**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
Харківський національний університет імені В. Н. Каразіна
Бахмутський навчально-науковий професійно-педагогічний інститут
Кафедра електромеханічних та комп'ютерних систем

До захисту допущено
**Завідувач кафедри**
_____ Інна НЕФЬОДОВА
(підпис) (ім'я, прізвище)
«07» грудня 2024 року

**КВАЛІФІКАЦІЙНА РОБОТА (ПРОЄКТ)**

рівень вищої освіти другий (магістерський)

спеціальність 015.39 Професійна освіта (Цифрові технології)

освітньо-професійна програма Професійна освіта. Комп'ютерні технології в управлінні та навчанні

тема «Професійна підготовка фахівців з цифрових технологій для викладання освітнього модулю «Структурні патерни проєктування» у закладах вищої освіти»

**Виконав(ла)**

здобувач(ка) групи БЗ-К23мг
(шифр групи)

Денис ГРИНАЦЬ _____
(ім'я, прізвище) (підпис)

**Керівник роботи**
к.т.н., доц. Павло ЧИКУНОВ _____
(науковий ступінь, вчене звання, ім'я, прізвище) (підпис)

**Рецензент роботи**
к.пед.н., доц. Наталія ЛОГІНОВА _____
(науковий ступінь, вчене звання, ім'я, прізвище) (підпис)

**Консультант**
д.пед.н., проф. Вікторія КУЛЕШОВА _____
(науковий ступінь, вчене звання, ім'я, прізвище) (підпис)

Засвідчую, що у цій роботі немає цитат та вилучень з праць інших авторів без відповідних посилань
здобувач (ка) _____
(підпис)

Харків – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В. Н. Каразіна

Факультет/ННІ Бахмутський навчально-науковий професійно-педагогічний інститут
Кафедра Електромеханічних та комп'ютерних систем
Рівень вищої освіти другий (магістерський)
Спеціальність 015.39 Професійна освіта (Цифрові технології)
Освітньо-професійна програма Професійна освіта. Комп'ютерні технології в управлінні та навчанні

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри**
(підпис) Інна НЕФЬОДОВА (ім'я, прізвище)
«08» жовтня 2024 року

# ЗАВДАННЯ
## НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЄКТ)

Гринаць Денис Романович
(прізвище, ім'я, по батькові здобувача)

1. Тема роботи Професійна підготовка фахівців з цифрових технологій для викладання освітнього модулю «Структурні патерни проєктування» у закладах вищої освіти
керівник роботи Чикунов Павло Олександрович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «08» жовтня 2024 року № 5101-5/3232

2. Строк подання здобувачем роботи «02» грудня 2024 р.

3. Перелік питань, які потрібно розробити: Актуальність професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Структурні патерни проєктування» у закладах вищої освіти. Характеристика об'єктів галузі: стан і стратегії розвитку. Вимоги до кадрового забезпечення об'єкту галузі. Методика професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Структурні патерни проєктування» у закладах вищої освіти.

4. План роботи

| № з/п | Назви етапів роботи |
|---|---|
| 1 | Огляд літературних джерел, нових розробок, опублікованих даних та іншої інформації, пов'язаної з темою роботи. |
| 2 | Дослідження теоретичних підходів до актуальності професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Структурні патерни проєктування» у закладах вищої освіти». |
| 3 | Характеристика об'єктів галузі: стан і стратегії розвитку. |
| 4 | Розробка методики професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Структурні патерни проєктування» у закладах вищої освіти. |
| 5 | Розробка вимог до кадрового забезпечення об'єкту галузі. |
| 6 | Оформлення першого варіанту тексту, подання його на ознайомлення науковому керівнику. |
| 7 | Усунення недоліків, написання остаточного варіанту тексту, оформлення дипломної роботи. |
| 8 | Подання роботи на кафедру, перевірка на плагіат та зовнішнє рецензування роботи. |
| 9 | Захист дипломної роботи у ЕК. |

5. Дата видачі завдання «08» жовтня 2024 р.

**Здобувач(ка)** _____ Денис ГРИНАЦЬ
(підпис) (ім'я, прізвище)

**Керівник роботи** _____ Павло ЧИКУНОВ
(підпис) (ім'я, прізвище)

# РЕФЕРАТ

Об'єктом дослідження роботи є процес професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Структурні патерни проєктування» у закладах вищої освіти.

Предметом дослідження роботи є методика професійної підготовки фахівців з цифрових технологій до розробки комплексу цифрових освітніх для викладання освітнього модуля «Структурні патерни проєктування».

Мета дослідження – теоретично обґрунтувати та частково перевірити методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Структурні патерни проєктування» у закладах вищої освіти.

Охарактеризовано систему професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Структурні патерни проєктування» у закладах вищої освіти.

Виконано аналіз характеристик об'єктів галузі проєктування програмного забезпечення. Виконана постановка 4 нових лабораторних робіт. Виконано аналіз вимог до кадрового забезпечення об'єкту ІТ-галузі в умовах цифрової трансформації суспільства.

Розроблено дидактичний проект консультативного заняття.

За основними результатами дослідження виконана публікація тези доповіді на VIII міжнародній науково-практичній конференції здобувачів вищої освіти та молодих учених «Студенти та молодь – для майбутнього країни» (м. Харків, 14-15 листопада 2024 р.).

Робота складається із вступу, чотирьох розділів, висновків, бібліографічного списку, що містить 46 джерел, 2 додатка, 5 таблиць, 18 рисунків.

ЦИФРОВІ ТЕХНОЛОГІЇ, ПРОФЕСІЙНА ПІДГОТОВКА, ЛАБОРАТОРНИЙ ПРАКТИКУМ, ПАТЕРНИ ПРОЄКТУВАННЯ, КАДРОВЕ ЗАБЕЗПЕЧЕННЯ, МЕТОДИЧНА РОЗРОБКА

# ABSTRACT

The object of the study is the process of professional training for specialists in digital technologies to teach the educational module "Structural Design Patterns" in higher education institutions.

The subject of the study is the methodology of professional training for specialists in digital technologies to develop a set of digital educational tools for teaching the educational module "Structural Design Patterns."

The purpose is to theoretically substantiate and partially test the methodology for the professional training of specialists in digital technologies to teach the educational module "Structural Design Patterns" in higher education institutions.

Key results:

- the system of professional training for specialists in digital technologies for teaching the "Structural Design Patterns" educational module in higher education institutions was characterized;

- an analysis of the characteristics of objects in the software design field was performed;

- four new laboratory works were developed;

- requirements for staffing in the IT industry under conditions of societal digital transformation were analyzed;

- a didactic project for a consultative lesson was developed.

The main results of the study were presented in a conference thesis at the VIII International Scientific and Practical Conference of Higher Education Students and Young Scientists "Students and Youth – For the Future of the Country" (Kharkiv, November 14-15, 2024).

Structure of the work: The work consists of an introduction, four chapters, conclusions, a bibliography of 46 sources, 2 appendices, 5 tables, and 18 figures.

DIGITAL TECHNOLOGIES, PROFESSIONAL TRAINING, LABORATORY PRACTICUM, DESIGN PATTERNS, STAFFING, METHODOLOGICAL DEVELOPMENT.

**ЗМІСТ**

# ВСТУП

Становлення і розвиток цифрового суспільства є першочерговою ознакою сьогодення. Саме зараз активно розвиваються цифрові технології, створюються умови для ефективного використання знань, необхідних у всіх сферах суспільного життя. На сучасному етапі розвитку освіти у закладах вищої освіти (ЗВО) відбувається стрімке використання цифрових технологій.

Отже, країні потрібні фахівці з цифрових технологій для викладання освітніх модулів. Саме такими фахівцями є інженери-педагоги зі спеціальності 015.39 Професійна освіта (Цифрові технології).

Науковці одностайні у думці, що розвиток і повсякденне, систематичне використання цифрових технологій створює умови для професійного удосконалення всіх суб'єктів ЗВО. Перспективним напрямом функціонування ЗВО є використання сучасних цифрових технологій у діяльності організаційно-навчальних підрозділів університету, які реалізують різноманітні функції щодо організації освітнього процесу, виконання основних статутних завдань ЗВО та посадових обов'язків.

Зазначене вимагає від випускників ЗВО відповідний розвиток цифрової компетентності.

Науковці акцентують на важливості розвитку цифрової компетентності фахівців ЗВО в Україні. Використання цифрових технологій є перспективним напрямом у функціонуванні вищих навчальних закладів (ЗВО), зокрема у сфері організації освітнього процесу та виконання статутних завдань. Розвиток цифрової компетентності вимагає від працівників постійного вдосконалення та адаптації до нових технологічних умов.

Потреба в адаптації освітніх програм до цифрових вимог зумовлює необхідність формування у здобувачів освіти навичок ефективного використання інформаційних технологій, а також здатності до постійного самоосвіти та розвитку.

У цьому контексті важливою є інтеграція інформаційно-цифрової

компетентності в навчальний процес, що дозволяє створити сучасне середовище для підготовки фахівців. Вона повинна охоплювати не тільки знання основних цифрових інструментів і платформ, але й уміння критично оцінювати інформацію, безпечно користуватися цифровими ресурсами, а також здійснювати творче вирішення проблем із застосуванням цифрових технологій.

Цифрова компетентність формує важливу основу для дослідження та застосування цифрових технологій в освіті. Це питання активно розробляється в працях українських і зарубіжних науковців, які зосереджуються на різних аспектах цієї проблеми.

У галузі філософії освіти, вивчення цифрової компетентності тісно пов’язане з формуванням сучасних підходів до навчання.

Науковці, такі як В. Андрущенко, І. Зязюн досліджували вплив цифрових технологій на концептуальні засади освіти, зокрема з огляду на еволюцію навчальних процесів у цифровому середовищі.

Реалізація компетентнісного підходу є важливим напрямом у підготовці фахівців у вищій освіті. С. Сисоєва, В. Ягупов акцентують увагу на розвитку цифрових компетентностей у контексті підготовки майбутніх професіоналів, які мають володіти не тільки технічними, а й соціальними та аналітичними вміннями для успішного використання цифрових технологій у своїй діяльності.

Розглядають застосування освітнього процесу до нових цифрових реалій О. Беспарточна, Р. Гуревич, Дж. Ендерсон, О. Коношевський, М. Хорн. Учені досліджують, як цифрові технології можуть впливати на когнітивні, емоційні та соціальні аспекти навчання.

О. Глазунова, Н. Морзе акцентують увагу на методичних аспектах використання цифрових інструментів у професійній діяльності, зокрема у підготовці майбутніх педагогів до використання технологій у своїй роботі.

Ці дослідження створюють комплексний підхід до розвитку цифрової компетентності, що охоплює різні аспекти освіти та підготовки фахівців у

умовах цифровізації.

Аналіз психолого-педагогічної літератури засвідчив, що у контексті наукових здобутків проблема проблема професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Структурні патерни проєктування) у закладах вищої освіти, а також вирішення низки суперечностей, а саме між:

− зростаючими вимогами суспільства до фахівців з цифрових технологій та реальним станом їх готовності до цього;

− сучасними викликами до якості професійної діяльності ЗВО та відсутністю відповідної системи роботи з розвитку цифрової компетентності.

Важливість вирішення зазначених суперечностей і зумовили вибір теми кваліфікаційної роботи.

Отже, актуальність, об’єктивна потреба закладів вищої освіти в пошуках нових підходів до змісту навчання в інженерно-педагогічному закладі вищої освіти, недостатня розробленість проблеми зумовили вибір теми дослідження: «Професійна підготовка фахівців з цифрових технологій для викладання освітнього модуля «Структурні патерни проєктування» у закладах вищої освіти».

Мета дослідження – теоретично обґрунтувати та експериментально перевірити методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Структурні патерни проєктування» у закладах вищої освіти.

Завдання дослідження:

1. Визначити ступінь актуальності проблеми професійної підготовки здобувачів освіти зі спеціальності 015.39 Професійна освіта (Цифрові технології).

2. Виконати аналіз характеристик об'єктів галузі, стану і стратегії розвитку.

3. Виконати розробку вимог до кадрового забезпечення об'єкту галузі.

4. Теоретично обґрунтувати, розробити перевірити методику

професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Структурні патерни проєктування» у закладах вищої освіти.

Об'єкт дослідження: процес професійної підготовки здобувачів освіти.

Предмет дослідження: методика професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Структурні патерни проєктування» у закладах вищої освіти.

Методи дослідження:

– загальнонаукові (аналіз, синтез, систематизація, зіставлення, узагальнення) з метою професійної підготовки фахівців з цифрових технологій.

– емпіричні (тестування, опитування);

– педагогічний експеримент з метою перевірки методики професійної підготовки фахівців з цифрових технологій.

# РОЗДІЛ 1 АКТУАЛЬНІСТЬ ПРОФЕСІЙНОЇ ПІДГОТОВКИ ФАХІВЦІВ З ЦИФРОВИХ ТЕХНОЛОГІЙ

Сьогодні багато вчених [38, с.67] відзначили, що сучасний світ переходить на новий рівень розвитку технологій, який отримав назву «діджиталізація», що є пріоритетним напрямком модернізації сучасної освіти, заміщаючи собою процес інформатизації. Продуктивне використання цифрових технологій в освіті, включення здобувачів освіти у самостійний пошук, підбір інформації, участь у формах проектної діяльності для майбутніх фахівців компетенцій XXI століття, включаючи інформаційні компетенції. Аналіз нормативних документів, дослідження в галузі цифровізації економіки, освіти дозволили виділити умови цифровізації освіти: цифрове покоління учнів; створення законодавчої бази для діджиталізації освіти; ресурсне забезпечення цифровізації освіти, в тому числі цифрового освітнього середовища освітньої організації; підготовка кадрового потенціалу цифрової освіти, що володіє інформаційно-компетентністю; використання цифрових інноваційних педагогічних технологій та освітньозначущих цифрових технологій. Підготовка інженерів-педагогів здійснюється у вищих навчальних закладах.

Основою стандартного інтерфейсу є професійна компетентність майбутнього інженера-педагога, включаючи інформаційну компетентність. Учені вклали в поняття «інформаційно-компетентність майбутнього інженера-педагога» різні характеристики: сукупність знань, умінь і навичок (досвід діяльності) [39, с. 113], особистісно-активна характеристика фахівця в галузі інженерно-педагогічної освіти [40, с.7], набір IT-компетенцій [41, с.7], готовність творчо використовувати IT у своїй професійній діяльності [42, с.56].

Сьогодні інформаційна компетентність як складова професійної компетентності включає цифрову грамотність, яка базується на рекомендаціях UNESCO ICT Competency Framework for Teachers [37, 45].

Цифрова грамотність – це здатність людини використовувати цифрові технології, комунікації або мережі для пошуку, оцінки, використання та створення інформації; розуміти і використовувати інформацію в різних форматах з широкого кола джерел; Ефективно виконувати завдання в цифровому середовищі. ICT CFT рекомендує використовувати сучасні освітні тренди: відкриті освітні ресурси, мобільні технології, віртуальну реальність та інші.

*Під інформаційною компетенцією студента інженерно-педагогічного інституту (БННППІ) ми будемо розуміти його цілісну особистісно-активну якість, що виявляється в уміннях на основі знань, умінь і досвіду діяльності, набутий в процесі навчання в Бахмутському навчально-науковому професійно-педагогічному інституті, вирішувати професійні проблеми за допомогою інформаційних технологій і на основі цифрової грамотності; високомотивоване застосування інформаційних технологій з урахуванням специфіки професійної сфери.*

На основі дослідження [38, с. 7], аналізу ОПП спеціальності 015.39 Професійна освіта (Цифрові технології) визначені складові інформаційної компетентності:

• мотиваційно-ціннісний;

• загальний,

• загально- педагогічний;

• предметно-педагогічний компоненти.

*Мотиваційно-ціннісна* складова визначає активність і потребу людини у використанні інформаційних технологій (IT), необхідність розуміння знань і набуття навичок у сфері IT-додатків. Компонент General User характеризує знання, навички роботи з сучасними інформаційно-комунікаційними та освітніми значними цифровими технологіями в інформаційно-освітньому середовищі (далі – IOS) освітньої організації.

*Загальний* компонент характеризує знання, навички роботи з сучасними інформаційними технологіями (IT).

*Загально-педагогічна складова* включає цифрову грамотність педагога й віддзеркалює підготовку до педагогічної діяльності та його постійне відображення в навколишньому середовищі відповідно до планування і організації освітньої діяльності.

*Предметно-педагогічна складова* визначає розширення та поглиблення сформованих знань, майбутніх навичок інженерів-педагогів з цифрових технологій з метою конкретної професійної педагогічної діяльності.

*Предметно-педагогічна складова* включає цифрову грамотність інженера-педагога та відображає підготовку до педагогічної діяльності в інформаційно-освітньому середовищі та її постійне відображення в цьому середовищі відповідно до планування та організації освітньої діяльності.

Визначено рівні формування ІТ-компетентності майбутніх інженерів-педагогів з цифрових технологій: репродуктивний, продуктивний, творчий.

Репродуктивний рівень характеризується: відтворенням раніше отриманих знань і навичок з цифрових технологій для застосування в загальних ситуаціях; застосування існуючих знань з цифрових технологій тощо.

Продуктивний рівень характеризується: недостатньою потребою в знаннях і навичках в цифрових технологіях; уміння застосовувати отриману інформацію в нестандартних ситуаціях і при вирішенні нестандартних завдань, у тому числі практичних, прикладних за консультативної підтримки викладача тощо.

Творчий рівень характеризується: усвідомленою потребою майбутнього інженера-педагога з цифрових технологій оволодіти ІТ та використовувати їх у професійній діяльності; уміння застосовувати доступні знання в галузі цифрових технологій при вирішенні нестандартних завдань в. Виділено критерії, показники сформованості інформаційної компетентності здобувачів освіти.

Розроблено та описано моделі, реалізація яких дозволяє забезпечити формування інформаційної компетентності здобувачів освіти БННППІ:

структурно-функціональна модель формування інформаційної компетентності здобувачів освіти як основи; додаткова - міждисциплінарна модель курсу, спрямована на формування інформаційної компетентності майбутніх інженерів-педагогів з цифрових технологій; моделі комплексів НЗ (навчальні завдання), НМЗ (навчально-методичні завдання), НПЗ (навчально-професійні завдання) як спеціальних дидактичних засобів формування ІТ-компетентності майбутніх інженерів-педагогів з цифрових технологій. Фундаментальна складова базується на концептуальній системі (професійно-педагогічних комплексів НЗ, НМЗ, НПЗ.

Основа завдання, представлена єдністю фундаментальних і професійних завдань.

Комплекс КЗ лежить в основі навчально-пізнавальної (або освітньої) діяльності, що спрямована на засвоєння змісту освіти.

Структурно-функціональна модель компетенції ІТ здобувачів освіти включає: теоретичний, цільовий, змістово-діяльнісний, організаційно-технологічний та оцінювально-результативний блоки.

Основи теоретичного блоку теоретичні засади формування інформаційної компетентності здобувачів освіти.

Основою цільового блоку є інтегрована мета, заснована на соціальному замовленні (освіта на основі сучасних стандартів), формуванні ІТ-компетентності здобувачів освіти інженерно-педагогічного ЗВО. Блок змістової дії включає компоненти компетентності ІТ (мотиваційна, ціннісно-орієнтована, загальнокористувацька, педагогічна та предметно-педагогічна) та етапи її формування (мотиваційна, оновлювальна, базова, узагальнююча).

Основою організаційного технологічного підрозділу є педагогічні умови формування ІКТ-компетентності здобувачів освіти, активні та інтерактивні методи, форми навчання, цифрові педагогічні, ІТ та освітні засоби в основі завдання лежить єдність фундаментальних і відповідно до розробленої моделі ефективність формування ІТ компетентності майбутніх інженерів-педагогів повинна забезпечуватися визначеними педагогічними умовами. Перша умова:

міждисциплінарна інтеграція як основа побудови дисципліни «Теорія та методологія використання ІТ у професійно-технічних навчальних закладах».

Друга умова: використання інтегрованого інформаційно-освітнього середовища Бахмутського навчально-наукового професійно-педагогічного інституту Харківського національного університету імені В.Н.Каразіна.

Третя умова: інтеграція формальної, неформальної та інформаційної освіти. Формальна освіта може бути надана, де студенти можуть працювати з новітнім обладнанням ІТ і застосовувати його в освітніх заходах з учнями професійно-технічних навчальних закладів.

Неформальна освіта може здійснюватися в позанавчальних заходах здобувачів освіти – участі в конкурсах і науково-практичних конференціях, в обраних чемпіонатах WorldSkills [46], в межах електронних навчальних курсів, вивчення яких є добровільним і незалежним від основного тренувального процесу.

**Висновки до розділу**

Виконані дослідження підтверджують, що цифровізація стає пріоритетним напрямком модернізації освіти, замінюючи інформатизацію. Вона сприяє розвитку компетенцій XXI століття, зокрема інформаційної грамотності, необхідної для майбутніх фахівців. Цифрова грамотність охоплює вміння шукати, оцінювати, створювати інформацію та ефективно використовувати цифрові технології. Згідно з рекомендаціями UNESCO ICT Competency Framework for Teachers, в освіті важливо впроваджувати відкриті ресурси, мобільні технології та віртуальну реальність.

У контексті підготовки інженерів-педагогів особлива увага приділяється формуванню інформаційної компетентності, яка поєднує знання, навички й досвід застосування ІТ для вирішення професійних завдань. Вона включає такі компоненти: мотиваційно-ціннісний, загальний, загально-педагогічний та

предметно-педагогічний. Залежно від рівня розвитку інформаційної компетентності виділяють репродуктивний, продуктивний і творчий рівні.

Ефективність формування ІТ-компетенції залежить від визначених педагогічних умов: міждисциплінарна інтеграція, використання інформаційно-освітнього середовища та поєднання формальної, неформальної й інформальної освіти. Інтегроване освітнє середовище сприяє використанню сучасних ІТ-рішень, а участь у конкурсах і конференціях мотивує до практичного застосування знань.

Розроблені моделі забезпечують поступове формування компетентності: структурно-функціональна модель, міждисциплінарна модель курсу та дидактичні комплекси навчальних завдань. Це дозволяє поєднати фундаментальні та професійні аспекти підготовки майбутніх фахівців. Зокрема, інтеграція формальної та позаформальної освіти сприяє підготовці до роботи в умовах швидко змінюваного цифрового середовища. Таким чином, сучасні підходи до освіти забезпечують високий рівень підготовки інженерів-педагогів з цифрових технологій.

# РОЗДІЛ 2 ХАРАКТЕРИСТИКА ОБ'ЄКТІВ ГАЛУЗІ: СТАН І СТРАТЕГІЇ РОЗВИТКУ

## 2.1 Огляд структурних патернів проєктування

Метою розробки лабораторного практикуму «Структурні патерни проєктування» є ознайомлення студентів цифрових технологій із принципами використання структурних шаблонів у процесі розробки програмного забезпечення, розвиток навичок їхнього практичного застосування, а також аналіз переваг та недоліків використання таких патернів.

Лабораторний практикум на тему «Структурні патерни проєктування» є важливим етапом вивчення програмної інженерії, який дозволяє студентам оволодіти принципами і методами створення ефективних та гнучких програмних рішень. Структурні патерни, або шаблони проєктування, представляють собою усталені підходи до організації взаємодії між класами і об'єктами в програмі [1]. Вони дозволяють оптимізувати структуру коду, знизити його складність і підвищити зрозумілість, що в кінцевому результаті сприяє легкості супроводу та модифікації програмного продукту. Структурні патерни забезпечують ефективну організацію коду, що дозволяє підвищити модульність, повторне використання компонентів, а також зменшити зв'язаність між ними.

Використання структурних патернів у проєктуванні програмного забезпечення має безліч переваг. Перш за все, це сприяє створенню більш гнучкої і масштабованої архітектури, що дозволяє легше адаптувати систему до нових вимог. По-друге, патерни допомагають розробникам ефективно взаємодіяти з великими обсягами коду, роблячи його більш зрозумілим і організованим. По-третє, структурні патерни сприяють повторному використанню коду, що знижує витрати на розробку та підтримку програмного продукту.

Серед основних структурних патернів, які розглядаються під час лабораторного практикуму, виділяються «Адаптер», «Декоратор», «Легковаговик», «Компонувальник».

Патерн «Адаптер» дозволяє поєднати класи, які мають несумісні інтерфейси, шляхом перетворення одного інтерфейсу в інший. Це дозволяє використати існуючі класи без необхідності їх змінювати, що знижує витрати на адаптацію коду. Існують два способи реалізації патерна «Адаптер»: адаптер об’єктів та адаптер класів.

Адаптер об’єктів використовує агрегацію: об’єкт адаптера «загортає», тобто містить посилання на службовий об’єкт. Такий підхід працює в усіх мовах програмування. Приклад ієрархії класів для адаптера об’єктів показаний на рис. 2.1.

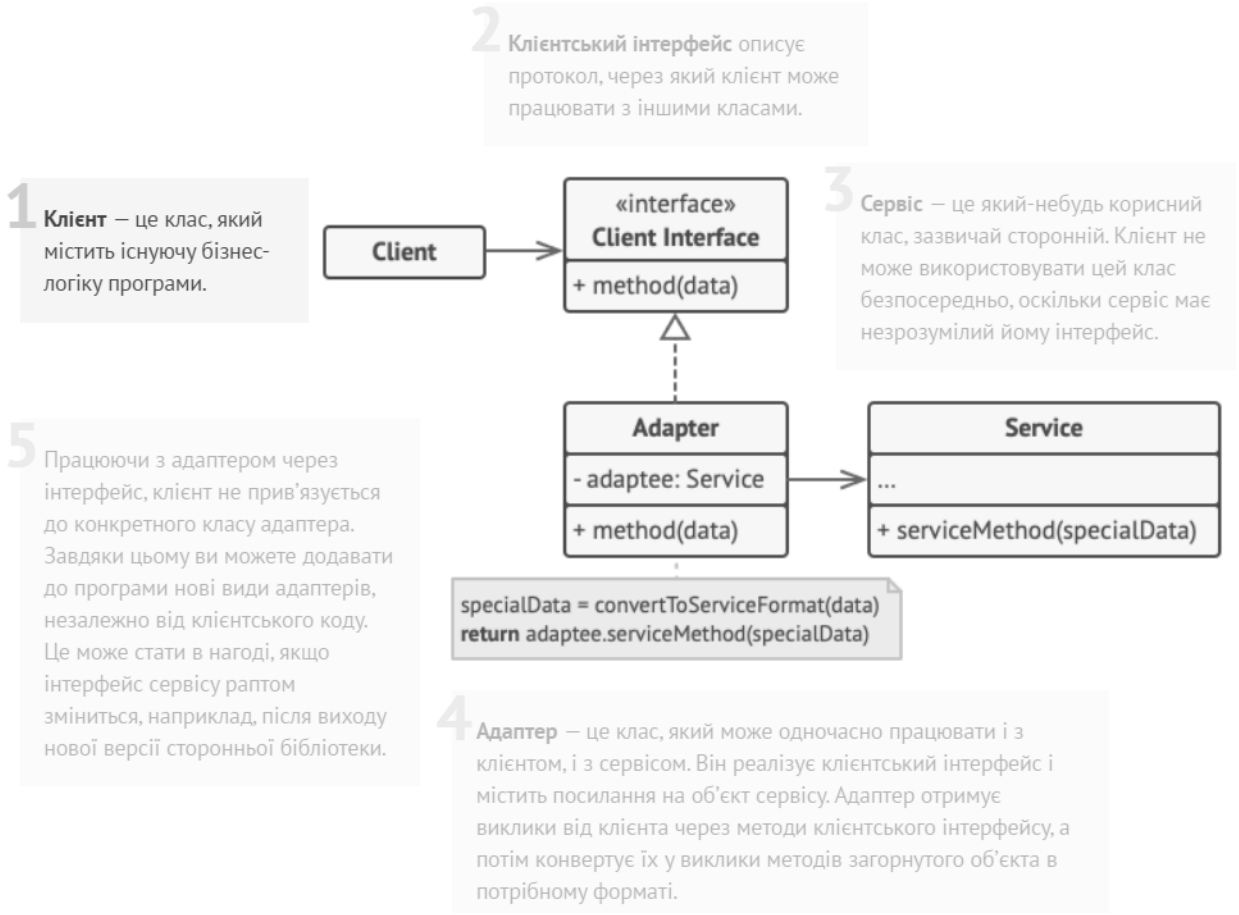



Рисунок 2.1 – Приклад структури класів відповідно патерну «Адаптер об’єктів»

Адаптер класів базується на спадкуванні: адаптер успадковує обидва інтерфейси одночасно. Такий підхід можливий тільки в мовах, які підтримують множинне спадкування, наприклад у C++. Приклад ієрархії класів для адаптера об’єктів показаний на рис. 2.2.

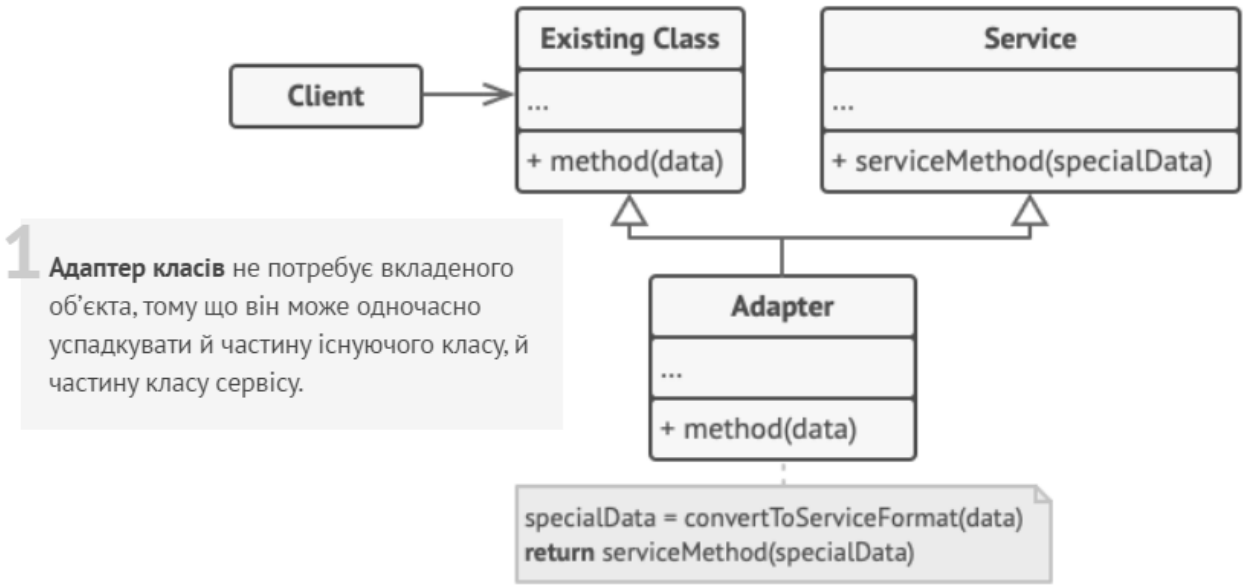



Рисунок 2.2 – Приклад структури класів відповідно патерну «Адаптер класів»

Патерн «Декоратор» дозволяє динамічно додавати нову функціональність до об'єкта, не змінюючи його структуру. Цей патерн є альтернативою наслідуванню, забезпечуючи гнучкіший спосіб розширення можливостей класів. Він використовує композицію для розширення функціональності класів, обгортаючи їх у додаткові об’єкти, звані декораторами. Це може бути зручним, коли потрібно додати нові можливості, не порушуючи принципів OCP (Open-Closed Principle) і SRP (Single Responsibility Principle).

Основна ідея патерну "Декоратор" полягає в тому, щоб створити інтерфейс для об’єктів, які можна обгорнути додатковими класами-декораторами, що реалізують той самий інтерфейс. Таким чином, об’єкт-декоратор додає нову поведінку, викликаючи при цьому методи початкового об’єкта, який він обгортає.

На рис. 2.3 показано структуру патерну «Декоратор»:

1. інтерфейс «Component» визначає інтерфейс для об’єктів, які можуть бути декоровані;

2. клас «Concrete Component» – це клас, який необхідно декорувати новими поведінками;

3.базовий клас декоратора «Base Decorator» має той самий інтерфейс, що й Компонент, і містить посилання на об’єкт компонента;

4. класи «Concrete Decorator» розширюють базовий декоратор і додають нову функціональність.

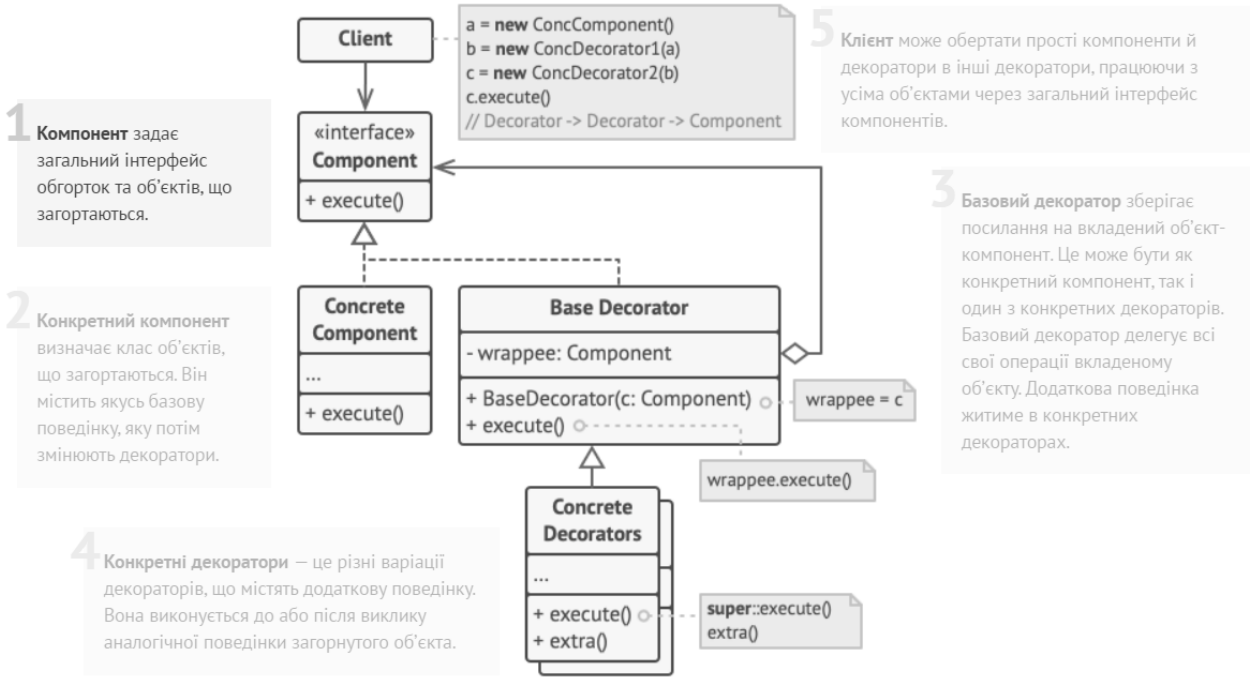



Рисунок 2.3 – Приклад структури класів відповідно патерну «Декоратор»

Патерн «Легковаговик» оптимізує використання пам'яті при створенні великої кількості однотипних об'єктів за рахунок спільного використання загальних даних. Це дозволяє зменшити кількість об'єктів у системі і, відповідно, знизити витрати на їх зберігання (рис. 2.4).

Легковаговик вирішує відразу дві проблеми, причому порушуючи принцип єдиного обов’язку класу:

1. гарантує наявність єдиного екземпляра класу, наприклад для доступу до якогось спільного ресурсу;

2. надає глобальну точку доступу, а не просто глобальну змінну, через яку можна дістатися до певного об’єкта;

3. приховує типовий конструктор та створює публічний статичний метод, який і контролюватиме життєвий цикл об’єкта-легковаговика.

На рис. 2.4 показано структуру патерну «Легковаговик»:

1. Flyweight – інтерфейс або абстрактний клас для легковагових об'єктів;

2. ConcreteFlyweight – конкретна реалізація легковагового об'єкта, що зберігає спільний стан;

3. FlyweightFactory – фабрика для створення та управління легковаговими об'єктами; вона перевіряє, чи є об'єкт зі спільним станом в пам'яті, і повертає його або створює новий, якщо такого немає;

4. Client – клас, що зберігає уникальний стан та використовує легковагові об'єкти.

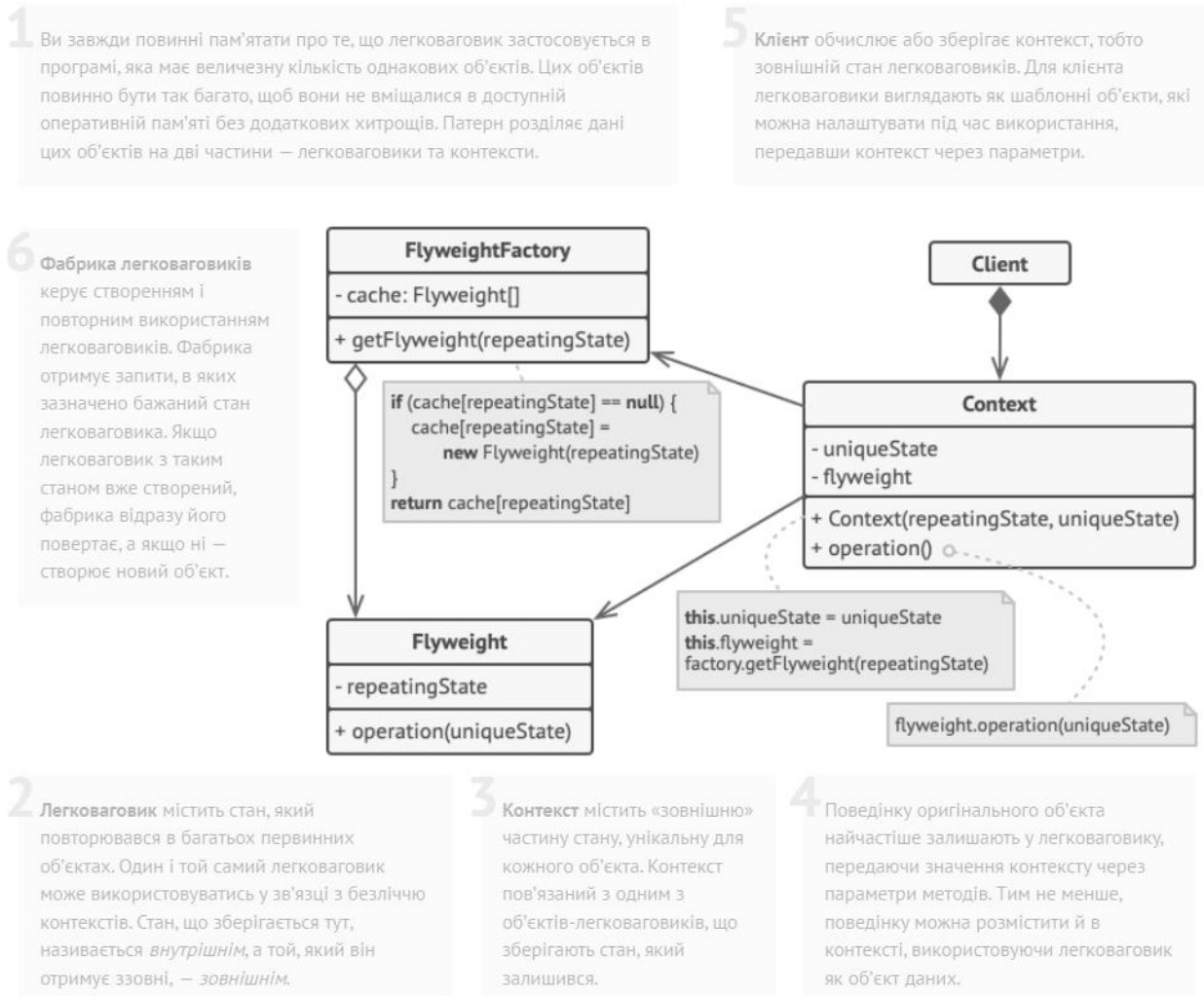



Рисунок 2.4 – Приклад структури класів відповідно патерну «Легковаговик»

Патерн «Компонувальник» є структурним патерном, який дозволяє створювати дерево об'єктів, де окремі об'єкти та їх групи можуть оброблятися однаково. Він визначає спільний інтерфейс для простих і складних об'єктів, завдяки чому код може працювати з ними, не звертаючи уваги на їхній тип.

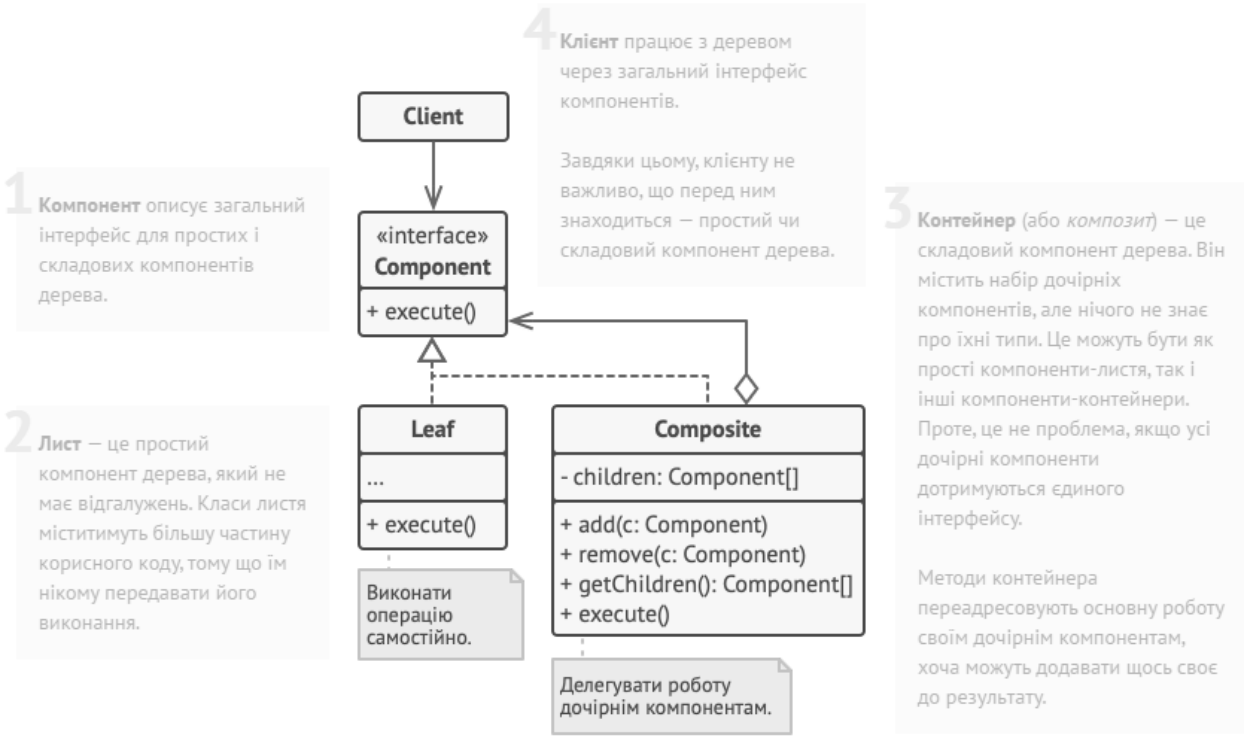



Рисунок 2.5 – Приклад структури класів відповідно патерну «Компонувальник»

1. Component – спільний інтерфейс для всіх об'єктів дерева, як простих, так і складних. Він може містити методи для управління дочірніми елементами та для виконання основної операції Operation.

2. Leaf – простий об'єкт без дочірніх елементів, який реалізує поведінку визначену в Component, та виконують фактичну роботу й надають кінцевий результат операції. Вони є кінцевими об'єктами в структурі дерева.

3. Composite – складний об'єкт, який може містити інші компоненти, як листки, так і інші композити. Він реалізує методи для керування дочірніми елементами (Add, Remove) та основну операцію, виконуючи її рекурсивно для всіх своїх дочірніх елементів, щоб повернути об'єднаний результат.

## 2.2 Постановка лабораторної роботи «Проєктування предметної області з використанням патерна “Адаптер”»

Постановка завдання лабораторної роботи.

1. Опрацюйте теоретичний матеріал. Ознайомтесь з специфікою структурного патерна «Адаптер».

2. Визначити вимоги до ПЗ згідно обраної предметної області. Побудувати власну ієрархію класів, яка базується на патерні «Адаптер». Описати та прокоментувати використання ієрархії класів, інтерфейси, розмежування атрибутів та методів конкретних класів.

3. Виконати програмну реалізацію ієрархії класів об’єктно-орієнтованою мовою (C#, C++, TypeScript, Java). Обов’язково повинен бути блок Main(), в якому створюються всі необхідні об’єкти. Код повинен бути працездатним та не містити помилки компіляції. Додати коментарі до ключових лементів.

4. Підготувати звіт до захисту лабораторної роботи.

Звіт до лабораторної роботи повинен містити:

– аналіз предметної області;

– UML-діаграма ієрархії класів;

– описи класів та їх атрибутів, методів, зв’язків та залежностей між класами;

– програмний код мовою C#, або C++, TypeScript, Java.

**Приклад виконання лабораторної роботи.** За допомогою патерна проєктування «Адаптер» реалізувати систему продажу у аптеці ліки різних виробників. Всі вони мають назву, групу ліків, до якої вони належать (антибіотики, протизапальні, шлункові тощо), ціну, термін зберігання. Розробити структуру класів, які можна використовувати для комп’ютерної обробки даних про ліки (як вітчизняні, так й імпортні) в аптеці.

На рис. 2.6 наведено UML-діаграму класів, що відповідає патерну «Адаптер».

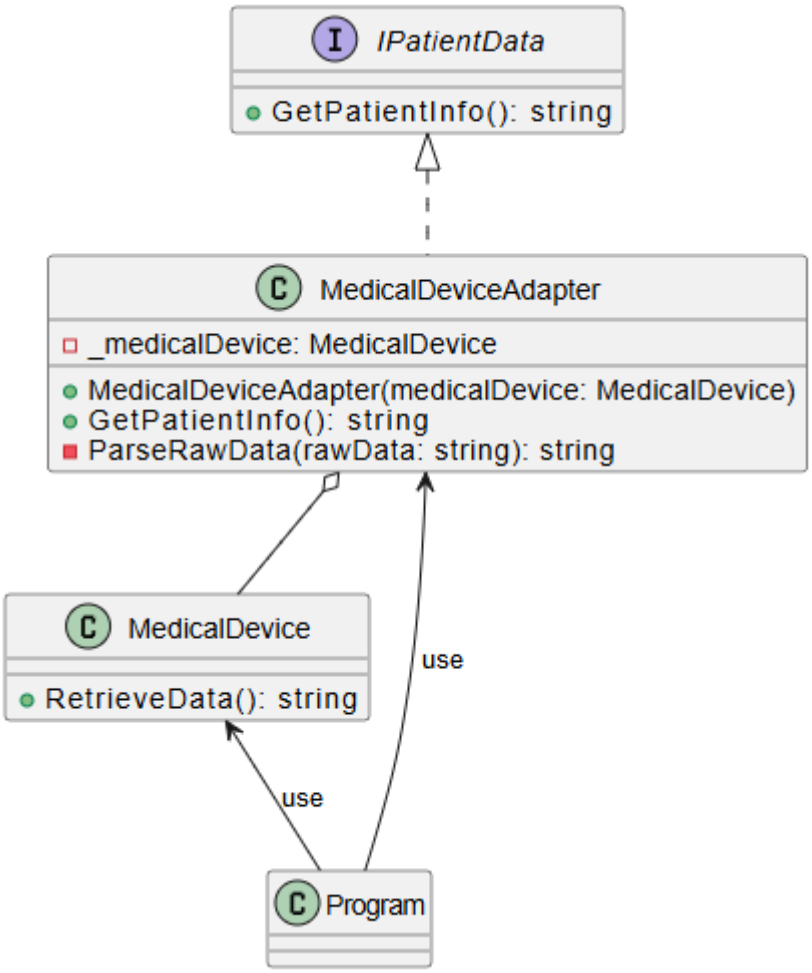



Рисунок 2.6 – UML-діаграма ієрархії класів

Далі наведено опис класів та їх атрибутів.

1. Інтерфейс IPatientData дозволяє клієнтському коду працювати зі структурами даних пацієнта в уніфікованому форматі, зокрема він визначає метод GetPatientInfo, який дозволяє отримувати інформацію про пацієнта.

2. Клас MedicalDeviceреалізує функціональність медичного приладу, який надає дані про пацієнта у специфічному форматі. Цей клас не сумісний з інтерфейсом IPatientData. Метод RetrieveData() повертає інформацію про пацієнта (такий як ідентифікатор пацієнта, частота серцебиття та артеріальний тиск) у спеціальному форматі, специфічному для приладу.

Клас MedicalDeviceAdapter реалізує інтерфейс IPatientData, тому він як клас-адаптер адаптує дані від MedicalDevice до формату, сумісного з IPatientData, перетворюючи спеціалізований формат на більш зручний для

клієнтського коду. Він містить поле _medicalDevice, з якого отримуються дані. Конструктор MedicalDeviceAdapter() приймає екземпляр MedicalDevice і зберігає його в полі _medicalDevice. Метод GetPatientInfo() реалізує інтерфейс IPatientData, отримує дані через RetrieveData та повертає їх у зручному форматі. Приватний метод ParseRawData() перетворює необроблені дані у відформатований рядок для зручного використання в клієнтському коді.

Клієнтський клас Program містить код для демонстрації використання шаблону «Адаптер». У ньому налаштовуються кодування консолі, створюється екземпляр MedicalDevice, а також його адаптер MedicalDeviceAdapter.

Програмний код реалізації патерна «Адаптер» для предметної області:

```
// Інтерфейс для отримання даних про пацієнта
public interface IPatientData
{
    // Метод для отримання інформації про пацієнта
    string GetPatientInfo();
}

// Клас, який має корисну функціональність для отримання даних
// в специфічному форматі, несумісний з IPatientData
public class MedicalDevice
{
    // Метод для отримання даних від медичного приладу в специфічному форматі
    public string RetrieveData()
    {
        // Логіка отримання даних в специфічному форматі від медичного приладу
        return "PATIENT_ID: 12345, HEART_RATE: 75, BLOOD_PRESSURE: 120/80";
    }
}

// Адаптер для забезпечення сумісності між MedicalDevice і IPatientData
public class MedicalDeviceAdapter : IPatientData
{
    // Поле для зберігання об'єкта медичного приладу
    private readonly MedicalDevice _medicalDevice;

    // Конструктор, що приймає об'єкт MedicalDevice для адаптації до IPatientData
    public MedicalDeviceAdapter(MedicalDevice medicalDevice)
    {
        _medicalDevice = medicalDevice;
    }

    // Реалізація методу GetPatientInfo з інтерфейсу IPatientData
    public string GetPatientInfo()
    {
        // Отримуємо дані від медичного приладу
        string rawData = _medicalDevice.RetrieveData();

        // Парсимо дані і повертаємо інформацію у більш зручному форматі
        return ParseRawData(rawData);
    }
```

```
    // Метод для обробки та перетворення сирих даних в зручний формат
    private string ParseRawData(string rawData)
    {
        // Формуємо відредагований рядок інформації про пацієнта
        return $"Ідентифікатор пацієнта: 12345, Частота серцебиття: " +
            $"75, Артеріальний тиск: 120/80";
    }
}

// Клієнтський код для демонстрації використання адаптера
class Program
{
    static void Main(string[] args)
    {
        // Налаштування кодування для правильного відображення українського тексту
        Console.InputEncoding = System.Text.Encoding.UTF8;
        Console.OutputEncoding = System.Text.Encoding.UTF8;

        // Створюємо медичний прилад
        MedicalDevice medicalDevice = new MedicalDevice();

        // Створюємо адаптер до нього
        IPatientData adapter = new MedicalDeviceAdapter(medicalDevice);

        // Клієнтський код використовує адаптер для отримання даних про пацієнта
        Console.WriteLine("Інформація про пацієнта, яку було отримано через " +
                          "адаптер:");
        Console.WriteLine(adapter.GetPatientInfo());
    }
}
```

```
Консоль отладки Microsoft Visual Studio
Інформація про пацієнта, яку було отримано через адаптер:
Ідентифікатор пацієнта: 12345, Частота серцебиття: 75, Артеріальний тиск: 120/80
```

Рисунок 2.7 – Приклад виконання програми

## 2.3 Постановка лабораторної роботи «Проєктування предметної області з використанням патерна “Декоратор”»

Постановка завдання лабораторної роботи.

1. Опрацюйте теоретичний матеріал. Ознайомтесь з специфікою структурного патерна «Декоратор».

2. Визначити вимоги до ПЗ згідно обраної предметної області. Побудувати власну ієрархію класів, яка базується на патерні «Декоратор». Описати та прокоментувати використання ієрархії класів, інтерфейси, розмежування атрибутів та методів конкретних класів.

3. Виконати програмну реалізацію ієрархії класів об’єктно-орієнтованою мовою (C#, C++, TypeScript, Java). Обов’язково повинен бути

блок Main(), в якому створюються всі необхідні об’єкти. Код повинен бути працездатним та не містити помилки компіляції. Додати коментарі до ключових лементів.

4. Підготувати звіт до захисту лабораторної роботи.

Звіт до лабораторної роботи повинен містити:

– аналіз предметної області;

– UML-діаграма ієрархії класів;

– описи класів та їх атрибутів, методів, зв’язків та залежностей між класами;

– програмний код мовою C#, або C++, TypeScript, Java.

**Приклад виконання лабораторної роботи.** За допомогою патерна проєктування «Декоратор» реалізувати систему продажу у аптеці ліки різних виробників. Всі вони мають назву, групу ліків, до якої вони належать (антибіотики, протизапальні, шлункові тощо), ціну, термін зберігання. Розробити структуру класів, які можна використовувати для комп’ютерної обробки даних про ліки (як вітчизняні, так й імпортні) в аптеці.

На рис. 2.8 наведено UML-діаграму класів, що відповідає патерну «Декоратор».

Далі наведено опис класів та їх атрибутів.

1. Абстрактний клас MedicalReport визначає основний інтерфейс для створення медичного звіту. Метод CreateReport() – абстрактний метод, який повинен бути реалізований у похідних класах для створення звіту.

2. Клас BasicMedicalReport реалізує базовий медичний звіт, який повертає просте повідомлення. Метод CreateReport() повертає базове повідомлення, яке представляє медичний звіт.

3. Абстрактний клас Decorator розширює функціональність об'єктів типу MedicalReport. Захищений атрибут _report зберігає об'єкт, до якого буде застосовано додаткові функціональні можливості. Метод CreateReport() викликає метод CreateReport базового об'єкта, якщо він присутній.

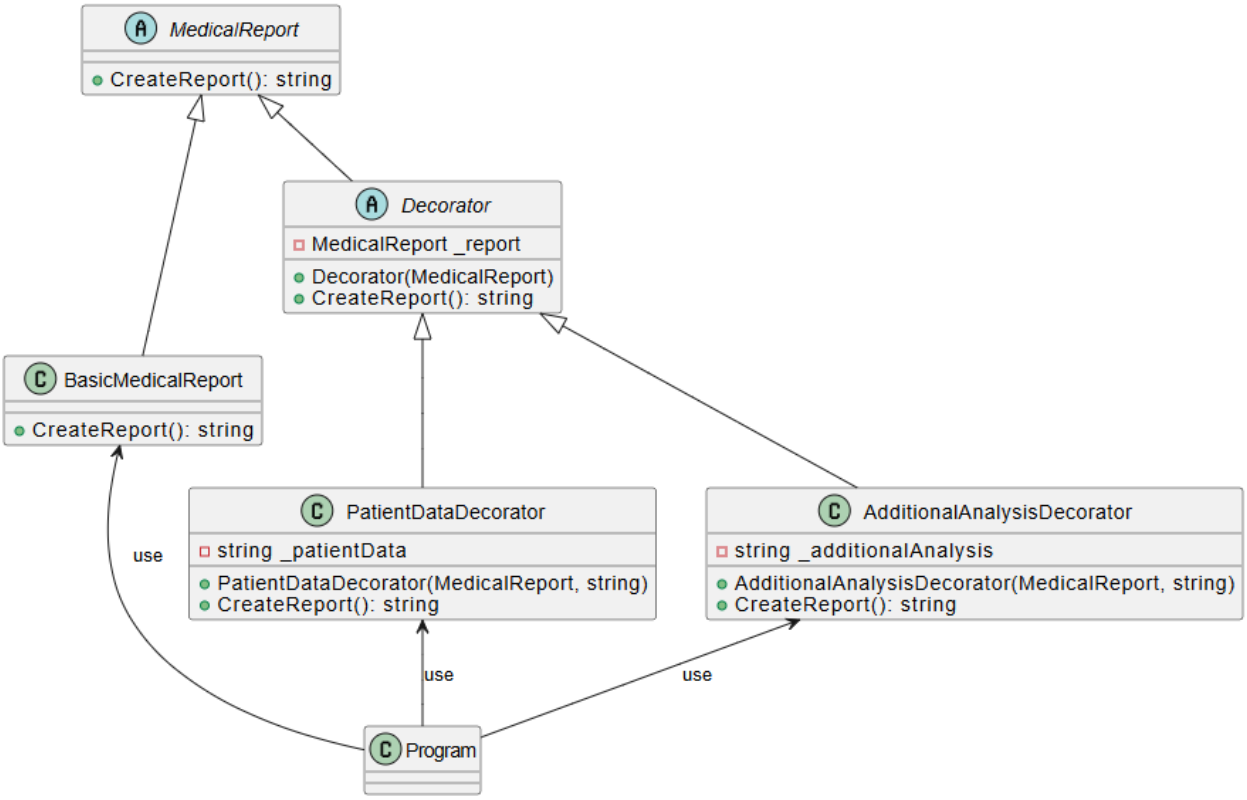



Рисунок 2.8 – UML-діаграма ієрархії класів

5. Клас AdditionalAnalysisDecorator є декоратором для додаткових аналізів, він додає до медичного звіту дані про додаткові аналізи. Приватний атрибут _additionalAnalysis зберігає дані про додаткові аналізи, які слід додати до звіту. Методи CreateReport()додає інформацію про додаткові аналізи до звіту, викликаючи CreateReport базового об'єкта і додаючи нові дані.

Програмний код реалізації патерна «Декоратор» для предметної області:

```
// Базовий клас компонента
public abstract class MedicalReport
{
    // Абстрактний метод, який буде реалізований в похідних класах
    public abstract string CreateReport();
}

// Конкретний компонент - базовий медичний звіт
public class BasicMedicalReport : MedicalReport
{
    // Реалізація методу для створення базового медичного звіту
    public override string CreateReport()
    {
        return "Базовий медичний звіт";
    }
}

// Загальний клас Decorator
abstract class Decorator : MedicalReport
{
```

```
    // Захищене поле для зберігання посилання на об’єкт MedicalReport
    protected MedicalReport _report;

    // Конструктор, який приймає об'єкт MedicalReport та ініціалізує ним поле
_report
    public Decorator(MedicalReport report)
    {
        _report = report;
    }

    // Реалізація методу CreateReport з базовою логікою створення звіту
    public override string CreateReport()
    {
        // Перевірка, чи існує базовий звіт, якщо так – повертає його результат
        if (_report != null)
        {
            return _report.CreateReport();
        }
        else
        {
            return string.Empty;
        }
    }
}

// Декоратор для додавання даних пацієнта до звіту
class PatientDataDecorator : Decorator
{
    // Поле для зберігання додаткової інформації про пацієнта
    private readonly string _patientData;

    // Конструктор, який приймає об'єкт звіту та дані пацієнта, ініціалізує їх
    public PatientDataDecorator(MedicalReport report, string patientData)
        : base(report)
    {
        _patientData = patientData;
    }

    // Метод створення звіту з додатковою інформацією про пацієнта
    public override string CreateReport()
    {
        return $"{base.CreateReport()}\nДодаткова інформація про пацієнта: " +
            $"{_patientData}";
    }
}

// Декоратор для додавання додаткових аналізів до звіту
class AdditionalAnalysisDecorator : Decorator
{
    // Поле для зберігання даних про додаткові аналізи
    private readonly string _additionalAnalysis;
    // Конструктор, який приймає об'єкт звіту та додаткові аналізи, ініціалізує їх
    public AdditionalAnalysisDecorator(MedicalReport report,
        string additionalAnalysis) : base(report)
    {
        _additionalAnalysis = additionalAnalysis;
    }

    // Метод створення звіту з додатковими аналізами
    public override string CreateReport()
    {
        return $"{base.CreateReport()}\nДодаткові аналізи: " +
            $"{_additionalAnalysis}";
    }
}
```

```
// Клієнтський код для демонстрації використання декоратора
class Program
{
    static void Main(string[] args)
    {
        // Налаштування кодування для правильного відображення українського тексту
        Console.InputEncoding = System.Text.Encoding.UTF8;
        Console.OutputEncoding = System.Text.Encoding.UTF8;

        // Створюємо базовий медичний звіт та виводимо його
        var basicReport = new BasicMedicalReport();
        Console.WriteLine("\n" + basicReport.CreateReport());

        // Додаємо дані пацієнта до звіту
        PatientDataDecorator reportWithPatientData =
            new PatientDataDecorator(basicReport, "ім'я - Олексій, вік: 35");
        Console.WriteLine("\n" + reportWithPatientData.CreateReport());

        // Додаємо додаткові аналізи до звіту
        AdditionalAnalysisDecorator finalReport =
            new AdditionalAnalysisDecorator(reportWithPatientData, "тести на віруси,
тести на кров");

        // Виводимо результат створеного звіту
        Console.WriteLine("\n" + finalReport.CreateReport());
    }
```

```
Консоль отладки Microsoft Visual Studio

Базовий медичний звіт

Базовий медичний звіт
Додаткова інформація про пацієнта: ім'я - Олексій, вік: 35

Базовий медичний звіт
Додаткова інформація про пацієнта: ім'я - Олексій, вік: 35
Додаткові аналізи: тести на віруси, тести на кров
}
```

Рисунок 2.9 – Приклад виконання програми

## 2.4 Постановка лабораторної роботи «Проєктування предметної області з використанням патерна “Легковаговик”»

Постановка завдання лабораторної роботи.

1. Опрацюйте теоретичний матеріал. Ознайомтесь з специфікою структурного патерна «Легковаговик».

2. Визначити вимоги до ПЗ згідно обраної предметної області. Побудувати власну ієрархію класів, яка базується на патерні «Легковаговик». Описати та прокоментувати використання ієрархії класів, інтерфейси,

розмежування атрибутів та методів конкретних класів.

3. Виконати програмну реалізацію ієрархії класів об’єктно-орієнтованою мовою (C#, C++, TypeScript, Java). Обов’язково повинен бути блок Main(), в якому створюються всі необхідні об’єкти. Код повинен бути працездатним та не містити помилки компіляції. Додати коментарі до ключових лементів.

4. Підготувати звіт до захисту лабораторної роботи.

Звіт до лабораторної роботи повинен містити:

– аналіз предметної області;

– UML-діаграма ієрархії класів;

– описи класів та їх атрибутів, методів, зв’язків та залежностей між класами;

– програмний код мовою C#, або C++, TypeScript, Java.

**Приклад виконання лабораторної роботи.** За допомогою патерна проєктування «Легковаговик» реалізувати систему продажу у аптеці ліки різних виробників. Всі вони мають назву, групу ліків, до якої вони належать (антибіотики, протизапальні, шлункові тощо), ціну, термін зберігання. Розробити структуру класів, які можна використовувати для комп’ютерної обробки даних про ліки (як вітчизняні, так й імпортні) в аптеці.

На рис. 2.10 наведено UML-діаграму класів, що відповідає патерну «Легковаговик».

Далі наведено опис класів та їх атрибутів.

1. Інтерфейс Medicine містить абстрактний метод Produce(), який використовується для виготовлення конкретних медикаментів, приймаючи номер партії.

2. Клас AllergySpray містить статичний атрибут dosage, що зберігає дозування спрею від алергії. Усі екземпляри цього класу спільно використовують це значення. Метод Produce є реалізацією методу з інтерфейсу Medicine, виготовляє спрей від алергії та виводить інформацію про партію та дозування.

3. Клас AllergyDrops містить статичний атрибут dosage, що зберігає дозування крапель від алергії. Усі екземпляри цього класу спільно використовують це значення. Метод Produce() є реалізацією методу з інтерфейсу Medicine. Виготовляє краплі від алергії та виводить інформацію про партію та дозування.

4. Клас PharmacyFactory містить словник medicines, який зберігає асоціацію між назвою медикаменту (ключ) і його екземпляром (значення). Дозволяє отримувати об'єкти Medicine за назвою. Метод GetMedicine() необхідне для отримання екземпляра медикаменту за вказаним ключем. Якщо медикамент не знайдено, повертає null.

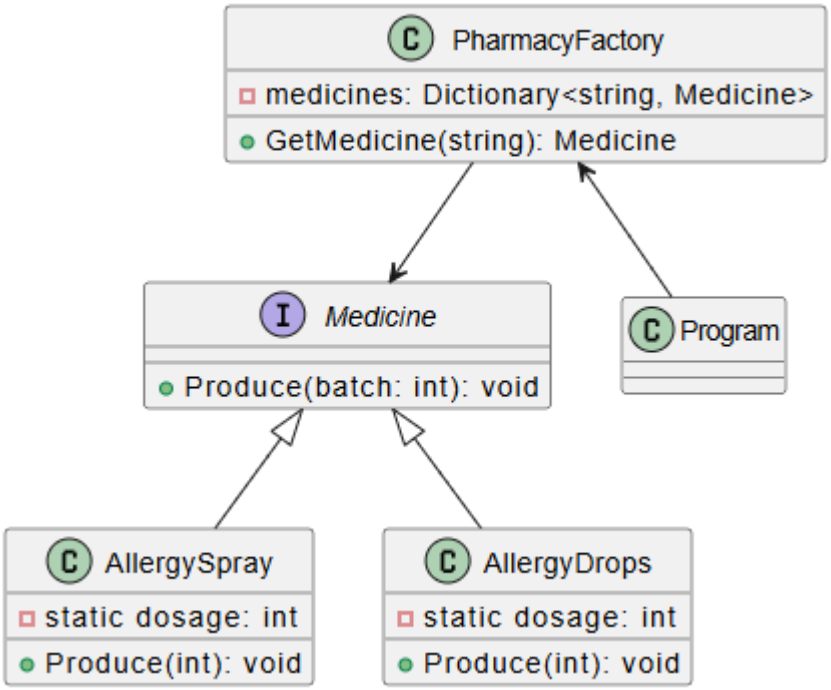



Рисунок 2.10 – UML-діаграма ієрархії класів

Програмний код реалізації патерна «Легковаговик» для предметної області:

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

// Легковаговик для медикаментів
class Program
{
    static void Main(string[] args)
    {
        // Встановлюємо кодування для вводу та виводу
        Console.InputEncoding = System.Text.Encoding.UTF8;
```

```
        Console.OutputEncoding = System.Text.Encoding.UTF8;

        // Ініціалізуємо змінну для номеру партії
        int batch = 1;
        // Створюємо фабрику аптечних засобів
        PharmacyFactory pharmacy = new PharmacyFactory();

        // Виготовляємо три партії спрею від алергії 'Віста'
        for (int i = 0; i < 3; i++)
        {
            // Отримуємо легковаговик спрею від алергії
            Medicine allergySpray = pharmacy.GetMedicine("Спрей від алергії" +
                                    "'Віста'");
            // Виготовляємо його
            allergySpray.Produce(batch);
            batch++; // Збільшуємо номер партії
        }

        // Виготовляємо дві партії крапель від алергії 'Лютик'
        for (int i = 0; i < 2; i++)
        {
            // Отримуємо легковаговик крапель від алергії
            Medicine allergyDrops = pharmacy.GetMedicine("Краплі від алергії "+
                                    "'Лютик'");
            // Виготовляємо їх
            allergyDrops.Produce(batch);
            batch++; // Збільшуємо номер партії
        }
    }
}

// Інтерфейс для медикаментів
interface Medicine
{
    void Produce(int batch);
}

// Клас для спрею від алергії
class AllergySpray : Medicine
{
    // Спільне дозування для всіх спреїв
    private static readonly int dosage = 30;
    public void Produce(int batch)
    {
        Console.WriteLine($"Виготовлено спрей від алергії 'Віста' " +
            $"дозуванням {dosage}мг з партії під №{batch}.");
    }
}

// Клас для крапель від алергії
class AllergyDrops : Medicine
{
    // Спільне дозування для всіх крапель
    private static readonly int dosage = 15;
    public void Produce(int batch)
    {
        Console.WriteLine($"Виготовлено краплі від алергії 'Лютик' " +
            $"дозуванням {dosage}мг з партії під №{batch}.");
    }
}

// Клас фабрики для виготовлення медикаментів
class PharmacyFactory
{
    // Словник для зберігання медикаментів
```

```
    private readonly Dictionary<string, Medicine> medicines =
        new Dictionary<string, Medicine>();

    // Конструктор, що ініціалізує словник медикаментами
    public PharmacyFactory()
    {
        medicines.Add("Спрей від алергії 'Віста'", new AllergySpray());
        medicines.Add("Краплі від алергії 'Лютик'", new AllergyDrops());
    }

    // Метод для отримання медикаменту за ключем
    public Medicine GetMedicine(string key)
    {
        // Перевіряємо, чи існує медикамент з таким ключем
        if (medicines.ContainsKey(key))
        {
            return medicines[key]; // Повертаємо медикамент
        }
        else
            return null; // Повертаємо null, якщо медикамент не знайдено
    }
}
```

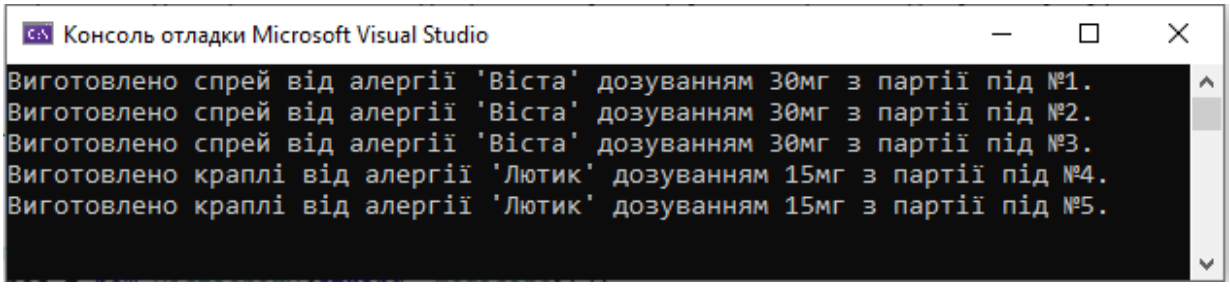



Рисунок 2.11 – Приклад виконання програми

## 2.5 Постановка лабораторної роботи «Проєктування предметної області з використанням патерна “Компонувальник”»

Постановка завдання лабораторної роботи.

1. Опрацюйте теоретичний матеріал. Ознайомтесь з специфікою структурного патерна «Компонувальник».

2. Визначити вимоги до ПЗ згідно обраної предметної області. Побудувати власну ієрархію класів, яка базується на патерні «Компонувальник». Описати та прокоментувати використання ієрархії класів, інтерфейси, розмежування атрибутів та методів конкретних класів.

3. Виконати програмну реалізацію ієрархії класів об’єктно-орієнтованою мовою (C#, C++, TypeScript, Java). Обов’язково повинен бути блок Main(), в якому створюються всі необхідні об’єкти. Код повинен бути працездатним та не містити помилки компіляції. Додати коментарі до

ключових лементів.

4. Підготувати звіт до захисту лабораторної роботи.

Звіт до лабораторної роботи повинен містити:

– аналіз предметної області;

– UML-діаграма ієрархії класів;

– описи класів та їх атрибутів, методів, зв’язків та залежностей між класами;

– програмний код мовою C#, або C++, TypeScript, Java.

**Приклад виконання лабораторної роботи.** За допомогою патерна проєктування «Компонувальник» реалізувати ієрархію армії країни у вигляді перевернутих дерев. На нижньому рівні знаходяться солдати, далі взводи, далі полки, а далі цілі армії. Накази віддаються зверху вниз структурою командування до тих пір, поки вони не доходять до конкретного солдата.

На рис. 2.12 наведено UML-діаграму класів, що відповідає патерну «Компонувальник».

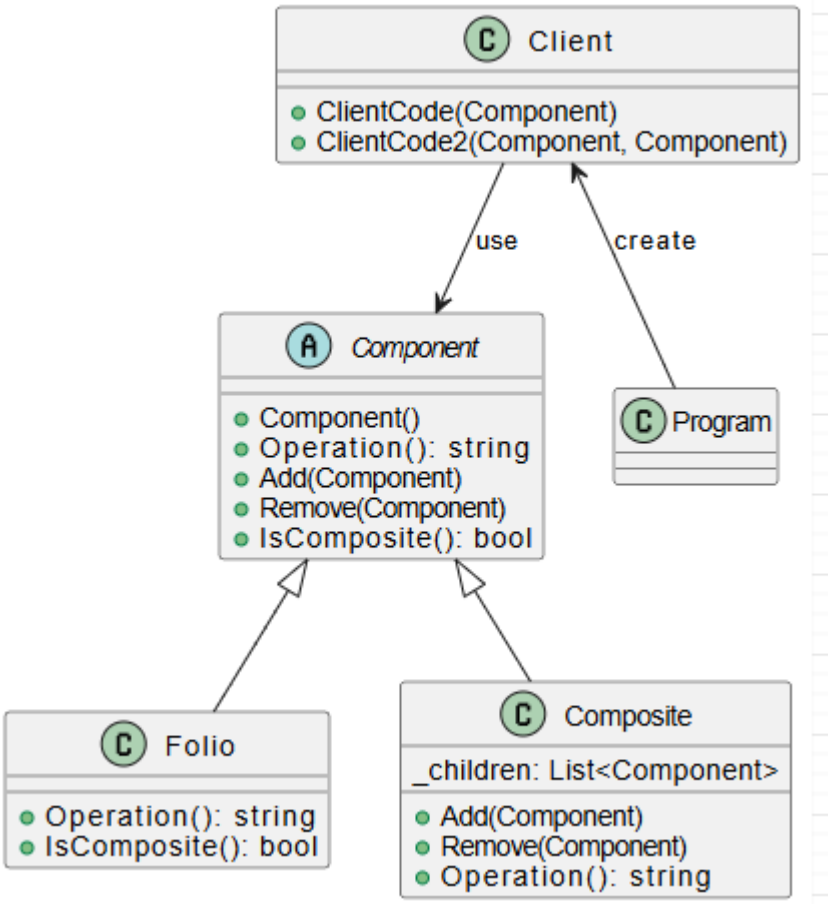



Рисунок 2.12 – UML-діаграма ієрархії класів

Далі наведено опис класів та їх атрибутів.

1. Component – це базовий клас, який визначає спільний інтерфейс для всіх елементів дерева. Він містить методи для виконання операцій та керування дочірніми елементами (наприклад, Add, Remove). Така структура дозволяє об'єднувати прості та складні об'єкти під спільний тип.

2. Folio – це клас, що представляє прості елементи дерева, які не мають дочірніх елементів. Об'єкти Folio зазвичай виконують конкретну роботу, але не містять інших об'єктів усередині. У прикладі вони реалізують метод Operation, який повертає результат для конкретного листка.

3. Composite – це клас, що представляє складні елементи, які можуть містити дочірні компоненти (Component). Він реалізує методи для додавання та видалення дочірніх елементів і обробляє їх рекурсивно. Це дозволяє збирати результати виконання дочірніх елементів, виконуючи основну роботу, орієнтовану на структуру.

4. Client – це клас, який працює з усіма компонентами через загальний інтерфейс. Йому не потрібно знати, який саме тип компоненту він використовує, оскільки він взаємодіє з ними однаково через базовий інтерфейс Component.

5. Program – це клас, що ініціює виконання програми, демонструючи приклади створення як простих компонентів Folio, так і складених компонентів Composite. У цьому класі також демонструється, як Client працює з компонентами, не перевіряючи їх тип, а лише використовуючи спільний інтерфейс.

Програмний код реалізації патерна «Компонувальник» для предметної області:

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

namespace Composite
{
    // Базовий клас Component оголошує спільні операції для як простих, так і
    // складних об'єктів композиції.
    abstract class Component
    {
        public Component() { }
```

```
    // Базовий компонент може реалізувати деяку поведінку за замовчуванням або
    // залишити її конкретним класам (шляхом оголошення методу як
    // "абстрактного").
    public abstract string Operation();

    // У деяких випадках доцільно визначити операції керування дочірніми
    // об'єктами прямо в базовому класі Component. Це дозволяє
    // не використовувати конкретні класи компонентів у клієнтському
    // коді навіть при збиранні дерева об'єктів. Мінусом є те,
    // що ці методи будуть порожніми для компонентів-фоліо.
    public virtual void Add(Component component)
    {
        throw new NotImplementedException();
    }

    public virtual void Remove(Component component)
    {
        throw new NotImplementedException();
    }

    // Можна надати метод, який дозволяє клієнтському коду визначити, чи може
    // компонент мати дочірні елементи.
    public virtual bool IsComposite()
    {
        return true;
    }
}

// Клас Folio представляє кінцеві об'єкти композиції. Folio не може
// мати жодних дочірніх об'єктів.
// Зазвичай об'єкти Folio виконують фактичну роботу, тоді як Composite
// лише делегують її своїм дочірнім елементам.
class Folio : Component
{
    public override string Operation()
    {
        return "Folio";
    }

    public override bool IsComposite()
    {
        return false;
    }
}

// Клас Composite представляє складні компоненти, які можуть мати
// дочірні елементи. Зазвичай об'єкти Composite делегують фактичну роботу
// своїм дочірнім елементам і потім "підсумовують" результат.
class Composite : Component
{
    protected List<Component> children = new List<Component>();

    public override void Add(Component component)
    {
        this.children.Add(component);
    }

    public override void Remove(Component component)
    {
        this.children.Remove(component);
    }

    // Composite виконує свою основну логіку особливим чином. Він
    // рекурсивно обходить усі свої дочірні елементи, збираючи та
    // підсумовуючи їх результати. Оскільки дочірні елементи композиту
```

```
    // передають ці виклики своїм дочірнім елементам і так далі,
    // в результаті обходиться все дерево об'єктів.
    public override string Operation()
    {
        int i = 0;
        string result = "Branch(";

        foreach (Component component in this.children)
        {
            result += component.Operation();
            if (i != this.children.Count - 1)
            {
                result += "+";
            }
            i++;
        }

        return result + ")";
    }
}

class Client
{
    // Клієнтський код працює з усіма компонентами через базовий
    // інтерфейс.
    public void ClientCode(Component folio)
    {
        Console.WriteLine($"RESULT: {folio.Operation()}\n");
    }

    // Завдяки тому, що операції керування дочірніми елементами оголошені
    // в базовому класі Component, клієнтський код може працювати з будь-яким
    // компонентом, простим чи складним, не залежачи від їх конкретних класів.
    public void ClientCode2(Component component1, Component component2)
    {
        if (component1.IsComposite())
        {
            component1.Add(component2);
        }

        Console.WriteLine($"RESULT: {component1.Operation()}");
    }
}

class Program
{
    static void Main(string[] args)
    {
        Console.OutputEncoding = System.Text.Encoding.UTF8;
        Client client = new Client();

        // Таким чином, клієнтський код може підтримувати прості
        // компоненти-Folio...
        Folio folio = new Folio();
        Console.WriteLine("Client: Отримую простий компонент:");
        client.ClientCode(folio);

        // ...а також складні Composite.
        Composite tree = new Composite();
        Composite branch1 = new Composite();
        branch1.Add(new Folio());
        branch1.Add(new Folio());
        Composite branch2 = new Composite();
        branch2.Add(new Folio());
        tree.Add(branch1);
```

```
            tree.Add(branch2);
            Console.WriteLine("Client: Тепер маю складене дерево:");
            client.ClientCode(tree);

            Console.Write("Client: Мені не потрібно перевіряти класи " +
                "компонентів навіть під час керування деревом:\n");
            client.ClientCode2(tree, folio);
        }
    }
}
```

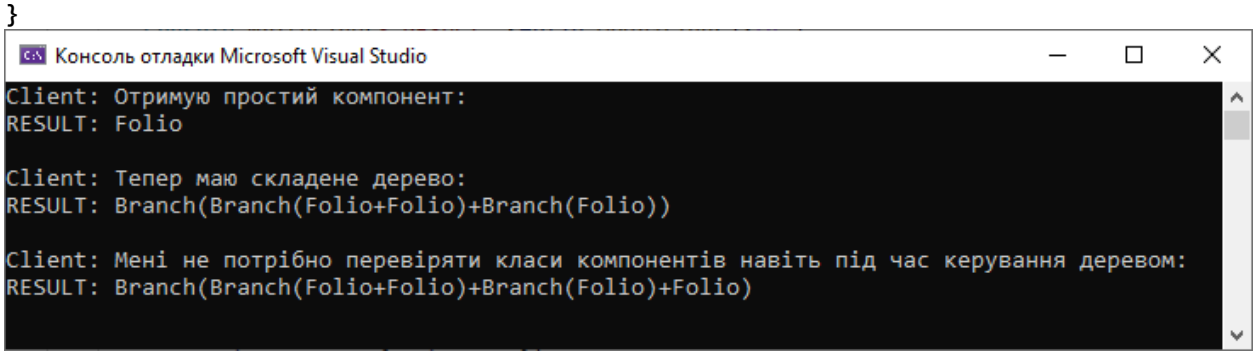



Рисунок 2.13 – Приклад виконання програми

## 2.6 Розробка предметних областей для лабораторних робіт

1. Використовуючи шаблон проєктування, розробити віртуальний автомат для приготування напоїв, таких як чай, кава, або сік. Напої повинні відрізнятися не лише за типом, але й за ціновою категорією. Наприклад, чай може бути заварним або пакетованим, кава – розчинною або меленим, а сік – 100% або нектаром. Налаштувати автомат на виготовлення напоїв відповідної якості залежно від обраної цінової політики.

2. Створити віртуальні меблеві фабрики за допомогою шаблону проєктування. Кожна фабрика повинна виробляти вітальні, м'які меблі та кухні. Виробничі характеристики фабрик визначаються матеріалами та вартістю продукції. Вибір фабрики здійснюється на основі побажань і фінансових можливостей замовника.

3. Використовуючи шаблон проєктування, організувати друк перепусток різних типів (постійних, тимчасових, одноразових) та ордерів на внесення-винесення техніки у поліграфічних фірмах. Фірми відрізняються якістю паперу, ступенем захисту та вартістю продукції. Замовлення спрямовуються на певну фірму залежно від вимог клієнта.

4. Реалізувати друк квитків на концерти за допомогою шаблону проєктування різними поліграфічними фірмами. Квитки можуть бути вхідними, звичайними або вкладеними у спеціальні запрошення. Друк театральних квитків здійснюється державними підприємствами, а квитків на гастролі – комерційними фірмами. Поліграфічні фірми розрізняються за кількістю кольорів, що використовуються при друку, та ціною послуг.

5. Організувати виробництво пакетів соку з різних фруктів і ягід на кількох заводах, застосовуючи шаблон проєктування. Заводи розрізняються сортами сировини, типом упаковки та вартістю продукції. Залежно від пори року клієнти замовляють сік у різних заводів.

6. Реалізувати процес збирання комп'ютерів і телевізорів у різних країнах за допомогою шаблону проєктування. Кожна збірка відрізняється деталями, комплектацією (базова або VIP) та вартістю. Вибір країни для виготовлення продукції здійснюється на основі вартості послуг.

7. Використовуючи шаблон проєктування, організувати виробництво пластикових читацьких квитків до бібліотеки. Квитки можуть бути студентськими, шкільними, науковими або особливими (для академіків). Різні апарати виготовляють картки з фото чи без, з голограмою або без. Вибір апарата залежить від бюджету клієнта.

8. Застосовуючи шаблон проєктування, організувати виробництво грамот, дипломів та інших відзнак для учасників конкурсу. Кожна грамота має бути незалежним об'єктом. Поліграфічні фірми, що виготовляють продукцію, розрізняються кількістю кольорів, якістю паперу та ціною послуг. Вибір фірми залежить від бюджету замовника.

9. Організувати вибір постачальників продуктів для ресторану залежно від пори року за допомогою шаблону проєктування. Постачальники працюють з різними виробниками: одні – з вітчизняними, інші – з зарубіжними. Виробники розрізняються за сортами та якістю продукції.

10. Використовуючи шаблон проєктування, організувати виробництво взуття кількома виробниками. Види взуття: жіночі туфлі, чоловічі туфлі та

кросівки. Перший виробник використовує натуральну шкіру, другий – шкірозамінник, третій – шкіру та хутро. Забезпечити постачання продукції від усіх трьох виробників до магазину.

**Висновки до розділу**

У розділі виконано аналіз характеристик об'єктів галузі проєктування об’єктно-орієнтованої ієрархії класів прикладних програмних систем. Зокрема, виконано аналіз структурних патернів проєктування «Адаптер», «Декоратор», «Легковаговик», «Компонувальник».

Виконана постановка нових лабораторних робіт:

1. проєктування предметної області з використанням патерна “Адаптер”;
2. проєктування предметної області з використанням патерна “Декоратор”;
3. проєктування предметної області з використанням патерна “Легковаговик”;
4. проєктування предметної області з використанням патерна “Компонувальник”.

Розроблено 10 предметних областей для лабораторного практикуму у якості індивідуальних варіантів для студентів.

Лабораторний практикум «Структурні патерни проєктування» є важливою складовою підготовки майбутніх фахівців з цифрових технологій. Він не лише надає можливість засвоїти основи застосування патернів на практиці, але й сприяє розвитку критичного мислення у виборі найбільш ефективних рішень для розв'язання конкретних завдань. Опанування структурних патернів є необхідним елементом професійного зростання кожного розробника, що прагне створювати якісне, надійне і масштабоване програмне забезпечення.

# РОЗДІЛ 3 ВИМОГИ ДО КАДРОВОГО ЗАБЕЗПЕЧЕННЯ ОБ'ЄКТУ ГАЛУЗІ

Підготовка викладачів з інформаційних технологій (ІТ) у Європі та США значною мірою визначається потребами цифрового суспільства, що активно розвивається. Технологічні інновації та впровадження цифрових навичок у процес навчання сприяють постійному оновленню та модернізації освітніх програм для викладачів. Далі розглянуто основні особливості підготовки викладачів з ІТ у цих регіонах, зокрема методологічні аспекти, структуру навчання, підхід до практичних навичок, акцент на інноваційні технології та партнерства з індустрією.

1. Структура освітніх програм для викладачів з ІТ.

Системи підготовки викладачів у Європі та США мають багато спільних рис, проте різняться за структурою та організацією. У більшості європейських країн професійна підготовка викладачів з ІТ організована у вигляді бакалаврських і магістерських програм, де велика увага приділяється методології викладання, психології навчання та цифровій грамотності. Наприклад, у Німеччині, Франції та Нідерландах навчальні програми викладачів з ІТ складаються з поєднання обов'язкових та факультативних курсів, де значний акцент робиться на педагогічних підходах до викладання інформаційних дисциплін.

У США структура підготовки викладачів має інші відмінності: викладання ІТ часто інтегроване в ширшу програму підготовки педагогів. Майбутні викладачі з ІТ можуть обрати спеціалізацію у вигляді додаткового курсу до основної програми. Американські університети, зокрема Стенфорд, Массачусетський технологічний інститут та інші, активно сприяють підготовці викладачів, які поєднують ІТ-навички з іншими предметами, наприклад, математикою чи фізикою. Це дозволяє підготувати фахівців, здатних інтегрувати ІТ-компетенції у широкий спектр дисциплін, роблячи навчання більш міждисциплінарним і практично орієнтованим.

2. Особливості методології навчання.

Європейські програми для викладачів з ІТ вважають за краще зосереджуватися на педагогічних інноваціях та сучасних методах викладання. Однією з ключових методологій у Європі є дослідницьке навчання (research-based learning), яке залучає студентів-викладачів до активної участі у дослідницьких проектах і розробках. Це сприяє розвитку критичного мислення, наукової інтуїції та аналітичних навичок у майбутніх викладачів. Наприклад, у Фінляндії викладачі з ІТ часто працюють над спільними проектами з учнями, допомагаючи їм розуміти структуру та процеси, які стоять за створенням цифрових технологій.

У США популярною методологією є проектне навчання (project-based learning), де студенти-викладачі працюють над реальними проектами та розвивають практичні навички на основі конкретних завдань. Цей підхід сприяє формуванню практичних компетенцій і дозволяє викладачам знайомитися з реальними викликами індустрії, одночасно отримуючи досвід у викладанні. Також, у США значний акцент робиться на змішаному навчанні (blended learning), де поєднується очне і дистанційне навчання. Такі інструменти допомагають викладачам на практиці опановувати сучасні технології та згодом адаптувати їх для студентів.

3. Підхід до практичних навичок та стажувань.

Одним із важливих аспектів підготовки викладачів з ІТ є розвиток практичних навичок. У Європі стажування та практичні заняття є невід’ємною частиною навчального процесу. Наприклад, у Норвегії та Швеції навчальні програми включають обов’язкові стажування у школах, коледжах чи технологічних компаніях, що дозволяє майбутнім викладачам працювати у реальному навчальному середовищі. Під час стажувань вони вивчають, як інтегрувати технології у викладацьку діяльність та як працювати з цифровими інструментами для навчання.

У США акцент на практичній підготовці не менш важливий, проте часто співпрацю з індустрією підтримують через партнерські відносини між

університетами та технологічними компаніями, такими як Google, Microsoft та Apple. Майбутні викладачі мають можливість проходити стажування у таких компаніях, працювати з новітніми технологіями і брати участь у спільних проектах. Це не лише підвищує рівень їхньої підготовки, але й сприяє розвитку підприємницьких навичок, таких як управління проектами та командна робота.

4. Інновації та технологічний акцент.

У підготовці викладачів з ІТ особлива увага приділяється впровадженню інноваційних технологій. У європейських університетах, наприклад, активно використовуються технології доповненої реальності, штучного інтелекту, робототехніки, хмарних обчислень та інтернету речей. Багато програм включають курси, присвячені вивченню основ програмування, аналітики даних та створенню навчальних платформ. Такі інновації сприяють розвитку творчого підходу та здатності адаптуватися до нових умов.

У США особливий акцент робиться на використанні штучного інтелекту та великих даних для поліпшення процесу навчання. Наприклад, освітні технологічні стартапи в університетах США активно досліджують можливості персоналізації навчання на основі аналізу великих даних, що дозволяє оптимізувати процес навчання, враховуючи індивідуальні потреби кожного учня. Більшість університетів пропонують спеціальні програми з розробки навчальних додатків і платформ, що допомагають викладачам ефективніше взаємодіяти зі студентами в цифровому середовищі.

5. Партнерства з індустрією.

Як у Європі, так і у США важливу роль у підготовці викладачів з ІТ відіграють партнерства з технологічними компаніями. В Європі університети співпрацюють з локальними технологічними фірмами та стартапами, що сприяє розширенню досвіду студентів та надає їм практичні знання, необхідні для сучасного ринку праці. Наприклад, у Великобританії та Нідерландах активно проводяться семінари, конференції та тренінги за участю

представників індустрії, що дозволяє майбутнім викладачам бути в курсі останніх тенденцій і технологій.

У США партнерства між університетами і гігантами технологічної галузі, такими як Google та Amazon, забезпечують викладачів доступом до новітніх технологій і методик. Крім того, багато університетів мають доступ до ресурсів і сертифікацій від великих компаній, які сприяють підвищенню якості освіти та дозволяють викладачам отримувати сертифікати за програмами, що підвищують їхню кваліфікацію.

Партнерства освітніх IT-закладів США та Європи з промисловою індустрією є ключовими у підготовці висококваліфікованих фахівців. Вони забезпечують студентам доступ до сучасних технологій, новітнього обладнання та реальних практичних проєктів, що значно підвищує якість навчання та наближує його до актуальних потреб ринку праці.

Массачусетський технологічний інститут має партнерство з Google, завдяки якому студенти отримують доступ до хмарної платформи Google Cloud, інструментів аналізу великих даних і технологій штучного інтелекту. Google надає технічну підтримку для дослідницьких проєктів MIT, а також регулярно проводить лекції та семінари для студентів.

Стенфордський університет співпрацює з Microsoft, зокрема через програму AI for Good, яка надає ресурси для розробки інноваційних проєктів у галузі штучного інтелекту. Microsoft не тільки підтримує фінансово дослідження, але й проводить воркшопи та сертифікаційні програми для студентів Стенфорду.

Університет Карнегі-Меллон має партнерство з Apple у галузі доповненої реальності (AR) та штучного інтелекту (AI). Apple надає студентам і викладачам доступ до обладнання і програмного забезпечення для розробки додатків у сфері AR/VR, а також проводить курси для підвищення кваліфікації.

Завдяки партнерству з Facebook, студенти Каліфорнійського університету у Берклі мають можливість брати участь у проектах з розробки

програмного забезпечення для соціальних мереж і аналізу даних. Facebook також фінансує дослідження в галузі штучного інтелекту та безпеки даних у співпраці з університетом.

Університет Оксфорда співпрацює з IBM у дослідженні квантових обчислень і штучного інтелекту. IBM надає доступ до своїх квантових комп'ютерів для дослідницьких проєктів університету, а також проводить навчальні програми для студентів з квантового програмування та машинного навчання.

Технічний університет Мюнхена співпрацює з Siemens у проєктах автоматизації та промислового Інтернету речей (IIoT). Siemens підтримує лабораторії університету, надаючи обладнання для досліджень у сфері робототехніки та кібербезпеки. Компанія також пропонує стажування для студентів TUM.

Політехніка Мілану має партнерство з Huawei у сфері телекомунікацій та мобільних технологій. Huawei надає доступ до лабораторій 5G і проводить серії лекцій з новітніх технологій у галузі бездротових мереж. Це допомагає студентам працювати з новими мережевими технологіями та розвивати практичні навички.

Університет Амстердама співпрацює з Philips у дослідженнях з біомедицини та аналізу даних у медицині. Philips підтримує дослідження, спрямовані на розробку медичних пристроїв і аналітичних інструментів, що базуються на штучному інтелекті для охорони здоров'я.

Такі партнерства створюють можливості для студентів отримувати передові знання і досвід роботи з інноваційними технологіями, сприяючи розвитку сучасної IT-освіти.

6. Виклики та перспективи викладачів з IT у Європі та США.

Сучасна підготовка викладачів з інформаційних технологій у Європі та США стикається з численними викликами, зумовленими швидким розвитком технологій, потребами освітньої системи та запитами ринку праці. Попри ці труднощі, IT-освіта залишається однією з найбільш перспективних і

затребуваних напрямів підготовки фахівців. Виклики та перспективи в галузі ІТ-освіти в Європі та США суттєво впливають на формування нових освітніх підходів і стратегій.

Одним із найбільших викликів у підготовці викладачів з ІТ є швидке оновлення технологій, через що навчальні програми швидко застарівають. Наприклад, з появою нових мов програмування, методик обробки даних і платформ для розробки програмного забезпечення викладачам складно залишатися в курсі всіх нововведень і передавати актуальні знання студентам.

Сучасні ІТ-викладачі мають володіти не лише технічними знаннями, але й здатністю викладати ІТ як міждисциплінарну дисципліну, поєднуючи її з математикою, фізикою, економікою тощо. Проте така інтеграція вимагає суттєвого перегляду навчальних програм і підвищення кваліфікації викладачів.

Незважаючи на важливість ІТ-освіти, багато європейських і американських університетів стикаються з фінансовими обмеженнями, що ускладнює доступ до сучасного обладнання та необхідних технологій. У країнах Європи фінансування навчальних закладів здебільшого державне, тому університети не завжди можуть швидко адаптуватися до нових тенденцій.

ІТ-освіта потребує великої кількості кваліфікованих викладачів, проте в Європі та США спостерігається дефіцит спеціалістів, здатних викладати ІТ-дисципліни. Багато потенційних викладачів обирають роботу в промисловості, де заробітні плати та кар'єрні можливості значно вищі, що створює конкуренцію для навчальних закладів.

Європейська та американська системи освіти мають різні підходи до акредитації ІТ-програм, що ускладнює взаємне визнання дипломів і сертифікатів. Наприклад, у Європі, де впроваджена Болонська система, програми підготовки часто стандартизовані, що забезпечує єдиний підхід до акредитації. У США ж стандарти можуть значно відрізнятися залежно від штату, що ускладнює роботу над міжнародними проєктами.

Дистанційне навчання та електронні платформи стали важливим інструментом під час пандемії COVID-19, і цей тренд продовжує розвиватися. Дистанційна освіта дозволяє викладачам і студентам мати доступ до якісних курсів незалежно від місця перебування, що значно розширює можливості для підготовки фахівців у віддалених районах. Це також сприяє зменшенню витрат на освітню інфраструктуру, знижуючи вартість підготовки викладачів.

**Висновки до розділу**

У результаті аналізу предметної області встановлено, що У Європі та США зростає попит на перекваліфікацію для викладачів, які прагнуть отримати нові IT-компетенції. Освітні заклади активно розробляють програми для викладачів, які хочуть інтегрувати IT у свою роботу. Це дозволяє залучати до освітнього процесу фахівців з інших галузей, які готові опанувати IT-навички та передавати їх студентам.

Встановлено, що підготовка викладачів з IT у Європі та США стикається з низкою викликів, серед яких швидкий розвиток технологій, недостатня кількість фахівців, проблеми фінансування та стандартизації. Розвиток таких ініціатив, як віртуальні лабораторії, міждисциплінарні програми та дистанційне навчання, дозволяє створювати ефективні моделі підготовки викладачів, здатних не лише викладати інформаційні технології, але й адаптувати їх до будь-якої дисципліни, роблячи навчальний процес більш ефективним і актуальним для потреб сучасного суспільства.

# РОЗДІЛ 4 МЕТОДИКА ПРОФЕСІЙНОЇ ПІДГОТОВКИ ФАХІВЦІВ З ЦИФРОВИХ ТЕХНОЛГІЙ ДИДАКТИЧНИЙ ПРОЕКТ КОНСУЛЬТАТИВНОГО ЗАНЯТТЯ З ТЕМИ «ФУНДАМЕНТАЛЬНІ СКЛАДОВІ ПРОЦЕСУ КОНСТРУЮВАННЯ ПЗ» ДИСЦИПЛІНИ «КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ» ДЛЯ ЗДОБУВАЧІВ ОСВІТИ СПЕЦІАЛЬНОСТІ ПРОФЕСІЙНА ОСВІТА(ЦИФРОВІ ТЕХНОЛОГІЇ)

**Вихідні дані:**

*навчальний заклад: Бахмутський* Навчально-науковий професійно-педагогічний інститут Харківського національного університету імені В.Н.Каразіна, ВНЗ III-IV рівнів акредитації;

*Галузь знань:* 01 Освіта /Педагогіка;

*спеціальність:* Професійна освіта (Цифрові технології);

*освітній рівень*: перший бакалаврський;

*Форма навчання*: заочна;

*назва навчальної дисципліни і теми, з якої проводиться консультативне заняття:* «Фундаментальні складові процесу конструювання ПЗ» дисципліни «Конструювання програмного забезпечення».

Отже**,** дисципліна містить такі характеристики як:

кількість кредитів – 4;

модулів – 1;

загальну кількість годин для вивчення дисципліни – 120 навчальних годин з яких 108 годин самостійної роботи та 12 години аудиторної роботи для заочної форми навчання.

Форма контролю: іспит, залік.

Великий обсяг навчального матеріалу, обширні, складні цілі навчання та великий відсоток часу, що відведено на самостійну роботу, обумовлюють необхідність в проведенні консультативних занять для уточнення та

пояснення навчального матеріалу з дисципліни «Конструювання програмного забезпечення».

Проектування цілей консультативного заняття представлені у табл. 4.1. [38].

Таблиця 4.1

Цілі консультативного заняття

| Цілі консультативного заняття | Цілі формування різних рівнів засвоєння навчального матеріалу | Умови досягнення | Результат у вигляді дій студентів |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 1 | З переліку визначень впізнавати основні поняття теми «Фундаментальні складові процесу конструювання ПЗ» такі, як технологія, утиліти, CASE системи; парадигми ТКПЗ; вміти називати сутність цих понять. | Знати визначення понять «технологія» та «утиліти», знання сутності поняття «CASE системи». | Правильно названі з переліку основні поняття теми «Фундаментальні складові процесу конструювання ПЗ» такі, як технологія, утиліти, CASE системи; парадигми ТКПЗ ; вміло названо сутність технологій та їх роль для створення програмного забезпечення, перераховано методи, процедури та засоби ТКПЗ. |
| 2 | Уміти розрізняти відмінності між поняттями «технологія», «утиліти», «CASE системи»; «парадигми ТКПЗ»; описувати методи, засоби та процедури. За схемою здійснювати вибір методів, засобів, процедур. | Виконання дій першого рівня: правильно названі з переліку основні поняття теми «Фундаментальні складові процесу конструювання ПЗ» такі, як «технологія», «утиліти», «CASE системи»; «парадигми ТКПЗ» «системний аналіз», «аналіз вимог», «кодування» «тестування» «супровід». . | Вміло розпізнано відмінності між між поняттями такими як «технологія», «утиліти», «CASE системи»; «парадигми ТКПЗ» «системний аналіз», «аналіз вимог», «кодування» «тестування» «супровід». За схемою здійснено вибір етапів класичного життєвого циклу ПЗ. |

Продовження табл. 4.1

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 3 | Уміти аналізувати класичний життєвий циклу програмного забезпечення. Характеризувати зміст основних етапів. | Виконання дій першого і другого рівнів: вміло розпізнано відмінності між поняттями «технологія», «утиліти», «CASE системи»; «парадигми ТКПЗ» «системний аналіз», «аналіз вимог», «кодування» «тестування» «супровід». | Правильно проаналізовано класичний життєвого циклу програмного забезпечення. |
| 4 | Уміти досліджені переваги та недоліки класичного життєвого циклу програмного забезпечення. | Виконання дій першого, другого і третього рівнів. | Правильно досліджено класичний життєвого циклу програмного забезпечення. |

Наведемо перелік джерел інформації для підготовки студентів до консультації згідно з робочою програмою дисципліни «Конструювання програмного забезпечення». Нижче представлено перелік основної та допоміжної літератури, а також інформаційні ресурси для вивчення дисципліни та підготовки до консультативного заняття:

Рекомендована література:

1. Lelek T., Jon Skeet J. Software Mistakes and Tradeoffs Software Mistakes and Tradeoffs: How to make good programming decisions". Manning Publications Co, Shelter Island, 2022. P. 416.

2. Rylander S. Patterns of Software Construction: How to Predictably Build Results. Apress, 2022. 156 p. ISBN: 9781484279359.

3. Бородкіна І. Інженерія програмного забезпечення: навч. посібник. Центр учбової літератури, 2021. 204 с. ISBN: 9786110112321.

4. Мартін Роберт. Чиста архітектура. 2 вид. Х.: Фабула, 2019. 368 с. ISBN: 978-6-17-095286-8.

5. Баран С. В. Розробка програмного забезпечення з використанням патернів проектування: навч. посіб. Кривий Ріг, 2023. 203 с.

6. Цибульник С. О., Барандич К. С. Технології розроблення програмного забезпечення. Ч. 1. Життєвий цикл програмного забезпечення [Електронний ресурс] : підручник. Київ : КПІ ім. Ігоря Сікорського, 2022. 270 с.

7. Швець О. Занурення в патерни проєктування : підручник. К.: Refactoring.Guru, 2021. 393 с. URL: https://refactoring.guru/files/design-patterns-ru-demo.pdf.

11. Рульєв В. А. Менеджмент [Текст] : навчальний посібник / В. А. Рульєв, С. О. Гуткевич. - К.: ЦУЛ, 2011. - 312 с.

12. Туленков М.В. Сучасні теорії менеджменту [Текст] : навчальний посібник / М. В. Туленков. - К.: Каравела, 2012. – 304 с.

Інформаційні ресурси

1. http://cyber.onua.edu.ua/ – робочі матеріали з курсу

2. http://dspace.onua.edu.ua – eNUOLAIR – депозитарій (архів) НУ «ОЮА»

3. http://sites.computer.org/ccse/SE2004Volume.pdf – Software Engineering. Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. A Volume of the Computing Curricula Series.

4. http://standards.ieee.org/reading/ieee/std_public/ description/se/610.12-1990_desc.html – IEEE Std 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology.

5. http://www.shellmethod.com/refs/ seglossary.pdf – Glossary of Software Engineering terms.

6.http://www.computer.org/portal/site/ieeecs/menuitem.c5efb9b8ade9096b8a9ca0108bcd45f3/index.jsp?&pName=ieeecs_level1&path=ieeecs/content&file=ethics.xml&xsl

=generic.xsl& – IEEE-CS/ACM Software Engineering Ethics and Professional Practices.

7. https://abitap.com/category/paterny-proektuvannya/ – ABout IT And Programming. Категорія: Патерни проєктування.

8. https://www.udemy.com/course/the-complete-guide-to-becoming-a-software-architect/ – The Complete Guide to Becoming a Software Architect – Онлайн-курс Udemy.

9. https://www.udemy.com/course/basics-of-software-architecture-design-in-java/ – Software Architecture (SOLID) & Design Patterns in Java – Онлайн-курс Udemy.

10. https://www.classcentral.com/course/edx-software-construction-data-abstraction-8200 – Software Construction: Data Abstraction. The University of British Columbia via edX.

8. https://www.pluralsight.com/blog/software-development/10-steps-to-clean-code – 10 Tips for Writing Clean Code.

Основним джерелом для підготовки студентів до консультації є навчальний посібник з дисципліни «Конструювання програмного забезпечення», оскільки він є найбільш адаптованим до змісту робочої програми.

Визначимо найбільш складних для розуміння та засвоєння питань (табл. 4.2) [37].

Оберемо методи активізації навчальної діяльності студентів на консультації (табл. 4.3) [38].

Далі необхідно здійснити вибір способів організації консультативного заняття. Він здійснюється з урахуванням даних, наведених в таблиці 4.4.

Згідно представленої таблиці обираємо 1 варіант організації консультативного заняття, на якому викладач пояснює питання, які здалися незрозумілими студентам.

Таблиця 4.2

Обрання питань для консультування та формулювання відповідей на можливі питання

| Теми (або тема) дисципліни | Зміст програми за кожною темою | Найбільш складні питання за темами (темою) | Відповіді на питання |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| Фундаментальні складові процесу конструювання ПЗ | 1. Фундаментальні складові процесу конструювання ПЗ: мінімізація складності, очікування змін, конструювання з можливістю перевірки, використання зовнішніх та внутрішніх стандартів конструювання. Життєвий цикл ПЗ. | 1. Які завдання вирішують методи? | 1. Методи забезпечують вирішення наступних завдань: планування і оцінка проекту; аналіз системних і програмних і програмних структур; кодування; тестування; супровід. вимог; проектування алгоритмів, структур даних |
| | | 2.Схарактеризуйте сутність процедур | **2.** Процедури є «клеєм», який сполучає методи і утиліти так, що вони забезпечують безперервний технологічний ланцюжок розробки. Процедури визначають: порядок застосування методів і утиліт; формування звітів, форм по відповідних вимогах; контроль, який допомагає забезпечувати якість і координувати зміни; формування «віх», по яких керівники оцінюють прогрес. |

Наводимо розробку сценарію проведення консультативного заняття у відповідності до обраного варіанту його організації (табл. 4.5) [37].

Таблиця 4.3

Методи активізації навчальної діяльності студентів на консультації

| Дидактичні методи | Реалізація методів при проведенні консультаційного заняття |
|---|---|
| Методи підвищення наочності | Використання інтерактивної дошки для демонстрації слайдів з теми «Конструювання програмного забезпечення» |
| Мотиваційні методи | Для реалізації мотивації використаємо:<br>тип: внутрішня мотивація;<br>вид: вступна мотивація;<br>метод: мотивуючий вступ;<br>прийом: віднесення до особистості.<br>Повідомлення важливості вивчення даної теми: «Тема «Конструювання програмного забезпечення». Вивчення цієї теми для вас, як майбутніх інженерів-педагогів з цифрових технологій, є на сьогоднішній час дуже актуальною. Відповідно до вашої професійної діяльності знання цієї теми знадобляться вам конструювання програмного забезпечення. Це питання особливо набуває значення для програмістів та фахівців з цифрових технологій. Усе це допоможуть Вам приймати ефективні виробничі, організаційні та наукові рішення в галузі програмного забезпечення. |
| Проблемні методи | Використання проблемного питання.<br>Проблемні питання:<br>«Яким чином визначити переваги та недоліки інженерної схеми? Обґрунтуйте доцільність. |
| Комунікативні методи | Імітація ситуацій з реального життя.<br>Яким чином ви як замовник будете формулювати вимоги до програмного продукту |

Таблиця 4.4

Варіанти організації консультативного заняття [36]

| № варіанта | Етапи організації заняття | Характеристика варіанта |
|---|---|---|
| 1 | 2 | 3 |
| 1 | - вступне слово лектора,<br>- відповіді на питання студентів і обговорення їх,<br>- заключне слово викладача | Недоліком цього варіанту проведення лекції-консультації є відсутність<br>послідовності, системи в питаннях, на які доводиться викладачу давати відповіді. Питання поступають хаотично, що знижує якість консультації. |
| 2 | - збір питань в письмовій формі до лекції, їх систематизація,<br>- відповіді на питання, що поступили,<br>- відповіді на додаткові питання,<br>- обмін думками,<br>- висновки | Цей варіант, на відміну від попереднього, дозволяє викладачу групувати відповіді, що сприяє кращому засвоєнню навчального матеріалу студентами. |

*Продовження табл. 4.4*

| 1 | 2 | 3 |
|---|---|---|
| 3 | - видача завдань на самостійне вивчення матеріалу теми.<br>- підготовка питань лектору.<br>- відповіді і їх обговорення | В цьому випадку консультування грає функцію додаткового інформування зі складних питань і пояснення незрозумілого навчального матеріалу. |
| 4 | - повідомлення теми,<br>- консультування декількома фахівцями в певній області науки і техніка з актуальних питань науки і нової техніки | Цей варіант лекції-консультації проводиться, як правило, зі спеціальних дисциплін, іноді для цієї мети використовуються наукові семінари. Такі заняття дають можливість зіставити думки різних учених на одну і ту ж проблему і є чудовою школою ведення дискусії. |

Таблиця 4.5

Сценарій консультативного заняття

| Етапи проведення консультативного заняття | Дії викладача | Дії учнів (студентів) |
|---|---|---|
| 1 | 2 | 3 |
| Організаційний момент | Викладач вітає студентів, робить перекличку, пропонує студентам розпочати роботу на консультації. | Студенти вітають викладача, беруть участь у перекличці, налаштовуються на роботу на консультації. |
| Повідомлення теми і мети уроку | Повідомлення теми заняття «Фундаментальні складові процесу конструювання ПЗ», сутність понять «технологія», «утиліти», «CASE системи»; «парадигми ТКПЗ» | Фіксація теми, сприйняття цілей, представлення результатів засвоєння матеріалу теми даного заняття |
| Мотивація мети | Повідомлення важливості вивчення даної теми: «Фундаментальні складові процесу конструювання ПЗ». Вивчення цієї теми для вас, як майбутніх фахівців з цифрових технологій, є на сьогоднішній час дуже актуальною. Відповідно до вашої професійної діяльності знання цієї теми знадобляться вам для ефективної реалізації професійних завдань. | Сприйняття важливості і актуальності вивчення теми, прояв інтересу до неї. |
| Актуалізація знань | Викладач проводить фронтальне усне опитування з метою перевірки базових знань:<br>1. Як ви розумієте значення процесу розробки програмного забезпечення?<br>2. Схарактеризуйте суть «системного аналізу»? | Здобувачі освіти беруть участь у опитуванні та відповідають на поставлені питання |

*Продовження табл. 4.5*

| 1 | 2 | 3 |
|---|---|---|
| Формування ООД | Викладач проводить консультацію згідно плану, за допомогою методу - пояснення:<br>План<br>1Визначення технології конструювання програмного забезпечення.<br>2. Класичний життєвий цикл. | Слухають пояснення, конспектують. |
| Визначення проблемних моментів під час вивчення питань теми та формування ВД | Викладач запитує здобувачів освіти про недоречності, які виникли у них під час самостійного вивчення теми. Викладач відповідає на поставлені запитання.<br>• Системний аналіз задає роль кожного елементу в комп'ютерній системі, взаємодія елементів один з одним. Оскільки програмне забезпечення є лише частиною великої системи, то аналіз починається з визначення вимог до всіх системних елементів і призначення підмножини цих вимог програмному «елементу». Необхідність системного підходу явно виявляється, коли формується інтерфейс програмного забезпечення з іншими елементами (апаратурою, людьми, базами даних). На цьому ж етапі починається рішення задачі планування проекту програмне забезпечення. | Студенти запитують:<br>1. Назвіть, що означає «Системний аналіз, як етап»?<br>2. Назвіть функції комунікацій?<br>3. Назвіть типи комунікацій? |
| Підведення підсумків | Викладач підводить підсумки проведення консультації: «Сьогодні ми розглянули незрозумілі вам питання теми. | Здобувачі освіти слухають, відповідають. |

Контурний конспект заняття з теми «Фундаментальні складові процесу конструювання ПЗ» представлено у Додатку Б.

**Висновки до розділу**

Розроблено дидактичний проєкт з теми «Фундаментальні складові процесу конструювання ПЗ» дисципліни «Конструювання програмного забезпечення» для здобувачів освіти спеціальності Професійна освіта (Цифрові технології).

Сформульовано цілі консультативного заняття. Обрано методи активізації навчальної діяльності студентів на консультації. Здійснено вибір

способів організації консультативного заняття. Розроблено сценарій проведення консультативного заняття у відповідності до обраного варіанту його організації. Проаналізовано джерела інформації для підготовки студентів до консультації згідно з робочою програмою дисципліни. Подано список використаних джерел та відповідні посилання.

# ВИСНОВКИ

У кваліфікаційній роботі відповідно до мети і завдань розкрито стан наукової проблеми. Установлено, що питання, що порушено у кваліфікаційній роботі до цього часу не були предметом вивчення науковців.

Автором теоретично обґрунтовано та експериментально перевірено методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Структурні патерни проєктування» у закладах вищої освіти засобами інноваційних технологій; вдосконалення методичного забезпечення на основі розроблення методичних рекомендацій для самостійного навчання фахівців; використання інноваційних технологій у процесі підготовки інженерів-педагогів цифрових технологій; подальшого розвитку набули зміст та засоби професійної перепідготовки інженера-педагога з цифрових технологій.

Виконана постановка нових лабораторних робіт по проєктуванню предметної області з використанням:

1. патерна «Адаптер»;
2. патерна «Декоратор»;
3. патерна «Легковаговик»;
4. патерна «Компонувальник».

Розроблено 10 предметних областей для лабораторного практикуму у якості індивідуальних варіантів для студентів.

У результаті аналізу предметної області встановлено, що У Європі та США зростає попит на перекваліфікацію для викладачів, які прагнуть отримати нові IT-компетенції. Освітні заклади активно розробляють програми для викладачів, які хочуть інтегрувати IT у свою роботу. Це дозволяє залучати до освітнього процесу фахівців з інших галузей, які готові опанувати IT-навички та передавати їх студентам.

Встановлено, що підготовка викладачів з IT у Європі та США стикається з низкою викликів, серед яких швидкий розвиток технологій, недостатня

кількість фахівців, проблеми фінансування та стандартизації. Розвиток таких ініціатив, як віртуальні лабораторії, міждисциплінарні програми та дистанційне навчання, дозволяє створювати ефективні моделі підготовки викладачів, здатних не лише викладати інформаційні технології, але й адаптувати їх до будь-якої дисципліни, роблячи навчальний процес більш ефективним і актуальним для потреб сучасного суспільства.

Розроблено дидактичний проєкт консультативного заняття з теми «Фундаментальні складові процесу конструювання ПЗ» дисципліни «Конструювання програмного забезпечення» для здобувачів інженерно-педагогічної спеціальності «Професійна освіта (Цифрові технології)».

Сформульовано цілі консультативного заняття. Обрано методи активізації навчальної діяльності студентів на консультації. Здійснено вибір способів організації консультативного заняття. Розроблено сценарій проведення консультативного заняття у відповідності до обраного варіанту його організації. Проаналізовано джерела інформації для підготовки студентів до консультації згідно з робочою програмою дисципліни. Подано список використаних джерел та відповідні посилання.

Наукова новизна одержаних результатів дослідження:

– вперше теоретично обґрунтовано та експериментально перевірено методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Структурні патерни проєктування» у закладах вищої освіти засобами інноваційних технологій; вдосконалення методичного забезпечення на основі розроблення методичних рекомендацій для самостійного навчання фахівців; використання інноваційних технологій у процесі підготовки інженерів-педагогів цифрових технологій;

– подальшого розвитку набули зміст та засоби професійної перепідготовки інженера-педагога з цифрових технологій.

Теоретичне та практичне значення одержаних результатів полягає в обґрунтуванні методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Структурні патерни

проєктування» у закладах вищої освіти засобами інноваційних технологій.

Апробація результатів дослідження виконана під час виконання лабораторного практикуму з дисципліни «Програмне забезпечення» бакалаврами спеціальності 015 Професійна освіта (Цифрові технології) Бахмутського Навчально-наукового професійно-педагогічного інституту ХНУ ім. В.Н. Каразіна у жовтні 2024 р.

За основними результатами дослідження виконана публікація тез доповідей на конференції:

– Лабораторний практикум «Структурні патерни проєктування» // Матеріали VIII міжн. наук.-практ. конф. здобувачів вищої освіти та молодих учених «Студенти та молодь – для майбутнього країни» (Харків, 14-15 листопада 2024 р.).

# СПИСОК ВИКОРИСТОВАНИХ ДЖЕРЕЛ

1. Швець О. Занурення в патерни проєктування : підручник. К.: Refactoring.Guru, 2021. 393 с. URL: https://refactoring.guru/files/design-patterns-ru-demo.pdf.

2. Ahmad A., et al. A survey on mining stack overflow: question and answering (Q&A) community. Data Technologies and Applications, 2018, 52.2. Pp. 190-247.

3. Bass L., Clements P., Kazman R. Software Architecture in Practice. Software Architecture in Practice: Software Architect Practice. Addison-Wesley, 2012.

4. Breivold H., Crnkovic I., Larsson M. A systematic review of software architecture evolution research. Information and Software Technology, 2012, 54.1: Pp. 16-40.

5. Code Club Україна – всеукраїнська мережа безкоштовних клубів кодування. URL: https://codeclub.com.ua/about.html

6. Design Patterns : elements of reusable object-oriented software / Erich Gamma ... [et al.]. Addison-Wesley professional computing series. 1994. P. 396.

7. Dingle A. Software Essentials. Design and Construction. Chapman & Hall, 2020. 436 p. ISBN: 9780367659134.

8. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2012.

9. Freeman E., Robson E. Head First Design Patterns. O'Reilly Media, 2020.

10. Glossary of Software Engineering terms. URL: http://www.shellmethod.com/refs/ seglossary.pdf.

11. Hackernoon. Software Architecture Basics: What, How & Why. URL: https://hackernoon.com/software-architecture-primer

12. IEEE Std 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology. URL: http://standards.ieee.org/reading/ieee/std_public/description/

13. IEEE-CS/ACM Software Engineering Ethics and Professional Practices. URL: https://ethics.acm.org/code-of-ethics/software-engineering-code/.

14. InfoQ Software Architecture and Design Trends Report. URL: https://www.infoq.com/articles/architecture-trends-2024/

15. IT Arena. URL: https://itarena.ua/ua/

16. Lelek T., Jon Skeet J. Software Mistakes and Tradeoffs Software Mistakes and Tradeoffs: How to make good programming decisions". Manning Publications Co, Shelter Island, 2022. P. 416.

17. Nugroho Yosep Novento, Kusumo Dana Sulistyo, Alibasa, Muhammad Johan. Clean architecture implementation impacts on maintainability aspect for backend system code base. In: 2022 10th International Conference on Information and Communication Technology (ICoICT). IEEE, 2022. Pp. 134-139.

18. Reddit. r/softwarearchitecture. URL: https://www.reddit.com /r/ softwarearchitecture/?rdt=41013

19. Rylander S. Patterns of Software Construction: How to Predictably Build Results. Apress, 2022. 156 p. ISBN: 9781484279359.

20. se/610.12-1990_desc.html.

21. Software Architecture (SOLID) & Design Patterns in Java – Онлайн-курс Udemy. URL: https://www.udemy.com/course/basics-of-software-architecture-design-in-java/.

22. Syeed M., Hammouda I., Systä T. Evolution of open source software projects: A systematic literature review. J. Softw., 2013, 8.11: Pp. 2815-2829.

23. The .NET Compiler Platform SDK – MS Learn. URL: https://learn.microsoft.com/en-us/dotnet/csharp/roslyn-sdk/.

24. ThoughtWorks. Software Architecture: The Hard Parts. URL: https://www.thoughtworks.com/insights/books/software-architecture-hard-parts

25. Баран С. В. Розробка програмного забезпечення з використанням патернів проєктування: навч. посіб. Кривий Ріг, 2023. 203 с.

26. Відео курси з програмування та IT професій. URL: https://itvdn.com/ua

27. Ключові методології розробки програмного забезпечення: робота команди зсередини. URL: https://wezom.com.ua/ua/blog/metodologija-razrabotki-programmnogo-obespechenija

28. Конспект лекцій з дисципліни «Конструювання програмного забезпечення» для здобувачів вищої освіти першого (бакалаврського) рівня спеціальності 121 «Інженерія програмного забезпечення» очної і заочної форм навчання / Уклад. К.В. Яшина, К.М. Ялова, Н.М. Лимар. Кам'янське: ДДТУ, 2019 р. 75 с.

29. Мартін Роберт. Чиста архітектура. 2 вид. Х.: Фабула, 2019. 368 с. ISBN: 978-6-17-095286-8.

30. Олійник А.В., Шацька В.М. Інформаційні системи і технології у фінансових установах. 2006.

31. Поліщук А.М., Ніколюк П.К. Технологія програмування та основні етапи її розвитку. Комп'ютерні технології обробки даних, 2022, 89-91.

32. Самчинська Я.Б., Вінник М.О. Просування й розповсюдження педагогічного програмного забезпечення на ринку України. Інформаційні технології в освіті, 2013. № 15. – С. 210-220.

33. Цибульник С. О., Барандич К. С. Технології розроблення програмного забезпечення. Ч. 1. Життєвий цикл програмного забезпечення [Електронний ресурс] : підручник. Київ : КПІ ім. Ігоря Сікорського, 2022.

34. Чикунов П.О. Рефакторинг коду при конструюванні програмного забезпечення. Актуальні тенденції розвитку освіти, науки та технологій : матер. VI Міжнар. наук.-практ. конф. у 2-х ч. Бахмут – Харків: ННППІ УІПА, 2023. Ч 1. С. 98-100.

35. Чикунов П.О. Силабус навчальної дисципліни «Проєктування програмного забезпечення» освітньо-професійної програми «Інженерія програмного забезпечення». Одеса, НУ «ОЮА», 2024. – С. 8.

36. Методика формування пошуково-дослідницьких умінь майбутніх інженерів-педагогів у процесі професійної підготовки: колективна монографія

/ В.В.Кулешова, В.В.Мальована. – Артемівськ: ННППІ УІПА, 2012. – 264 с. (власний внесок: Р1; Р2 с.86-92; Р3; 9,8 д.а.).

37. Коваленко О. Е. Концепція професійно-педагогічної підготовки студентів інженерно-педагогічних спеціальностей / О. Е. Коваленко, Н. О. Брюханова, О. О. Мельниченко // Проблеми інженерно-педагогічної освіти: зб. наук. праць. − Х.: УІПА, 2005. − № 10. − С. 7–20.

38. Кулешова В. В. Техніка управлінської діяльності: [навч.посіб.] (рекомендовано МОН України лист № 1/II-10232 від 04.11.11р.) /Д. В. Коваленко, І. М. Шалімова, А. С. Шалімова В. В. Кулешова – Харків: «ФОП Шевченко», 2011. – С.4–43 (особ. вн.: 2 д.а.).

39. Information and Computer Support for Adaptability of Learning in Higher Education Institutions Kovalenko, O., Briukhanova, N., Bondarenko, T., Yaschun, T. Advances in Intelligent Systems and Computing, 2020, 1135 AISC, с. 145-153

40. Формування психолого-педагогічної компетентності викладачів технічних дисциплін у системі післядипломної освіти: Монографія для студентів інженерно-педагогічних, технічних вищих навчальних закладів, науково-педагогічних працівників у галузі інженерно-педагогічної та технічної освіти, аспірантів, керівників ПТНЗ, Харків: : Вид-во ТОВ «Щедра садиба плюс», 2014., 442с.

41. Формування професійної компетентності викладачів технічних дисциплін: колективна монографія / В.В.Кулешова, В.В.Мальована, Ю.С.Бобрикова. – Х., 2020. – 206 с. (власний внесок: Р1 с.8-94; Р2 с.95-100; Р3с.146-159; 6,5 д.а.).

42. Viktoriia Kuleshova, Viktoriia Malovana. Methodological approaches to the formation of psychological and pedagogical competence of teachers of technical disciplines // SCHOLA 2019 R&E-SOURCE https://journal.ph-noe.ac.at Online Journal for Research and Education Special Issue 17, Dec. 2019, ISSN 2313-1640 – P.196 -206.

43. Кулешова В., Разумовська Н. (2020). Спрямованість як інтегрований показник цінності професійно-педагогічної орієнтації особистості. Vol. 7, No. 4. Ternopil-Aberdeen, 2020. pp. 524-536. DOI: 10.25128/2520-6230.20.4.9.

44. Кулешова В.В. Формування соціальної компетентності у майбутніх фахівців педагогічних спеціальностей. Педагогічна академія: наукові записки. 2024 № 11.https://doi.org/10.5281/zenodo.14052575.

45. UNESCO's ICT Competency Framework for Teachers. URL: https://www.unesco.org/en/digital-competencies-skills/ict-cft

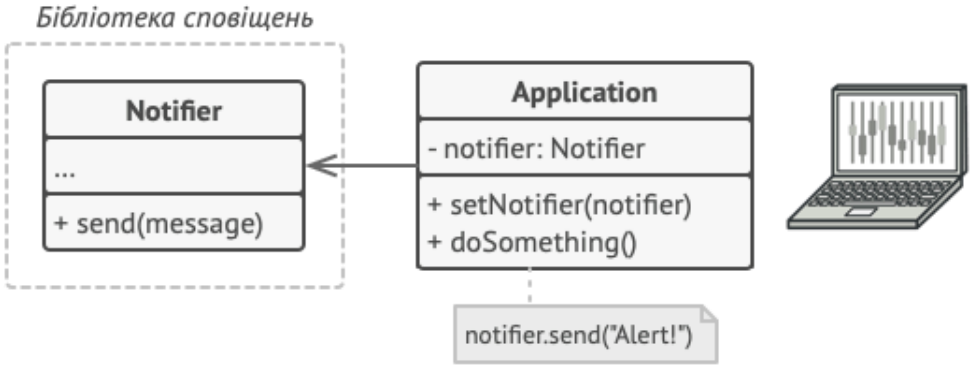
46. Moving TVET to the top of the global agenda. URL: https://worldskills2024.com/en/conference/

# ДОДАТОК А КОНТУРНИЙ КОНСПЕКТ НА ТЕМУ «ФУНДАМЕНТАЛЬНІ СКЛАДОВІ ПРОЦЕСУ КОНСТРУЮВАННЯ ПЗ»

**Декоратор** – це структурний патерн проектування, що дає змогу динамічно додавати об'єктам нову функціональність, загортаючи їх у корисні «обгортки».

Ви працюєте над бібліотекою сповіщень, яку можна підключати до різноманітних програм, щоб отримувати сповіщення про важливі події.

Основою бібліотеки є клас Notifier з методом send, який приймає на вхід рядок-повідомлення і надсилає його всім адміністраторам електронною поштою. Стороння програма повинна створити й налаштувати цей об'єкт, вказавши, кому надсилати сповіщення, та використовувати його щоразу, коли щось відбувається.

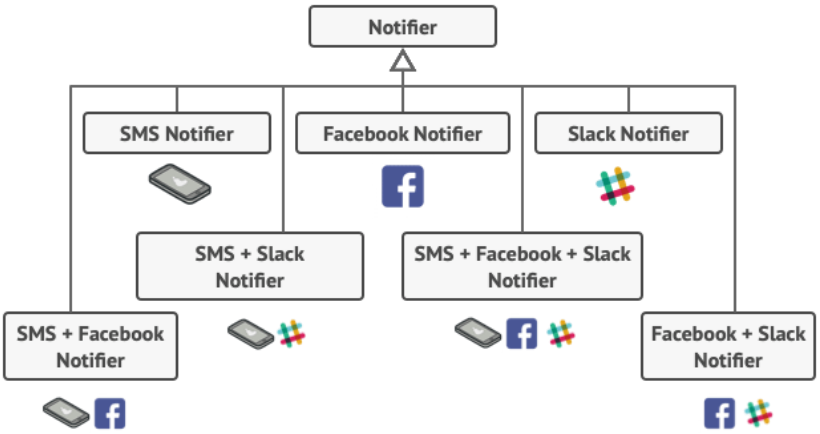



В якийсь момент стало зрозуміло, що користувачам не вистачає одних тільки email-сповіщень. Деякі з них хотіли б отримувати сповіщення про критичні проблеми через SMS. Інші хотіли б отримувати їх у вигляді Facebook-повідомлень. Корпоративні користувачі хотіли би бачити повідомлення у Slack.


Спершу ви додали кожен з типів сповіщень до програми, успадкувавши їх від базового класу Notifier. Тепер користувачі могли вибрати один з типів сповіщень, який і використовувався надалі.

Але потім хтось резонно запитав, чому не можна увімкнути кілька типів сповіщень одночасно? Адже, якщо у вашому будинку раптом почалася пожежа, ви б хотіли отримати сповіщення по всіх каналах, чи не так?

Ви зробили спробу реалізувати всі можливі комбінації підкласів сповіщень, але після того, як додали перший десяток класів, стало зрозуміло, що такий підхід неймовірно роздуває код програми.




Отже, потрібен інший спосіб комбінування поведінки об’єктів, який не призводить до збільшення кількості підкласів.

Спадкування – це перше, що приходить в голову багатьом програмістам, коли потрібно розширити яку-небудь чинну поведінку. Проте механізм спадкування має кілька прикрих проблем.

- Він **статичний**. Ви не можете змінити поведінку об’єкта, який вже існує. Для цього необхідно створити новий об’єкт, вибравши інший підклас.
- Він **не дозволяє наслідувати поведінку декількох класів одночасно**. Тому доведеться створювати безліч підкласів-комбінацій, щоб досягти поєднання поведінки.

Одним зі способів, що дозволяє обійти ці проблеми, є заміна спадкування *агрегацією* або *композицією*. Це той випадок, коли один об’єкт *утримує* інший і делегує йому роботу, замість того, щоб самому *успадкувати* його поведінку. Саме на цьому принципі побудовано патерн Декоратор.


Рисунок – Спадкування проти Агрегації

Декоратор має альтернативну назву – *обгортка*. Вона більш вдало описує суть патерна: ви розміщуєте цільовий об’єкт у іншому об’єкті-обгортці, який запускає базову поведінку об’єкта, а потім додає до результату щось своє.

Обидва об’єкти мають загальний інтерфейс, тому для користувача немає жодної різниці, з чим працювати – з чистим чи загорнутим об’єктом. Ви можете використовувати кілька різних обгорток одночасно – результат буде мати об’єднану поведінку всіх обгорток.

В нашому прикладі зі сповіщеннями залишимо в базовому класі просте надсилання сповіщень електронною поштою, а розширені способи зробимо декораторами.

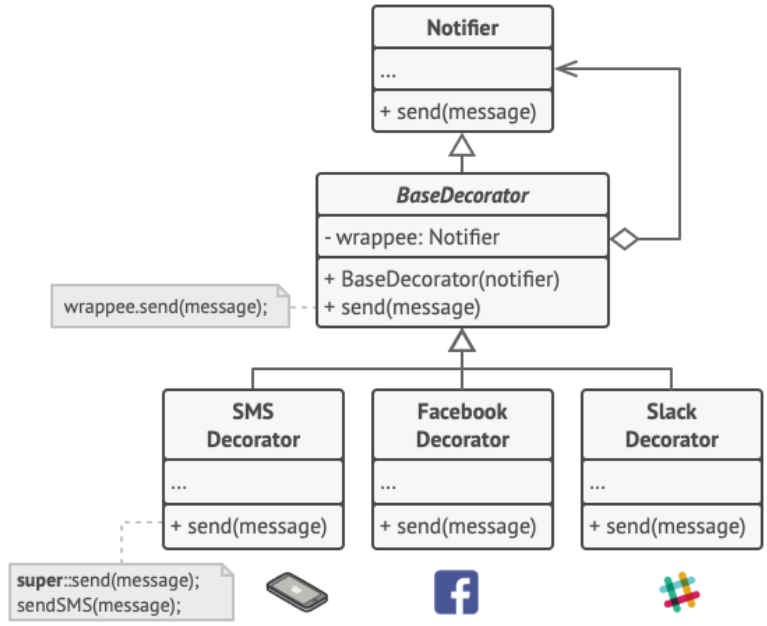



Рисунок – Розширені способи надсилання сповіщень стають декораторами

Стороння програма, яка виступає клієнтом, під час початкового налаштовування буде загортати об’єкт сповіщення в ті обгортки, які відповідають бажаному способу сповіщення.

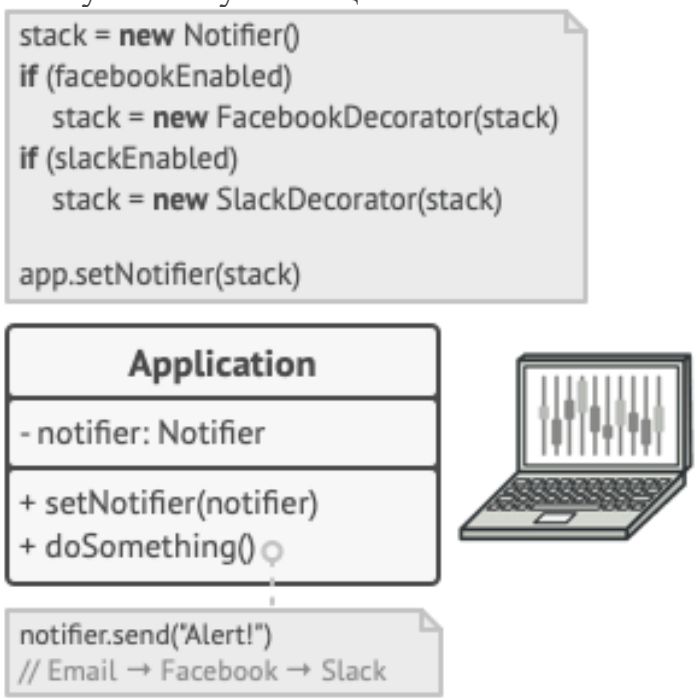



Рисунок – Програма може збирати складові об’єкти з декораторів

Остання обгортка у списку буде саме тим об’єктом, з яким клієнт працюватиме увесь інший час. Для решти клієнтського коду нічого не зміниться, адже всі обгортки мають такий самий інтерфейс, що і базовий клас сповіщень.

Так само можна змінювати не тільки спосіб доставки сповіщень, але й форматування, список адресатів і так далі. До того ж клієнт зможе «дозагорнути» об’єкт у будь-які інші обгортки, якщо йому цього захочеться.

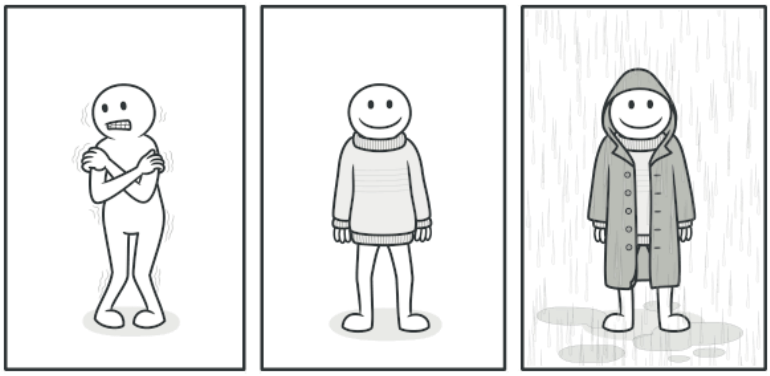


Рисунок – Одяг можна одягати кількома шарами

Будь-який одяг – це аналог Декоратора. Застосовуючи Декоратор, ви не змінюєте початковий клас і не створюєте дочірніх класів. Так само з одягом: вдягаючи светра, ви не перестаєте бути собою, але отримуєте нову властивість – захист від холоду. Ви можете піти далі й одягти зверху ще один декоратор – плащ, щоб захиститися від дощу.

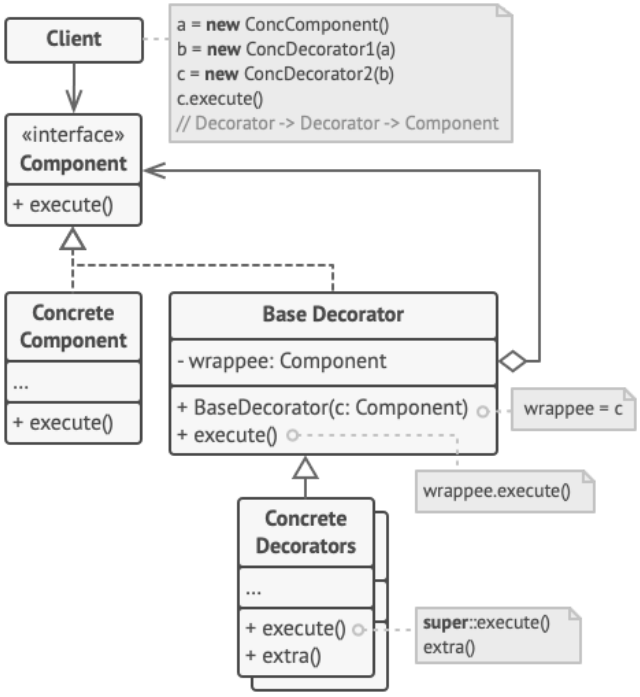



Рисунок – Структура ієрархії

Компонент задає загальний інтерфейс обгорток та об’єктів, що загортаються.

Конкретний компонент визначає клас об’єктів, що загортаються. Він містить якусь базову поведінку, яку потім змінюють декоратори.

Базовий декоратор зберігає посилання на вкладений об’єкт-компонент. Це може бути як конкретний компонент, так і один з конкретних декораторів. Базовий декоратор делегує всі свої операції вкладеному об’єкту. Додаткова поведінка житиме в конкретних декораторах.

Конкретні декоратори – це різні варіації декораторів, що містять додаткову поведінку. Вона виконується до або після виклику аналогічної поведінки загорнутого об’єкта.

Клієнт може обертати прості компоненти й декоратори в інші декоратори, працюючи з усіма об’єктами через загальний інтерфейс компонентів.

# ДОДАТОК Б ПУБЛІКАЦІЯ ЗА РЕЗУЛЬТАТАМИ ДОСЛІДЖЕННЯ

## ЛАБОРАТОРНИЙ ПРАКТИКУМ «СТРУКТУРНІ ПАТЕРНИ ПРОЄКТУВАННЯ»

*Автор: Гринаць Д.Р., магістр*
*Науковий керівник: Чикунов П.О., к.т.н., доц.*
*Бахмутський навчально-науковий професійно-педагогічний інститут ХНУ імені В. Н. Каразіна*

Метою розробки лабораторного практикуму «Структурні патерни проєктування» є ознайомлення студентів цифрових технологій із принципами використання структурних шаблонів у процесі розробки програмного забезпечення, розвиток навичок їхнього практичного застосування, а також аналіз переваг та недоліків використання таких патернів.

Лабораторний практикум на тему «Структурні патерни проєктування» є важливим етапом вивчення програмної інженерії, який дозволяє студентам оволодіти принципами і методами створення ефективних та гнучких програмних рішень. Структурні патерни, або шаблони проєктування, представляють собою усталені підходи до організації взаємодії між класами і об'єктами в програмі [1]. Вони дозволяють оптимізувати структуру коду, знизити його складність і підвищити зрозумілість, що в кінцевому результаті сприяє легкості супроводу та модифікації програмного продукту. Структурні патерни забезпечують ефективну організацію коду, що дозволяє підвищити модульність, повторне використання компонентів, а також зменшити зв'язаність між ними.

Серед основних структурних патернів, які розглядаються під час лабораторного практикуму, виділяються такі.

Патерн «Адаптер» дозволяє поєднати класи, які мають несумісні інтерфейси, шляхом перетворення одного інтерфейсу в інший. Це дозволяє використати існуючі класи без необхідності їх змінювати, що знижує витрати на адаптацію коду.

Патерн «Декоратор» дозволяє динамічно додавати нову функціональність до об'єкта, не змінюючи його структуру. Цей патерн є альтернативою наслідуванню, забезпечуючи гнучкіший спосіб розширення можливостей класів.

Патерн «Легковаговик» оптимізує використання пам'яті при створенні великої кількості однотипних об'єктів за рахунок спільного використання загальних даних. Це дозволяє зменшити кількість об'єктів у системі і, відповідно, знизити витрати на їх зберігання.

Використання структурних патернів у проєктуванні програмного забезпечення має безліч переваг. Перш за все, це сприяє створенню більш гнучкої і масштабованої архітектури, що дозволяє легше адаптувати систему до нових вимог. По-друге, патерни допомагають розробникам ефективно взаємодіяти з великими обсягами коду, роблячи його більш зрозумілим і організованим. По-третє, структурні патерни сприяють повторному використанню коду, що знижує витрати на розробку та підтримку програмного продукту.

Виконана постановка трьох лабораторних робіт для опанування навичок проєктування предметної області з використанням структурних патернів «Адаптер», «Декоратор» та «Легковаговик».

Лабораторний практикум «Структурні патерни проєктування» є важливою складовою підготовки майбутніх фахівців з цифрових технологій. Він не лише надає можливість засвоїти основи застосування патернів на практиці, але й сприяє розвитку критичного мислення у виборі найбільш ефективних рішень для розв'язання конкретних завдань. Опанування структурних патернів є необхідним елементом професійного зростання кожного розробника, що прагне створювати якісне, надійне і масштабоване програмне забезпечення.

## Список використаних джерел

1. Швець О. Занурення в патерни проєктування : підручник. К.: Refactoring.Guru, 2021. 393 с. URL: https://refactoring.guru/files/design-patterns-ru-demo.pdf.