група, і факультет мають ім’я або назву. Також студент характеризується масивом оцінок, які він отримує протягом сесії. Потік студентів складається з масиву груп, курс – з масиву потоків, факультет – з масиву курсів. Організувати виведення оцінок студентів, згрупувавши їх по групам, потокам, курсам. Також реалізувати метод обчислення середнього балу для студента, групи тощо.

Варіант 24

1. У штаті вищого навчального закладу є багато різних типів працівників, зокрема науково-педагогічні працівники, адміністративні працівники та методисти. Всі вони мають спільні характеристики: наприклад, ПІБ, корпус та кімната, де знаходиться робоче місце, факультет тощо. Але крім того всі вони мають і свої власні обов’язки: наприклад, читання лекцій, проведення іспитів, складання розкладу занять, моніторинг успішності навчального процесу тощо. Розробити базовий клас «Працівник вузу», а також похідні від нього класи, що реалізують характерні функції працівників та містять додаткову інформацію про них.

2. За допомогою шаблона проектування реалізувати механізм видання заробітної плати у навчальному закладі. Спочатку гроші

надходять на рахунок самого закладу, потім на рахунки підрозділів, потім конкретному співробітнику. Відповідно кожний підрозділ має відомість, що містить список його співробітників та розмір їх заробітної плати.

Варіант 25

1. Гра «Морський бій». Реалізувати шаблон проектування, який дозволяє розміщувати на гральному полі одно-, дво- та трипалубні кораблі. Також в залежності від типу корабля забезпечити визначення його стану (цілий, поранений, вбитий).

2. За допомогою шаблона проектування забезпечити виведення на екран власних ініціалів, які складаються з примітивних графічних об’єктів (ліній, дуг і т.ін.).

Контрольні запитання

1. Які шаблони ми називаємо структурними шаблонами проектування?
2. Дайте визначення шаблону «Декоратор» («Компонувальник», «Міст»).
3. У чому полягає призначення шаблону «Декоратор» («Компонувальник», «Міст»)?
4. Назвіть переваги та недоліки застосування шаблону «Декоратор» («Компонувальник», «Міст»).
5. Наведіть UML-діаграму класів, які утворюють структуру шаблона «Декоратор» («Компонувальник», «Міст»).

2.2. Лабораторна робота №2

Тема роботи: Реалізація складних структурних шаблонів проектування.

Мета роботи: ознаойомлення з основними характеристиками шаблонів «Заступник», «Фасад», «Адаптер» та «Легковаговик», запам’ятовування поширених ситуацій, коли використання цих шаблонів є доцільним, набуття вмінь та навичок реалізації шаблонів під час створення програмного коду.

Теоретичні відомості

Заступник

Це структурний шаблон проектування, який забезпечує створення заступника об’єкта для контролю доступу до останнього через перехоплення всіх викликів.

Надає об’єкт-заступник (surrogate) або об’єкт-замінник (placeholder).

Обгортаючи доступ до реального компонента, «Заступник»зменшує складність роботи з ним.

Структурний шаблон «Заступник» потрібно використовувати, коли:

- створення об’єкту є “дорогою” операцією – можна створювати об’єкт тільки при першому зверненні до нього;
- потрібно контролювати доступ до об’єкту: для різних об’єктів можна встановлювати різні рівні доступу;
- потрібно організувати віддалений функціонал – заступник надає локального представника замість цільового об’єкта, що знаходиться в іншому адресному просторі;
- потрібно виконувати додаткові дії при доступі до об'єкта – «розумне посилання».

Види шаблону «Заступник»

– **віртуальний заступник**: управляє створенням одного об’єкта через інший тоді, коли цей об’єкт реально потрібний (доцільно у випадках, якщо процес створення досить повільний або може бути визнаним непотрібним). Має назву ще **lazy loading, on-demand loading, just-in-time loading.** Може кешувати частину інформації про реальний Суб’єкт, щоб відкласти його створення;

– **заступник для аутентифікації (заступник-захисник)**: перевіряє правильність умов доступу до об’єкту;

– **віддалений заступник**: забезпечує зв’язок з Суб’єктом, який знаходиться у іншому адресному просторі (домені застосунку, процесі чи комп’ютері); кодує/декодує повідомлення, які надсилаються мережею;

– **розумний заступник:** додає щось до запиту перед його надсиланням до мережі або змінює запит;

– **кешуючий заступник**: забезпечує тимчасове зберігання результатів розрахунків до надсилання їх клієнтам, які можуть розділити ці результати.

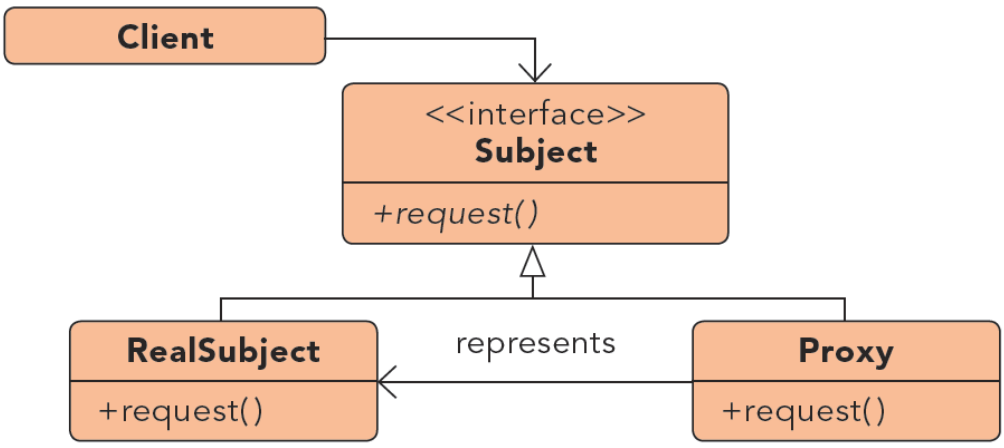

Рис. 7. Структурна схема шаблону «Заступник»

Учасники шаблону:

–**Proxy :**

- зберігає посилання, яке дозволяє йому звертатися до реального суб'єкту, використовуючи інтерфейс класу Subject;
- надає інтерфейс, ідентичний інтерфейсу Subject, тому заступник завжди може бути підставлений замість реального суб'єкта;
- контролює доступ до реального суб'єкту і може відповідати за його створення і видалення;
- виконує інші обов'язки, які залежать від виду заступника (remote proxies, virtual proxies, protection proxies etc.);

–**Subject** – визначає спільний для **RealSubject** і **Proxy** інтерфейс, так що клас **Proxy** можна використовувати скрізь, де очікується **RealSubject;**

–**RealSubject** – визначає реальний об'єкт, який подається за допомогою заступника.

Контрольний приклад

Завдання

За допомогою шаблону проектування змоделювати роботу охорони нічного клубу, яка не пускає до закладу людей напідпитку.

Розв’язок

```
namespace ProxyExample
{
    class Program
    {
        static void Main(string[] args)
        {
            FaceControlledClub Prime_Facecontrol = new FaceControlledClub();
            Man m1 = new Man() { name="John", drunk=true};
            Man m2 = new Man() { name = "Carl", drunk = false };
            Prime_Facecontrol.Enter_Club(m1);
            Prime_Facecontrol.Enter_Club(m2);
            Console.ReadKey();
        }
    }
```

```
    class Man
    {
        public string name;
        public bool drunk;
    }

    //interface
    abstract class Abst_Club
    {
        public abstract void Enter_Club(Man m);
    }

    //realsubject
    class Club : Abst_Club
    {
        public override void Enter_Club(Man m)
        {
            Console.WriteLine("Welcome to our club, {0}", m.name);
        }
    }
    //proxy
    class FaceControlledClub : Abst_Club
    {
        Club club = new Club();

        public override void Enter_Club(Man m)
        {
            if (!m.drunk)
                club.Enter_Club(m);
            else
                Console.WriteLine("Not today, man :(");
        }
    }
}
/*Результат:
Not today, man :(
Welcome to our club, Carl
*/
```

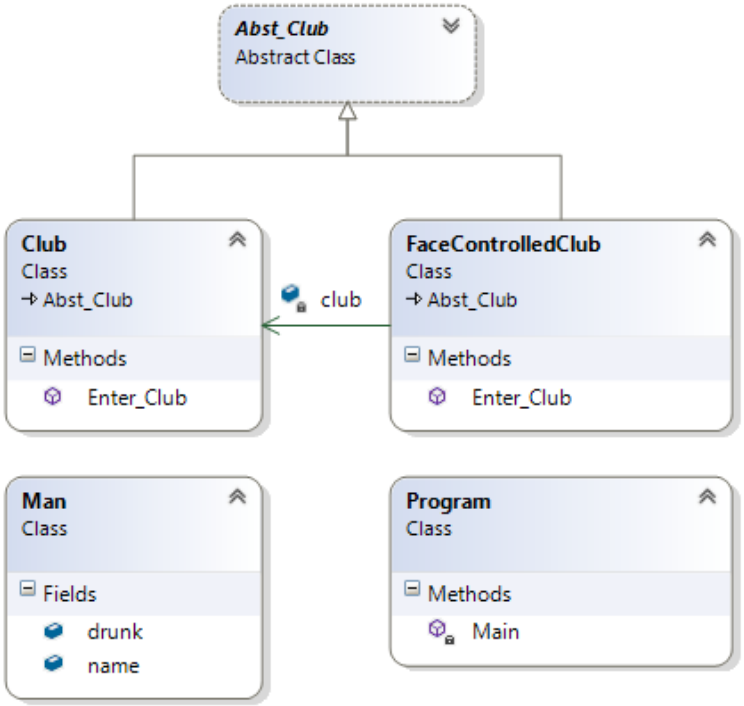

Рис. 8. Діаграма класів шаблону «Заступник»

Фасад

Це структурний шаблон проектування, який структурує об’єкти, надаючи до них всіх доступ через єдиний шлюз. Надає єдиний, уніфікований інтерфейс до всієї підсистеми замість набору окремих та багаточисельних інтерфейсів. Фактично, “Фасад” визначає інтерфейс більш високого рівня, який спрощує використання системи.

Структурний шаблон «Фасад» потрібно використовувати, коли:

–є необхідність у наданні простого інтерфейсу доступу до складної системи;

–між клієнтами та класами реалізації абстракції існує багато залежностей і треба зменшити їхню кількість;

–є необхідність розкласти підсистему на окремі шари, створити різні рівні доступу до підсистеми, розшарувати її.

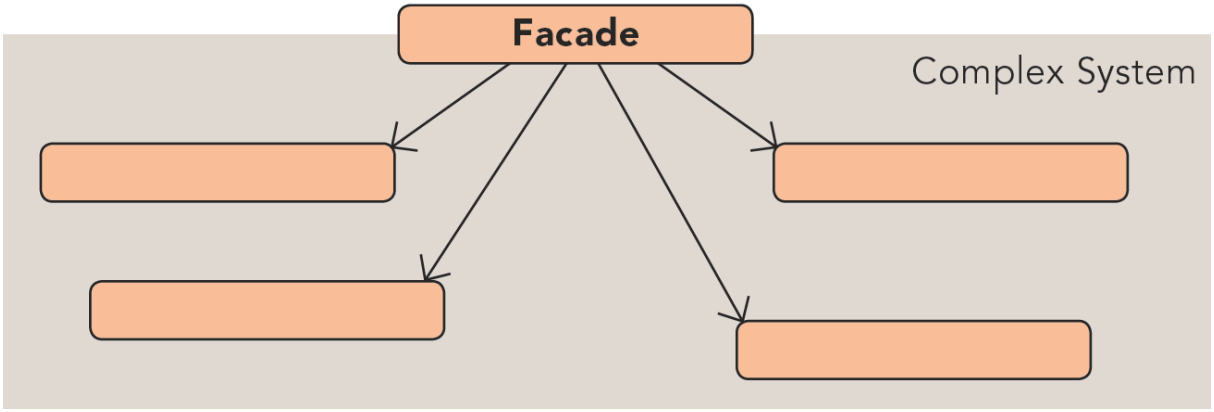

Рис. 9. Структурна схема шаблону «Фасад»

Учасники шаблону:

– **Facade** – фасад:
  - проінформований про те, яким складовим системи потрібно переадресовувати запит;

- делегує запити клієнтів відповідним об’єктам всередині підсистеми;

– **Класи підсистеми**:
  - реалізують функціональність підсистеми;
  - виконують дії, які потребує «Фасад» і які, в свою чергу, «Фасад» отримав від одного з клієнтів;
  - нічого не “знають” про існування самого «Фасаду» (не зберігають посилань на нього);
  - реалізація компонентів підсистеми закрита та не видна зовнішнім компонентам.

Контрольний приклад

Завдання

У будівельній бригаді працюють маляр, лицювальник, штукатур та 2 різнорабочих, які допомагають майстрам. Накази про виконання конкретної роботи (фарбувати, штукатурити, лицювати) надходять від бригадира. Використовуючи шаблон проектування створити структуру класів, шо моделює роботу даної будівельної бригади.

Розв’язок

```
namespace FacadeExample
{
    class Program
    {
        static void Main(string[] args)
        {
            Brigadier Mamonov = new Brigadier("Mbenga", "Zelmambekov", "Niguzaev", "Menshikov", "Ryazantcev");
            Mamonov.To_Coat();
            Console.ReadKey();
        }
    }
    //Бригадир - Фасад
    class Brigadier
    {
        Plasterer plasterer;
        Painter painter;
        Tiler tiler;
        Swamper swamper1;
        Swamper swamper2;
```

```
    public Brigadier(string plasterer_name, string painter_name,
        string tiler_name, string swamper1_name, string swamper2_name)
    {
        plasterer = new Plasterer(plasterer_name);
        painter = new Painter(painter_name);
        tiler = new Tiler(tiler_name);
        swamper1 = new Swamper(swamper1_name);
        swamper2 = new Swamper(swamper2_name);
    }
    public void To_Coat()//лицювати
    {
        tiler.Do_Work();
        swamper1.Help(tiler);
    }
    public void To_Paint()//фарбувати
    {
        painter.Do_Work();
    }
    public void To_Plaster()//штукатурити
    {
        plasterer.Do_Work();
        swamper2.Help(plasterer);
    }
}

abstract class SpecWorker
{
    public string Name;
    public abstract void Do_Work();
}

//Штукатур
class Plasterer : SpecWorker
{
    public Plasterer(string N)
    {
        Name = N;
    }
    public override void Do_Work()
    {
        Console.WriteLine("Plasterer {0} is doing his work", Name);
    }
}

//Маляр
class Painter : SpecWorker
{
    public Painter(string N)
    {
        Name = N;
    }
    public override void Do_Work()
    {
        Console.WriteLine("Painter {0} is doing his work", Name);
    }
}

//Лицювальник
class Tiler : SpecWorker
{
    public Tiler(string N)
    {
        Name = N;
    }
    public override void Do_Work()
    {
        Console.WriteLine("Tiler {0} is doing his work", Name);
    }
}
```

```
        //Різноробочий
        class Swamper
        {
            string Name;
            public Swamper(string N)
            {
                Name = N;
            }
            public void Help(SpecWorker master)
            {
                Console.WriteLine("Swamper {0} is helping master {1}", this.Name,
master.Name);
            }
        }
    }

    /*Результат:
     Tiler Niguzaev is doing his work
    Swamper Menshikov is helping master Niguzaev
    */
```

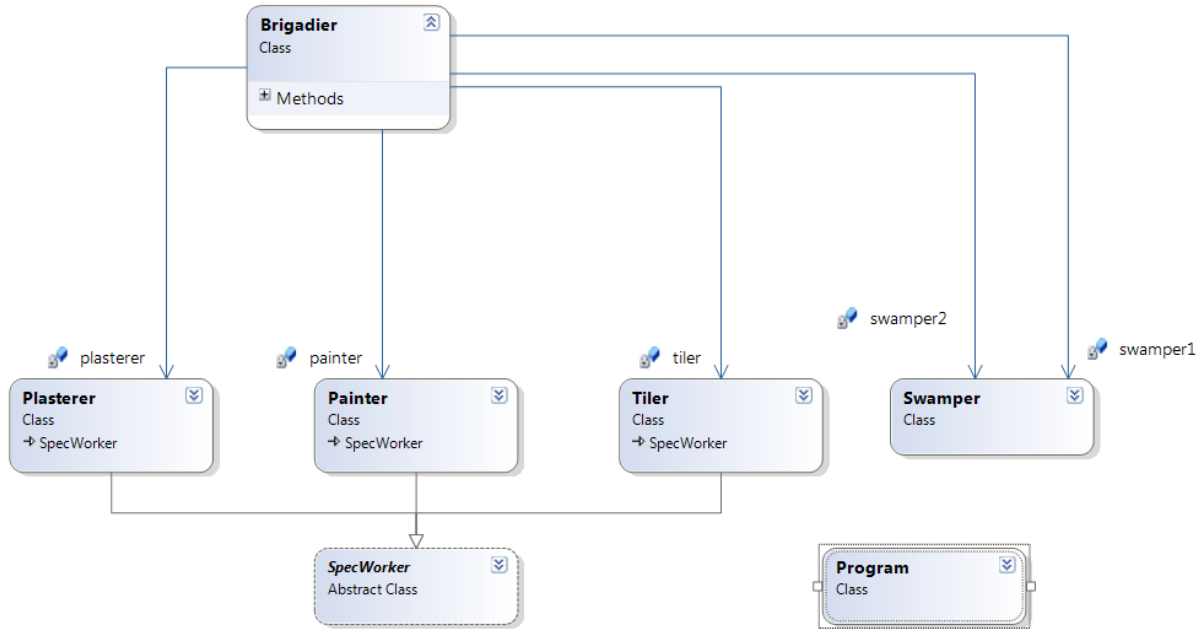

Рис. 10. Діаграма класів шаблону «Фасад»

Адаптер

Це структурний шаблон проектування, призначений для організації використання функцій об'єкта, недоступного для модифікації, через спеціально створений інтерфейс.

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

«Адаптер» – шаблон, що уніфікує класи та об’єкти.

Структурний шаблон «Адаптер» потрібно використовувати, коли:

– існуючий клас підтримує необхідні дані і поведінку, але має інтерфейс, який не відповідає певним вимогам (наприклад, при використанні існуючої бібліотеки, до класів якої немає доступу);
– необхідно створити повторно використовуваний клас, який повинен взаємодіяти із зазделегідь невідомими або непов’язаними з ним класами, які мають несумісні інтерфейси;
– необхідно об’єднати інтерфейси декількох класів, причому в наявності можуть бути тільки об’єкти-підкласи кожного з цих класів, а не примірники цих класів.

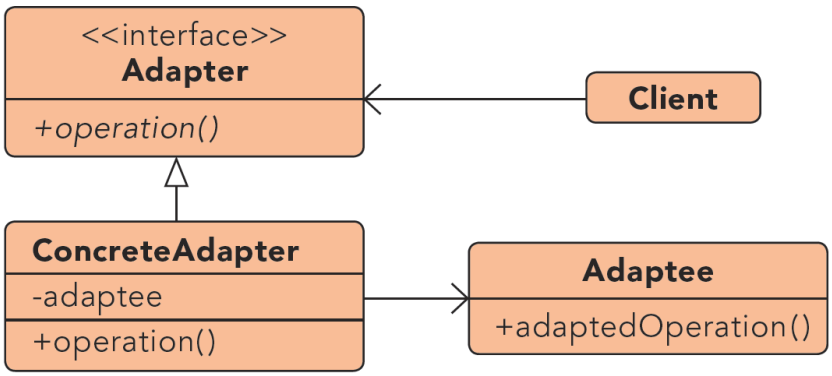

Рис. 11. Структурна схема шаблону «Адаптер»

Учасники шаблону:

– **Target** - цільовий інтерфейс. Визначає інтерфейс, який залежить від предметної області і яким користується **Client**;
– **Client** - взаємодіє з об'єктами, що задовольняють інтерфейсу **Target**;

– **Adaptee** - *а*даптований інтерфейс. Визначає існуючий інтерфейс класу (бібліотеки), який потребує адаптації;
– **Adapter** - клас, що адаптує інтерфейс **Adaptee** до інтерфейсу **Target.**

Контрольний приклад

Завдання

За допомогою шаблона проектування забезпечити ефективну роботу з банком хімічних даних. Об’єкти, що містять дані про хімічні сполуки, мають доступ до банку даних через спеціальний адаптований інтерфейс.

Розв’язок

```
namespace AdapterExample
{
  class MainApp
  {
    static void Main()
    {
      // Non-adapted chemical compound
      Compound unknown = new Compound("Unknown");
      unknown.Display();

      // Adapted chemical compounds
      Compound water = new RichCompound("Water");
      water.Display();
      Compound benzene = new RichCompound("Benzene");
      benzene.Display();
      Compound ethanol = new RichCompound("Ethanol");
      ethanol.Display();
      // Wait for user
      Console.ReadKey();
    }
  }

  // The 'Target' class
  class Compound
  {
    protected string _chemical;
    protected float _boilingPoint;
    protected float _meltingPoint;
    protected double _molecularWeight;
    protected string _molecularFormula;

    // Constructor
    public Compound(string chemical)
    {
      this._chemical = chemical;
    }

    public virtual void Display()
    {
      Console.WriteLine("\nCompound: {0} ------ ", _chemical);
    }
  }
```

```
// The 'Adapter' class
class RichCompound : Compound
{
  private ChemicalDatabank _bank;

  // Constructor
  public RichCompound(string name)
    : base(name)
  {
  }

  public override void Display()
  {
    // The Adaptee
    _bank = new ChemicalDatabank();

    _boilingPoint = _bank.GetCriticalPoint(_chemical, "B");
    _meltingPoint = _bank.GetCriticalPoint(_chemical, "M");
    _molecularWeight = _bank.GetMolecularWeight(_chemical);
    _molecularFormula = _bank.GetMolecularStructure(_chemical);

    base.Display();
    Console.WriteLine(" Formula: {0}", _molecularFormula);
    Console.WriteLine(" Weight : {0}", _molecularWeight);
    Console.WriteLine(" Melting Pt: {0}", _meltingPoint);
    Console.WriteLine(" Boiling Pt: {0}", _boilingPoint);
  }
}

// The 'Adaptee' class
class ChemicalDatabank
{
  // The databank 'legacy API'
  public float GetCriticalPoint(string compound, string point)
  {
    // Melting Point
    if (point == "M")
    {
      switch (compound.ToLower())
      {
        case "water": return 0.0f;
        case "benzene": return 5.5f;
        case "ethanol": return -114.1f;
        default: return 0f;
      }
    }
    // Boiling Point
    else
    {
      switch (compound.ToLower())
      {
        case "water": return 100.0f;
        case "benzene": return 80.1f;
        case "ethanol": return 78.3f;
        default: return 0f;
      }
    }
  }

  public string GetMolecularStructure(string compound)
  {
    switch (compound.ToLower())
    {
      case "water": return "H20";
      case "benzene": return "C6H6";
      case "ethanol": return "C2H5OH";
      default: return "";
    }
  }
```

```
    public double GetMolecularWeight(string compound)
    {
      switch (compound.ToLower())
      {
        case "water": return 18.015;
        case "benzene": return 78.1134;
        case "ethanol": return 46.0688;
        default: return 0d;
      }
    }
  }
}

/*Результат:

Compound: Unknown ------

Compound: Water ------
 Formula: H20
 Weight : 18.015
 Melting Pt: 0
 Boiling Pt: 100

Compound: Benzene ------
 Formula: C6H6
 Weight : 78.1134
 Melting Pt: 5.5
 Boiling Pt: 80.1

Compound: Alcohol ------
 Formula: C2H6O2
 Weight : 46.0688
 Melting Pt: -114.1
 Boiling Pt: 78.3
*/
```

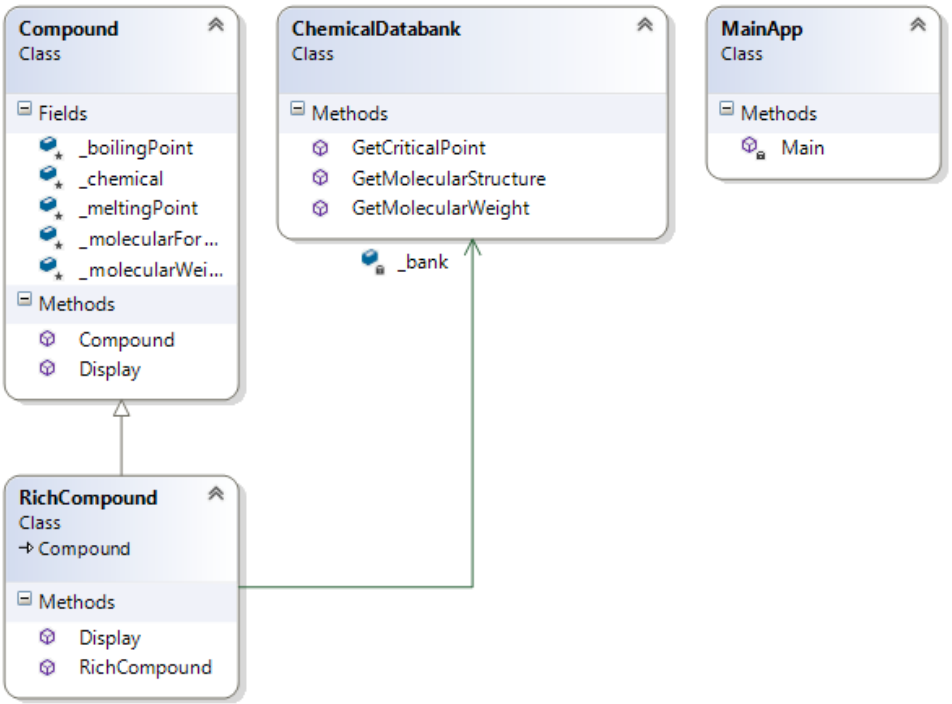

Рис. 12. Діаграма класів шаблону «Адаптер»

Легковаговик

Це структурний шаблон проектування, який структурує об'єкти таким чином, що з них створюється лише обмежений набір екземплярів замість великої множини об’єктів.

Полегшує повторне використання багатьох малих об’єктів, роблячи використання великої кількості об’єктів більш ефективною.

Структурний шаблон «Легковаговик» потрібно використовувати, коли:

– у додатку використовується велика кількість подібних об'єктів, при цьому накладні витрати на їх зберігання є високими (може не вистачити пам’яті для їх одночасного розміщення в режимі runtime);
– більшу частину станів об'єктів можна винести назовні (у зону відповідальності клієнтів, які створюють і використовують ці об'єкти);
– багато нероздільних (unshared) об'єктів можна замінити невеликою кількістю поділюваних (shared) об'єктів, оскільки їх стан винесено назовні;
– ідентичність кожного об’єкту не має значення.

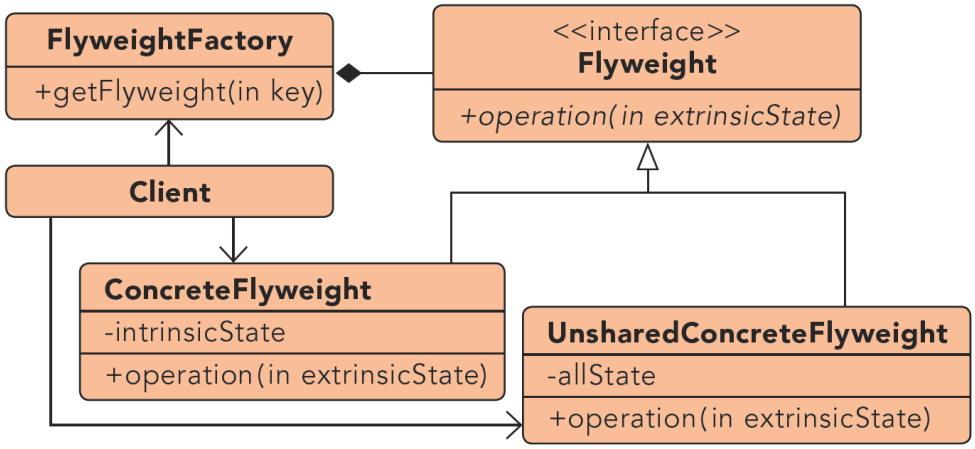

Рис. 13. Структурна схема шаблону «Легковаговик»

Учасники шаблону:

–**Flyweight** визначає інтерфейс, за допомогою якого пристосуванці можуть отримувати зовнішній стан або певним чином на нього впливати;

–**ConcreteFlyweight** реалізує інтерфейс **Flyweight** та додає при необхідності внутрішній стан. Об'єкт класу **ConcreteFlyweight** повинен бути розділюваним (shared). Будь-який збережуваний ним стан повинен бути внутрішнім, тобто таким, що не залежить від контексту;

–**UnsharedConcreteFlyweight** – нерозділюваний (unshared) конкретний пристосуванець. Не всі підкласи **Flyweight** обов'язково повинні бути shared. Інтерфейс **Flyweight** допускає розділення, але не нав'язує його. Часто у об'єктів **UnsharedConcreteFlyweight** на деякому рівні структури пристосуванця є нащадки у вигляді об'єктів класу **ConcreteFlyweight;**

–**FlyweightFactory** створює об'єкти-пристосуванці та управляє ними. Забезпечує необхідне розділення пристосуванців. Коли клієнт звертається до пристосуванця, об'єкт **FlyweightFactory** надає існуючий екземпляр або створює новий, якщо готового ще немає;

–**Client** зберігає посилання на одного або декількох пристосуванців. Обчислює або зберігає зовнішній стан пристосуванців.

Контрольний приклад

Завдання

За допомогою шаблона проектування забезпечити ефективну роботу з символами в тексті, де вони зустрічаються кожен багато разів.

Розв'язок

```
namespace FlyWeightExample
{
    class MainApp
    {
        static void Main()
        {
            // Build a document with text
            string document = "AAZZBBZB";
            char[] chars = document.ToCharArray();

            CharacterFactory factory = new CharacterFactory();

            // extrinsic state
            int pointSize = 10;
            // For each character use a flyweight object
            foreach (char c in chars)
            {
                pointSize++;
                Character character = factory.GetCharacter(c);
                character.Display(pointSize);
            }
            Console.ReadKey();
        }
    }

    // The 'FlyweightFactory' class
    class CharacterFactory
    {
        private Dictionary<char, Character> _characters = new Dictionary<char,
Character>();

        public Character GetCharacter(char key)
        {
            // Uses "lazy initialization"
            Character character = null;
            if (_characters.ContainsKey(key))
            {
                character = _characters[key];
            }
            else
            {
                switch (key)
                {
                    case 'A': character = new CharacterA(); break;
                    case 'B': character = new CharacterB(); break;
                    //...
                    case 'Z': character = new CharacterZ(); break;
                }
                _characters.Add(key, character);
            }
            return character;
        }
    }

    // The 'Flyweight' abstract class
    abstract class Character
    {
        protected char symbol;
        protected int width;
        protected int height;
        protected int ascent;
        protected int descent;
        protected int pointSize;

        public abstract void Display(int pointSize);
    }
```

```
    // A 'ConcreteFlyweight' class
    class CharacterA : Character
    {
        // Constructor
        public CharacterA()
        {
            this.symbol = 'A';
            this.height = 100;
            this.width = 120;
            this.ascent = 70;
            this.descent = 0;
        }

        public override void Display(int pointSize)
        {
            this.pointSize = pointSize;
            Console.WriteLine(this.symbol + " (pointsize " + this.pointSize + ")");
        }
    }

    /// A 'ConcreteFlyweight' class
    class CharacterB : Character
    {
       // Constructor
        public CharacterB()
        {
            this.symbol = 'B';
            this.height = 100;
            this.width = 140;
            this.ascent = 72;
            this.descent = 0;
        }

        public override void Display(int pointSize)
       {
            this.pointSize = pointSize;
            Console.WriteLine(this.symbol + " (pointsize " + this.pointSize + ")");
        }
    }

    // ... C, D, E, etc.

    // A 'ConcreteFlyweight' class
    class CharacterZ : Character
    {
        // Constructor
        public CharacterZ()
       {
            this.symbol = 'Z';
            this.height = 100;
            this.width = 100;
            this.ascent = 68;
            this.descent = 0;
        }

        public override void Display(int pointSize)
       {
            this.pointSize = pointSize;
            Console.WriteLine(this.symbol + " (pointsize " + this.pointSize + ")");
        }
    }
}

/*Результат:
A (pointsize 11)
A (pointsize 12)
Z (pointsize 13)
Z (pointsize 14)
B (pointsize 15)
B (pointsize 16)
Z (pointsize 17)
B (pointsize 18)
*/
```

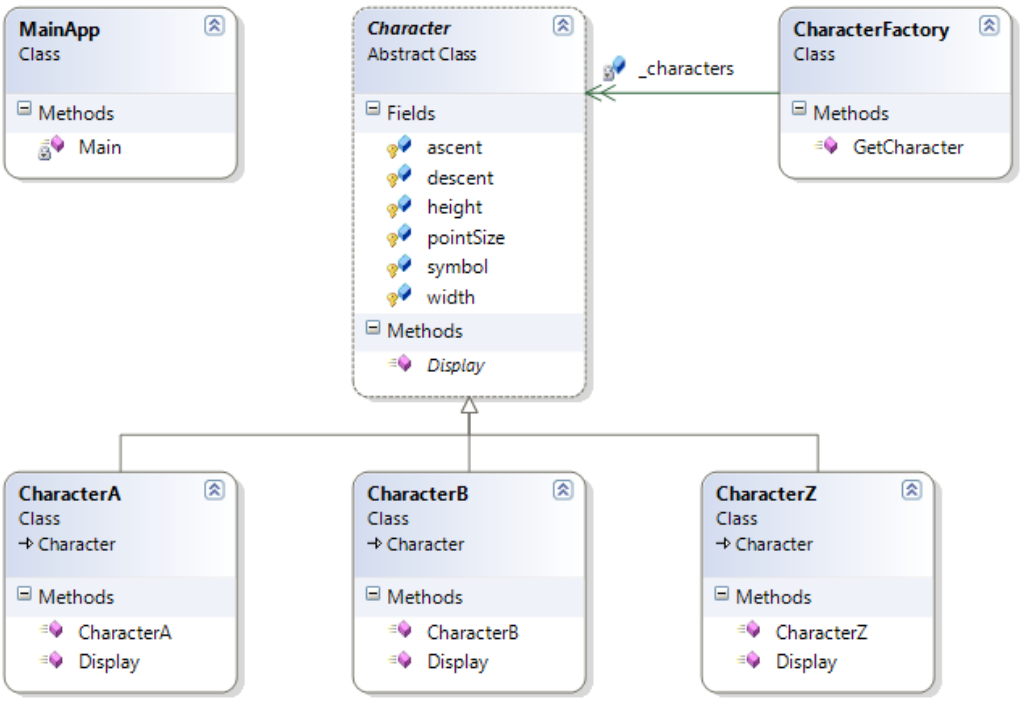

Рис. 14. Діаграма класів шаблону «Легковаговик»

Індивідуальні завдання

Варіант 1

1. Розробити механізм реєстрації користувача у деякій програмній системі, яким передбачено надавання можливості додавати фотографії до профілю користувача (створювати альбоми для фото і т.д.) тільки тим людям, у яких заповнені поля профілю щодо особистих даних (ПІБ, вік, адреса).

2. Розробити модуль програмного забезпечення роботи ЖЕКу, який буде формувати різноманітні види довідок. Наприклад, довідку про те, що людина дійсно прописана за вказаною адресою, довідку про кількість прописаних людей в певній квартирі тощо.

Варіант 2

1. В системі дистанційної освіти реалізувати механізм, який буде дозволяти користувачу он-лайн курсу виконувати завдання з поточного модуля (генерувати список завдань з наявного загального масиву завдань) тільки за умови наявності виконаних завдань (позитивних оцінок) з минулих модулів.

2. Стара система оцінювання знань студентів характеризувалась такими рисами: 100-бальна шкала, до 60 балів – незадовільно; 60-69 – трійка, 70-84 – четвірка; 85-100 – п’ятірка. Оцінка при цьому ставилась цифрою (2, 3, 4, 5). Нова система оцінювання має 60-бальну систему та вимагає проставлення оцінки прописом. Нова шкала балів змінена пропорційно старої шкали. За допомогою шаблона проектування забезпечити виведення оцінки студента на основі суми набраних ним балів як у старій, та і у новій системі оцінок.

Варіант 3

1. Розробити програмні засоби, за допомогою яких користувачу платного веб-порталу доступ до функцій щодо завантаження та обробки відео (наприклад, «Створити відеокаталог» тощо) буде надаватися тільки в тому випадку, якщо у нього є гроші на рахунку.

2. Розробити програмне забезпечення для створення та відправлення електронних вітальних листівок різним людям: наприклад, до дитячої листівки необхідно вставити картинку з мульфільма та зробити шрифт кольоровим; до листівки для підлітків потрібно вставити фото відомого актора чи зірки естради тощо. Тип листівки, яку потрібно генерувати, визначати за віком та статтю адресата..

Варіант 4

1. Розробити модуль до банківської програмної системи, який забезпечує друк деталізованої виписки щодо руху коштів на рахунку тільки у разі, якщо користувач підключений до спеціального пакету банківських послуг.

2. У старій системі реєстрації користувача необхідно було обов’язково вказувати ім’я, прізвище, місто проживання, вік, адресу електронної пошти, ключове слово, а також логін та пароль. Для прискорення процесу реєстрації необхідно вдосконалити систему: зробити таку надбудову над системою, щоб користувач повинен був вводити тільки ім’я, прізвище, логін, пароль та ключове слово. Всі інші незаповнені поля повинні отримати значення за замовчанням від системи.

Варіант 5

1. Розробити модуль до програмного забезпечення, що використовується у бібліотеці, який буде дозволяти читачу здійснювати пошук в електронному каталозі тільки за умови, якщо він не має незданих книжок на своєму абонементі.

2. Створити програмне забезпечення автоматичної генерації грамот для учасників конкурсу юних піаністів, а також дипломів лауреатів та дипломантів. Список учасників з вказаними школою та класом, а також з поміткою щодо зайнятого місця задається користувачем перед початком роботи програми.

Варіант 6

1. Розробити модуль до програмного забезпечення, яке використовується в міліції. Даний модуль повинен забезпечити

автоматичне створення електронної справи затриманого (окрім загальної анкети, де міститься загальна інформація про злочинця, дата та причини його затримання) у разі, якщо його затримано більше, ніж на мінімальний термін ув’язнення.

2. Система оплати за проїзд в метро. Пасажири можуть сплачувати за допомогою жетонів, проїзних квитків на місяць та карток з певною кількістю поїздок. Турнікет на вході дає можливість проходу пасажиру, якщо він отримує дозвільний сигнал від системи контролю проїзний документів. За допомогою шаблона проектування розробити програмний модуль, який буде перетворювати отриманий платіж на сигнал «1» («так») чи «0» («ні») для турнікету.

Варіант 7

1. Розробити модуль до програмного забезпечення, що використовується у приймальній комісії вузу, який буде автоматично створювати папку з бланками електронних документів абітурієнта (анкета, оцінки на іспитах тощо) тільки у разі, якщо останній набрав прохідний бал.

2. Кухонний комбайн має функції кавниці, сокодавильниці, м’ясорубки, вимішувальника тіста тощо. За допомогою шаблона проектування реалізувати функціональність кожного побутового пристрою окремо (у вигляді окремих класів), а потім організувати новий клас, який дозволить об’єднати функції цих пристроїв разом.

Варіант 8

1. Розробити модуль до програмного забезпечення роботи бухгалтерії, який буде автоматично формувати пакет електронних фінансових документів (акт виконаних робіт, рахунок-фактура тощо), що

підтверджують виконання обов’язків компанії перед замовником, тільки у разі наявності оплати останнім послуги чи товару.

2. Стара система автоматичного вітання користувачів сайту раніше всіх вітала однаково: «Привіт, » + ім’я людини. Тепер вирішено, що таким чином система повинна вітати тільки людей, молодше 35 років. Користувачі, старші 35 років, повинні побачити повідомлення «Доброго дня, » + ПІБ. За допомогою шаблона проектування забезпечити диференціацію виведення вітального рядка (передачу до відповідного метода цілого об’єкта Користувач, або значень його полів ПІБ та вік). У разі, якщо вік користувача невідомий, за замовчанням вважати його більшим, ніж 35 років.

Варіант 9

1. Розробити модуль чата, який буде дозволяти користувачу приєднуватися до чата тільки у разі, якщо не перевищена припустима кількість підключених учасників чата.

2. Всі квіти мають таку характеристику як відношення до вологи. Якщо квітка любить вологу, її треба поливати більше звичайної норми на 30%, а якщо не любить вологу – на 50% менше за звичайну норму. Для автоматичного поливання трьох видів квітів раніше існувало 3 окремих прилади. Потім їх об’єднали в один та надали новому пристрою можливість розпізнавати тип квітки. За допомогою шаблона проектування реалізувати програмний прототип нового приладу для поливання квітів.

Варіант 10

1. Розробити модуль програмного забезпечення, що використовується у бібліотеці, який буде дозволяти читачу користуватися

послугами електронного каталогу тільки у випадку, якщо у даного читача є власний логін у цьому програмному забезпеченні. У протилежному випадку модуль повинен пропонувати читачу зареєструватися у програмі.

2. Для проведення обчислень програмний модуль приймає 10-розрядні дані у вісімковій системі числення. На вхід метода Calculate подається 2 таких числа, метод повертає одне таке число як результат роботи. Інший модуль, який повинен працювати з описаним вище модулем, виконує обчислення у шістнадцятковій системі числення та обробляє 7-розрядні числа. За допомогою шаблону проектування забезпечити взаємодію даних програмних модулів.

Варіант 11

1. Розробити модуль програмної системи обліку абонентів оператора мобільного зв’язку, який при спробі абонента перейти на інший тариф буде запитувати в нього дані щодо PUK-коду телефону і відображати перелік доступних тарифів тільки у разі коректного введення цього коду.

2. За допомогою шаблона проектування забезпечити виведення на шахову дошку фігур в їх початковому розташуванні.

Варіант 12

1. Розробити модуль програмного забезпечення, що використовується у банку, який буде дозволяти клієнту користуватися послугами інтернет-банкінгу тільки у випадку, якщо у нього підключена дана послуга. У протилежному випадку модуль повинен пропонувати клієнту зареєструватися у системі інтернет-банкінгу.

2. У старій програмній системі «Деканат» для виведення відомостей про студентів на екран як аргумент методу Print раніше передавався весь список групи. У новій версії програми списки груп впорядковані та зберігаються у вигляді словника з полями «шифр групи», «список групи». Для виконання друку тепер необхідно передавати методу Print тільки шифр групи. За допомогою шаблона проектування забезпечити підтримку переходувід функціональності старої версії програми до нової.

Варіант 13

1. Розробити модуль програмної системи обліку абонентів оператора мобільного зв’язку, який буде дозволяти абоненту здійснювати дзвінок з телефону тільки у випадку, коли IMEI-код телефону присутній у базі «білих» телефонних апаратів.

2. Деякій програмній системі на вхід подаються команди англійською мовою (“run”, “stop”, “pause”). Після виконання кожної команди система повертає код результату її виконання: 0 – успішно виконано, 1 – аварійно завершено. Розробники українського інтерфейсу до даної програми не забезпечили повного збігу з вихідним інтерфейсом системи: інтерфейс передбачає команди українською мовою (“старт”, “стоп”, “пауза”) та може приймати такі коди результатів виконання команди: 1 – успішно завершено, 0 – виконано неуспішно. За допомогою шаблона проектування «Адаптер» забезпечити двосторонній перехід від команд одного інтерфейсу до іншого та у зворотному напрямку.

Варіант 14

1. Розробити модуль програмної системи «Деканат», який дозволятиме вносити оцінки з екзаменаційної відомості до бази даних тільки користувачам з правами Адміністратора. Всі інші користувачі системи повинні мати змогу лише продивлятися виставлені оцінки.

2. За допомогою шаблона проектування забезпечити розміщення на ігровому полі різнотипних кораблів гравців у грі «Морський бій».

Варіант 15

1. Розробити програмний механізм, який буде перевіряти текстові повідомлення на перевищення припустимої кількості символів у них перед відправленням до мережі.

2. В магазині існує таблиця відповідностей розмірів взуття (див. нижче). У програмній системі, яка працює в цьому магазині реалізовано дві функції: переведення довжини стопи в український розмір, а також переведення українського розміру в англійський розмір. За допомогою шаблона проектування реалізувати функцію, яка буде здійснювати переведення довжини стопи до англійського розміру взуття.

Довжина стопи (сантиметри)	23,6	24,3	25	25,7	26,4	27	27,7	28,4	29	29,7
Розмір (УкраЇна)	37	38	39	40	41	42	43	44	45	46
UK	4	5	6	6,5	7,5	8	9	9,5	10,5	11,5

Варіант 16

1. Розробити програмний механізм розсилання повідомлень, який буде пересилати повідомлення, яке спочатку було адресоване всім користувачам системи, тільки тим адресатам, які мають певний рівень прав на отримання повідомлень.

2. За допомогою шаблона проектування забезпечити голосування на виборах в електронному вигляді з метою уникнення створення мільйонів бюлетенів для голосування. Список кандидатів один для всіх виборців.

Варіант 17

1. У месенджері при розсиланні повідомлення по цілому контакт-листу виключити зі списку розсилки ті контакти, які занесені до «чорного списку» (або invisible-листа), у випадку, якщо включений відповідний режим роботи програми.

2. За допомогою шаблона проектування забезпечити заповнення головної форми вашої програми геометричними фігурами (квадратами, колами, прямокутниками тощо) довільного розміру та кольору. .

Варіант 18

1. При збереженні тексту документа організувати додавання до нього інформації щодо дати останньої зміни тексту та щодо автора.

2. За допомогою шаблона проектування забезпечити виведення на екран веселки, яка складається з дуг різного радіуса та кольору.

Варіант 19

1. При введенні тексту до певного текстового поля забезпечити автоматичну заміну всіх літер, які стоять після комбінації символів «. », на великі.

2. На старому сервері програмної системи вичерпався набір розрядів для зберігання паролів. Раніше пароль формувався за заданим пін-кодом з 4 символів за допомогою спеціального алгоритму, який на основі пін-коду генерував 8-розрядний унікальний ключ. У новій версії системи ключ розширено до 12 розрядів. За допомогою шаблона проектування розробити модуль підключення до сервера нової версії системи шляхом формування на основі старого 8-розрядного ключа його нового 12-розрядного замінника.

Варіант 20

1. Реалізувати програмний механізм перевірки реєстраційної форми, заповненої користувачем. Зберігати надані дані тільки у разі, якщо заповнено всі обов’язкові поля, відмічені «*».

2. Мобільний телефон має функції будильника, органайзера, фотоапарату, плеєра, радіо тощо. За допомогою шаблона проектування реалізувати функціональність кожного згаданого вище пристрою окремо (у вигляді окремих класів), а потім організувати новий клас, який дозволить об’єднати функції цих пристроїв разом.

Варіант 21

1. Створити програмний модуль, який при обчисленні ціни товару буде автоматично надавати скидку на нього за карткою постійного

покупця у тому випадку, якщо ціна на цей товар не є акційною. Список кодів акційних товарів міститься у системі.

2. За допомогою шаблону проектування розробити програмну реалізацію гри «Мозаїка», не створюючи весь масив її елементів. Для прикладу скласти з мозаїки прапор будь-якої країни. Розмір поля мозаїки встановити мінімум 500 на 1000 елементів.

Варіант 22

1. Запровадити додаткову перевірку віку користувача при реєстрації його у певній програмній системі. Якщо поле «Вік» було залишене незаповненим, програма має перепитати користувача, чи виповнилося йому 18 років. Якщо ні – відмовляти у реєстрації.

2. У першій версії комп'ютерної гри управління відбувалося через джойстик. У новій версії гри за допомогою шаблона проектування необхідно реалізувати управління грою за допомогою миші.

Варіант 23

1. Розробити програмний модуль сервера з медіа-контентом, який буде перевіряти адреси вузлів, що підключаються до сервера. Якщо вузол, який намагається встановити з'єднання, має заборонену адресу, треба відмовити йому у наданні каналу зв'язку – надіслати повідомлення про неможливість функціонування ресурсу.

2. За допомогою шаблона проектування забезпечити розміщення на ігровому полі різнотипних кораблів гравців у грі «Морський бій».

Варіант 24

1. У програмній системі «Деканат» забезпечити формування списку студентів за певними характеристиками у такий спосіб: при першому зверненні до списку сформувати його та запам’ятати у «кеші»; при наступних зверненнях до цього ж списку студентів, повертати вже збережений результат.

2. Створити програмне забезпечення автоматичної генерації дипломів бакалаврів: звичайного диплому та диплому з відзнакою. Список студентів з вказаними ПІБ, спеціальністю та поміткою щодо типу видаваного диплому задається користувачем перед початком роботи програми.

Варіант 25

1. Розробити модуль програмної системи «Деканат», який дозволятиме вносити оцінки з екзаменаційної відомості до бази даних тільки користувачам з правами Адміністратора. Всі інші користувачі системи повинні мати змогу лише продивлятися виставлені оцінки.

2. За допомогою шаблона проектування забезпечити розміщення на ігровому полі різнотипних кораблів гравців у грі «Морський бій».

Контрольні запитання

1. Дайте визначення шаблону «Заступник» («Адаптер», «Легковаговик», «Фасад»).
2. У чому полягає призначення шаблону «Заступник» («Адаптер», «Легковаговик», «Фасад»)?
3. Назвіть переваги та недоліки застосування шаблону «Заступник» («Адаптер», «Легковаговик», «Фасад»).

4. Наведіть UML-діаграму класів, які утворюють структуру шаблона «Заступник» («Адаптер», «Легковаговик», «Фасад»).

2.3. Лабораторна робота №3

Тема роботи: реалізація породжуючих шаблонів проектування.

Мета роботи: ознаойомлення з основними характеристиками всіх породжуючих шаблонів («Прототипу», «Фабричного методу», «Абстрактної фабрики», «Одинака», «Будівельника»), запам’ятовування поширених ситуацій, коли використання цих шаблонів є доцільним, набуття вмінь та навичок реалізації шаблонів під час створення програмного коду.

Теоретичні відомості

Прототип

Прототип (Prototype) – шаблон, що задає види створюваних об’єктів за допомогою екземпляра-прототипу і створює нові об’єкти шляхом копіювання цього прототипу.

Це єдиний шаблон з серії Породжуючих, котрий для створення нових об’єктів використовує не явне інстанціювання (ключове слово new), а клонування.

Породжуючий шаблон «Прототип» потрібно використовувати, коли:

- система не повинна залежати від способу створення та реалізації об’єктів, які до неї входять;
- класи, які породжуються, визначаються під час виконання програми, наприклад, за допомогою динамічного завантаження;

– бажано уникнути успадкування створювача об’єкта (тоді “Прототип” є конкуретном “Абстрактної фабрики”);
– екземпляри класу можуть знаходитися в одному зі станів, кількість яких є невеликою. Тому може бути зручніше встановити відповідну кількість прототипів та клонувати їх, а не створювати об’єкт кожного разу в потрібному стані;
– створення об’єкта заново (new + встановлення значень полів) є “дорогим”.

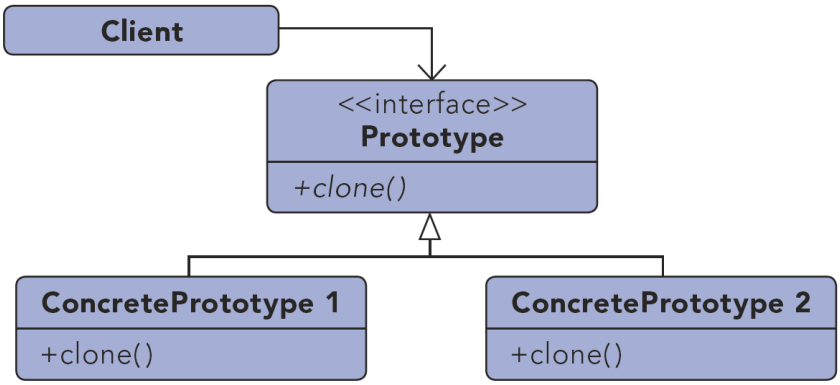

Рис. 15. Структурна схема шаблону «Прототип»

Учасники шаблону:

– **Prototype** — визначає інтерфейс для клонування самого себе;
– **ConcretePrototype** — реалізує операцію клонування самого себе;
– **Client** — створює новий об’єкт, звертаючись до прототипу з запитом клонувати себе, потім, користуючись його інтерфейсом, виконує з отриманим клоном потрібні маніпуляції.

Контрольний приклад

Завдання

ПЗ електронної комерції збирає інформацію про продукти, використовуючи складні запити до БД. Вміст БД оновлюється через

певний, відомий інтервал часу. Кількість продуктів є такою, що може бути закешованою.

Коли користувач запитує інформацію щодо обраного продукту, система може сформувати дані двома способами:

– виконати запит до БД, отримати інформацію та створити новий об’єкт;
– створити об’єкти та тримати їх у кеші. Коли користувач буде запитувати дані щодо об’єкта, вони будуть вилучені з кешу та клоновані. Коли БД оновлюється, вміст кешу буде замінюватися новими об’єктами.

Розв’язок

```
namespace PrototypeExample
{
    class Program
    {
        static void Main(string[] args)
        {
            ProductCache.loadCache();
            Book clonedBook = (Book)ProductCache.getProduct("B1");
            Console.WriteLine("SKU = " + clonedBook.getSKU());
            Console.WriteLine("SKU = " + clonedBook.getDescription());
            Console.WriteLine("SKU = " + clonedBook.getNumberOfPages());
            Console.WriteLine();
            DVD clonedDVD = (DVD)ProductCache.getProduct("D1");
            Console.WriteLine("SKU = " + clonedDVD.getSKU());
            Console.WriteLine("SKU = " + clonedDVD.getDescription());
            Console.WriteLine("SKU = " + clonedDVD.getDuration());
            Console.ReadKey();
        }
    }

    //IPrototype
    public abstract class Product:ICloneable
    {
        private String SKU; // stock-keeping unit
           private String description;

           public Object Clone()
        {
                    Object clone = null;
                    try
                    {
                            clone = this.MemberwiseClone();
                    }
                    catch (Exception e) {
                            Console.WriteLine(e.Message);
                    }
                    return clone;
           }

           public String getDescription() {
                    return description;
           }
```

```
        public String getSKU() {
                return SKU;
        }
        public void setDescription(String desc) {
                description = desc;
        }
        public void setSKU(String _sku) {
                SKU = _sku;
        }
    }

    //Concrete Prototype1
    public class Book: Product {
        private int numberOfPages;

        public int getNumberOfPages() {
                return numberOfPages;
        }
        public void setNumberOfPages(int i) {
                numberOfPages = i;
        }
    }

    //Concrete Prototype2
    public class DVD : Product {

        private int duration;

        public int getDuration()
       {
                return duration;
        }
        public void setDuration(int i)
       {
                duration = i;
        }
    }

    //PrototypeManager
    public class ProductCache {
        private static Hashtable productMap = new Hashtable();

        public static Product getProduct(String productCode) {
                Product cachedProduct = (Product) productMap[productCode];
                return (Product) cachedProduct.Clone();
        }

        public static void loadCache() {
                // for each product run expensive query and instantiate product
                // productMap.put(productKey, product);
                // for exemplification, we add only two products
                Book b1 = new Book();
                b1.setDescription("Oliver Twist");
                b1.setSKU("B1");
                b1.setNumberOfPages(100);
                productMap[b1.getSKU()] = b1;
                DVD d1 = new DVD();
                d1.setDescription("Superman");
                d1.setSKU("D1");
                d1.setDuration(180);
                productMap[d1.getSKU()] = d1;
        }
    }
}
/*Результат:
SKU = B1
SKU = Oliver Twist
SKU = 100

SKU = D1
SKU = Superman
SKU = 180
*/
```

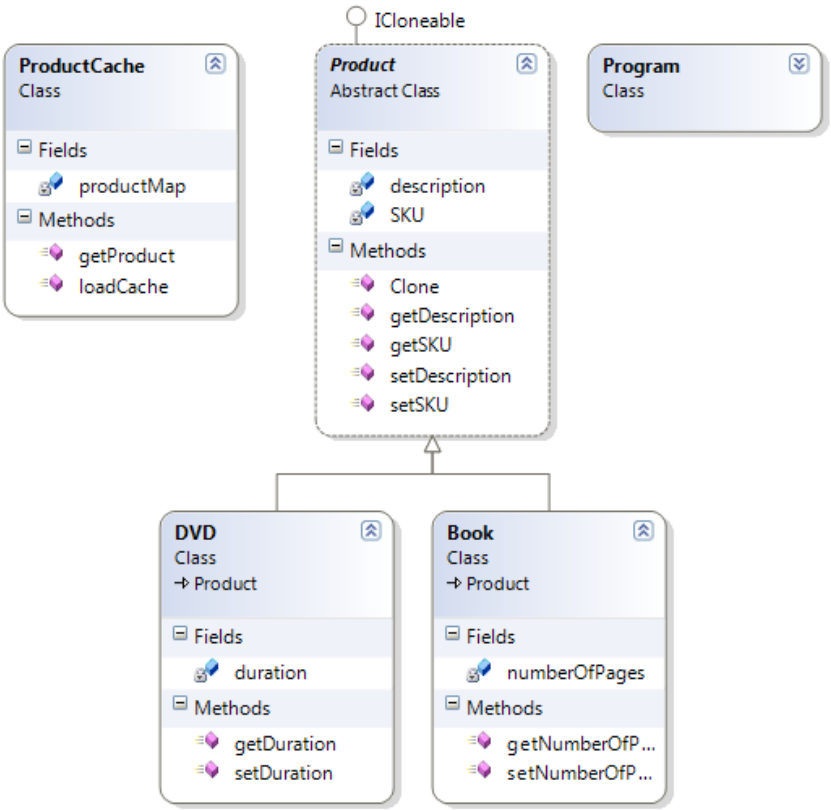

Рис. 16. Діаграма класів шаблону «Прототип»

Фабричний метод

Шаблон, який визначає інтерфейс для створення об'єкта, але рішення про те, який саме об'єкт створювати, залишає за підкласами. Таким чином, «*Фабричний метод*» дозволяє класу делегувати дію інстанціювання підкласам.

Це шаблон рівні класу, а не об'єкта!

Відомий ще під назвою "*Віртуальний конструктор (Virtual Constructor)*".

Породжуючий шаблон «*Фабричний метод*» потрібно використовувати, коли:

– класу наперед невідомо, об'єкти яких класів йому потрібно створювати, оскільки планується багато різних варіантів функціонування;

– клас спроектовано таким чином, щоб специфікація породжуваного об’єкту визначається тільки в нащадках;
– клас делегує свої обов’язки одному з декількох допоміжних підкласів і планується локалізувати знання про те, який клас приймає ці обов’язки на себе;
– це дозволяє використовувати у коді програми не специфічні класи, а маніпулювати абстрактними об’єктами на більш високому рівні.

Шаблон “Фабричний метод” часто застосовується в інструментальних бібліотеках і каркасах.

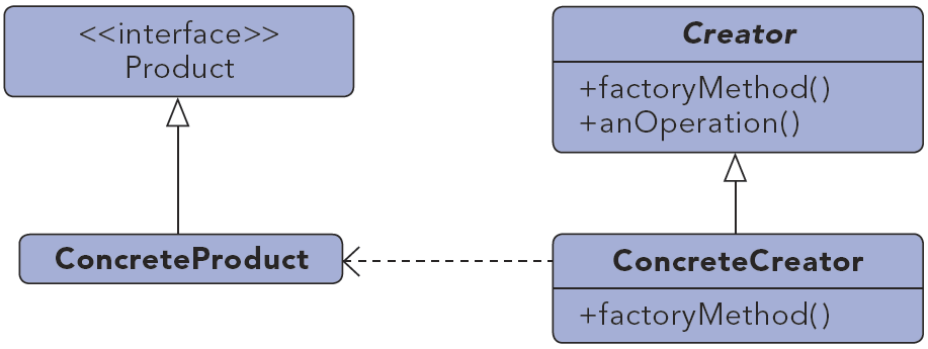

Рис. 17. Структурна схема шаблону «Фабричний метод»

Учасники шаблону:
– **Product** — продукт, абстрактний клас; визначає інтерфейс об’єктів, створюваних фабричним методом;
– **ConcreteProduct** — конкретний продукт, клас; реалізує інтерфейс Product;
– **Creator** — творець; оголошує фабричний метод, що повертає об’єкт типу Product; може викликати реалізацію за замовчанням Фабричного методу, який повертає об’єкт ConcreteProduct; може викликати Фабричний метод для створення об’єкту Product;

– **ConcreteCreator** —конкретний творець; заміщує фабричний метод, що повертає об’єкт ConcreteProduct.

Контрольний приклад

Завдання

У різні пори року постачальник продуктів до ресторану замовляє їх у різних регіонах. Влітку та восени він отримує продукти з України; взимку та навесні – з Іспанії, Африки тощо. Реалізувати механізм постачання продуктів та продемонструвати, що в залежності від пори року місце поставки змінюється.

Розв’язок

```
namespace FactoryPatternExample
{
    interface IProduct
    {
        string ShipFrom();
    }

    class ProductA : IProduct
    {
        public String ShipFrom()
        {
            return " from Ukraine";
        }
    }

    class ProductB : IProduct
    {
        public String ShipFrom()
        {
            return "from Spain";
        }
    }

    class DefaultProduct : IProduct
    {
        public String ShipFrom()
        {
            return "not available";
        }
    }

    class Creator
    {
        public IProduct FactoryMethod(int month)
        {
            if (month >= 4 && month <= 11)
                return new ProductA();
            else
                if (month == 1 || month == 2 || month == 12)
                    return new ProductB();
                else return new DefaultProduct();
        }
    }
```

```
    class Program
    {
        static void Main()
        {
            Creator c = new Creator();
            IProduct product;

            for (int i = 1; i <= 12; i++)
            {
                product = c.FactoryMethod(i);
                Console.WriteLine("Avocados " + product.ShipFrom());
            }
            Console.ReadKey();
        }
    }
}

/* Output
 Avocados from Spain
 Avocados from Spain
 Avocados not available
 Avocados from Ukraine
 Avocados from Ukraine
 Avocados from Ukraine
 Avocados from Ukraine
 Avocados from Ukraine
 Avocados from Ukraine
 Avocados from Ukraine
 Avocados from Ukraine
 Avocados from Spain
 */
```

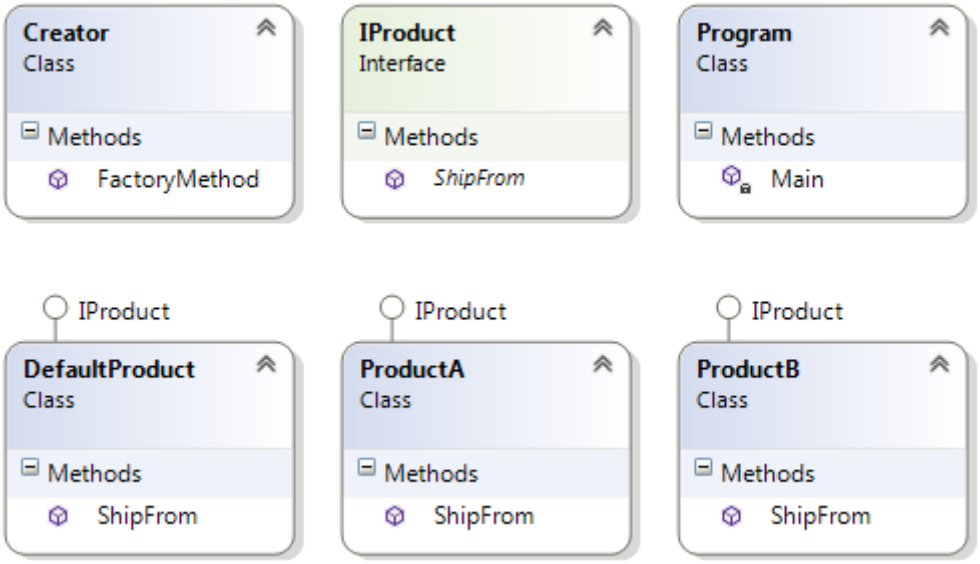

Рис. 18. Діаграма класів шаблону «Фабричний метод»

Абстрактна фабрика

Породжуючий шаблон, який дозволяє змінювати поведінку системи, варіюючи створювані об’єкти, але при цьому залишаючи незмінніми інтерфейси.

Надає інтерфейс для створення сімейств взаємнопов’язаних або незалежних об’єктів, не визначаючи при цьому їх конкретних класів.

Породжуючий шаблон «*Абстрактна фабрика*» потрібно використовувати, коли:

– система не повинна залежати від того, яким чином створюються, компонуються та подаються вхідні об’єкти;
– конкретний варіант бажаної поведінки системи визначають не окремі об’єкти, а ціле сімейство пов’язаних об’єктів. Об’єкти одного сімейства повинні використовуватися разом;
– на вхід системи подається тільки ціле сімейство об’єктів, система конфігурується одним із сімейств;
– виконуються певні умови конфіденційності, коли небажано надавати весь API бібліотеки об’єктів, їх реалізацію. Замість цього надаються тільки їх інтерфейси.

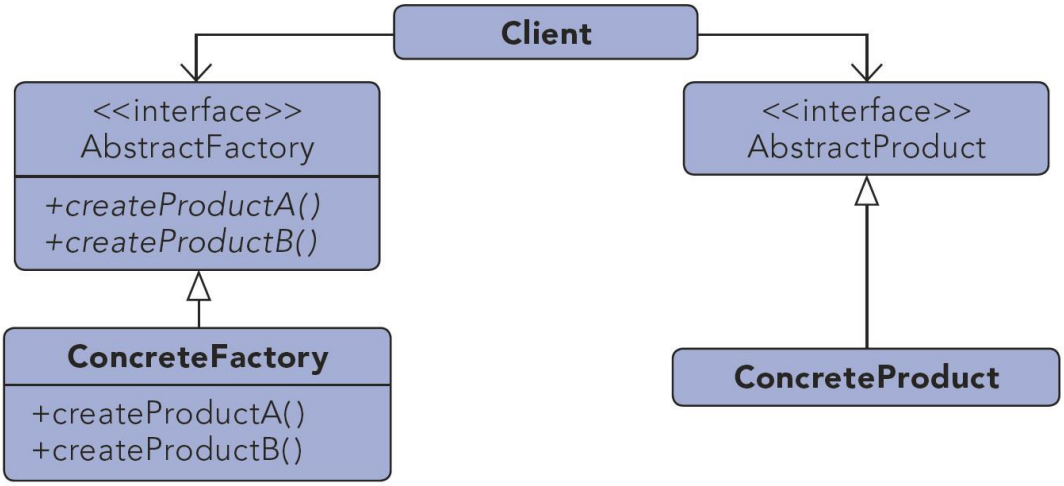

Рис. 19. Структурна схема шаблону «Абстрактна фабрика»

Учасники шаблону:

– **AbstractFactory** – визначає загальний інтерфейс для операцій створення абстрактних об’єктів – продуктів. Надалі в програмі замість

абстрактних будуть створюватися вже конкретні об’єкти певного сімейства;

- **ConcreteFactory[N]** – визначає, реалізує операції, які створюють всі об’єкти одного конкретного сімейства;
- **AbstractProduct[A-Z]** – визначає загальний інтерфейс;
- для типу об’єкта-продукта. A-Z в даному випадку – множина таких продуктів, які складають абстрактне сімейство;
- **ConcreteProduct[A-Z][N]** - визначає конкретний об’єкт-продукт типу [A-Z], створюваний відповідною конкретною фабрикою;
- **Client** – програмний модуль, який віддає команди на отримання конкретного сімейства продуктів, користуючись виключно відомими інтерфейсами класів AbstractFactory та AbstractProduct.

Контрольний приклад

Завдання

Тваринний світ на всіх континентах ділиться на Хижаків та Травоїдних, але разом з тим на кожному континенті мешкають свої представники кожного з цих двох класів тварин. Створити віртуальну модель заселення континентів тваринами.

Розв’язок

```
namespace AbstractFactoryExample
{
  class MainApp
  {
    public static void Main()
    {
      // Create and run the African animal world
      ContinentFactory africa = new AfricaFactory();
      AnimalWorld world = new AnimalWorld(africa);
      world.RunFoodChain();
      // Create and run the American animal world
      ContinentFactory america = new AmericaFactory();
      world = new AnimalWorld(america);
      world.RunFoodChain();
      // Wait for user input
      Console.ReadKey();
    }
  }
```

```
// The 'AbstractFactory' abstract class
abstract class ContinentFactory
{
  public abstract Herbivore CreateHerbivore();
  public abstract Carnivore CreateCarnivore();
}

// The 'ConcreteFactory1' class
class AfricaFactory : ContinentFactory
{
  public override Herbivore CreateHerbivore()
  {
    return new Wildebeest();
  }
  public override Carnivore CreateCarnivore()
  {
    return new Lion();
  }
}

// The 'ConcreteFactory2' class
class AmericaFactory : ContinentFactory
{
  public override Herbivore CreateHerbivore()
  {
    return new Bison();
  }
  public override Carnivore CreateCarnivore()
  {
    return new Wolf();
  }
}

// The 'AbstractProductA' abstract class
abstract class Herbivore
{
}

// The 'AbstractProductB' abstract class
abstract class Carnivore
{
  public abstract void Eat(Herbivore h);
}

// The 'ProductA1' class
class Wildebeest : Herbivore {}
  // The 'ProductB1' class
class Lion : Carnivore
{
  public override void Eat(Herbivore h)
  {
    // Eat Wildebeest
    Console.WriteLine(this.GetType().Name +
      " eats " + h.GetType().Name);
  }
}

// The 'ProductA2' class
class Bison : Herbivore
{
}

// The 'ProductB2' class
class Wolf : Carnivore
{
  public override void Eat(Herbivore h)
  {
    // Eat Bison
    Console.WriteLine(this.GetType().Name +
      " eats " + h.GetType().Name);
  }
}
```

```
  // The 'Client' class
  class AnimalWorld
  {
    private Herbivore _herbivore;
    private Carnivore _carnivore;

    // Constructor
    public AnimalWorld(ContinentFactory factory)
    {
      _carnivore = factory.CreateCarnivore();
      _herbivore = factory.CreateHerbivore();
    }

    public void RunFoodChain()
    {
      _carnivore.Eat(_herbivore);
    }
  }
 }

/*Результат:
Lion eats Wildebeest
 Wolf eats Bison
*/
```

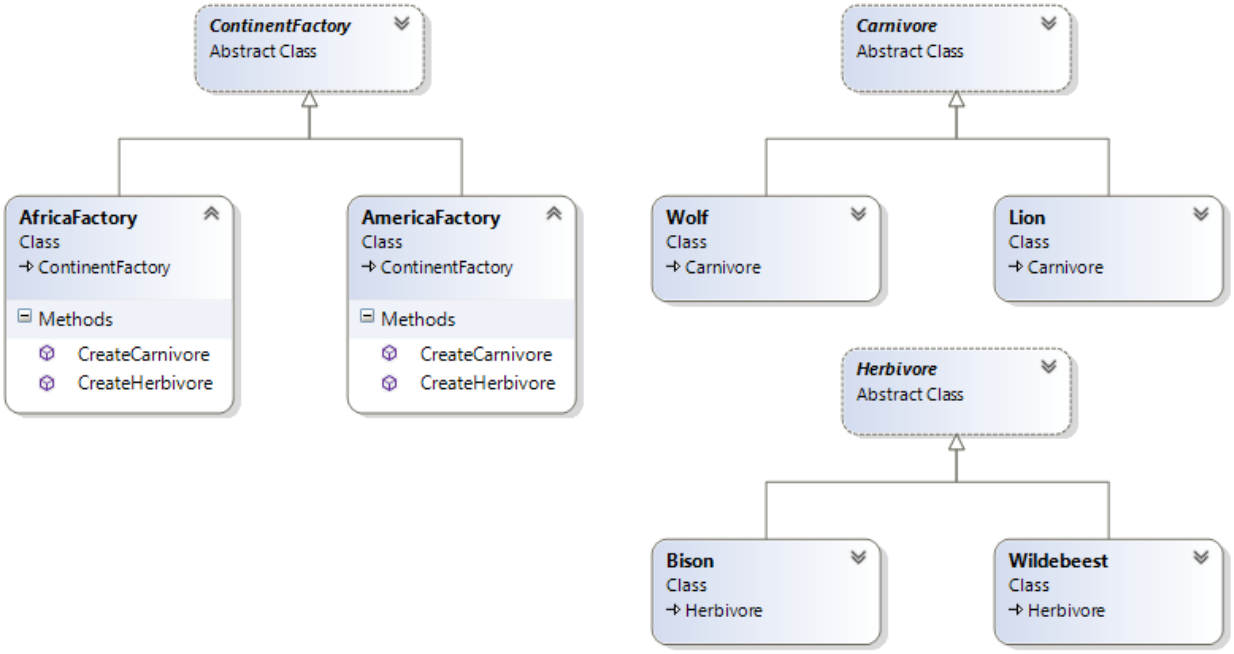

Рис. 20. Діаграма класів шаблону «Абстрактна фабрика»

Одинак

Породжуючий шаблон, який гарантує, що існує тільки один екземпляр певного класу і надає до нього глобальну точку доступу (зазвичай статичний метод).

Найбільш поширений спосіб використання шаблона “**Одинак**” - використання його як місця для зберігання та підтримки доступу до глобальних змінних.

Породжуючий шаблон «*Одинак*» потрібно використовувати, коли:

– повинен бути рівно один екземпляр певного класу, легко доступний для всіх клієнтів;
– єдиний екземпляр повинен розширюватися шляхом породження підкласів, і клієнтам потрібно мати можливість працювати з розширеним екземпляром без модифікації свого коду.

Якщо створення та надання глобального доступу до єдиного об’єкту класу, час і способи створення даного об’єкту не є проблемними питаннями, розглядання шаблона “Одинак” не є актуальним.

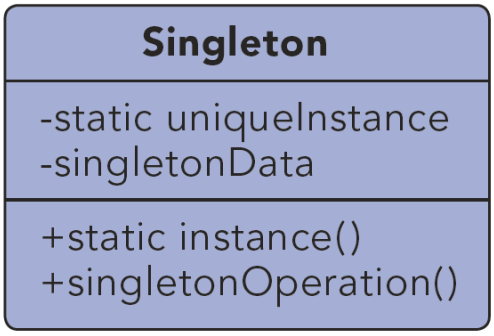

Рис. 21. Структурна схема шаблону «Одинак»

Учасники шаблону:

– **Singleton** — одинак: визначає операцію(статичний метод) Instance, яка дозволяє клієнтам отримувати доступ до єдиного екземпляру; може нести відповідальність за створення та маніпулювання власним унікальним екземпляром;

– клієнти отримують доступ до екземпляру класу Singleton тільки через його операцію Instance (публічний статичний метод класу);
– всі конструктори класу мають бути protected або private.

Контрольний приклад

Завдання

Оскільки задач на окрему реалізацію даного шаблону немає, нижче наведемо просто приклад коду «*Одинака*» з відкладеною ініціалізацією.

Розв’язок

```
namespace SingletonExample
//Lazy_instanse
{

  class MainApp
  {
    static void Main()
    {
      // Constructor is protected -- cannot use new
      Singleton s1 = Singleton.Instance();
      Singleton s2 = Singleton.Instance();

      // Test for same instance
      if (s1 == s2)
      {
        Console.WriteLine("Objects are the same instance");
      }
      Console.ReadKey();
    }
  }

  // The 'Singleton' class
  class Singleton
  {
    private static Singleton _instance;

    // Constructor is 'protected'
    protected Singleton()
    {
    }

    public static Singleton Instance()
    {
      // Uses lazy initialization.
      // Note: this is not thread safe.
      if (_instance == null)
      {
        _instance = new Singleton();
      }

      return _instance;
    }
  }
}

/*Результат:
Objects are the same instance
*/
```

Будівельник

Породжуючий шаблон, який відокремлює конструювання складного об’єкта від його подання (моделі), тому в результаті одного і того ж процесу конструювання можуть утворю-ватися різні подання.

Породжуючий шаблон «*Будівельник*» потрібно використовувати, коли:

– клієнт повинен створювати складені об’єкти. При цьому процес створення об’єкта можна розділити на етапи;
– алгоритм створення складного об’єкта не повинен залежати від того, з яких частин складається об’єкт і яким чином вони між собою з’єднані;
– процес конструювання повинен забезпечувати створення різних видів подання об’єкта, що конструюється.

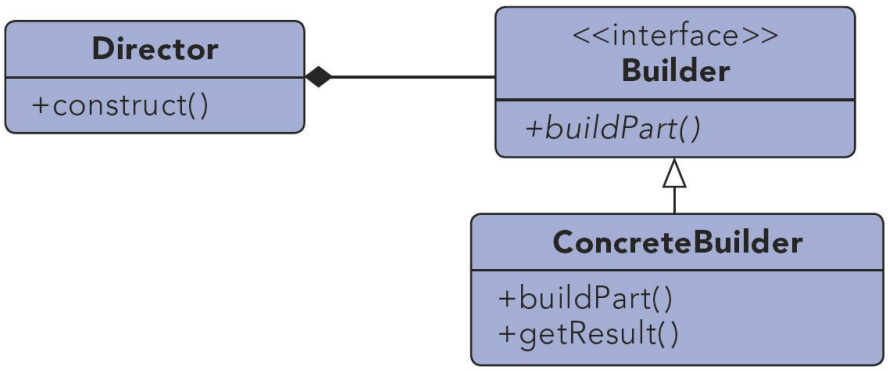

Рис. 22. Структурна схема шаблону «Будівельник»

Учасники шаблону:

– **Builder** – містить перелік **всіх можливих абстрактних** методів для створення та поєднання частин різноманітних об’єктів (продуктів);
– **ConcreteBuilder** – конкретний будівельник, який займається створенням **якогось одного** об’єкту (продукту): онструює та зв’язує разом частини продукту за допомогою реалізації тільки тих абстрактних методів інтерфейсу **Builder**, які потрібні для створення частин цього

продукту; визначає внутрішнє подання цього складного об’єкту та надає інтерфейс для отримання останнього;

- **Director** – розпорядник: фіксований, єдиний алгоритм створення будь-яких продуктів, для яких є у наявності відповідний **Builder**; конструює продукт, користуючись загальним інтерфейсом **Builder-a**;
- **Product** – кінцевий продукт, побудований за допомогою конкретного **Builder-a**: являє собою складний конструйований об’єкт;
- **ConcreteBuilder** будує внутрішнє подання продукта та визначає процес його збирання. Крім цього може включати класи, які визначають складові частини (у т.ч. інтерфейси) для збірки кінцевого результату з частин.

Контрольний приклад

Завдання

Побудувати документи різних форматів (PDF, Excel) з однаковою структурою (header, body, footer).

Розв’язок

```
namespace BuilderExample
{
    // Client invoking class
    public class Client
    {
        public static void Main()
        {
            // Create a PDF report
            ReportBuilder pdfBuilder = new PDFBuilder();
            Director dir = new Director();
            Report pdfReport = dir.GenerateReport(pdfBuilder);
            // Print content
            Console.WriteLine(pdfReport.Header);
            Console.WriteLine(pdfReport.Content);
            Console.WriteLine(pdfReport.Footer);
            // Create a Excel report
            ReportBuilder excelBuilder = new ExcelBuilder();
            Report excelReport = dir.GenerateReport(excelBuilder);
            // Print content
            Console.WriteLine(excelReport.Header);
            Console.WriteLine(excelReport.Content);
            Console.WriteLine(excelReport.Footer);
            Console.ReadLine();
        }
    }
```

```
// Report or Product
public class Report
{
    public string ReportType;
    public string Header;
    public string Footer;
    public string Content;
}

// Report Builder - Builder is responsible for defining
// the construction process for individual parts. Builder
// has those individual processes to initialize and
// configure the report.
public abstract class ReportBuilder
{
    public Report report;
    public void CreateReport()
    {
        report = new Report();
    }
    public abstract void SetReportType();
    public abstract void SetHeader();
    public abstract void SetFooter();
    public abstract void SetContent();
    public Report DispatchReport()
    {
        return report;
    }
}

// PDF Report class
public class PDFBuilder : ReportBuilder
{
    public override void SetReportType()
    {
        report.ReportType = "PDF";
    }
    public override void SetHeader()
    {
        report.Header = "PDF Header";
    }
    public override void SetFooter()
    {
        report.Footer = "PDF Footer";
    }
    public override void SetContent()
    {
        report.Content = "PDF Content";
    }
}

// Excel Report class
public class ExcelBuilder : ReportBuilder
{
    public override void SetReportType()
    {
        report.ReportType = "Excel";
    }
    public override void SetHeader()
    {
        report.Header = "Excel Header";
    }
    public override void SetFooter()
    {
        report.Footer = "Excel Footer";
    }
    public override void SetContent()
    {
        report.Content = "Excel Content";
    }
}
```

```
    /// Director takes those individual processes from the builder
    /// and defines the sequence to build the report.
    public class Director
    {
        public Report GenerateReport(ReportBuilder reportBuilder)
        {
            reportBuilder.CreateReport();
            reportBuilder.SetReportType();
            reportBuilder.SetHeader();
            reportBuilder.SetContent();
            reportBuilder.SetFooter();
            return reportBuilder.DispatchReport();
        }
    }
}

/*Результат:
PDF Header
PDF Content
PDF Footer
Excel Header
Excel Content
Excel Footer
*/
```

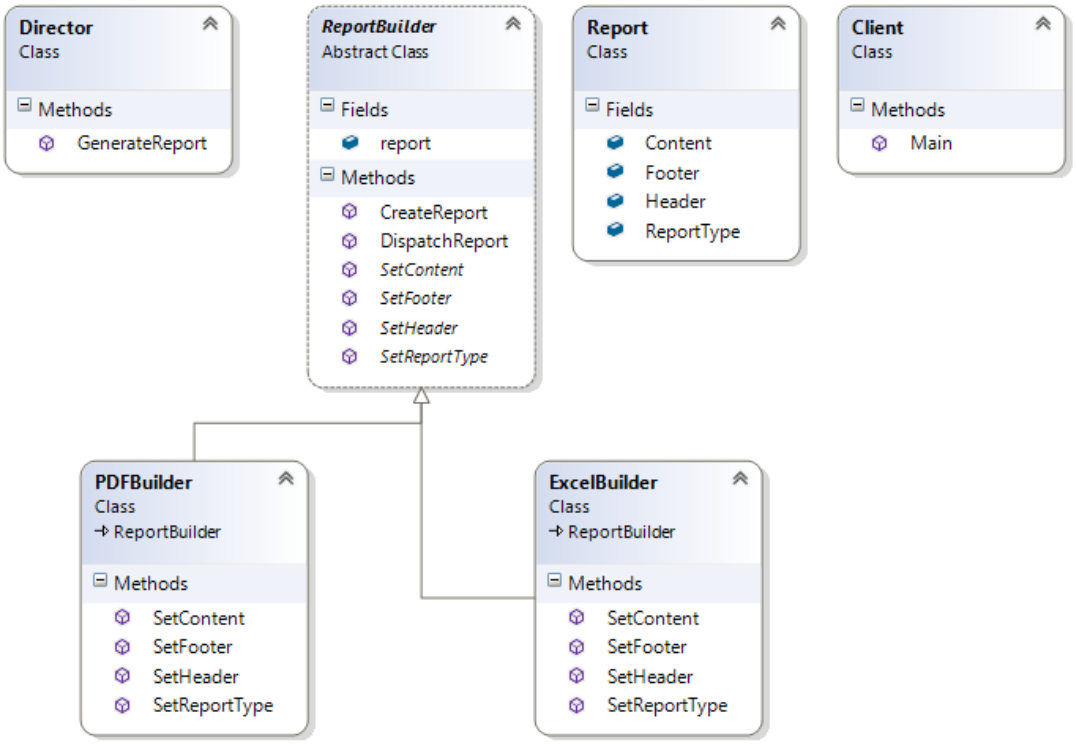

Рис. 23. Діаграма класів шаблону «Будівельник»

Індивідуальні завдання

Увага! При розв’язанні хоча б однієї задачі з варіанту необхідно також використати шаблон «*Одинак*». Але не замість основного шаблону, якому присвячена тематика задачі.

Варіант 1

1) За допомогою шаблона проектування імітувати процес випуску автомобілів відповідно до заданих прототипів. Кожне авто має модель, колір, тип кузова. Номер машині присвоюється тоді, коли екземпляр є готовим. Застосувати для цього завдання «легке копіювання».

2) За допомогою шаблона «Абстрактна фабрика» реалізувати віртуальний автомат, який виробляє (або просто розливає) напої, наприклад: чай, каву, сік. Напої характеризуються не тільки типом, а й ціновою категорією. Наприклад, чай може бути заварним або в пакетиках; кава розчиненою або меленою; сік 100% або нектар. В залежності від обраної цінової політики налаштувати автомат на випуск напоїв певної якості.

Варіант 2

1) За допомогою шаблона проектування реалізувати процес випуску книжок за заготовленими раніше матрицями. Кожна книга характеризується іменем автора, видавництвом, де вона була надрукована, серією та номером видання. Масив сторінок є тим об’єктом, який необхідно розтиражувати для кожної книги.

2) За допомогою шаблона «Абстрактна фабрика» реалізувати віртуальні меблеві фабрики. Кожна фабрика виготовляє вітальні, м’які меблі, кухні. Характеризуються фабрики матеріалами, які використовуються для виготовлення меблів, та вартістю продукції. В залежності від побажань та фінансових можливостей замовника обирати ту фабрику, яка буде створювати для нього меблі в даний момент часу.

Варіант 3

1) За допомогою шаблона проектування розробити механізм клонування групи студентів, яка складається з окремих об’єктів класу «Студент». Дане клонування необхідне для обробки масиву даних про студентів різними службами університету: поліклінікою, деканатом, студентською радою і т.д.

2) За допомогою шаблона «Абстрактна фабрика» реалізувати друк перепусток різних видів (постійних, тимчасових, одноразових тощо), а також ордерів на внесення-винесення техніки на певну територію в різних поліграфічних фірмах. Фірми відрізняються між собою якістю паперу, на якому здійснюється друк (звичайний або щільний), ступенем захисту (наявністю водяних знаків), вартістю партії виробів. В залежності від побажань клієнта перенаправляти замовлення на ту чи іншу фірму.

Варіант 4

1) У наявності маємо список файлів (посилання на файли, які зберігаються на комп’ютері). За допомогою шаблона проектування забезпечити копіювання даного списку файлів без перенесення самих файлів. Також реалізувати копіювання списку файлів з перенесенням самих файлів на нове місце. Вважати, що всі файли зі списку знаходяться в одній теці, а не розкидані по диску.

2) За допомогою шаблона «Абстрактна фабрика» забезпечити друк квитків на концерти різними фірмами. Білети можуть бути просто вхідними (без місця), звичайними (з місцем) та вкладеними у спеціальне запрошення. Білети на театральні вистави друкують на державних підприємствах, а білети на гастролі та приватних театрів – в комерційних фірмах. Поліграфічні фірми відрізняються кількістю кольорів, які

використовуються при друці білетів, та відповідно ціною на послуги. В залежності від типу білетів перенаправляти їх для друку на ту чи іншу фірму.

Варіант 5

1) Всі користувачі одного комп'ютера працюють з одним і тим самим з'єднанням з Інтернетом. Це з'єднання зберігається одна з характеристик кожного користувача. За допомогою шаблона проектування забезпечити створення нових користувачів комп'ютера шляхом клонування базового екземпляра профілю користувача, при чому Інтернет-з'єднання повинне залишатися єдиним об'єктом, без копій.

2) За допомогою шаблона «Абстрактна фабрика» організувати випуск пакетів соку з різних фруктів та ягід декількома заводами. Заводи відрізняються сортами сировини, упаковкою та ціною продукції. В залежності від пори року клієнти замовляють сік у різних заводів.

Варіант 6

1) За допомогою шаблона проектування допоможіть Діду Морозу сформувати подарунковий набір з іграшок, цукерок тощо та його розтиражувати. Всі складові набору повинні бути реалізовані як самостійні об'єкти та скопійовані до кожного наступного повністю.

2) За допомогою шаблона «Абстрактна фабрика» реалізувати процес віртуального збирання комп'ютерів та телевізорів у різних країнах. Кожна збірка відрізняється деталями, які вбудовуються до прибору, комплектацією (є базова, є vip-моделі тощо) та відповідно вартістю. В залежності від ціни послуги забезпечити запуск виготовлення продукції у тій чи іншій країні.

Варіант 7

1) Гра у шахи. За допомогою шаблона проектування забезпечити заповнення шахової дошки. Для цього використати відповідні прототипи шахових фігур. У випадку консольної реалізації програми інформацію про фігури виводити на екран у форматі: «позиція фігури» (наприклад, E2), «колір фігури» та «назва фігури».

2) За допомогою шаблона «Абстрактна фабрика» організувати виробництво пластикових читацьких квитків до бібліотеки. Читацькі квитки можуть бути студентські, шкільні, наукові, особливі (для академіків тощо). Різні апарати можуть виготовляти пластикові картки з фото або без фото, з голограмою або без. В залежності від складності виробництва змінюється і вартість продукції. Забезпечити у програмі вибір апарата з виготовлення квитків у залежності від наявної суми грошей.

Варіант 8

1) Гра у карти. За допомогою шаблона проектування створити колоду карт. Для цього використати відповідні прототипи грального карт кожної масті. У створених карт змінювати тільки їх ранг. На екран вивести список отриманих таким чином карт.

2) Організувати виробництво грамот для лауреатів міжшкільного конкурсу, дипломів для дипломантів та грамот для всіх інших учасників на основі використання шаблону «Абстрактна фабрика». Список учасників конкурсу містить такі дані: школа та клас учня, а також місце яке він посів (або примітка про присудження диплому конкурсу). Всі створені грамоти повинні бути незалежними об’єктами. Поліграфічні фірми, які займаються виготовленням продукції такого типу, відрізняються кількістю кольорів, що використовуються при друці, якістю паперу та відповідно ціною на

послуги. Забезпечити у програмі вибір фірми з виготовлення грамот у залежності від наявної суми грошей.

Варіант 9

1) Учні класу пішли до бібліотеки за підручниками. Кожний підручник має свій унікальний бібліотечний шифр. Сформувати прототип набору підручників; за допомогою «глибокого копіювання» створити набори підручників для учнів. Забезпечити виведення на екран списку учнів, а також списку підручників, які дісталися кожному учню (указати шифр та назву кожної книги, яка видана учню).

2) У різні пори року ресторан замовляє продукти у різних постачальників. Кожний постачальник працює з виробниками з різних регіонів. Одні постачальники орієнтовані на співпрацю з вітчизняними виробниками продуктів, інші – із зарубіжними виробниками. Виробники продуктів відрізняються сортами та ґатунком продукції. За допомогою шаблона «Абстрактна фабрика» організувати вибір постачальника в залежності від пори року.

Варіант 10

1) За допомогою шаблона проектування реалізувати віртуальний автомат, який виробляє напої. В залежності від того, які параметри задані покупцем (обрано чай/каву/какао, вказано «додатковий цукор», «лимон» і т.ін.) згенерувати напій. Забезпечити виробництво як мінімум трьох різних напоїв.

2) За допомогою шаблона «Абстрактна фабрика» організувати виробництво взуття декількома виробниками. Види взуття – жіночі туфлі, чоловічі туфлі та кросівки. Перший виробник працює тільки з

натуральною шкірою, другий – із шкірозамінником, третій використовую шкіру та хутро для оздоблення виробів. Забезпечити поставку до магазину продукції всіх трьох виробників.

Варіант 11

1) За допомогою шаблона проектування реалізувати друк перепусток різних видів (постійних, тимчасових, одноразових тощо), а також ордерів на внесення-винесення техніки на території компанії. В залежності від замовника (одні фірми завжди замовляють перепустки, а інші – бланки ордерів) здійснювати друк тих чи інших документів.

2) Дід Мороз та Снігуронька формують новорічні подарунки з іграшок, цукерок тощо. Дід Мороз робить подарунки для хлопчиків, куди як іграшку кладе пістолет або машинку. Снігуронька робить подарунки для дівчаток, куди як іграшку кладе ляльку ведмедика. Цукерки для хлопчиків загорнуті у блакитну обгортку, а для дівчаток – у рожеву. За допомогою шаблону «Абстрактна фабрика» забезпечити новорічне свято подарунками для всіх дітей.

Варіант 12

1) За допомогою шаблона проектування забезпечити створення квитків на концерти. Друк здійснювати з урахуванням уведених даних про концерт та місця (ряд, місце у залі).

2) За допомогою шаблона «Абстрактна фабрика» реалізувати заповнення екрана геометричними фігурами. Для цього створити декілька конструкторів фігур, які відповідають за один певний вид фігур. Наприклад, тільки за квадрати чи кола. Колір та розмір фігур може змінюватися.

Варіант 13

1) За допомогою шаблона проектування реалізувати програму, яка в залежності від введеного цінового діапазону формує віртуальний комп’ютер зі складових, що підходять до заданої ціни.

2) За допомогою шаблона «Абстрактна фабрика» організувати процес формування порцій у столовій. Кожна порція повинна містити м’ясне блюдо та гарнір. Крім того, порція може бути двох типів: дієтична або звичайна. Для формування кожного типу порцій використовується окремий конвеєр.

Варіант 14

1) За допомогою шаблона проектування реалізувати віртуальний автомат з виробництва пластикових читацьких квитків до бібліотеки. Читацькі квитки можуть бути студентські, шкільні, наукові, особливі (для академіків тощо). В залежності від того, який тип квитка вибраний на електронному таблі автомата, згенерувати читацький квиток відповідного типу.

2) У кіоску є 2 автомати з коктейлями. Один містить декілька алкогольних напоїв, інший – їх безалкогольні аналоги. В залежності від віку покупця наповнювати стаканчики тими чи іншими видами коктейлів (програму виконати на основі шаблона проектування «Абстрактна фабрика»).

Варіант 15

1) Реалізувати заповнення шахової дошки за допомогою шаблона проектування. Виведення на екран фігур забезпечити в форматі «позиція фігури» (E2) + колір фігури + її назва.

2) У місті існує три Інтернет провайдери. Перший забезпечує з’єднання із всесвітньою мережею за допомогою телефонної лінії, другий – телевізійного кабелю, третій підтримує безпровідний зв’язок. При цьому кожний з провайдерів пропонує користувачам пакети, що відрізняються по швидкості передачі даних, об’єму трафіку та відповідно вартості. Деякий сервісний центр з підключення населення міста до Інтернету заключив договори з усіма вищезазначеними провайдерами. За допомогою шаблона «Абстрактна фабрика» реалізувати механізм «підключення до інтернету» з підтримкою можливості вибору за певними критеріями провайдера, послугами якого хоче скористуватися клієнт.

Варіант 16

1) За допомогою шаблона проектування побудувати генератор комплектів пам’яток для студентів, які можуть роздаватися у приймальній комісії. Наприклад, для студентів-першокурсників до комплекту включають карту університету, список необхідних телефонів служб університету та факультету, інформацію щодо програми навчання до ступеня «бакалавра», основні правила навчання в вузі (за що відраховують, за що видають червоний диплом тощо). Для студентів-магістрів до комплекту включають програму навчання до ступеня «магістр», загальні поради щодо написання наукових статей, магістерської дисертації тощо. Сформувати по N комплектів обох видів.

2) Забезпечити створення нових користувачів програми за допомогою клонування базового прототипу користувача. Полями, які повинна містити особова картка користувача, можуть бути: логін, пароль, ПІБ, вік, місто, колір шрифту для спілкування у чаті тощо.

Варіант 17

1) За допомогою шаблона проектування допоможіть Діду Морозу сформувати подарункові набори з іграшок, цукерок тощо. Всі складові наборів повинні бути реалізовані як самостійні об’єкти. Подарункові набори для хлопчиків та дівчаток розрізняються.

2) За допомогою шаблона проектування реалізувати процес випуску меблів за прототипами. Наприклад, типами меблів можуть бути «кухня», «вітальня», «спальня» тощо. Кожний гарнітур характеризується типом, матеріалом та складається з модулів (шухляди, полиці і т.д.). Забезпечити реалізацію складових частин меблів як окремих об’єктів, а також організувати «глибоке копіювання» об’єктів при створенні нового екземпляра гарнітура.

Варіант 18

1) За допомогою шаблона запрограмувати конвеєр формування порцій у столовій. Кожна порція повинна містити м’ясне блюдо та гарнір. Якщо одне блюдо закінчилося, необхідно автоматично перейти на формування порцій з іншого блюда.

2) Телефонний довідник складається з різних списків контактів. Контакти можуть повторюватись в різних частинах довідника. Наприклад, одна людина може бути записана в алфавітному покажчику, а також у списку контактів певної компанії. За допомогою шаблона проектування «Прототип» забезпечити створення контакту один раз з його «легким копіюванням» в інші місця.

Варіант 19

1) У різні пори року постачальник продуктів до ресторану

замовляє їх у різних регіонах. Влітку та восени він отримує продукти з України; взимку та навесні – з Туреччини, Африки тощо. Реалізувати механізм постачання продуктів та продемонструвати, що в залежності від пори року місце поставки змінюється.

2) За допомогою шаблона проектування розробити код, який буде створювати складене меню в ресторані швидкого харчування. До складу однієї порції повинні входити м'ясне блюдо, гарнір, напій, десерт. Створити меню для мереж ресторанів швидкого харчування МакДональдс, Швидко та Пузата хата.

Варіант 20

1) Гра у карти. За допомогою шаблона проектування створити колоду карт. Для цього використати відповідні прототипи гральних карт кожної масті. У створених карт змінювати тільки їх ранг. На екран вивести список отриманих таким чином карт.

2) Створити структуру класів, яка б дозволяла виконувати такі дії: 1. створювати масив (одновимірний, двовимірний, тривимірний тощо); 2. заповнювати його даними, 3. виконувати над елементами масиву різні операції, наприклад, сортувати, інвертувати тощо.

Варіант 21

1) За допомогою шаблона запрограмувати конвеєр формування порцій у столовій. Кожна порція повинна містити м'ясне блюдо та гарнір. Якщо одне блюдо закінчилося, необхідно автоматично перейти на формування порцій з іншого блюда.

2) За допомогою шаблона проектування реалізувати автомат, який робить напої. В залежності від того, які параметри задані покупцем,

згенерувати каву (декілька видів) або чай, або какао. Виробництво напою складається з декількох етапів: вибір стаканчика, насипання чаю або кави, додавання води, додавання цукру, видача зробленого напою. Для кожного напою у автоматі є механізм, який його готує.

Варіант 22

1) За допомогою шаблона проектування створити будівельника будинку, який працює в такі етапи: закладення фундаменту, побудова стін та перекриттів, встановлення даху, внутрішні роботи. Будинки можуть бути панельними, цегляними, монолітно-каркасними. Панельні та цегляні будинки не перевищують 9 поверхів. Монолітно-каркасні можуть бути вище. В панельних та цегляних будинках встановлюються дерев'яні вікна, а в монолітних – пластикові.

2) Реалізувати заповнення шахової дошки за допомогою шаблона проектування. Виведення на екран фігур забезпечити в форматі «позиція фігури» (E2) + колір фігури + її назва.

Варіант 23

1) Забезпечити створення нових користувачів програми за допомогою клонування базового прототипу користувача. Полями, які повинна містити особова картка користувача, можуть бути: логін, пароль, ПІБ, вік, місто, колір шрифту для спілкування у чаті тощо.

2) Шкільна бібліотека. За допомогою шаблона проектування створити формувач наборів підручників для різних класів. Підручники діляться на групи: математичні (математика – молодші класи; алгебра і геометрія – середні класи; тригонометрія, стереометрія - старші), природничі (природознавство – молодші, географія – середні, фізика-хімія

- старші), мовні (різні види літератури та мов). Для кожного класу викликається свій комплектатор, який формує набори підручників кожного типу у комплект.

Варіант 24

1) За допомогою шаблона проектування реалізувати процес випуску меблів за прототипами. Наприклад, типами меблів можуть бути «кухня», «вітальня», «спальня» тощо. Кожний гарнітур характеризується типом, матеріалом та складається з модулів (шухляди, полиці і т.д.). Забезпечити реалізацію складових частин меблів як окремих об’єктів, а також організувати «глибоке копіювання» об’єктів при створенні нового екземпляра гарнітура.

2) За допомогою шаблона «Абстрактна фабрика» організувати випуск пакетів соку з різних фруктів та ягід декількома заводами. Заводи відрізняються сортами сировини, упаковкою та ціною продукції. В залежності від пори року клієнти замовляють сік у різних заводів.

Варіант 25

1) Реалізувати заповнення шахової дошки за допомогою шаблона проектування. Виведення на екран фігур забезпечити в форматі «позиція фігури» (E2) + колір фігури + її назва.

2) У місті існує три Інтернет провайдери. Перший забезпечує з’єднання із всесвітньою мережею за допомогою телефонної лінії, другий – телевізійного кабелю, третій підтримує безпровідний зв’язок. При цьому кожний з провайдерів пропонує користувачам пакети, що відрізняються по швидкості передачі даних, об’єму трафіку та відповідно вартості. Деякий сервісний центр з підключення населення міста до Інтернету заключив

договори з усіма вищезазначеними провайдерами. За допомогою шаблона «Абстрактна фабрика» реалізувати механізм «підключення до інтернету» з підтримкою можливості вибору за певними критеріями провайдера, послугами якого хоче скористуватися клієнт.

Контрольні запитання

1. Які шаблони ми називаємо породжужчими шаблонами проектування?
2. Дайте визначення шаблону «Прототип» («Фабричний метод», «Абстрактна фабрика», «Одинак», «Будівельник»).
3. У чому полягає призначення шаблону «Прототип» («Фабричний метод», «Абстрактна фабрика», «Одинак», «Будівельник»)?
4. Назвіть переваги та недоліки застосування шаблону «Прототип» («Фабричний метод», «Абстрактна фабрика», «Одинак», «Будівельник»).
5. Наведіть UML-діаграму класів, які утворюють структуру шаблона «Прототип» («Фабричний метод», «Абстрактна фабрика», «Одинак», «Будівельник»).

2.4. Лабораторна робота №4

Тема роботи: Реалізація поведінкових шаблонів проектування.

Мета роботи: ознаойомлення з основними характеристиками шаблонів «Стратегія», «Шаблонний метод» та «Стан», запам’ятовування поширених ситуацій, коли використання цих шаблонів є доцільним, набуття вмінь та навичок реалізації шаблонів під час створення програмного коду.

Теоретичні відомості

Стратегія

Поведінковий шаблон, який визначає сімейство алгоритмів, інкапсулює кожен з них та робить їх взаємозамінними. Дозволяє змінювати алгоритми незалежно від коду клієнтів. Також відомий як *Policy*. Якщо в системі є алгоритми, які часто можуть використовуватися повторно в різних частинах програми, зручно їх виділити в окрему сутність, параметризувати та запускати там, де це потрібно, не дублюючи сам код.

Поведінковий шаблон «*Стратегія*» потрібно використовувати, коли:

– програма повинна забезпечувати різні варіанти алгоритму або поведінки;
– у наявності є багато споріднених класів, які відрізняються тільки поведінкою (мають схожі інтерфейси, але різну логіку);
– потрібно змінювати поведінку кожного екземпляра класу;
– необхідно змінити поведінку об'єктів на стадії виконання;
– введення інтерфейсу дозволяє класам-клієнтам нічого не знати про класи, що реалізують цей інтерфейс і інкапсулюють в собі конкретні алгоритми (метод “чорного ящика”).

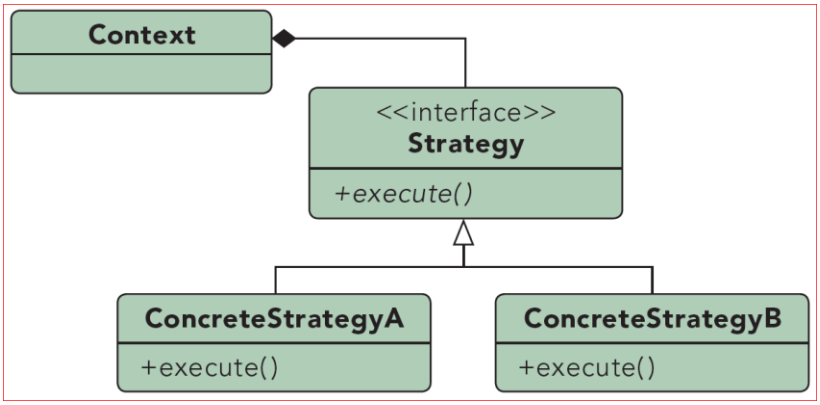

Рис. 24. Структурна схема шаблону «Стратегія»

Учасники шаблону:

- **Strategy** – оголошує загальний для всіх підтримуваних алгоритмів інтерфейс. Клас Context користується цим інтерфейсом для виклику конкретного алгоритму, визначеного в класі ConcreteStrategy.
- **ConcreteStrategy** - реалізує алгоритм, що використовує інтерфейс, оголошений у класі Strategy.
- **Context**: конфігурується об'єктом класу ConcreteStrategy; зберігає посилання на об'єкт класу Strategy; може визначати інтерфейс, який дозволяє об'єкту Strategy отримати доступ до даних контексту.

Контрольний приклад

Завдання

Підприємець має бізнес у різних країнах, які характеризуються різними податковими системами. Розробити програмний продукт, який дозволив би швидко налаштовуватися на обчислення податків у будь-якій країні.

Розв’язок

```
namespace StrategyExample
{
    //Taxes in different countries
    class Program
    {
        static void Main(string[] args)
        {
            InventoryItem itm;
            itm = new InventoryItem();
            itm.ItemAmount = (double)10;

            USATax usaTax;
            usaTax = new USATax();

            UKTax ukTax;
            ukTax = new UKTax();

            itm.SetTax(usaTax);
            Console.WriteLine(itm.ItemWithTax.ToString());
            itm.SetTax(ukTax);

            Console.WriteLine(itm.ItemWithTax.ToString());
            Console.ReadKey();
        }
    }
```

```
    public interface ITaxStrategy
    {
        double CalculateTax(double amount);
    }

    public class USATax : ITaxStrategy
    {
        public USATax()
        {
        }

        public double CalculateTax(double amount)
        {
            return amount * 1.05;
        }
    }

    public class UKTax : ITaxStrategy
    {
        public UKTax()
        {
        }

        public double CalculateTax(double amount)
        {
            return amount * 1.07;
        }
    }


    public class InventoryItem
    {
        private ITaxStrategy _ItemTax;
        private double _ItemAmount;

        public InventoryItem()
        {
        }

        public void SetTax(ITaxStrategy tax)
        {
            _ItemTax = tax;
        }

        public double ItemAmount
        {
            get { return _ItemAmount; }
            set { _ItemAmount = value; }
        }
        public double ItemWithTax
        {
            get { return _ItemTax.CalculateTax(_ItemAmount); }
        }
    }
}
/*Результат:
10.5
10.7
*/
```

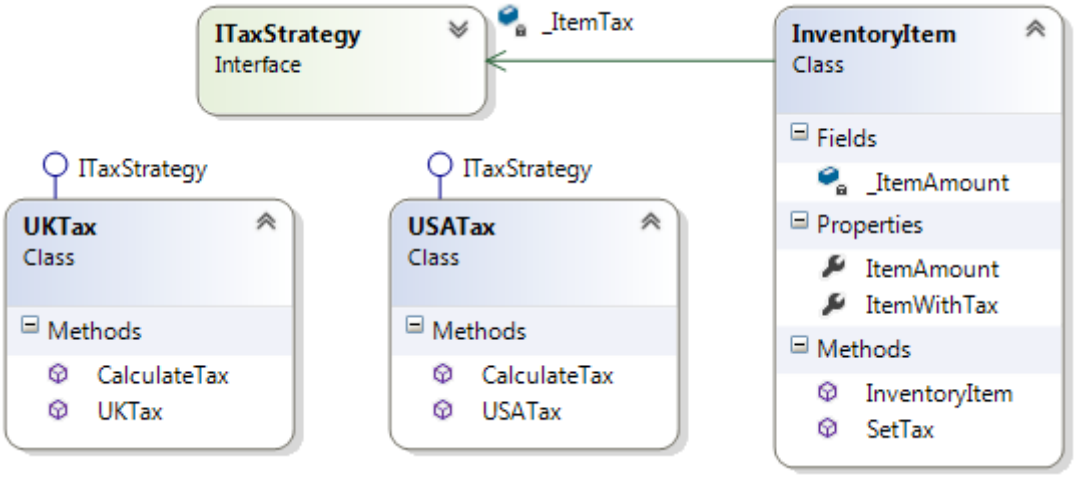

Рис. 25. Діаграма класів шаблону «Стратегія»

Шаблонний метод

Поведінковий шаблон, який визначає функціональність конкретних методів в рамках лише абстрактних сутностей.

Визначає основу алгоритму та дозволяє підкласам перевизначити деякі кроки алгоритму, не змінюючи структуру в цілому.

Поведінковий шаблон «*Шаблонний метод*» потрібно використовувати, коли:

– необхідно забезпечити одноразову реалізацію інваріантних частин алгоритму, залишаючи реалізацію поведінки, що змінюється, на розсуд підкласів;
– треба відокремити та локалізувати в одному класі поведінку, що є загальною для усіх підкласів, щоб запобігти дублювання коду. Це хороший приклад техніки “винесення за лапки з метою узагальнення”;
– батьківські класи повинні мати змогу уніфіковано звертатися до поведінки підкласів;

– необхідний дозвіл на розширення коду нащадками тільки в чітко визначених місцях.

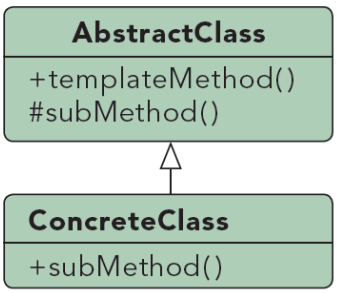

Рис. 26. Структурна схема шаблону «Шаблонний метод»

Учасники шаблону:

– **AbstractClass** – визначає абстрактні примітивні операції, що заміщуються у конкретних підкласах для реалізації кроків алгоритму; реалізує шаблонний метод, що визначає скелет алгоритму;
– **ConcreteClass** – реалізує примітивні операції, що виконують кроки алгоритму у спосіб, необхідний підкласу. ConcreteClass припускає, що інваріантні (зафіксовані) кроки алгоритму будуть виконані в AbstractClass.

Контрольний приклад

Завдання

Для «Шаблонного методу» не наводиться тематичний приклад для того, щоб полегшити сприйняття програмної реалізації власне обов’язкових складових шаблону. Нижче розміщено абстрактний приклад.

Розв’язок

```
interface IPrimitives
{
    string Operation1();
    string Operation2();
}
```

```
class Algorithm
{
    public void TemplateMethod(IPrimitives a)
    {
        string s =
        a.Operation1() +
        a.Operation2();
        Console.WriteLine(s);
    }
}
class ClassA : IPrimitives
{
    public string Operation1()
    {
        return "ClassA:Op1 ";
    }
    public string Operation2()
    {
        return "ClassA:Op2 ";
    }
}
class ClassB : IPrimitives
{
    public string Operation1()
    {
        return "ClassB:Op1 ";
    }
    public string Operation2()
    {
        return "ClassB.Op2 ";
    }
}
class TemplateMethodPattern
{

    static void Main()
    {
        Algorithm m = new Algorithm();

        m.TemplateMethod(new ClassA());
        m.TemplateMethod(new ClassB());
    }
}

/* Output
ClassA:Op1 ClassA:Op2
ClassB:Op1 ClassB.Op2
*/
```

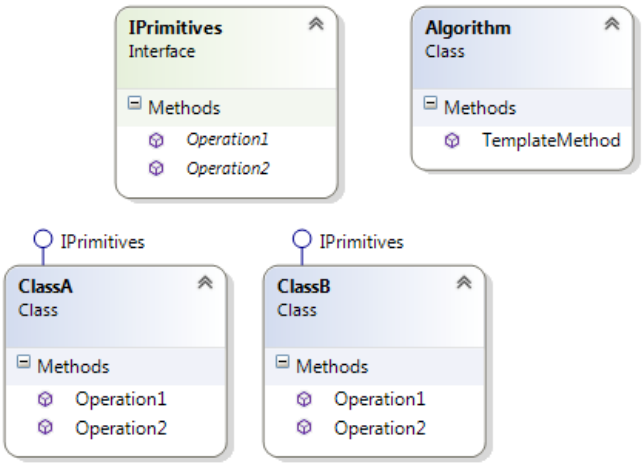

Рис. 27. Діаграма класів шаблону «Шаблонний метод»

Стан

Поведінковий шаблон, який дозволяє об'єкту варіювати свою поведінку залежно від внутрішнього стану.

Часто називається *динамічною Strategy*.

Передбачає реалізацію поведінки, асоційованої з певним станом об'єкту, а також забезпечення зміни поведінки відповідно до зміни внутрішнього стану.

Поведінковий шаблон «*Стан*» потрібно використовувати, коли:

- поведінка об'єкта залежить від його стану і повинна змінюватися під час виконання;
- в коді операцій зустрічаються умовні оператори, які складаються з багатьох гілок і у яких вибір гілки залежить від стану (зазвичай стан подається зліченними константами);
- одна і та сама структура умовного оператора повторюється в кількох операціях, патерн "*Стан*" передбачає розміщення кожної гілки в окремому класі. Це дозволяє трактувати стан об'єкту як самостійний об'єкт, який може змінюватися незалежно від інших;
- переходи між станами об'єкту повинні бути явними.

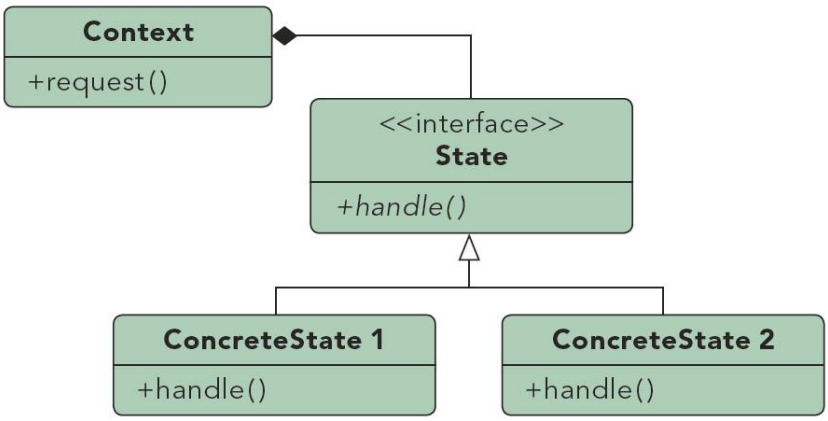

Рис. 28. Структурна схема шаблону «Стан»

Учасники шаблону:

- **Context** – визначає інтерфейс, що представляє інтерес для клієнтів. Зберігає екземпляр підкласу ConcreteState, яким визначається поточний стан;
- **State** – визначає інтерфейс для інкапсуляції поведінки, асоційованого з конкретним станом контексту Context;
- **ConcreteState** – кожен підклас реалізує поведінку, асоційовану з деяким станом контексту Context.

Контрольний приклад

Завдання

За допомогою шаблона проектування "**Стан**" змоделювати ситуацію, коли ми намагаємось налити воду з пляшки, яка є закритою/відкритою.

Розв'язок

```
namespace StateExample
{
    abstract class State
    {
        protected string strStatename;
        abstract public void Pour();
    }

    class OpenedState : State
    {
        public OpenedState()
        {
            strStatename = "Opened";
        }
        override public void Pour()
        {
            Console.WriteLine("...pouring...");
            Console.WriteLine("...pouring...");
            Console.WriteLine("...pouring...");
        }
    }

    class ClosedState : State
    {
        public ClosedState()
        {
            strStatename = "Closed";
        }
        override public void Pour()
        {
            Console.WriteLine("ERROR - bottle is closed - cannot pour");
        }
    }
```

```
    class ContextColaBottle
    {
        public enum BottleStateSetting
        {
            Closed,
            Opened
        };

        OpenedState openedState = new OpenedState();
        ClosedState closedState = new ClosedState();

        public ContextColaBottle()
        {
            // Initialize to closed
            CurrentState = closedState;
        }

        private State CurrentState;

        public void SetState(BottleStateSetting newState)
        {
            if (newState == BottleStateSetting.Closed)
            {
                CurrentState = closedState;
            }
            else
            {
                CurrentState = openedState;
            }
        }

        public void Pour()
        {
            CurrentState.Pour();
        }
    }

    public class Client
    {
        public static int Main(string[] args)
        {
            ContextColaBottle contextColaBottle = new ContextColaBottle();

            Console.WriteLine("initial state is closed");

            Console.WriteLine("Now trying to pour");
            contextColaBottle.Pour();

            Console.WriteLine("Open bottle");
            contextColaBottle.SetState(ContextColaBottle.BottleStateSetting.Opened);

            Console.WriteLine("Try to pour again");
            contextColaBottle.Pour();

            Console.ReadKey();
            return 0;
        }
    }
}
/*Результат:
initial state is closed
Now trying to pour
ERROR - bottle is closed - cannot pour
Open bottle
Try to pour again
...pouring...
...pouring...
...pouring...
*/
```

Індивідуальні завдання

Варіант 1

1) За допомогою шаблона проектування реалізувати роботу автомата з продажу напоїв у різних ситуаціях. Передбачити такі ситуації, коли внесено готівку, обрано товар та: товар є та готівки вистачає; товар є – готівки замало; товар є – готівки забагато; товару нема.
2) За допомогою шаблона проектування реалізувати декілька режимів прання у пральній машині. Кожний режим по-своєму впливає на стан речей з різних матеріалів. Наприклад, один режим добре пере вовняні речі, але псує шовкові. Інший режим діє на тканини навпаки тощо. Відповідно до налаштувань пральної машини випрати заданий набір речей з різних тканин та вивести на екран їх стан до та після прання.

Варіант 2

1) За допомогою шаблона проектування у грі «Морський бій» реалізувати зміну зображення кораблів на ігровому полі в залежності від їх стану: цілий, поранений, убитий.
2) За допомогою шаблона проектування реалізувати декілька способів переміщення нагору на вниз у торговому центрі. Наприклад, можна ходити сходами, їздити на ескалаторі та їздити на ліфті. В залежності від стану устаткування або від часу доби активувати той чи інший спосіб пересування відвідувачів. Наприклад, з 22.00 до 10.00 кожного дня ескалатори та ліфти перестають працювати. А під час профілактичних робіт з ескалатором має працювати ліфт та навпаки.

Варіант 3

1) За допомогою шаблона проектування змоделювати зміну кольору та кількості листя дерева в залежності від пори року. Взимку кількість листя на деревах (крім вічнозелених) дорівнює 0; навесні – половина максимальної кількості + колір листя – зелений; влітку – максимум листя + колір листя – зелений; восени – половина максимальної кількості листя + колір – жовтий.
2) У компанії практикуються різні методи розрахунку показників якості роботи. Наприклад, розрахунок показників на основі врахування відношення прибутку до витрат; або врахування якості підготовки співробітників (досвід роботи, освіта тощо). За допомогою шаблона проектування реалізувати можливість визначати якість роботи компанії різними способами.

Варіант 4

1) За допомогою шаблона проектування забезпечити підтримку проведення розрахунків у ресторані з використанням клієнтської накопичувальної картки клієнта ресторану. В залежності від типу картки клієнту надається знижка відповідно 5%, 10% та 15%. Сума, сплачена за замовлення, додається до суми балансу картки. При перевищенні даною сумою значення 5000 грн., картка з 5% стає 10%. При перевищенні 10000грн. – 15%.
2) За допомогою шаблона проектування реалізувати декілька різноманітних способів зберігання тексту до файлу. Перший спосіб – звичайний: зберігаємо текст без внесення жодних змін. Другий спосіб – видаляємо всі зайві пробіли з тексту перед збереженням. Третій спосіб – застосовуємо кодування тексту (або архівацію). В залежності від

налаштування програми забезпечити збереження тексту до файлу найбільш доцільним способом.

Варіант 5

1) За допомогою шаблону проектування реалізувати перемикання режимів роботи сервера, до якого через мережу мають змогу підключатися різні користувачі. У випадку перевищення припустимого числа підключень, сервер перестає надавати доступ до своїх ресурсів та видає повідомлення про це всім користувачам, які роблять спроби встановити з ним з'єднання. У разі наявності можливості підключення до нього, сервер приймає запит на встановлення зв'язку.
2) За допомогою шаблону проектування реалізувати декілька комплексів спортивних занять для людей з різними фізичними можливостями: підвищеної складності, нормальний, полегшений. В залежності від фізичної підготовленості людини призначати їй той чи інший набір вправ, які вона буде виконувати під час ранкової зарядки.

Варіант 6

1) За допомогою шаблона проектування реалізувати підкажчик щодо приймання ліків хворими за чітким графіком: в залежності від дня тижня хворий повинен приймати різні ліки, наприклад:
    а) в понеділок зранку - ліки №1, ввечері – ліки №4;
    б) у вівторок зранку – ліки №2, ввечері - №4;
    в) у середу зранку - ліки №3, ввечері - №4;
    г) у четвер зранку – ліки №1, ввечері - №4;
    д) в п'ятницю зранку – ліки №2, ввечері - №4;
    е) в суботу зранку – ліки №3, ввечері - №4;

ж) у неділю перерва у прийомі ліків.

2) За допомогою шаблону проектування реалізувати різні алгоритми подання документів для отримання візи такими категоріями людей: службовці, приватні підприємці, діти. Як відомо, для кожної групи потрібні різні документи. Наприклад, ПП має подавати копію звіту про прибуток за останні 2 квартали з податкової інспекції; діти – дитячий проїзний документ тощо.

Варіант 7

1) За допомогою шаблону проектування відтворити процес вивчення студентом певного предмету та здачі ним екзамену. За кожне самостійне заняття студент опановує 5% матеріалу. При опануванні студентом більше 60% матеріалу, він переходить у категорію потенційних «трійочників», при опануванні більше 80% - у категорію студентів, що претендують на «четвірку», та при опануванні більше 95% матеріалу – стає потенційним відмінником.
2) В залежності від тканини, з якої пошито одяг, його можна прати та прасувати за певних умов або чистити у хімчистці. За допомогою шаблону проектування реалізувати різні алгоритми чистки одягу: прання при різних температурах + прасування в різних режимах праски, а також хімічна чистка одягу.

Варіант 8

1) За допомогою шаблону проектування реалізувати автоматичний розрахунок знижок у сплаті за абонементи у спортивному комплексі. Якщо людина є клієнтом більше півроку, вона має знижку 5% (статус «Постійний клієнт»). Якщо людина оформлює місячний абонемент до

спорткомплексу регулярно (рівно через місяць від дати попереднього придбання абонементу), вона має додатково 10% знижки (статус «VIP клієнт»).

2) В залежності від кількості часу, який домогосподарка може витратити на прибирання квартири, існує 3 варіанти прибирання: легке прибирання (скласти розкидані речі, підмести підлогу), звичайне прибирання (витерти пил, використати пилосос) та генеральне прибирання (вологе прибирання підлоги, а також миття вікон). За допомогою шаблону проектування реалізувати дані види прибирання.

Варіант 9

1) За допомогою шаблона проектування реалізувати зміну зображення погонів у персонажів комп'ютерної військової гри в залежності від їх чину.
2) За допомогою шаблона проектування реалізувати декілька варіантів роздрукування документа. Сам процес складається з етапу завантаження паперу до друкарського пристрою та етапу друку. Завантаження паперу може здійснюватися як вручну, так і автоматично. Друк може здійснюватися вручну (друкарська машинка), струменевим або лазерним принтером. Передбачити такі варіанти роздрукування документа: на друкарській машинці, на струменевому принтері з ручною подачею паперу, на струменевому принтері з автоматичною подачею паперу, а також на лазерному принтері з ручною та автоматичною подачею паперу.

Варіант 10

1) За допомогою шаблона проектування реалізувати функції ввімкнення та вимкнення певного пристрою, у якому для цього є одна єдина кнопка, повторне натиснення на яку і призводить до ввімкнення виключеного пристрою та на вимкнення включеного.
2) За допомогою шаблона проектування реалізувати декілька варіантів здійснення покупки в магазині. Процес купівлі складається з таких етапів: прихід до магазину, вибір товару, оплата, доставка. В залежності від типу магазину дані етапи можуть бути реалізовані по-різному. Етап приходу до магазину може бути як реальним, так і віртуальним (у випадку, якщо мова йдеться про Інтернет-магазин). Таким чином, для того, щоб дістатися до магазину в одному випадку треба знати його фізичну адресу, в іншому – його адресу у мережі Інтернет. Етап вибору товару в реальному магазині здійснюється за допомогою продавця-консультанта; в Інтернет-магазині – за допомогою служби підтримки сайту за телефоном або онлайн. Оплата може здійснюватися в обох видах магазинів як готівкою, так і платіжними картками. Доставка купленого товару може виконуватися як безпосередньо покупцем, так і службою доставки магазину (незалежно від його виду). Реалізувати такі варіанти процесу купівлі: покупка в інтернет-магазині з оплатою готівкою кур’єру служби доставки; покупка в Інтернет-магазині з оплатою кредитною карткою та само вивезенням товару зі складу; покупка в реальному магазині з оплатою готівкою та доставкою товару до дому покупця силами служби доставки магазину.

Варіант 11

1) За допомогою шаблона проектування відтворити процес споживання певного продукту. Даний продукт має визначений термін зберігання. У випадку перевищення терміну зберігання, продукт стає шкідливим для здоров’я людини. У випадку використання продукту, коли термін його зберігання ще не сплив (тобто, коли продукт є корисним), загрози здоров’ю споживача немає.
2) За допомогою шаблона проектування реалізувати отримання різних форм списку навчальної групи відповідно до таких етапів: формування списку студентів, які мають потрапити до списку, за заданим критерієм; сортування списку за заданим параметром; друк списку. Критеріями відбору студентів можуть бути: наявність навчальної заборгованості; наявність фінансової заборгованості. Параметри, за якими можна сортувати список, - за прізвищем студента в алфавітному порядку; за кількістю предметів, які складають заборгованість або за сумою фінансового боргу. Реалізувати три варіанти таких списків.

Варіант 12

1) За допомогою шаблона проектування реалізувати систему клімат-контролю приміщення: в залежності від обраного кліматичного режиму та стану повітря у приміщенні система підвищує чи знижує вологість повітря, піднімає чи понижує його температуру.
2) За допомогою шаблона проектування реалізувати декілька режимів прання у пральній машині. Кожний режим по-своєму впливає на стан речей з різних матеріалів. Наприклад, один режим добре пере вовняні речі, але псує шовкові. Інший режим діє на тканини навпаки тощо.

Процес прання складається з таких етапів: завантаження білизни до машинки, засипання миючих засобів до неї, безпосередньо прання (результатом якого є стан випраної речі), віджим (або його відсутність). Відповідно до налаштувань пральної машини випрати заданий набір речей з різних тканин та вивести на екран їх стан до та після прання

Варіант 13

1) За допомогою шаблону проектування відтворити процес складання студентом екзамену з предмету. У разі нездачі студентом екзамену, він має право на 2 перездачі. Усі наступні спроби здати екзамен є незаконними і спроба внести оцінку у відомість такому студенту буде викликати помилку програми обліку успішності студентів. Студент, який має заборгованість, має статус такого, що не може бути переведений на наступний курс.
2) Процес проведення співбесід у it-компаніях складається з таких етапів: 1 – співбесіда з відділом кадрів (представниками HR) та менеджерами, 2 – співбесіда з технічними спеціалістами, 3 – співбесіда з замовником. Співбесіда з відділом кадрів та замовником, як правило, носить стандартний характер. Співбесіда з технічними спеціалістами може складатися з усної розмови, з письмового виконання певних тестів. Також даний етап може розділятися на декілька кроків (наприклад, співбесіда зі спеціалістами з .Net та співбесіда з розробниками баз даних). Реалізувати 3 варіанти проведення співбесіди робітника.

Варіант 14

1) За допомогою шаблону проектування реалізувати призначення відвідувачу певної конференції різних статусів. Якщо людина більше 3

років відвідує даних захід, вона має статус «Постійний відвідувач», якщо більше 5 років – статус «Суперпостійний відвідувач», якщо більше 7 років – «Почесний гість». В залежності від статусу людини, вона отримує запрошення до участі у різних закритих засіданнях секцій, урочистому відкритті та закритті конференції тощо.

2) Виробництво сиркових продуктів. В загальному вигляді процес складається з таких етапів: завантаження сировини до агрегату, вибір параметрів виробництва (маса сирків, наявність доповнень до сиркової маси таких як курага чи ізюм тощо, наявність шоколадної глазурі, упаковка у папір або у фольгу), безпосередньо сам процес виробництва товару, упаковка, формування ящиків з товарами. В залежності від налаштувань програми процес виробництва товару та його упаковки може бути різним. Завантаження сировини, вибір параметрів та формування ящиків з товаром – етапи, спільні для всіх варіантів процесу. Реалізувати на вибір 3 варіанти організації процесу (за допомогою встановлення потрібних значень параметрів).

Варіант 15

1) За допомогою шаблону проектування реалізувати спрощений процес встановлення програмного забезпечення на комп’ютер. Зазвичай, одні програми потребують наявності інших на комп’ютері. Нехай, необхідно встановити 3 програми. І програма ставиться без будь-яким додаткових умов, ІІ программа потребує наявності І-ї, ІІІ-я програма потребує наявності ІІ-ї. У випадку спроби встановлення одразу ІІ чи ІІІ програми, видавати відповідні повідомлення користувачу про неможливість виконання даної дії.

2) Покупка комп’ютера. В загальному вигляді процес покупки складається з етапів: вибір комплектуючих відповідно до заданих умов (грошей, якості/потужності, специфічного призначення), виконання розрахунку вартості комп’ютера, оплата покупки готівкою. Реалізувати 3 варіанти покупки комп’ютера з різними стратегіями вибору комплектуючих.

Варіант 16

1) Дано список балів – рейтинг студентів наприкінці семестру. За допомогою шаблона проектування «Стратегія» реалізувати механізм його перетворення у список оцінок або список заліків-незаліків в залежності від того, яка форма контролю передбачена у навчальному плані для цього предмета.
2) Процес приготування різних блюд з однакового набору продуктів складається з таких етапів: взяти певну (невелику) кількість продуктів, помити їх, нарізати – це спільні етапи. Сам процес приготування: або просто змішати нарізане, або зажарити, або зварити тощо – даний етап для кожного рецепту свій . Останній етап – додати сіль, цукор, різни приправи за смаком – також може бути реалізований по-різному. Блюда для приготування з одного і того самого набору продуктів придумати самостійно. Показати реалізацію як мінімум трьох блюд.

Варіант 17

1) Одна й та сама тварина у різному настрої може видавати різні звуки (від муркотіння до ричання у кішок і т.п.). За допомогою шаблона проектування («Стратегія» або «Стан») реалізувати метод «Подати голос» у об’єкта «Тварина», враховуючи його стан.

2) Приготування напою. Процес складається з таких етапів: закип’ятити воду – спільний етап, покласти те, з чого буде робитися напій (заварку в листочках, каву, листя рослин тощо) – може реалізовуватися по-різному, додавання цукру за смаком – спільний етап. За допомогою шаблону проектування реалізувати дану дію.

Варіант 18

1) За допомогою шаблона проектування забезпечити виведення текстового повідомлення у віконці чату звичайним шрифтом чорного кольору або у форматі «Ім’я користувача» (червоним жирним шрифтом) «Час надходження повідомлення» (синім звичайним шрифтом): «Повідомлення» (звичайним чорним шрифтом) в залежності від налаштувань програми.
2) Маршрут з дому до роботи складається з таких етапів: дійти до транспорту (у гараж, на зупинку громадського транспорту, в метро), доїхати на цьому виді транспорту до роботи (в метро пробок не буває, на наземному транспорті - бувають), припаркуватися (для власного транспортного засобу), дійти до робочого місця (від парковки, зупинки тощо). Реалізувати 3 варіанти діставання до робочого місця за допомогою шаблона проектування.

Варіант 19

1) Туристична компанія надає послугу трансферу з аеропорту до готелю. В залежності від того, тур якого класу куплений туристом (люкс, стандарт або економ), для цього викликається таксі, мікроавтобус або стандартний великий автобус. В залежності від якості трансферу варіюється і вартість послуги. За допомогою шаблона проектування

«Стратегія» реалізувати всі три види метода-трансфера, який відповідно до подоланої відстані повинен як результат повертати загальну суму до сплати туристом.

2) Процес відвідування лікаря має такі етапи: запис у реєстратурі на прийом, первинний огляд лікаря (можливо діагностика) – спільні етапи; в залежності від результатів первинного огляду подальше лікування є повністю різним. За допомогою шаблонів проектування змоделювати даний процес.

Варіант 20

1) За допомогою шаблона проектування «Стратегія» реалізувати сортування списку студентів за різними критеріями (не менш 3 критеріїв). Наприклад, за прізвищем, за номером залікової книжки, за датою народження тощо. В залежності від налаштування програми активувати той чи інший метод сортування.
2) Етапи оформлення послуги у РАГСі можуть бути такими: подати заяву та сплатити за послугу (природно, що заяви тут мають бути різні та плати за них також); провести саму процедуру , яка може бути весіллям, розлучення, реєстрацією дитини; в результаті отримати свідоцтво про виконання послуги (свідоцтво про брак, розлучення, народження дитини). За допомогою шаблону проектування реалізувати три вищезгадані варіанти послуг РАГСу.

Варіант 21

1) За допомогою шаблона стратегія забезпечити видачу карт гравцям у комп’ютерній карточній грі. В залежності від гри, реалізувати відповідну стратегію роздачі карт (не менше 3 ігор).

2) • Похід до перукарні. Процес обслуговування у перукарні складається з таких етапів: помив голови, розчісування, виконання зачіски, фінальна сушка/лакування тощо. Під етапом «Виконання зачіски» може розумітися укладка, фарбування волосся, стрижка. За допомогою шаблону проектування реалізувати програмну модель перукарського сервісу.

Варіант 22

1) За допомогою шаблона проектування реалізувати різноманітні методи лікування пацієнта при заданих симптомах: традиційної медицини, народної медицини, екстрасенсорні тощо. В залежності від уподобань пацієнта або від успішності лікування іншими засобами забезпечити реалізацію тієї чи іншої стратегії лікування.
2) Процес ремонту в квартирі складається з таких етапів:
    а) демонтажні роботи: зняття старих шпалер, покриттів підлоги, дверних блоків, а також ламання перегородок;
    б) винос будівельного сміття;
    в) заміна вікон;
    г) укладка електричних кабелів;
    д) штукатурні роботи, вирівнювання стін та стелі;
    е) підготовка підлоги: монтаж теплої підлоги, стяжка, вирівнювання.
    ж) встановлення дверей
    з) оздоблювальні роботи тощо.

В залежності від суми, виділеної на проведення ремонту, різні етапи ремонту можуть бути виконані по-різному. Наприклад, у більш дешевому варіанті ремонту тепла підлога може бути відсутньою, заміна вікон також

необов’язкова. Інші етапи, наприклад, винесення сміття – є спільними для всіх видів ремонту. За допомогою шаблона проектування реалізувати декілька видів ремонту.

Варіант 23

1) За допомогою шаблона проектування реалізувати декілька способів організації чаювання у кафе. Наприклад, дешевий спосіб – заварювання чаю в пакетиках. Більш дорогий спосіб – заварювання листового чаю. Найдорожчий спосіб – влаштування чайної церемонії. В залежності від вибору клієнта забезпечити процес чаювання у кафе.
2) Навчальний процес складається з таких етапів: прослуховування лекційного матеріалу; практикування: виконання лабораторної роботи або практикуму; перевірка знань з пройденого матеріалу: контрольна робота, домашня контрольна робота, тест. За допомогою шаблона проектування змоделювати процес вивчення матеріалу з декількох предметів, якими передбачається різна реалізація зазначених етапів.

Варіант 24

1) За допомогою шаблона проектування реалізувати декілька стратегій приготування їжі в ресторані. Одні клієнти принципово вживають лише сирі овочі, інші – лише парові чи варені, треті - печені. В залежності від уподобань клієнта зорієнтувати кухаря, яким чином приготувати страву для нього.
2) Процес розробки програмного забезпечення складається етапів проведення бізнес-аналізу, визначення вимог до ПЗ, розробки архітектури програми, реалізації архітектури, тестування, впровадження розробленого ПЗ, підтримки його функціонування. В залежності від

масштабності компанії-розробника дані етапи можуть бути реалізовані по-різному. Наприклад, етап тестування може складатися тільки з однієї вибіркової перевірки випадкових кусків програмного продукту на випадковому наборі вхідних даних, а може бути реалізований у вигляді спланованої довготривалої діяльності. За допомогою шаблону проектування реалізувати декілька варіантів процесу розробки ПЗ.

Варіант 25

1) За допомогою шаблону проектування реалізувати автоматичний розрахунок знижок у сплаті за абонементи у спортивному комплексі. Якщо людина є клієнтом більше півроку, вона має знижку 5% (статус «Постійний клієнт»). Якщо людина оформлює місячний абонемент до спорткомплексу регулярно (рівно через місяць від дати попереднього придбання абонементу), вона має додатково 10% знижки (статус «VIP клієнт»).
2) В залежності від кількості часу, який домогосподарка може витратити на прибирання квартири, існує 3 варіанти прибирання: легке прибирання (скласти розкидані речі, підмести підлогу), звичайне прибирання (витерти пил, використати пилосос) та генеральне прибирання (вологе прибирання підлоги, а також миття вікон). За допомогою шаблону проектування реалізувати дані види прибирання.

Контрольні запитання

1. Які шаблони ми називаємо поведінковими шаблонами проектування?
2. Дайте визначення шаблону «Стратегія» («Шаблонний метод», «Стан»).

3. У чому полягає призначення шаблону «Стратегія» («Шаблонний метод», «Стан»)?
4. Назвіть переваги та недоліки застосування шаблону «Стратегія» («Шаблонний метод», «Стан»).
5. Наведіть UML-діаграму класів, які утворюють структуру шаблона «Стратегія» («Шаблонний метод», «Стан»).

2.5. Лабораторна робота №5

Тема роботи: Реалізація складних поведінкових шаблонів проектування.

Мета роботи: ознаойомлення з основними характеристиками шаблонів «Ланцюжок обов'язків», «Команда» та «Посередник», запам'ятовування поширених ситуацій, коли використання цих шаблонів є доцільним, набуття вмінь та навичок реалізації шаблонів під час створення програмного коду.

Теоретичні відомості

Ланцюжок обов'язків

Поведінковий шаблон, призначений для організації в системі рівнів відповідальності.

Будує об'єкти складених частин програми зв'язаними між собою ланцюжком для передачі запиту на обробку від більш низьких, деталізованих прошарків до більш глобальних.

Дозволяє уникнути зв'язування відправника запиту та його отримувача.

Поведінковий шаблон «*Ланцюжок обо'язків*» потрібно використовувати, коли:

– у розроблюваній системі є група об'єктів, які можуть оброблювати запит певного типу, причому справжній обробник запиту заздалегідь невідомий;
– набір об'єктів, які обробляють запит, повинен визначатися динамічно;
– потрібно відправити запит одному з декількох об'єктів, не вказуючи явно, якому саме;
– всі повідомлення повинні бути оброблені хоча б одним об'єктом системи;
– повідомлення в системі оброблюються за схемою "оброби сам або передай іншому", тобто одні повідомлення обробляються на тому ж рівні, де вони були отримані, а інші – передаються об'єктам іншого рівня.

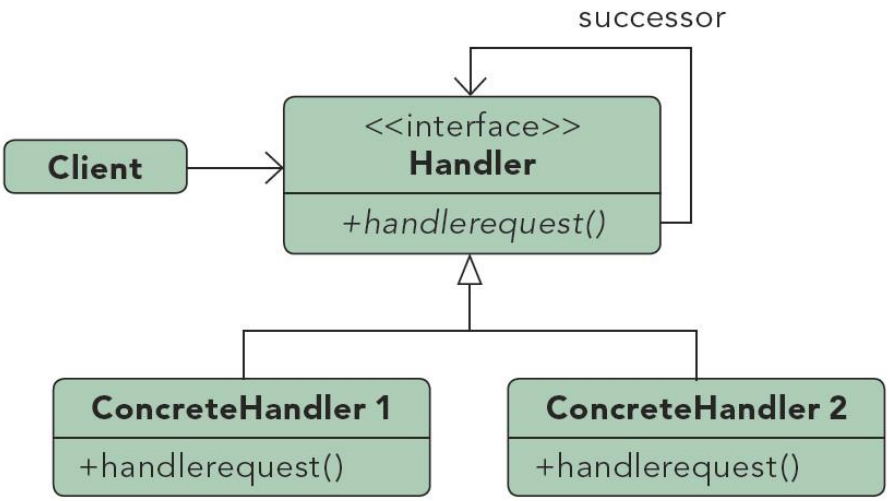

Рис. 29. Структурна схема шаблону «Ланцюжок обов'язків»

Учасники шаблону:

– **Handler** – визначає інтерфейс для обробки запитів, реалізує зв'язок зі спадкоємцем (необов'язково);

– **ConcreteHandler** –оброблює запит, за який відповідає. Має доступ до свого спадкоємця. Якщо може самостійно обробити запит – робить це, якщо ні – передає його своєму спадкоємцю;
– **Client** – відправляє запит деякому об’єкту ConcreteHandler у ланцюжку.

Коли клієнт ініціює запит, він починає просуватися ланцюжком, поки якийсь об’єкт ConcreteHandler не візьме на себе відповідальність за його обробку.

Контрольний приклад

Завдання

За допомогою шаблона проектування організувати передачу значення поточного часу у ланцюжок обробників. Який саме обробник візьме на себе відповідальність за обробку запиту, залежить від того, в яку частину хвилини цей запит був надісланий.

Розв’язок

```
namespace ChainOfResponsibilitiesExample
{
    class Program
    {
        static void Main(string[] args)
        {
            /* Set up a chain of candidate handlers to handle the
            request. Chain will be (1st considered -> last):
            ConcreteHandler1, ConcreteHandler2. */

            HandlerBase chain = new ConcreteHandler3();
            HandlerBase more = new ConcreteHandler2();
            more.Successor = chain;
            chain = new ConcreteHandler1();
            chain.Successor = more;
            // hand the request to the chain
            Console.WriteLine(chain.SayWhen(DateTime.Now));
            Console.ReadKey();
        }
    }

    public interface IHandler
    {
        // properties
        HandlerBase Successor { set; }
    }
```

```
          // --- Abstract Handler
          abstract public class HandlerBase
          {
              // fields
              protected HandlerBase _successor;
              // properties
              public HandlerBase Successor
              {
                  set { _successor = value; }
              }
              // methods
              abstract public string SayWhen(DateTime localTime);
          }

           // --- Concrete handlers
         public class ConcreteHandler1: HandlerBase {
              // methods
              override public string SayWhen(DateTime localTime) {
                   if (localTime.Second < 15)
                        return  String.Format("I  am  {0}.\nThe  time  is  {1:T}\nWe  are  just
starting on a shiny new minute.", this.ToString(),localTime);
                   else {
                        if (_successor != null) {
                            return _successor.SayWhen(localTime);
                        }
                        else
                              throw  new  ApplicationException("ChainOfResponsibility  object
exhausted all successors without call being handled.");
                   }
              }
         }

         public class ConcreteHandler2: HandlerBase {
              // methods
              override public string SayWhen(DateTime localTime) {
                   if ((localTime.Second >= 15) &
                      (localTime.Second < 45))
                         return String.Format("I am {0}.\nThe time is {1:T}\nWe are into the
middle half of this minute.", this.ToString(),localTime);
                   else {
                         if (_successor != null) {
                              return _successor.SayWhen(localTime);
                         }
                         else
                              throw  new  ApplicationException("ChainOfResponsibility  object
exhausted all successors without call being handled.");
                   }
              }
         }


         public class ConcreteHandler3 : HandlerBase
         {
             // methods
             override public string SayWhen(DateTime localTime)
             {
                 if ((localTime.Second >= 45) &
                    (localTime.Second < 60))
                     return String.Format("I am {0}.\nThe time is {1:T}\nWe are into the third
quarter of this minute.", this.ToString(), localTime);
                 else
                 {
                     if (_successor != null)
                     {
                         return _successor.SayWhen(localTime);
                     }
                     else
                         throw    new    ApplicationException("ChainOfResponsibility    object
exhausted all successors without call being handled.");
                 }
             }
         }
      }
```

```
/*Результат:
I am ChainResponsibilityExample.ConcreteHandler3.
The time is 3:00:58
We are into the third quarter of this minute.
*/
```

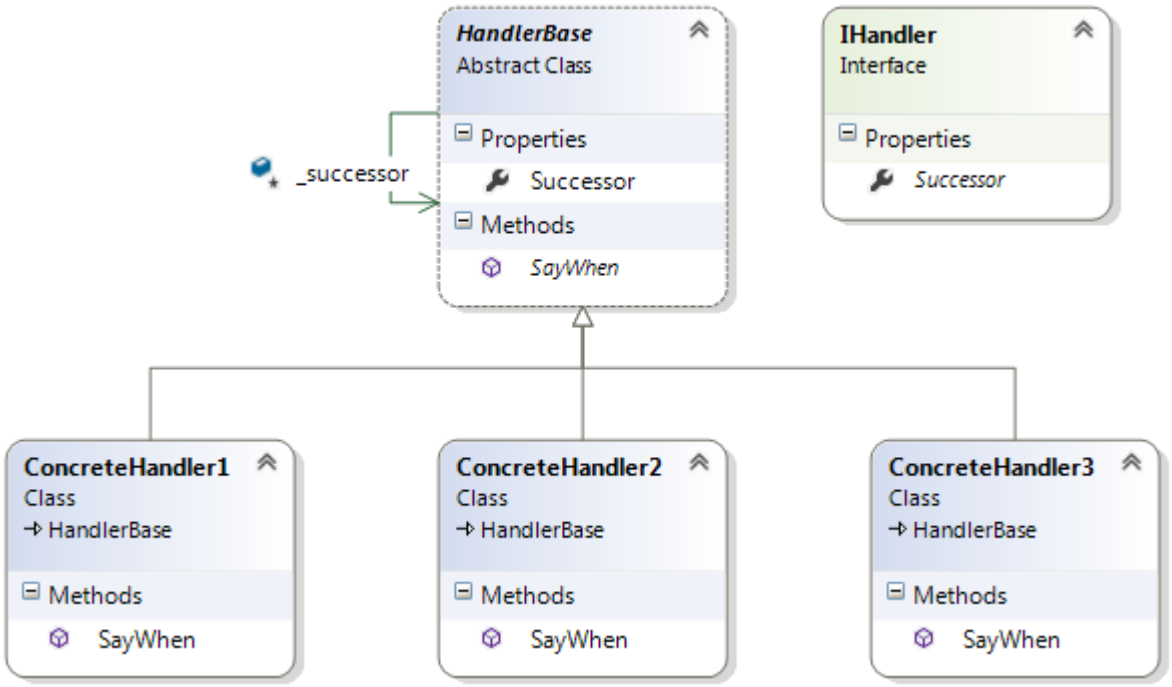

Рис. 30. Діаграма класів шаблону «Ланцюжок обов'язків»

Команда

Поведінковий шаблон, який інкапсулює різні алгоритми в єдину сутність (об'єкт), завдяки чому можна параметризувати клієнтів різними запитами, вести історію виконаних операцій та підтримувати скасування операцій. Відомий також під іменем *Action (дія)*, *Transaction (транзакція)*.

Забезпечує обробку команди у вигляді об'єкту, що дозволяє зберігати її, передавати у якості параметра методам, а також повертати її як результат методу.

Поведінковий шаблон «*Команда*» потрібно використовувати, коли:

– треба параметризувати об’єкти виконуваною дією;
– потрібно визначати, ставити в чергу та виконувати запити у різний час;
– потрібно підтримати відміну операцій;
– треба підтримати протоколювання змін (лог);
– структурувати систему на основі високорівневих операцій, побудованих з примітивних;
– команди для різних отримувачів повинні бути оброблені у різний спосіб.

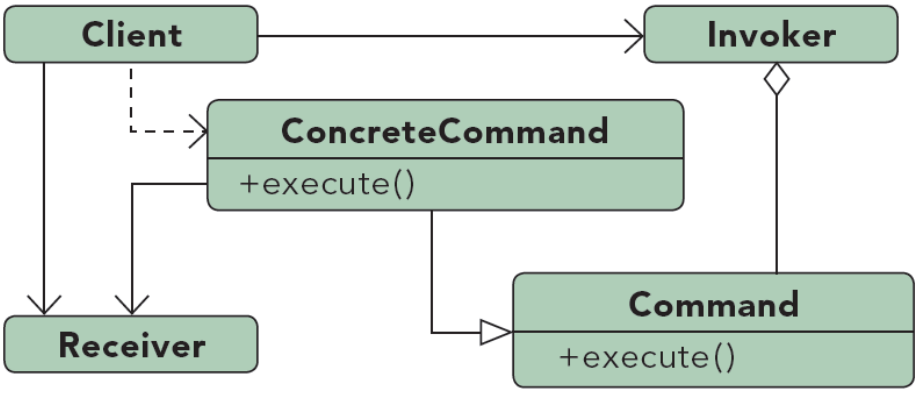

Рис. 31. Структурна схема шаблону «Команда»

Учасники шаблону:

– **Command** – визначає інтерфейс для виконання операцій;
– **ConcreteCommand** – визначає зв’язок між об’єктом-отримувачем Receiver та дією; реалізує операцію Execute шляхом виклику відповідних операцій об’єкту Receiver;
– **Client** – створює об’єкти класу ConcreteCommand та встановлює їх отримувачів;
– **Invoker** – ініціатор. Звертається до команди для виконання запиту;
– **Receiver** – отримувач. Має інформацію про способи виконання операцій, необхідних для задоволення запиту. Отримувачем може бути будь-який клас.

Контрольний приклад

Завдання

Виконання домашньої роботи. У випадку, якщо учень сам не може розв'язати задачі, які йому задані додому, він просить рідних допомогти йому. В залежності від того, хто допомогає учневі, він отримує відповідну оцінку. За допомогою шаблону проектування реалізувати виконання домашньої роботи учнем та його сім'єю.

Розв'язок

```
namespace CommandExample
{
    class Program
    {
        static void Main(string[] args)
        {
            Mother m = new Mother();
            Father f = new Father();
            Granny g = new Granny();


            Pupil[] klass = new Pupil[3];
            klass[0] = new Pupil();
            klass[1] = new Pupil();
            klass[2] = new Pupil();

            Task[] tasks = new Task[3];
            tasks[0] = new Task();
            tasks[1] = new Task();
            tasks[2] = new Task();

            klass[0].SetExecutor(tasks[0], m);
            klass[1].SetExecutor(tasks[1], f);
            klass[2].SetExecutor(tasks[2], g);

            for (int i=0;i<3;i++)
                klass[i].ExecuteTaskByHimself(tasks[i]);

            Console.ReadKey();
        }
    }

    class Pupil
    {
        public void SetExecutor(Task t, Executor e)
        {
            t._Executor = e;
        }
        public void ExecuteTaskByHimself(Task t)
        {
            t.Execute();
        }
    }
    class Task
    {
        protected Executor executor;
        public Executor _Executor
        {
            set
```

```
            {
                executor = value;
            }
        }

        public int Execute()
        {
            if (executor != null)
                return executor.DoHomeTask();
            else
                return 2;
        }
    }

    //receiver
    abstract class Executor
    {
        public abstract int DoHomeTask();
    }

    class Father : Executor
    {
        public override int DoHomeTask()
        {
            Console.WriteLine("Father has done task for 3 points");
            return 3;
        }
    }

    class Mother : Executor
    {
        public override int DoHomeTask()
        {
            Console.WriteLine("Mother has done task for 4 points");
            return 4;
        }
    }

    class Granny : Executor
    {
        public override int DoHomeTask()
        {
            Console.WriteLine("Granny has done task for 5 points");
            return 5;
        }
    }
}
/*Результат:
Mother has done task for 4 points
Father has done task for 3 points
Granny has done task for 5 points
*/
```

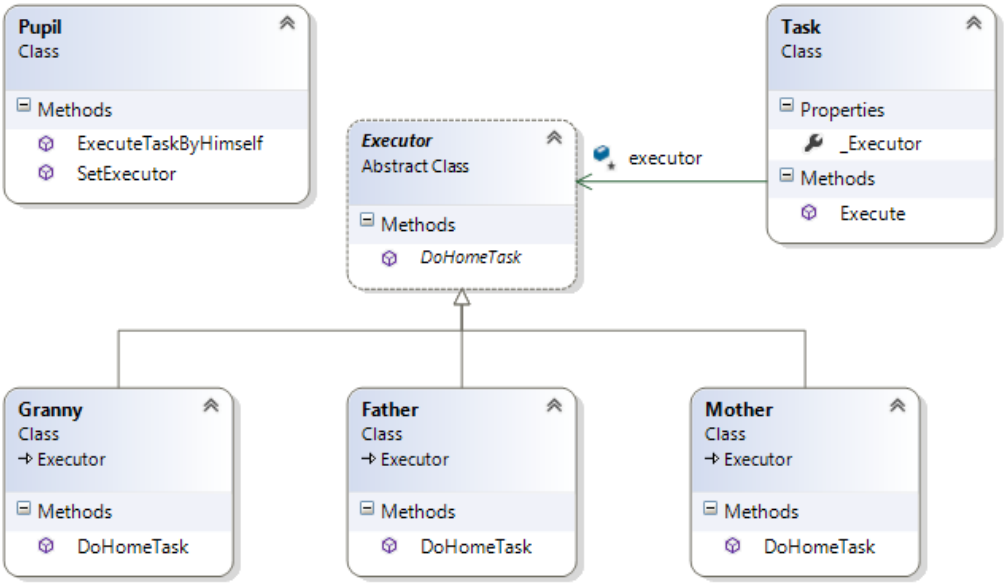

Рис. 32. Діаграма класів шаблону «Команда»

Посередник

Поведінковий шаблон, який надає **єдиний центр взаємодії** певної групи об’єктів, які повинні бути взаємнопов’язаними між собою. Визначає об'єкт, що інкапсулює спосіб взаємодії множини об'єктів.

Поведінковий шаблон «*Посередник*» потрібно використовувати, коли:

– потрібно послабити зв’язність системи, позбавляючи об'єкти від необхідності явно посилатися один на одного і дозволяючи тим самим незалежно змінювати взаємодії між ними;
– існують об'єкти, зв'язки між котрими досить складні та чітко задані. Отримані при цьому залежності не структуровані та важкі для розуміння;
– не можна повторно використовувати об'єкт, оскільки він обмінюється інформацією з багатьма іншими об'єктами;
– поведінка, розподілена між кількома класами, повинна піддаватися налагодженню без створювання множини підкласів.

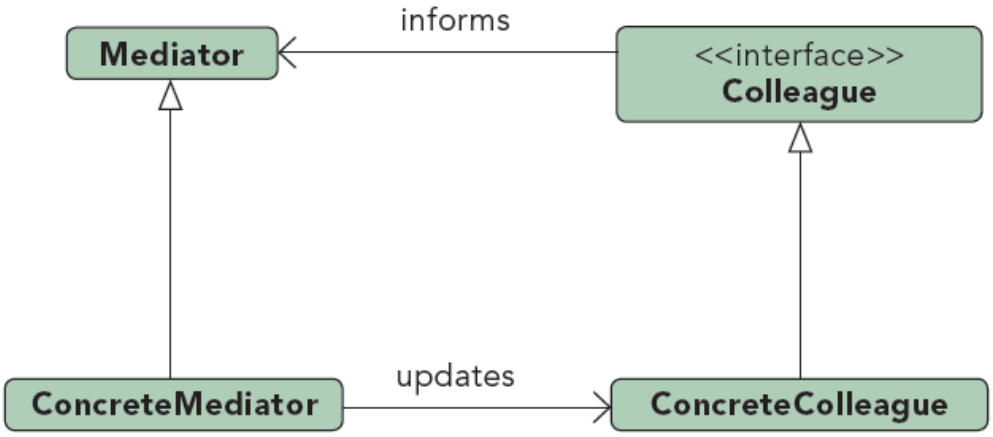

Рис. 33. Структурна схема шаблону «Посередник»

Учасники шаблону:

– **Mediator** – визначає інтерфейс для обміну інформацією з об'єктами Colleague;
– **ConcreteMediator** – реалізує кооперативне поведінку, координуючи дії об'єктів Colleague; володіє інформацією про колег і підраховує їх;
– Класи **Colleague** – кожен клас Colleague «знає» про свій об'єкті Mediator; всі колеги обмінюються інформацією лише з посередником, так як за його відсутності їм довелося б спілкуватися між собою безпосередньо.

Колеги посилають запити посереднику та отримують повідомлення від нього. Посередник реалізує кооперативну поведінку шляхом переадресації кожного запиту відповідному колезі (або декільком колегам).

Контрольний приклад

Завдання

За допомогою шаблону проектування реалізувати чат, що є центральним узлом, через який відбувається спілкування користувачів.

Розв'язок

```
namespace MediatorExample
{
  class MainApp
  {
    static void Main()
    {
      // Create chatroom
      Chatroom chatroom = new Chatroom();

      // Create participants and register them
      Participant George = new Beatle("George");
      Participant Paul = new Beatle("Paul");
      Participant Ringo = new Beatle("Ringo");
      Participant John = new Beatle("John");
      Participant Yoko = new NonBeatle("Yoko");

      chatroom.Register(George);
      chatroom.Register(Paul);
      chatroom.Register(Ringo);
      chatroom.Register(John);
      chatroom.Register(Yoko);

      // Chatting participants
      Yoko.Send("John", "Hi John!");
      Paul.Send("Ringo", "All you need is love");
      Ringo.Send("George", "My sweet Lord");
      Paul.Send("John", "Can't buy me love");
      John.Send("Yoko", "My sweet love");

      // Wait for user
      Console.ReadKey();
    }
  }

  // The 'Mediator' abstract class
  abstract class AbstractChatroom
  {
    public abstract void Register(Participant participant);
    public abstract void Send(
      string from, string to, string message);
  }

  // The 'ConcreteMediator' class
  class Chatroom : AbstractChatroom
  {
    private Dictionary<string,Participant> _participants =
      new Dictionary<string,Participant>();

    public override void Register(Participant participant)
    {
      if (!_participants.ContainsValue(participant))
      {
        _participants[participant.Name] = participant;
      }

      participant.Chatroom = this;
    }

    public override void Send(
      string from, string to, string message)
    {
      Participant participant = _participants[to];

      if (participant != null)
      {
        participant.Receive(from, message);
      }
    }
  }
```

```
  // The 'AbstractColleague' class
  class Participant
  {
    private Chatroom _chatroom;
    private string _name;

    // Constructor
    public Participant(string name)
    {
      this._name = name;
    }

    // Gets participant name
    public string Name
    {
      get { return _name; }
    }

    // Gets chatroom
    public Chatroom Chatroom
    {
      set { _chatroom = value; }
      get { return _chatroom; }
    }

    // Sends message to given participant
    public void Send(string to, string message)
    {
      _chatroom.Send(_name, to, message);
    }

    // Receives message from given participant
    public virtual void Receive(
      string from, string message)
    {
      Console.WriteLine("{0} to {1}: '{2}'",
        from, Name, message);
    }
  }

  // A 'ConcreteColleague' class
  class Beatle : Participant
  {
    // Constructor
    public Beatle(string name)
      : base(name)
    {
    }

    public override void Receive(string from, string message)
    {
      Console.Write("To a Beatle: ");
      base.Receive(from, message);
    }
  }

  // A 'ConcreteColleague' class
  {
    // Constructor
    public NonBeatle(string name)
      : base(name)
    {
    }

    public override void Receive(string from, string message)
    {
      Console.Write("To a non-Beatle: ");
      base.Receive(from, message);
    }
  }
}
```

```
/*Результат:
To a Beatle: Yoko to John: 'Hi John!'
To a Beatle: Paul to Ringo: 'All you need is love'
To a Beatle: Ringo to George: 'My sweet Lord'
To a Beatle: Paul to John: 'Can't buy me love'
To a non-Beatle: John to Yoko: 'My sweet love'
*/
```

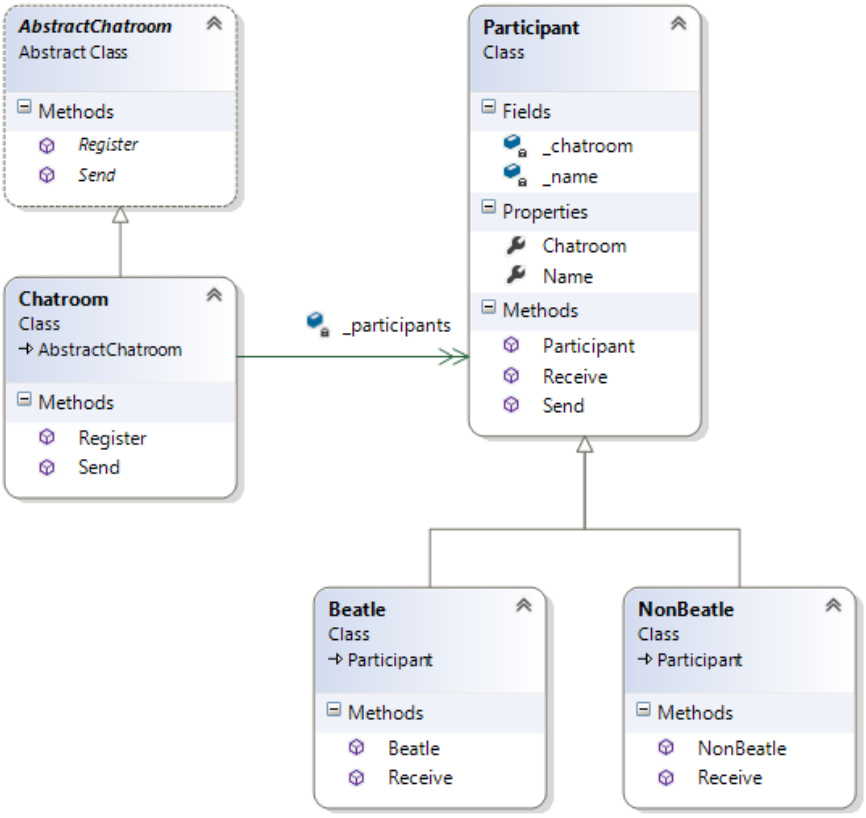

Рис. 34. Діаграма класів шаблону «Посередник»

Індивідуальні завдання

Варіант 1

1) За допомогою шаблону проектування змоделювати роботу операторів call- центру. На дзвінок клієнта спочатку відповідає пересічний оператор. Якщо він не володіє інформацією, яка цікавить клієнта, він переадресовує дзвінок на старшого оператора, який у свою чергу може переключити дзвінок клієнта на начальника відділу.
2) Виконання домашшколньої роботи. Вчитель на уроці визначає задачі, які необхідно розв’язати вдома учням. Учні просять допомоги у рідних. В залежності від того, хто з членів родини буде допомагати учню,

завдання буде виконано з різною успішністю. За допомогою шаблону проектування реалізувати виконання завдання обраним учнем членом родини.

Варіант 2

1) За допомогою шаблону проектування змоделювати роботу викладачів, які перевіряють екзаменаційні роботи студентів з певного предмету. Кожна робота перевіряється тільки одним викладачем. Якщо перший викладач зайнятий перевіркою роботи, вона передається для перевірки другому і т.д. У разі, якщо були зайняті всі викладачі і екзаменаційна робота залишилась неперевіреною, вона надходить на перевірку повторно до тих пір, поки її не перевірить якийсь викладач.
2) Для роботи в інформаційній системі користувач повинен заповнити додаткову інформацію про себе у своєму профілі в системі. Система пропонує ввести дані щодо ПІБ та дати народження користувача. Крім того, система надає можливість відмінити збереження цих даних. За допомогою шаблону проектування реалізувати поведінку такої системи.

Варіант 3

1) За допомогою шаблону проектування змоделювати роботу викладачів, які перевіряють екзаменаційні роботи студентів з декількох предметів. Кожен викладач відповідає тільки за перевірку одного завдання у роботі. Таким чином, перевіривши завдання з математики, математик передає роботу на перевірку фізику, той – викладачу з програмування тощо. Робота вважається перевіреною, якщо у ній виставлені бали з усіх завдань.
2) Перш, ніж свердлити стіну, робітник обирає свердло, яким він це

збирається робити, вставляє його у дриль та після цього вмикає пристрій. В залежності від того, яке свердло працює, робота може бути виконана за різний проміжок часу. За допомогою шаблону проектування змоделювати процес свердлення стіни робітником з дрилем.

Варіант 4

1) На певному заводі цілодобово працюють 3 зміни. І зміна – з 6.00 до 15.00, ІІ зміна – з 14.00 до 23.00 та ІІІ зміна – з 22.00 до 7.00. Якщо завдання робітникам певної зміни надходить менш, ніж за годину до закінчення зміни, вони передають його на виконання наступній зміні. За допомогою шаблону проектування реалізувати механізм обробки завдань робітниками описаного заводу.
2) Для роботи з даними у інформаційній системі передбачена можливість виконання таких дій над елементами масиву записів: «додати», «видалити», «редагувати». Для ініціювання виконання кожної з цих команд передбачений окремий пункт меню з відповідною назвою. За допомогою шаблону проектування реалізувати обробку зазначених вище команд у порядку, визначеному користувачем.

Варіант 5

1) Під час надходження замовлення на кухню ресторану його послідовно обробляють декілька кухарів. Наприклад, один кухар бере на себе відповідальність за приготування салату та передає замовлення наступному кухарю, який може відповідати за приготування м’ясної страви тощо. Таким чином, замовлення обходить всіх кухарів, які з нього вибирають ту частину блюд, яка лежить у зоні їх

відповідальності. За допомогою шаблону проектування реалізувати механізм обробки замовлення кухарями ресторану.

2) Програмні засоби віддаленої роботи з певною інформаційною системою надають можливість зайти до іншого клієнта системи та, ввівши потрібний пароль, віддавати команди на виконання іншому клієнту, запущеному на іншій машині. За допомогою шаблону проектування реалізувати механізм пересилання команд на виконання до іншого клієнта (Необов'язково створювати мережеву версію програми. Достатньо імітувати роботу двох клієнтів одразу).

Варіант 6

1) У автоматі з приготування гарячих напоїв (чай, кава тощо) є три баки з водою. На початку роботи автомату для обробки запиту клієнту використовується вода з першого баку. Якщо для виконання замовлення води, яка залишилася у баку, недостатньо, відбувається перемикання автомату на наступний бак. У випадку, якщо не залишилося води навіть у третьому баку, на екран автомата виводиться повідомлення про неможливість виконання замовлення клієнта. За допомогою шаблону проектування реалізувати роботу такого автомату.
2) Під час наукових досліджень зірку по небу ведуть декілька обсерваторій (адже у певний проміжок доби її краще видно у якомусь одному місці планети). В залежності від часу Центр спостереження за зірками передає контроль над зіркою від однієї обсерваторії до іншої. За допомогою шаблону проектування реалізувати механізм спостереження за зіркою, якщо відомо, що у Центра спостереження є тільки одна команда «Почати спостереження» і можливість змінювати виконавців даної команди.

Варіант 7

1) Деякий галузевий концерн об’єднує декілька фабрик з виробництва устаткування для підприємств важкої промисловості. Замовлення на виготовлення продукції надходять з вказівкою, яка саме фабрика бажано повинна його виконати. У разі, якщо вказаній фабриці не вистачає ресурсів для виконання замовлення, вона передає замовлення іншій фабриці концерну. У випадку, якщо знайдено фабрику, яка може виконати замовлення, у клієнта перепитують, чи погоджується він спрямувати замовлення іншій фабриці. У випадку, якщо жодна фабрика не має ресурсів для виконання замовлення, про це сповіщається клієнт. За допомогою шаблону проектування змоделювати обробку замовлення концерном.
2) User Component містить текстове поле для введення тексту та три кнопки для його збереження до файлу. У відповідність кожній кнопці поставлений окремий спосіб збереження тексту. Перший спосіб – звичайний: зберігаємо текст без внесення жодних змін. Другий спосіб – видаляємо всі зайві пробіли з тексту перед збереженням. Третій спосіб – застосовуємо кодування тексту (або архівацію). За допомогою шаблону проектування реалізувати описаний User Component, якщо відомо, що метод «Зберегти у файл» у нього один (немає жодних override варіантів методу).

Варіант 8

1) При покупці коштовної речі у магазині покупець виявив, що не має достатньої суми грошей і він повинен вирішити чи позичити гроші, щоб заплатити за покупку одразу, чи брати її у кредит. Він телефонує своєму другу з проханням позичити суму, якої не вистачає. У випадку, якщо

друг має необхідну суму і може її позичити, покупець оформлює покупку одразу. Якщо друг не має потрібної суми, він може в свою чергу зателефонувати комусь зі своїх друзів з аналогічним проханням. У випадку, якщо суми не знайдено, покупець оформлює покупку у кредит. За допомогою шаблону проектування змоделювати описану ситуацію.

2) Для продажу/купівлі квартири люди звертаються до агентств нерухомості, де у випадку продажу квартиру реєструють у базі квартир та допомагають знайти на неї покупця, у випадку ж купівлі – реєструють у базі покупців, фіксують побажання покупця до квартири та підбирають множину квартир, які відповідають даним вимогам. За допомогою шаблона проектування змоделювати процес продажу/купівлі квартир за участі безпосередніх продавців та покупців, а також агентства нерухомості. При підборі пропозицій квартир агентство нерухомості може звертатися по додаткову інформацію до інших подібних агентств.

Варіант 9

1) Медичний огляд. Пацієнт звертається з певними скаргами до терапевта. Терапевт може поставити діагноз самостійно, якщо картина хвороби є для нього ясною. У випадку ж, якщо терапевт не може поставити діагноз, він направляє хворого на обстеження до профільного лікаря (окуліста, хірурга, невропатолога тощо). Якщо профільний лікар також не може поставити діагноз, він може рекомендувати пацієнту звернутися до іншого лікаря такої ж спеціалізації, але вищої кваліфікації, або пройти додаткове обстеження у іншому медичному закладі (у іншого лікаря). За допомогою шаблону проектування

змоделювати проходження пацієнтом медичного огляду за умови, що хтось з лікарів обов’язково ставить діагноз.

2) Для формування команди спеціалістів на новий проект компанія звертається до рекрутингової агенції та надає інформацію щодо вимог до кандидатів на вакантні посади. Рекрутингова агенція, в свою чергу, має певну базу спеціалістів з інформацією про їх професійні якості та бажані позиції, з якої і вибирає кандидатів на задані посади. Рекрутингова агенція може звертатися до іншої подібної агенції з метою проведення пошуку кандидатів чи вакансій в базі даних останньої. Іноді спеціалісти за власною ініціативою відправляють відомості про себе до банка резюме рекрутингової компанії, щоб та повідомляла про нові вакансії, які можуть бути для них цікавими. За допомогою шаблона проектування змоделювати процес взаємодії фахівців та компаній-замовників через рекрутингову агенцію.

Варіант 10

1) Виконання домашньої роботи. У випадку, якщо учень сам не може розв’язати задачі, які йому задані додому, він просить маму допомогти йому. Мама або розв’язує задачі, або просить допомогти батька і т.д. У випадку, якщо жоден член сім’ї не зміг розв’язати задачі, учень отримує оцінку «незадовільно» за домашню роботу. За допомогою шаблону проектування реалізувати виконання домашньої роботи учнем та його сім’єю.
2) Головна форма інформаційної системи відображається по-різному в залежності від прав користувача системи (simple user, advanced user, administrator). Наприклад, для simple user деякі поля введення текстових даних (textbox) та кнопки (buttons) повинні бути невидимими та

недоступними, для advanced user ці елементи управління мають бути видимими, але все одно недоступними, і тільки administrator може у повній мірі використовувати ці текстові поля та кнопки. За допомогою шаблона проектування забезпечити виокремлення механізму відображення головної форми системи та інкапсуляцію її в одному об’єкті.

Варіант 11

1) Міжбібліотечний абонемент (МБА). У разі, якщо читач не може знайти книжку у певній бібліотеці, він може замовити її по МБА у іншій бібліотеці, яка є партнером даної. Якщо цієї книги немає і в іншій бібліотеці, запит може бути спрямований другою бібліотекою до якоїсь третьої. Якщо жодна з бібліотек не має потрібної книги, читачу буде про це сповіщено. У протилежному випадку, він отримає цю книгу. За допомогою шаблону проектування реалізувати механізм пошуку книги у бібліотеках.
2) Для того, щоб отримати товари для продажу, супермаркет не звертається напряму до виробників, а направляє відповідний запит постачальнику товарів, який має координати виробників товарів та допомагає супермаркету замовити товари, які характеризуються потрібною якістю та ціною. За допомогою шаблона проектування змоделювати процес надходження товарів на прилавки супермаркету.

Варіант 12

1) У випадку, якщо жінка, яка завжди купує хліб до вечері після роботи, не може цього зробити, вона просить свого чоловіка купити хліб замість неї. Чоловік може або купити хліб, якщо у нього є така можливість, або

просить це зробити свого сина і т.д. Якщо жоден з членів родини не може купити хліб, останній, до кого звернулися з таким проханням, замовляє хліб з доставкою з супермаркету. За допомогою шаблону проектування реалізувати алгоритм покупки хліба у родині.

2) Сайт знайомств. Користувач може розмістити свою анкету в базі даних сайту, а також здійснювати там пошук людей за певними характеристиками (вік, стать, місто, хобі тощо). У разі знаходження людей, які відповідають пошуковому запиту користувача, він отримує список цих людей, а обрані користувачі отримують повідомлення про те, хто цікавився їхніми даними. За допомогою шаблона проектування реалізувати механізм роботи сайту знайомств.

Варіант 13

1) Білети у театр не підлягають поверненню у касу. Тому у випадку, якщо у людини немає можливості піти на виставу, вона повинна намагатися перепродати квитки самостійно знайомим та друзям, або віддати рідним. У свою чергу рідні, друзі та знайомі людини можуть спробувати знайти бажаючих відвідати виставу серед своїх власних рідних, друзів та знайомих. У випадку, якщо бажаючі піти на виставу не знайдені, квитки пропадають і за них не отримують грошей. В іншому випадку продавець білетів повертає собі гроші. За допомогою шаблону проектування змоделювати ситуацію з продажем квитків до театру.
2) При спробі увійти до системи можливі 2 варіанти розвитку подій: система дає дозвіл на вхід або доступ до системи є закритим для даного користувача. У випадку, коли доступ дозволений, відбувається налаштування головної форми системи: виконується виведення певних списків даних на форму, розблокуються кнопки та меню. У випадку,

коли доступ заборонений, відбувається очищення списків, які відображаються на головній формі, а також блокується натиснення кнопок та робота меню. За допомогою шаблону проектування забезпечити виокремлення механізму відображення головної форми системи та інкапсуляцію її в одному об’єкті.

Варіант 14

1) За допомогою шаблону проектування реалізувати алгоритм визначення купюр для видачі замовленої суми в банкоматі. У випадку, якщо не всю бажану суму можна видати великими купюрами (наприклад, по 200 грн.), залишок, який є менше 200 грн., буде виданий купюрами меншого номіналу. У випадку, якщо неможливо визначити необхідну комбінацію купюр, банкомат видає про це відповідне повідомлення та просить перевизначити іншу суму з урахуванням наявних купюр в банкоматі.
2) До обов’язків секретаря компанії входить організація мітингів різних видів, від загальних зібрань всіх робітників компанії до термінових нарад начальників відділів з керівником компанії. В залежності від наказу секретар телефонує необхідним учасникам мітингу, сповіщає їх про зустріч та отримує підтвердження їх участі у зустрічі. Список співробітників, які підтвердили свою участь у мітингу, секретар передає керівнику компанії. За допомогою шаблона проектування реалізувати механізм організації мітингів у компанії.

Варіант 15

1) За допомогою шаблону проектування змоделювати витягування ріпки з землі у порядку, описаному у відомій народній казці «Ріпка».
2) Фірма, яка займається зборкою комп’ютерів. Клієнт звертається до

фірми із замовленням, у якому описує, яку конфігурацію комп’ютера він бажає отримати. В залежності від побажань клієнта фірма звертається до своїх постачальників, які продають комплектуючі з різними характеристиками (різною якістю, різними цінами), та з отриманих комплектуючих збирає комп’ютер для клієнта. За допомогою шаблона проектування реалізувати описаний механізм замовлення комп’ютера клієнтом.

Варіант 16

1) Текстовий редактор. З метою підвищення надійності редактора всі дії, які виконуються над текстом, протоколюються (записуються до history), щоб їх можна було виконати повторно після аварійної зупинки системи. За допомогою шаблону проектування реалізувати можливість ведення history. Продемонструйте роботу програми шляхом повторного автоматичного внесення змін у текст, який був відредагований, але не збережений, перед збоєм редактора.
2) Продюсерський центр, що співпрацює з багатьма артистами, забезпечує проведення свят різних типів: від невеликих концертів у клубах на дні народження (чи іншому святі) замовника до великих концертів, присвячених державним святам, що проводяться на головних концертних майданчиках країни. У замовленні клієнт вказує свої побажання щодо артистів, які повинні брати участь, місця проведення події тощо. Продюсерський центр обробляє дану інформацію та формує концертну програму для замовника, отримуючи згоду на участь у концерті від артистів. За допомогою шаблону проектування реалізувати механізм організації концерту за участі продюсерського центу.

Варіант 17

1) Вікно для здійснення перекладу тексту з однієї мови на іншу має текстове поле для введення вихідного тексту, combobox для вибору мови, на яку необхідно робити переклад, та кнопку «Перекласти». В залежності від того, яка мова вибрана на формі, при натисненні на кнопку підключається потрібний модуль перекладу, який і здійснює переклад тексту. За допомогою шаблону проектування реалізувати описану форму перекладу.
2) Диспетчерська вежа в аеропорті. Пілоти літаків, які вилітають або приземляються, спілкуються з диспетчерами, а не один з одним, щодо своїх дій з управління літаком. Дозвіл на виліт та приземлення видає вежа. За допомогою шаблона проектування реалізувати механізм взаємодії пілотів з диспетчерами аеропорту.

Варіант 18

1) В ЖЕК надходять заявки від мешканців будинку на виклик різних спеціалістів додому. Диспетчер приймає дані заявки та викликає потрібних майстрів для виконання цих заявок. За допомогою шаблону проектування реалізувати даний механізм роботи ЖЕКу.
2) Поштова розсилка. Кожний учасник поштової розсилки не знає переліку всіх людей, які на неї підписалися. Розсилка є єдиною точкою доступу, через яку людина може відправляти інформацію певному списку адресатів. Поштова система отримує повідомлення та самостійно пересилає його потрібним адресатам. За допомогою шаблона проектування реалізувати механізм поштової розсилки повідомлень.

Варіант 19

1) У період, коли одна з клієнтських програм філіалу банку знаходиться в оффлайні, сервер головного офісу генерує список задач, які буде повинна виконати програма, коли вона з’явиться в он-лайні. Наприклад, роздрукувати виписки щодо руху коштів за вказаними рахунками, обновити курси валют на поточний день тощо. За допомогою шаблону проектування реалізувати даний механізм накопичення задач та їх подальшого виконання на клієнтській програмі філіалу банку.
2) Аптеки не працюють напряму з виробниками ліків, а замовляють їх у фірм-постачальників. У замовлення аптеки як правило входять ліки, які виробляються різними фармацевтичними компаніями. Фірма-постачальник за свої послуги бере певний відсоток від вартості товару. За допомогою шаблону проектування реалізувати механізм замовлення ліків аптекою.

Варіант 20

1) За допомогою шаблону проектування реалізувати роботу зі сценарієм обробки текстових даних. Сценарій складається з окремих команд, в яких зберігається посилання на модуль системи, який буде виконувати дану команду (наприклад, орфокоректор, архіватор тощо). Забезпечити збереження сценарію в окремому файлі.
2) У заможних людей, які володіють великими маєтками, зазвичай, є керуючі справами, які відповідають за облаштування побуту власника у цьому маєтку (закупівля товарів, приготування їжі, облаштування території маєтку, прибирання у домі тощо). Тому, коли власнику маєтку щось потрібно, він не звертається напряму до фермерів по продукти чи садівника по квіти: він навіть може не знати, хто саме у нього працює. Натомість він ставить задачу керуючому, який має всю необхідну

інформацію щодо маєтку та людей, які в ньому працюють. За допомогою шаблону проектування змоделювати механізм взаємодії власника маєтку та людей, які його обслуговують.

Варіант 21

1) Під час виконання дзвінка у велику компанію виклик приймається АТС, автоматичний диктор розповідає абоненту, які цифри треба натиснути на телефоні, щоб додзвонитися до потрібного відділу компанії, який зможе обробити запит абонента. Узагальнений алгоритм обробки дзвінків виглядає таким чином: прийняття дзвінка абонента (зняття трубки АТС), вибір абонентом потрібного відділу та виконання дії «обробити запит абонента», яка виконується по-різному в залежності від того, який відділ був обраний абонентом для цього. За допомогою шаблона проектування реалізувати механізм обробки дзвінків до компанії.
2) На розширеному засіданні студентської ради за участі старост всіх груп факультету заступник декана з навчально-виховної роботи сповістив всіх про необхідність подання документів на отримання соціальних чорнобильських виплат у заданий термін. При цьому заступник декана не знає повного переліку осіб, які є такими, що постраждали внаслідок Чорнобильської трагедії. Натомість такими даними володіють старости груп. Таким чином, старости складають потрібні списки студентів та сповіщають їх про необхідність надати певні документи у визначений термін. За допомогою шаблону проектування реалізувати описаний механізм сповіщення студентів та надання ними документів для отримання чорнобильських виплат.

Варіант 22

1) Для перескладання іспиту студент пише заяву на ім’я декана факультету. Декан в свою чергу складає комісію, яка повинна приймати даний екзамен. В залежності від складу комісії студент може скласти іспит на різні оцінки. За допомогою шаблону проектування реалізувати виконання дії «Прийняти іспит» різними комісіями.
2) Інтернет-магазин пропонує покупцям широкий вибір товарів, які він замовляє у виробників або на складах. Покупці не працюють з виробниками чи складами напряму, а звертаються до інтернет-магазину з замовленням певної кількості товарів. У разі, якщо даний товар є доступним на складі магазину, він одразу продається покупцеві. У разі, якщо товару немає, магазин терміново зв’язується із потрібним виробником чи складом та домовляється про доставку потрібного товару. Коли товар, на який чекає покупець, надходить на склад інтернет-магазину, останній сповіщає про це покупця. За допомогою шаблону проектування змоделювати механізм роботи інтернет-магазину.

Варіант 23

1) За допомогою шаблону проектування записати перелік обстежень, які повинен пройти студент під час загального медогляду, та реалізувати їх у потрібний момент. У перелік входить відвідування різних лікарів та здача визначеної множини аналізів. За кожний огляд чи обстеження відповідає окремий лікар. Кожне обстеження завершується видачею висновку лікаря щодо працездатності студента. Передбачити можливість змінювати обстеження, які входять до переліку, а також редагувати, які лікарі відповідають за те чи інше обстеження.
2) Виконання домашньої роботи. У випадку, якщо учень сам не може

розв’язати задачі, які йому задані додому, він просить маму допомогти йому. Мама або розв’язує задачі, або просить допомогти батька і т.д. У випадку, якщо жоден член сім’ї не зміг розв’язати задачі, учень отримує оцінку «незадовільно» за домашню роботу. За допомогою шаблону проектування реалізувати виконання домашньої роботи учнем та його сім’єю.

Варіант 24

1) У автоматі з приготування гарячих напоїв (чай, кава тощо) є три баки з водою. На початку роботи автомату для обробки запиту клієнту використовується вода з першого баку. Якщо для виконання замовлення води, яка залишилася у баку, недостатньо, відбувається перемикання автомату на наступний бак. У випадку, якщо не залишилося води навіть у третьому баку, на екран автомата виводиться повідомлення про неможливість виконання замовлення клієнта. За допомогою шаблону проектування реалізувати роботу такого автомату.
2) При спробі увійти до системи можливі 2 варіанти розвитку подій: система дає дозвіл на вхід або доступ до системи є закритим для даного користувача. У випадку, коли доступ дозволений, відбувається налаштування головної форми системи: виконується виведення певних списків даних на форму, розблокуються кнопки та меню. У випадку, коли доступ заборонений, відбувається очищення списків, які відображаються на головній формі, а також блокується натиснення кнопок та робота меню. За допомогою шаблону проектування забезпечити виокремлення механізму відображення головної форми системи та інкапсуляцію її в одному об’єкті.

Варіант 25

1) У період, коли одна з клієнтських програм філіалу банку знаходиться в оффлайні, сервер головного офісу генерує список задач, які буде повинна виконати програма, коли вона з’явиться в он-лайні. Наприклад, роздрукувати виписки щодо руху коштів за вказаними рахунками, обновити курси валют на поточний день тощо. За допомогою шаблону проектування реалізувати даний механізм накопичення задач та їх подальшого виконання на клієнтській програмі філіалу банку.
2) Аптеки не працюють напряму з виробниками ліків, а замовляють їх у фірм-постачальників. У замовлення аптеки як правило входять ліки, які виробляються різними фармацевтичними компаніями. Фірма-постачальник за свої послуги бере певний відсоток від вартості товару. За допомогою шаблону проектування реалізувати механізм замовлення ліків аптекою.

Контрольні запитання

1. Дайте визначення шаблону «Ланцюжок обов’язків» («Команда», «Посередник»).
2. У чому полягає призначення шаблону «Ланцюжок обов’язків» («Команда», «Посередник»)?
3. Назвіть переваги та недоліки застосування шаблону «Ланцюжок обов’язків» («Команда», «Посередник»).
4. Наведіть UML-діаграму класів, які утворюють структуру шаблона «Ланцюжок обов’язків» («Команда», «Посередник»).

3. СПИСОК ВИКОРИСТАНОЇ ТА РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Методичні рекомендації щодо виконання курсових робіт для студентів, які навчаються за освітньо-професійною програмою підготовки спеціаліста за напрямом підготовки 0601 Право, спеціальністю 6.060101 Правознавство денної та заочної форм навчання [Текст] / уклад. Ю.В. Білоусов, Я.С. Бляхарський, Д.Л. Виговський [та ін.]. – Хмельницький : Хмельницький університет управління та права, 2009. – 21 с.
2. Положення про організацію навчального процесу в НТУУ “КПІ” [Текст] / уклад. Г.Б. Варламов, В.П. Головенкін, В.І. Тимофєєв, В.І. Шеховцов; за заг. ред. Ю.І. Якименка. – К. : ІВЦ “Видавництво «Політехніка»”, 2004. – 72 с.
3. Заболотня Т.М. Програма навчальної дисципліни «Об’єктно-орієнтоване програмування» для студентів факультету прикладної математики (напряму підготовки 6.050103 «Програмна інженерія»). // К.: НТУУ «КПІ», 2014. - 7с.
4. Заболотня Т.М. Робоча програма кредитного модуля «Шаблони проектування» дисципліни «Об’єктно-орієнтоване програмуванн» для студентів факультету прикладної математики (напряму підготовки 6.050103 «Програмна інженерія»). // К.: НТУУ «КПІ», 2014. - 12с.
5. Гамма, Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования [Текст] / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб : Питер, 2009. – 366 с.

6. Леоненков, А.В. Самоучитель UML [Текст] / А.В. Леоненков. – СПб.: БХВ, 2007. – 2-е изд. – 576 с.
7. Буч, Г. UML. Классика CS [Текст] / Г. Буч, А. Якобсон, Дж. Рамбо; пер.с англ.; под общей редакцией проф. С. Орлова. – СПб. : Питер, 2006. – 2-е изд. – 736 с.
8. Bishop, J. C# 3.0 Design Patterns [Text] / Judith Bishop. – O'Reilly, 2008. – 290 p. – ISBN-10 0-596-52773-X; ISBN-13 978-0-596-52773-0.
9. Shalloway, Alan. Design Patterns Explained A New Perspective on Object-Oriented Design [Text] / Alan Shalloway, James R. Trott. – Addison Wesley Professional, 2004. – 2nd Edition. – 480 p. – ISBN 0-321-24714-0.
10. Павловская, Т.А. C#. Программирование на языке высокого уровня: Учебник [Текст] / Т.А. Павловская. – СПб. : Питер, 2009. – 432 с.

ДОДАТКИ

Додаток А

Зразок оформлення титульного аркуша лабораторної роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

ЛАБОРАТОРНА РОБОТА №1

з дисципліни «Об’єктно-орієнтоване програмування»

на тему

Реалізація структурних шаблонів проектування

Варіант №13

Виконав студент
ІІ курсу групи КП-ХХ
Іваненко Іван Іванович
залікова книжка КП-ХХХХ

Перевірив
доцент, к.т.н. Заболотня Т.М.

Оцінка

_____

(дата, підпис)

КИЇВ 2014

ЗМІСТ