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 «Породжувальні патерни проєктування» у закладах вищої освіти»

Виконав(ла)

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

Олександр РЯБУХА _____
(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
к.пед.н., доц. Юлія БОБРИКОВА _____
(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)

Рябуха Олександр Віталійович
(прізвище, ім'я, по батькові здобувача)

1. Тема роботи Професійна підготовка фахівців з цифрових технологій для викладання освітнього модулю «Породжувальні 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 № 5101-5/3236

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 «Породжувальні 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 «Породжувальні 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 «Породжувальні 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 «Породжувальні 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) _____ Олександр РЯБУХА
(підпис) (ім'я, прізвище)

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

РЕФЕРАТ

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

Об'єктом дослідження роботи є процес професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Породжувальні 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 «Породжувальні патерни проєктування».

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

Виконано аналіз характеристик об'єктів галузі проєктування програмного забезпечення. Виконана постановка 3 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 НПК «Студенти та молодь – для майбутнього країни» (Бахмут-Харків, 14-15 листопада 2024 р.).

Обсяг дипломної роботи становить: пояснювальна записка, презентація доповіді. Пояснювальна записка складається з вступу, чотирьох розділів, висновків, списку використаних джерел, додатків. Загальний обсяг роботи 80 сторінок, з яких 60 сторінок основного тексту. Список використаних джерел становить 60 найменувань, 5 таблиць, 16 рисунків.

ПРОФЕСІЙНА ПІДГОТОВКА, ЛАБОРАТОРНИЙ ПРАКТИКУМ, ПАТЕРНИ, КАДРОВЕ ЗАБЕЗПЕЧЕННЯ, МЕТОДИЧНА РОЗРОБКА

ABSTRACT

Purpose of the study: to theoretically substantiate and partially test the methodology for the professional training of digital technology specialists to teach the educational module "Creational Design Patterns" in higher education institutions.

Object of the study: the process of professional training of digital technology specialists to teach the educational module "Creational Design Patterns" in higher education institutions.

Subject of the study: the methodology for the professional training of digital technology specialists to develop a set of digital educational tools for teaching the educational module "Creational Design Patterns."

The study characterizes the system of professional training for digital technology specialists to teach the educational module "Creational Design Patterns" in higher education institutions.

An analysis of the characteristics of objects in the field of software design was conducted. Three new laboratory exercises were proposed. An analysis of staffing requirements for IT sector entities under conditions of societal digital transformation was carried out.

A didactic project for a consultative lesson was developed.

Based on the main research results, a thesis of the report was published at the VIII International Scientific and Practical Conference "Students and Youth – for the Future of the Country" (Bakhmut-Kharkiv, November 14-15, 2024).

The diploma project consists of: an explanatory note and a presentation of the report. The explanatory note includes an introduction, four chapters, conclusions, a list of references, and appendices. The total length of the work is 80 pages, including 60 pages of main text. The list of references includes 60 items, 5 tables, and 16 figures.

Keywords: professional training, laboratory practicum, patterns, staffing, methodological development.

ЗМІСТ

Вступ 7
Розділ 1 Актуальність професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Породжувальні патерни проєктування» у закладах вищої освіти 11
Висновки до розділу 16
Розділ 2 Характеристика об'єктів галузі: стан і стратегії розвитку 18
2.1 Огляд породжувальних патернів проєктування 18
2.2 Постановка лабораторної роботи «Проєктування предметної області з використанням патерна “Фабричний метод”» 21
2.3 Постановка лабораторної роботи «Проєктування предметної області з використанням патерна “Абстрактна фабрика”» 29
2.4 Постановка лабораторної роботи «Проєктування предметної області з використанням патерна “Одинак”» 37
Висновки до розділу 44
Розділ 3 Вимоги до кадрового забезпечення об'єкту галузі 46
Висновки до розділу 51
Розділ 4 Методика професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Породжувальні патерни проєктування» у закладах вищої освіти 52
4.1 Вихідні дані до проєкту 52
4.2 Проектування цілей консультативного заняття 53
4.3 Перелік джерел інформації 55
4.4 Визначення найбільш складних для розуміння та засвоєння питань 58
4.5 Вибір дидактичних методів активізації 59
4.6 Вибір способів організації консультативного заняття 60
4.7 Розробка сценарію проведення консультативного заняття 61
Висновки до розділу 65
Висновки 66
Список використаних джерел 68
Додатки 74

ВСТУП

Швидкий розвиток сучасного суспільства, цифрових технологій та інноваційних засобів навчання вимагають від майбутніх фахівців нових професійних знань та вмінь, перегляду підходів щодо формування їх професійної компетентності. Активне використання цифрових технологій в освіті сприяє ефективності освітнього процесу на всіх його рівнях і формуванню професійних компетентностей майбутніх фахівців. Формування компетентностей нерозривно пов’язане з інформатизацією освіти.

Це висуває нові вимоги до професійної підготовки інженера-педагога: він повинен бути компетентним, освіченим, проінформованим, мати розвинене мислення. Професіоналізм майбутніх інженерів-педагогів має пріоритетне значення та його формування є необхідною умовою інтенсифікації професійної освіти.

Отже, виникає потреба пошуку нових підходів у вирішенні проблеми формування професіоналізму у майбутніх інженерів-педагогів, а саме фахівців з цифрових технологій для викладання освітнього модуля «Породжувальні патерни проєктування» у закладах вищої освіти.

Наукові дослідження та наявні з цих питань публікації в теорії та практиці професійної освіти показують, наскільки ця проблема складна та багатогранна. У педагогічній та психолого-педагогічній літературі існує величезна кількість досліджень щодо підвищення рівня професіоналізму.

У дослідженні професіоналізму діяльності ми спираємося на педагогічні теорії професійного навчання (Н. Астаф'єва, С. Батишев, А. Бєляєва, І. Богачек, Б. Гершунський, В. Кричевський, Н. Соколова та ін); теорію та методологію професійного розвитку особистості (Р. Бабанський, А. Бєляєва, Н. Гейжен, Є. Головаха, В. Давидов, І. Жданов, В. Загвязинський, О . Конопкін, Е. Мілерян, А. Рудкевич, Є. Рибалко, П.СХейфіц); систему формування педагогічної майстерності (Ю. Азаров, В. Аніськін, Ю.Бабанський, М. Губанова, В. Ворошилова, З. Єсарєва, Е. Зеєр, І. Зимова, І. Зязюн, В. Іванов, Н.

Касаткіна, Н. Кузьміна, Ю. Кулюткін, А. Макаренко, А. Маркова, Б. Невзоров, В. Рябцев, Г. Скок та ін.).

Питанням педагогічного покликання та її розвитку присвячені праці Л. Ахмедянової, В. Єлманової, З. Жуковський, Н. Кузьміною та ін; успішності діяльності педагога (З. Єсарєва, Н. Кухарєв, Г. Лаврентьєв, Н. Лаврентьєва, А. Маркова, Г. Скок та ін.). Основні положення педагогіки вищої школи та дидактики розглянуті у роботах С. Архангельського, Ю. Бабанського, В. Беспалько, Н. Кузьміної, Н. Нікандрова, В. Сластеніна, Т. Шалавіної, А. Щербакова та ін.

У сучасній теорії та практиці професійної освіти не отримали належного обґрунтування питання формування професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Породжувальні патерни проєктування» у закладах вищої освіти.

Враховуючи об'єктивні потреби підготовки таких спеціалістів та відсутність наукового обґрунтування в умовах інтегративного, системного підходів до вирішення проблеми, визначено тему дослідження: «Професійна підготовка фахівців з цифрових технологій для викладання освітнього модуля «Породжувальні патерни проєктування» у закладах вищої освіт».

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

Завдання дослідження:

1. Визначити ступінь актуальності проблеми професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Породжувальні патерни проєктування» у закладах вищої освіт».

2. Теоретично обґрунтувати, розробити методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Породжувальні патерни проєктування» у закладах вищої освіт».

Об'єкт дослідження: процес професійної підготовки фахівців з цифрових

технологій для викладання освітнього модуля «Породжувальні патерни проєктування» у закладах вищої освіт».

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

Методи дослідження:

– загальнонаукові (аналіз, синтез, систематизація, зіставлення, узагальнення) з метою систематизації теоретичних ідей та узагальнення досвіду формування компетентностей фахівців інженерно-педагогічного профілю зі спеціальності 015 Професійна освіта (Цифрові технології);

– емпіричні (навчання та узагальнення педагогічного досвіду, педагогічне спостереження, анкетування, тестування, педагогічний експеримент), для діагностування рівня сформованості професійних компетентностей у здобувачів вищої освіти зі спеціальності 015 Професійна освіта (Цифрові технології);

– педагогічний експеримент з метою перевірки методики професійної перепідготовки спеціаліста в галузі цифрових технологій з метою формування його інноваційної стратегії.

Наукова новизна одержаних результатів дослідження: полягає у постановці та вирішенні проблеми професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Породжувальні патерни проєктування» у закладах вищої освіт», що стала предметом спеціального наукового дослідження, під час якого:

– обґрунтовано мету професійної підготовки у вигляді формування професіоналізму у педагогів професійного навчання як його професійно-особистісної характеристики та феномена безперервної професійної підготовки;

– розроблено модель цілісного керованого процесу формування професіоналізму у освітян професійного навчання;

– реалізовано процес формування професіоналізму через єдність

галузевої, психолого-педагогічної підготовки та підготовки за робітничою професією, що визначає систему наступних технологій (виробничих, педагогічних);

– виділено особливості формування професіоналізму, що визначаються послідовністю досягнення цілей на кожному етапі професійної підготовки.

Практична значущість дослідження полягає у розробці та реалізації навчального освітнього модуля «Породжувальні патерни проєктування», що забезпечує підвищення рівня професіоналізму майбутніх фахівців з цифрових технологій для викладання у закладах вищої освіти».

Теоретичне та практичне значення одержаних результатів полягає в тому, що досліджено та уточнено зміст понять «професіоналізм», «формування професіоналізму у майбутніх педагогів у ЗВО», виявлено зовнішні чинники (соціально-економічні умови розвитку суспільства, громадські організації, сім'я, соціокультурні інститути) та внутрішні фактори (ціннісні орієнтації, потреби та здібності особистості педагога), що впливають на процес формування у фахівців з цифрових технологій навичок викладання освітнього модуля «Породжувальні патерни проєктування» у закладах вищої освіти, визначено структурні компоненти моделі формування професіоналізму у фахівців з цифрових технологій у ЗВО (цільовий, змістовний, процесуальний та результативний блоки), що забезпечують безперервний процес розвитку професіоналізму у майбутніх у фахівців з цифрових технологій у ЗВО.

Матеріали дослідження використовувались при підготовці здобувачів вищої освіти зі спеціальності 015 Професійна освіта (Цифрові технології) у Бахмутському навчально-науковому професійно-педагогічному інституті ХНУ імені В. Н. Каразіна.

За основними результатами дослідження виконана доповідь на міжнародній науково-практичній конференції здобувачів вищої освіти та молодих учених «Студенти та молодь – для майбутнього країни» (Бахмут-Харків, 17 листопада 2024 р.).

РОЗДІЛ 1 АКТУАЛЬНІСТЬ ПРОФЕСІЙНОЇ ПІДГОТОВКИ ФАХІВЦІВ З ЦИФРОВИХ ТЕХНОЛОГІЙ ДЛЯ ВИКЛАДАННЯ ОСВІТНЬОГО МОДУЛЯ «ПОРОДЖУВАЛЬНІ ПАТЕРНИ ПРОЄКТУВАННЯ» У ЗАКЛАДАХ ВИЩОЇ ОСВІТИ

Реформування освіти в Україні на сучасному етапі нерозривно пов’язане з переходом до інформаційного суспільства та інтеграцією в глобальну освітню спільноту. Ця спільнота характеризується активним використанням і впровадженням цифрових технологій у навчальний процес та управління освітою.

Цифрові технології в освіті виконують декілька ключових функцій: вони стають предметом вивчення в рамках загальних навчальних курсів і спеціалізованих дисциплін, засобом викладання та розробки методичних матеріалів. Їх використання збагачує зміст навчальних курсів, демонструючи сучасні підходи до вирішення фундаментальних і прикладних задач за допомогою комп’ютерних технологій. Цифрові засоби сприяють реалізації мовних стратегій, оптимізують управління освітніми установами та виступають інструментом інформаційного забезпечення педагогічної діяльності.

Сучасні заклади вищої освіти (ЗВО) зосереджують свої зусилля на розвитку професійних компетентностей студентів у сфері цифрових технологій, що є невід’ємною ознакою висококваліфікованого фахівця. Їхнє завдання полягає не лише у забезпеченні функціонування освітнього процесу, але й у формуванні у студентів навичок аналізу, структурування й передачі інформації.

Особливо важливим аспектом є розвиток у студентів мотивації до постійного самовдосконалення, здатності до самостійного пошуку нових знань і готовності адаптуватися до швидких змін у професійному середовищі. Завдяки таким підходам освіта стає інструментом формування фахівців, здатних ефективно діяти в умовах цифрової трансформації суспільства.

Ключовою характеристикою фахівця з інформаційних технологій є його компетентність, що відображає рівень готовності та здатності виконувати професійні завдання на високому рівні [46].

Науковці визначають компетентність як багатогранну властивість особистості, яка базується не лише на здобутих знаннях, вміннях і навичках, але й на досвіді, обізнаності та здатності застосовувати їх у професійній діяльності. Згідно з позицією О. І. Бондарчука [52], В. О. Болотова та В. В. Сєрікова, компетентність – це узагальнений термін, що позначає здатність до компетентної діяльності, тобто діяльності, яка виконується зі знанням справи. Вона визначає відповідність знань і вмінь спеціаліста змісту та складності завдань, що перед ним постають.

Водночас поняття «компетентність» часто використовується як синонім терміна «кваліфікація». Однак кваліфікація є офіційним результатом оцінювання та визнання досягнутих компетентностей (результатів навчання) відповідно до вимог стандартів вищої освіти. Її підтвердженням виступає документ про вищу освіту [50].

Таким чином, компетентність і кваліфікація є взаємопов’язаними поняттями: компетентність відображає фактичний рівень знань, умінь і навичок, тоді як кваліфікація є формальним підтвердженням цих досягнень. Забезпечення розвитку обох аспектів є одним із пріоритетних завдань сучасної системи вищої освіти.

Філософському осмисленню компетентнісного підходу та формуванню особистості фахівця присвятили свої праці багато науковців і практиків, зокрема В. Андрущенко, Н. Бібік, Л. Бірюк, В. Бобрицька, Л. Васильченко, Л. Ващенко, К. Віаніс-Трофименко, І. Галяміна, С. Гончаренко, О. Дубасенюк, І. Зимня, І. Зязюн, В. Кремень, О. Локшина, О. Овчарук, В. Огнев’юк, Л. Паращенко, О. Пометун, Дж. Равен, О. Савченко, С. Сисоєва, В. Ягупов та інші.

На думку цих дослідників, сучасні заклади освіти мають забезпечувати підготовку компетентних фахівців, здатних ефективно вирішувати навчально-

виховні завдання, спрямовані на формування особистості здобувачів освіти. Компетентність, у свою чергу, розкриває здатність до активних дій, вирішення проблемних ситуацій, мобілізації знань, досвіду, цінностей та умінь [38].

Розвиток цифрових технологій супроводжується зростанням потреби у спеціалізованих знаннях з цифрових комунікацій, що зумовлює нові вимоги до фахівців. Вони повинні бути готові надавати здобувачам освіти поглиблені знання у сфері цифрових технологій, розвивати їхні навички роботи з сучасними цифровими інструментами та технологіями.

Сучасний освітній процес вимагає активного використання інтерактивних методів навчання та цифрових інструментів. Педагогічні працівники мають володіти навичками використання цифрових комунікацій, що сприятиме підвищенню ефективності та якості навчального процесу. Це передбачає постійне вдосконалення професійної майстерності педагогів, а також адаптацію освітніх методів до нових умов цифрової трансформації суспільства.

Фахівці, які можуть ефективно викладати цифрові комунікації, стають більш конкурентоспроможними на ринку праці. Роботодавці шукають випускників з високим рівнем цифрової грамотності та навичками в області цифрових комунікацій. Разом із цим у педагогічній теорії та практиці недостатньо розроблено проблему яка б враховувала сучасні тенденції розвитку професійних компетентностей фахівців з цифрових технологій [40].

Фахівці з цифрових технологій повинні не лише розуміти сучасні технології цифрових комунікацій, але і бути здатними ефективно передавати ці знання здобувачам освіти. Сучасні здобувачі освіти вирізняються високим рівнем цифрової грамотності та очікують від навчання високотехнологічних підходів, включаючи використання цифрових комунікацій. Тобто, за такого підходу, важливим стає не наявність внутрішньої організації знань, а здатність застосовувати компетентності в професійній діяльності [37].

Педагогічна система формування професійної компетентності фахівців із цифрових технологій у процесі їхньої підготовки повинна базуватися на

інтеграції сучасних методів навчання, орієнтованих на актуальні потреби галузі. Основними компонентами такої системи є таке.

1. Актуальність змісту навчання. Забезпечення актуальності навчальних програм шляхом їх розробки та регулярного оновлення відповідно до сучасних тенденцій у цифрових технологіях. Це включає вивчення інноваційних технологій, розгляд викликів галузі та виконання практичних завдань, що відображають реальні сценарії професійної діяльності.

2. Інтерактивні методи навчання. Використання інтерактивних підходів для активного залучення здобувачів освіти до навчального процесу. Це можуть бути лекції з елементами дискусій, практичні лабораторні роботи, моделювання реальних ситуацій, робота з кейсами, а також навчання через гру, що стимулює творче мислення.

3. Практична орієнтація. Організація практик, стажувань і проєктної діяльності, спрямованих на набуття практичного досвіду. Важливо створювати умови для того, щоб студенти могли застосовувати отримані теоретичні знання у вирішенні практичних задач та розробці цифрових рішень.

4. Розвиток критичного мислення. Формування навичок критичного аналізу, необхідних для оцінювання інформації, прогнозування результатів і вирішення проблем у сфері цифрових технологій. Для цього доцільно використовувати аналіз реальних кейсів, розбір помилок і роботу з альтернативними сценаріями.

5. Індивідуалізований підхід. Урахування індивідуальних особливостей здобувачів освіти, їхніх темпів навчання, інтересів і потреб. Надання можливостей для самостійної роботи, зокрема виконання індивідуальних проєктів, що відповідають особистим уподобанням у сфері цифрових технологій.

6. Інтеграція сучасних технологій у навчання. Використання сучасних цифрових інструментів у навчальному процесі, таких як віртуальна та доповнена реальність, інтерактивні відеолекції, онлайн-курси, симуляції та

платформи для командної роботи. Це дозволяє створити максимально реалістичні умови, що відповідають майбутній професійній діяльності здобувачів освіти.

7. Мотивація до постійного навчання. Сприяння розвитку у студентів мотивації до безперервного вдосконалення знань і навичок у швидкозмінному світі цифрових технологій. Важливим є акцент на самостійному пошуку нових рішень та інновацій, що стане основою для їхньої конкурентоспроможності на ринку праці.

Таким чином, ефективна система підготовки фахівців із цифрових технологій має поєднувати сучасні підходи до викладання, інноваційні технології та гнучкість, що дозволяє адаптувати освітній процес до потреб кожного здобувача освіти.

Стажування та партнерства з промисловістю: організація стажувань та забезпечення партнерства з компаніями та промисловими представниками для надання здобувачам освіти можливостей отримати практичний досвід та розвивати мережі професійних зв'язків.

Оцінка та звітність: використання систем оцінювання, які відображають якість засвоєння здобувачами освіти матеріалу та їх готовність до вирішення завдань у сфері цифрових технологій.

Підтримка кар'єрного розвитку: забезпечення допомоги у плануванні кар'єри, підготовці до працевлаштування та розвитку ключових м'яких навичок, які важливі для успіху у сфері цифрових технологій.

Ці елементи утворюють комплексну систему, спрямовану на формування висококваліфікованих та конкурентоспроможних фахівців з цифрових технологій [41].

Все це обумовлює необхідність професійної підготовки фахівців з цифрових технологій.

Якісна професійна підготовка фахівців з цифрових технологій не може здійснюватися лише за традиційною системою організації навчального процесу, яка не враховує нових факторів і тим самим обмежує її розвиток.

Якщо процес професійної підготовки фахівців з цифрових технологій здійснюється в умовах спеціально створеного освітнього середовища, важливим елементом якого є зміст післядипломної безперервної освіти, то професійна підготовка фахівців з цифрових технологій здійснюватиметься успішніше [45].

«Цифрові» технології дозволяють зробити процес навчання мобільним, диференційованим та індивідуальним. При цьому технології не замінюють викладача, а доповнюють його. Таким заняттям властиві адаптивність, керованість, інтерактивність, поєднання індивідуальної та групової роботи, часова необмеженість навчання. Метою освіти є не лише передача здобувачам вищої освіти сукупності знань, умінь та навичок у певній сфері, а й розвиток кругозору, міждисциплінарного чуття, здатності до індивідуальних креативних рішень, самонавчання, а також формування гуманістичних цінностей [41].

Виходячи із зазначеного вище, з урахуванням сучасних тенденцій та процесів, пов'язаних з розвитком професійної підготовки фахівців з цифрових технологій, особистісної потреби у професійній підготовці фахівців, можна виділити проблему: підвищення рівня ефективності професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Породжувальні патерни проєктування» у закладах вищої освіти.

Отже, професійна підготовка фахівців з цифрових технологій зі спеціальності 015 Професійна освіта (Цифрові технології) потребує вирішення проблеми, яку ми вбачаємо через теоретичне обґрунтування методики професійної підготовки.

Висновки до розділу

У кваліфікаційній роботі відповідно до мети і завдань розкрито стан наукової проблеми. У ході дослідження уточнено та систематизовано основні поняття, що характеризують сутність та структуру процесу професійної

підготовки фахівців з цифрових технологій для викладання освітнього модуля «Породжувальні патерни проєктування» у закладах вищої освіти та виділено пріоритетні напрямки удосконалення професійних компетентностей.

З’ясовано, що професійна підготовка фахівців з цифрових технологій для викладання освітнього модуля «Породжувальні патерни проєктування» у закладах вищої освіти є актуальною у професійній педагогіці.

Проаналізовано ступінь актуальності проблеми професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Породжувальні патерни проєктування» у закладах вищої освіти.

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

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

2.1 Огляд породжувальних патернів проєктування

Породжувальні патерни дозволяють забезпечувати гнучкість і підтримувати розширення програмного забезпечення шляхом виділення процесу створення об'єктів в окремі класи. Кожен з цих патернів вирішує певні проблеми проектування і може бути використаний залежно від конкретних потреб вашого проекту [1].

Фабричний метод – це породжувальний патерн проектування, який визначає загальний інтерфейс для створення об’єктів у суперкласі, дозволяючи підкласам змінювати тип створюваних об’єктів [2]. Породжувальний патерн «Фабричний метод» дозволяє класу делегувати процес створення об'єктів підкласам. Це вирішує проблему, коли класу потрібно визначити, який саме підклас створити. Допоки всі класи реалізують спільний інтерфейс, їхні об’єкти можна змінювати один на інший у клієнтському коді (рис. 2.1).

Клієнт фабричного методу не відчує різниці між цими об’єктами, адже він трактуватиме їх як якийсь абстрактний клас. Для нього буде важливим, щоб об’єкт мав метод доставити, а не те, як конкретно він працює.

Патерн «Абстрактна фабрика» дозволяє створювати сімейства взаємопов'язаних об'єктів, не прив'язуючись до конкретних класів об'єктів [3]. Це досягається за допомогою інтерфейсу для створення об'єктів. Патерн пропонує виділити загальні інтерфейси для окремих продуктів, що складають одне сімейство, і описати в них спільну для цих продуктів поведінку. У випадку меблів абстрактною фабрикою є загальний інтерфейс, який містить методи створення всіх продуктів сімейства (рис. 2.2). Ці операції повинні повертати абстрактні типи продуктів, представлені інтерфейсами, які виділили раніше («Крісло», «Диван» і «Столик»).

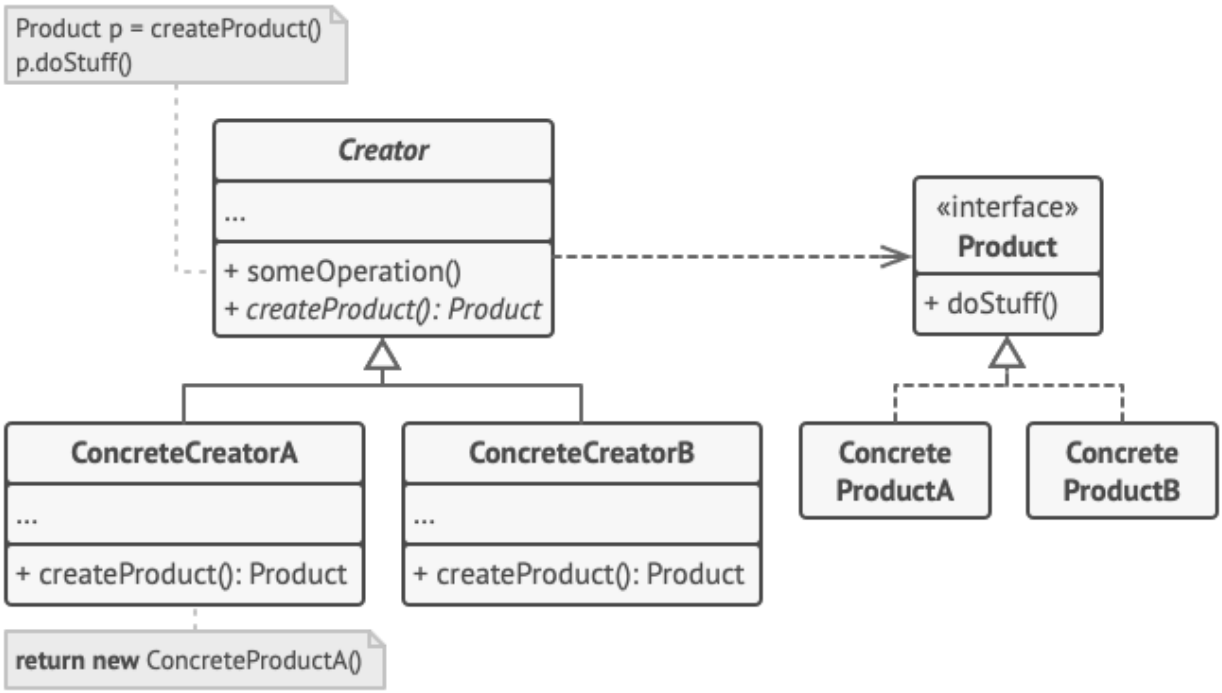

Рисунок 2.1 – Приклад структури класів відповідно патерну «Фабричний метод»

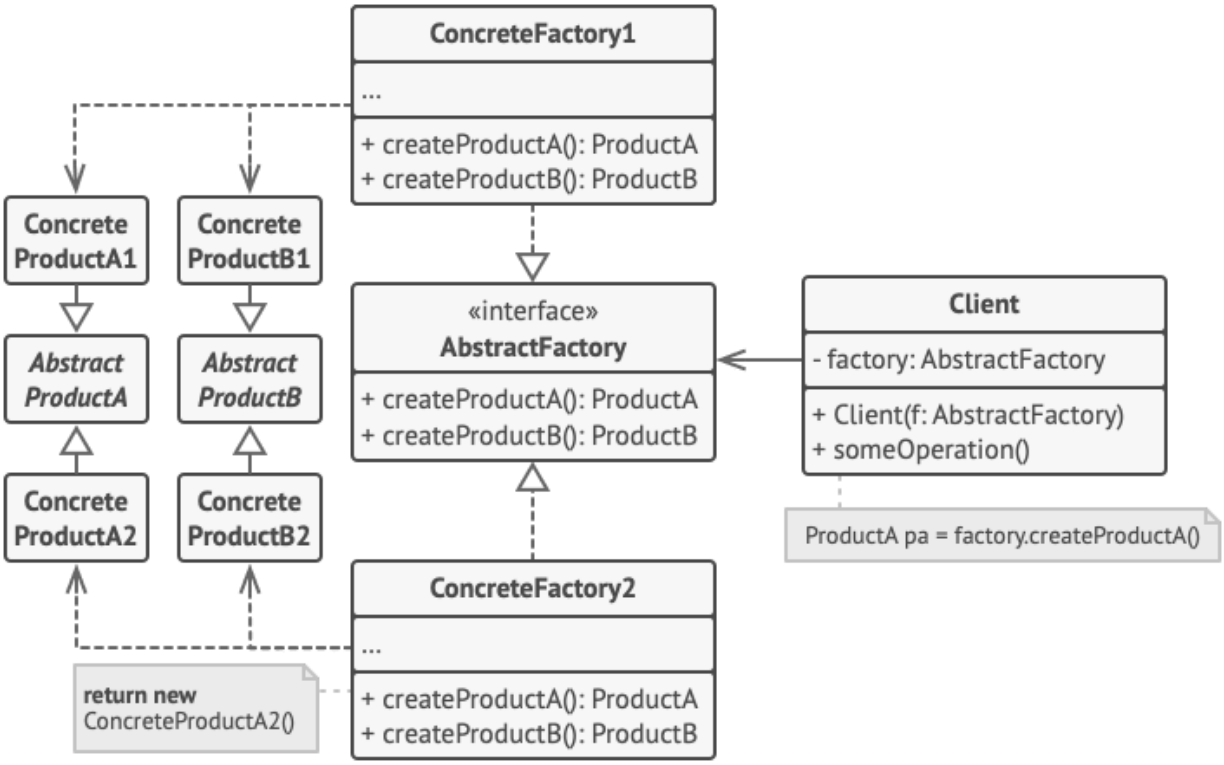

Рисунок 2.2 – Приклад структури класів відповідно патерну «Абстрактна фабрика»

Усі варіації крісел отримають спільний інтерфейс «Крісло», усі дивани реалізують інтерфейс «Диван» тощо. Абстрактна фабрика дозволяє клієнтам створювати об'єкти з різних сімейств (Modern, Victorian), не змінюючи код. Всі варіації одного й того самого об’єкта мають жити в одній ієрархії класів.

Одинак – це породжуючий патерн, який гарантує існування тільки одного об’єкта певного класу, а також дозволяє дістатися цього об’єкта з будь-якого місця програми [4].

Одинак має такі ж переваги та недоліки, що і глобальні змінні. Його неймовірно зручно використовувати, але він порушує модульність вашого коду. Не можна просто взяти і використовувати клас, залежний від одинака, в іншій програмі. Для цього доведеться емулювати там присутність одинака (рис. 2.3). Патерн «Одинак» гарантує, що клас має лише один екземпляр і надає глобальну точку доступу до нього. Це корисно, коли потрібен єдиний об'єкт для координації дій у системі.

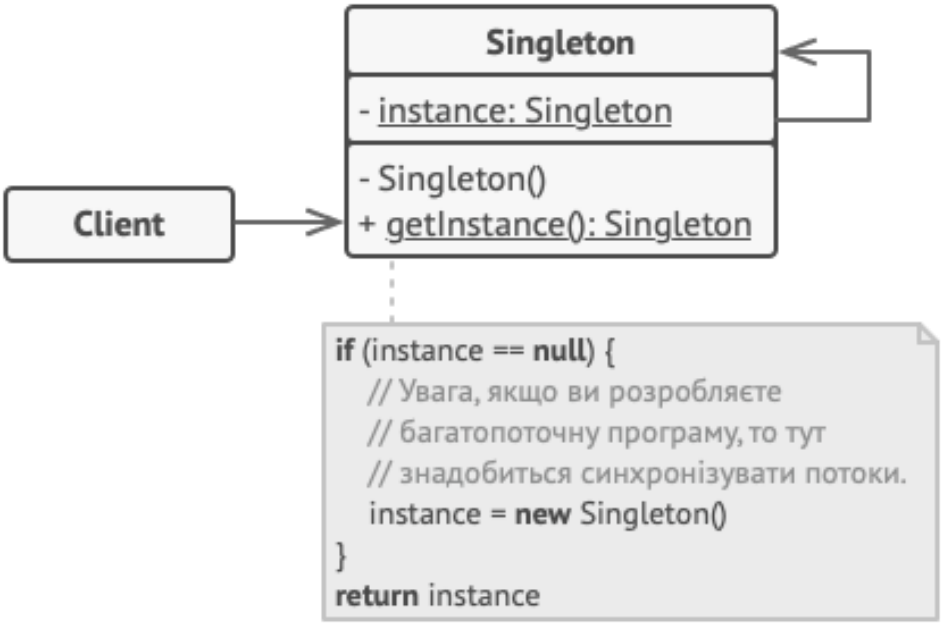

Рисунок 2.3 – Приклад структури класів відповідно патерну «Одинак»

Одинак 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.

Всі реалізації «Одинака» зводяться до того, аби приховати типовий конструктор та створити публічний статичний метод, який і контролюватиме життєвий цикл об’єкта-одинака. Якщо у є доступ до класу одинака, отже, буде й доступ до цього статичного методу. З якої точки коду б його не викликали, він завжди віддаватиме один і той самий об’єкт.

2.2 Постановка лабораторної роботи «Проєктування предметної області з використанням патерна “Фабричний метод”»

Постановка завдання.

1. Опрацюйте теоретичний матеріал. Ознайомтесь з специфікою породжувального патерна «Фабричний метод (Factory Method)».

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

3. Згенерувати програмний код реалізації спроектованої системи згідно розглянутого патерна проєктування.

4. Підготувати звіт до захисту лабораторної роботи.

Звіт до лабораторної роботи повинен містити:

– аналіз предметної області:

– UML-діаграма ієрархії класів (необхідний мінімум класів).

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

– код згенерований по UML-діаграмі, мова програмування C#, C++, TypeScript, або Java;

– в оформленні коду дотримуватись єдиного стилю та доповнювати код коментарями.

Приклад №1 виконання лабораторної роботи. За допомогою патерна проєктування «Фабричний метод» реалізувати систему моніторингу міського транспорту.

Предметною областю є система управління транспортом, де ми маємо різні типи громадського транспорту (автобус, трамвай, тролейбус, фунікулер). Основною метою цієї системи є створення об'єктів, що представляють різні види транспорту, з використанням фабричного методу.

На рис. 2.4 наведено UML-діаграму класів, що відповідає патерну «Абстрактна фабрика».

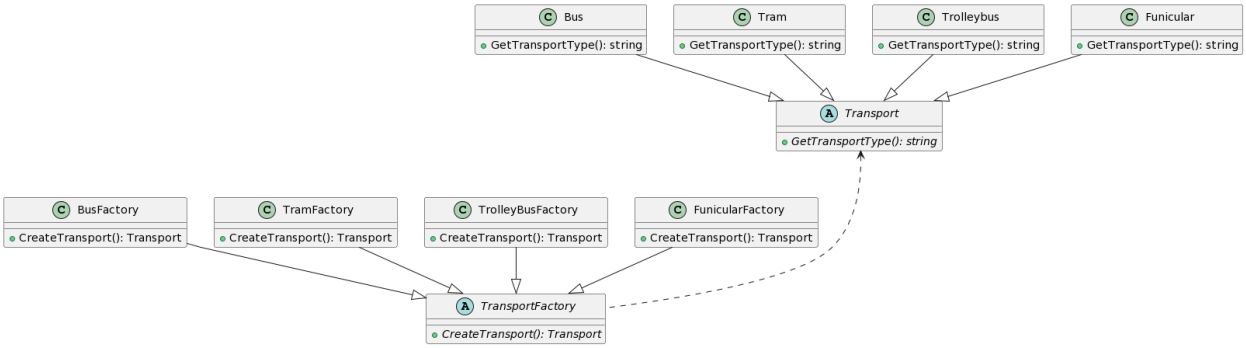

Рисунок 2.4 – UML-діаграма ієрархії класів

Описи класів та їх атрибутів.

Базовий клас Transport є абстрактним класом, який представляє спільні характеристики всіх видів транспорту. Він містить абстрактний метод GetTransportType, який повинен бути реалізований у підкласах для повернення типу транспорту.

Bus, Tram, Trolleybus, і Funicular – це конкретні класи, які наслідують базовий клас Transport. Кожен із цих класів реалізує метод GetTransportType, який повертає рядок з назвою типу транспорту.

Абстрактний клас TransportFactory є базовим класом для фабрик, які створюють об'єкти транспорту. Він містить абстрактний метод CreateTransport, який повинен бути реалізований у підкласах для створення конкретних об'єктів транспорту.

BusFactory, TramFactory, TrolleyBusFactory, FunicularFactory – це

фабрики, які наслідують абстрактний клас TransportFactory. Кожна фабрика реалізує метод CreateTransport, створюючи конкретний об'єкт транспорту.

Основний клас програми Program містить метод Main, в якому створюються екземпляри кожної фабрики. Потім фабрики використовуються для створення відповідних об'єктів транспорту, а тип кожного транспорту виводиться на консоль.

Програмний код реалізації патерна «Фабричний метод» для предметної області:

```
using System;

/**
 * Інтерфейс ITransport представляє набір функціональності,
 * яка повинна бути реалізована у кожному конкретному транспорті.
 * Метод GetTransportType повинен повертати тип транспорту у вигляді рядка.
 */
public interface ITransport
{
    string GetTransportType();
}

/**
 * Інтерфейс ITransportFactory представляє набір функціональності,
 * яка повинна бути реалізована кожною фабрикою транспорту.
 * Метод CreateTransport повинен створювати екземпляр конкретного транспорту.
 */
public interface ITransportFactory
{
    ITransport CreateTransport();
}

/**
 * Клас Bus є конкретним класом, який наслідує інтерфейс ITransport і визначає
автобус (маршрутку) як тип транспорту.
 * Він реалізує метод GetTransportType, який повертає тип транспорту.
 */
public class Bus : ITransport
{
    public string GetTransportType()
    {
        return "Bus";
    }
}

/**
 * Клас Tram є конкретним класом, який наслідує інтерфейс ITransport і визначає
трамвай як тип транспорту.
 * Він реалізує метод GetTransportType, який повертає тип транспорту.
 */
public class Tram : ITransport
{
    public string GetTransportType()
    {
        return "Tram";
    }
}
```

```
/**
 * Клас Trolleybus є конкретним класом, який наслідує інтерфейс ITransport і
визначає тролейбус як тип транспорту.
 * Він реалізує метод GetTransportType, який повертає тип транспорту.
 */
public class Trolleybus : ITransport
{
    public string GetTransportType()
    {
        return "Trolleybus";
    }
}

/**
 * Клас Funicular є конкретним класом, який наслідує інтерфейс ITransport і визначає
фунікулер як тип транспорту.
 * Він реалізує метод GetTransportType, який повертає тип транспорту.
 */
public class Funicular : ITransport
{
    public string GetTransportType()
    {
        return "Funicular";
    }
}

/**
 * Клас BusFactory реалізує інтерфейс ITransportFactory і є фабрикою для створення
об'єктів класу Bus.
 * Метод CreateTransport створює новий екземпляр транспорту типу Bus.
 */
public class BusFactory : ITransportFactory
{
    public ITransport CreateTransport()
    {
        return new Bus();
    }
}

/**
 * Клас TramFactory реалізує інтерфейс ITransportFactory і є фабрикою для створення
об'єктів класу Tram.
 * Метод CreateTransport створює новий екземпляр транспорту типу Tram.
 */
public class TramFactory : ITransportFactory
{
    public ITransport CreateTransport()
    {
        return new Tram();
    }
}

/**
 * Клас TrolleybusFactory реалізує інтерфейс ITransportFactory і є фабрикою для
створення об'єктів класу Trolleybus.
 * Метод CreateTransport створює новий екземпляр транспорту типу Trolleybus.
 */
public class TrolleybusFactory : ITransportFactory
{
    public ITransport CreateTransport()
    {
        return new Trolleybus();
    }
}
```

```
/**
 * Клас FunicularFactory реалізує інтерфейс ITransportFactory і є фабрикою для
створення об'єктів класу Funicular.
 * Метод CreateTransport створює новий екземпляр транспорту типу Funicular.
 */
public class FunicularFactory : ITransportFactory
{
    public ITransport CreateTransport()
    {
        return new Funicular();
    }
}

/**
 * Головний клас програми, який демонструє використання фабрик для створення
об'єктів транспорту
 * і виведення типу кожного створеного транспорту на консоль.
 */
class Program
{
    static void Main(string[] args)
    {
        // Створення фабрики для автобуса та отримання об'єкта Bus
        ITransportFactory busFactory = new BusFactory();
        ITransport bus = busFactory.CreateTransport();
        Console.WriteLine(bus.GetTransportType());

        // Створення фабрики для трамвая та отримання об'єкта Tram
        ITransportFactory tramFactory = new TramFactory();
        ITransport tram = tramFactory.CreateTransport();
        Console.WriteLine(tram.GetTransportType());

        // Створення фабрики для тролейбуса та отримання об'єкта Trolleybus
        ITransportFactory trolleybusFactory = new TrolleybusFactory();
        ITransport trolleybus = trolleybusFactory.CreateTransport();
        Console.WriteLine(trolleybus.GetTransportType());

        // Створення фабрики для фунікулера та отримання об'єкта Funicular
        ITransportFactory funicularFactory = new FunicularFactory();
        ITransport funicular = funicularFactory.CreateTransport();
        Console.WriteLine(funicular.GetTransportType());
    }
}
```

```
C:\Users\Wik\Desktop\Стаціонар (121)\Конструювання\1\Fabric Method>dotnet run
Bus
Tram
Trolleybus
Funicular
```

Рисунок 2.5 – Приклад виконання програми

Приклад №2 виконання лабораторної роботи. За допомогою патерна проєктування «Фабричний метод» реалізувати систему.

Предметна область відноситься до системи бронювання та продажу автобусних квитків, де використовуються різні типи квитків (економ-клас та бізнес-клас). Система дозволяє створювати квитки, резервувати їх, а також

здійснювати продаж.

В основі коду програмної системи використовується шаблон проектування «Фабричний метод». Цей шаблон дозволяє створювати об'єкти, не вказуючи конкретний клас створюваного об'єкта. Це корисно в тих випадках, коли існують різні варіанти об'єктів, що створюються, і їхні конкретні реалізації можуть змінюватися.

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

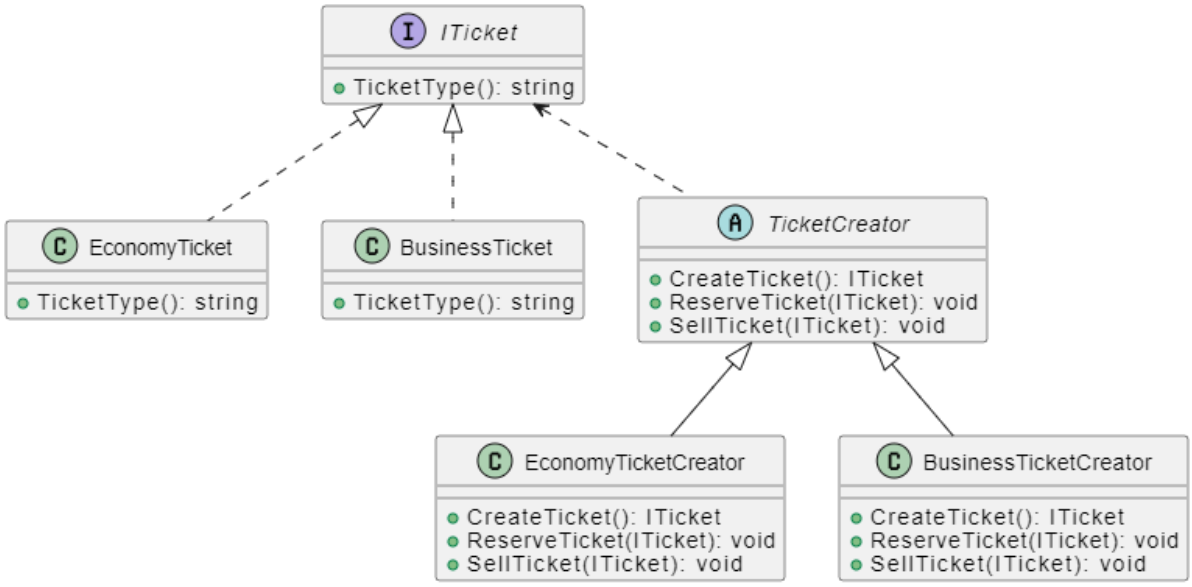

Рисунок 2.6 – UML-діаграма ієрархії класів

Описи класів та їх атрибутів.

1. Інтерфейс ITicket визначає загальний контракт для всіх типів квитків, що можуть бути створені в системі. Інтерфейс містить метод TicketType(), який повертає тип квитка.

2. Конкретні реалізації квитків (EconomyTicket та BusinessTicket) відповідають інтерфейсу ITicket і повертають відповідні значення для методу TicketType(), залежно від типу квитка (економ або бізнес).

3. Абстрактний клас TicketCreator визначає шаблон для створення квитків та їхньої обробки (бронювання та продаж). Клас містить абстрактний метод CreateTicket(), який відповідає за створення квитка. Абстрактні методи

ReserveTicket() та SellTicket(), які відповідають за бронювання та продаж квитків відповідно.

4. Конкретні творці квитків (EconomyTicketCreator та BusinessTicketCreator) реалізують клас TicketCreator, надаючи конкретну логіку для створення, бронювання та продажу квитків для економ-класу та бізнес-класу.

5. Клас Program містить метод Main, який є точкою входу в програму. У цьому методі створюються екземпляри творців квитків (EconomyTicketCreator та BusinessTicketCreator) і демонструється процес бронювання та продажу квитків для кожного з класів.

Програмний код реалізації патерна «Фабричний метод» для предметної області:

```
using System;
using System.Text;

namespace BusTicketBooking
{
    // Інтерфейс для квитків
    public interface ITicket
    {
        // Метод для отримання типу квитка
        string TicketType();
    }

    // Конкретна реалізація квитка для економ-класу
    public class EconomyTicket : ITicket
    {
        public string TicketType()
        {
            return "Економ-клас";
        }
    }

    // Конкретна реалізація квитка для бізнес-класу
    public class BusinessTicket : ITicket
    {
        public string TicketType()
        {
            return "Бізнес-клас";
        }
    }

    // Абстрактний клас Creator, який містить фабричний метод
    abstract class TicketCreator
    {
        // Фабричний метод для створення квитків
        public abstract ITicket CreateTicket();

        // Метод для бронювання квитка
        public abstract void ReserveTicket(ITicket ticket);
```

```
    // Метод для продажу квитка
    public abstract void SellTicket(ITicket ticket);
}

// Конкретний творець для економ-класу
class EconomyTicketCreator : TicketCreator
{
    // Реалізація фабричного методу для створення економ-класу
    public override ITicket CreateTicket()
    {
        return new EconomyTicket();
    }

    // Реалізація методу для бронювання економ-класу
    public override void ReserveTicket(ITicket ticket)
    {
        Console.WriteLine("Бронювання економ-класу: " + ticket.TicketType());
    }

    // Реалізація методу для продажу економ-класу
    public override void SellTicket(ITicket ticket)
    {
        Console.WriteLine("Продаж економ-класу: " + ticket.TicketType());
    }
}

// Конкретний творець для бізнес-класу
class BusinessTicketCreator : TicketCreator
{
    // Реалізація фабричного методу для створення бізнес-класу
    public override ITicket CreateTicket()
    {
        return new BusinessTicket();
    }

    // Реалізація методу для бронювання бізнес-класу
    public override void ReserveTicket(ITicket ticket)
    {
        Console.WriteLine("Бронювання бізнес-класу: " + ticket.TicketType());
    }

    // Реалізація методу для продажу бізнес-класу
    public override void SellTicket(ITicket ticket)
    {
        Console.WriteLine("Продаж бізнес-класу: " + ticket.TicketType());
    }
}

class Program
{
    static void Main(string[] args)
    {
        // Встановлення кодування консолі на UTF-8
        Console.OutputEncoding = Encoding.UTF8;

        // Приклад використання творця для економ-класу
        TicketCreator creator = new EconomyTicketCreator();
        ITicket ticket = creator.CreateTicket();
        creator.ReserveTicket(ticket);
        creator.SellTicket(ticket);

        // Приклад використання творця для бізнес-класу
        creator = new BusinessTicketCreator();
        ticket = creator.CreateTicket();
        creator.ReserveTicket(ticket);
```

```
            creator.SellTicket(ticket);
        }
    }
}
```

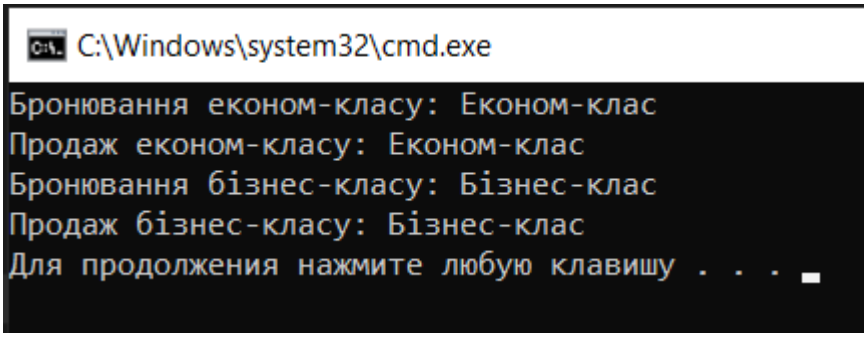

Рисунок 2.7 – Приклад виконання програми

2.3 Постановка лабораторної роботи «Проєктування предметної області з використанням патерна “Абстрактна фабрика”»

Постановка завдання.

1. Опрацюйте теоретичний матеріал. Ознайомтесь з специфікою породжувального патерна «Абстрактна фабрика (Abstract Factory)».

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

3. Згенерувати програмний код реалізації спроектованої системи згідно розглянутого патерна проєктування.

4. Підготувати звіт до захисту лабораторної роботи.

Звіт до лабораторної роботи повинен містити:

– аналіз предметної області:

– UML-діаграма ієрархії класів (необхідний мінімум класів).

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

– код згенерований по UML-діаграмі, мова програмування C#, C++, TypeScript, або Java;

– в оформленні коду дотримуватись єдиного стилю та доповнювати код

коментарями.

Приклад №1 виконання лабораторної роботи. За допомогою патерна проєктування «Абстрактна фабрика» реалізувати систему моніторингу міського транспорту.

Абстрактна фабрика дозволяє створювати об'єкти без необхідності вказувати конкретні класи об'єктів, які створюються. Це полегшує модифікацію коду, додає гнучкість і покращує підтримку. Завдяки використанню інтерфейсів створюється можливість розширення предметної області новими типами транспорту без значних змін у коді.

Предметна область орієнтована на моделювання транспортних засобів у вигляді класів, де кожен клас представляє певний вид транспорту. Основними елементами предметної області є:

– транспортні засоби: моделюються за допомогою інтерфейсу ITransport та його конкретних реалізацій (Bus, Tram, Trolleybus, Funicular).

– фабрики транспорту: створюють екземпляри транспортних засобів за допомогою інтерфейсу ITransportFactory та його реалізацій (BusFactory, TramFactory, TrolleybusFactory, FunicularFactory).

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

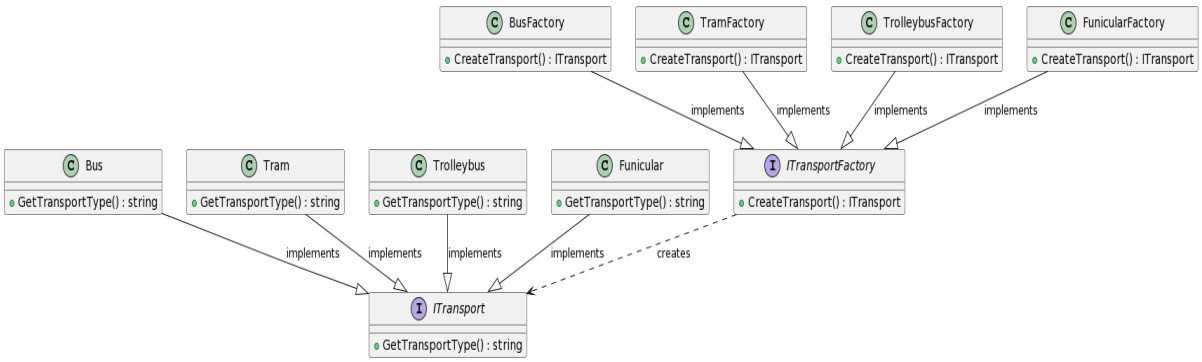

Рисунок 2.8 – UML-діаграма ієрархії класів

Описи класів та їх атрибутів.

1. Інтерфейси ITransport визначає функціонал, який повинен реалізовувати будь-який тип транспорту (метод GetTransportType). Інтерфейс

ITransportFactory визначає метод CreateTransport, який відповідає за створення конкретного транспортного засобу.

2. Класи Bus, Tram, Trolleybus, Funicular є реалізаціями інтерфейсу ITransport. Вони визначають, який саме тип транспорту представляють, реалізуючи метод GetTransportType.

3. Кожна фабрика (BusFactory, TramFactory, TrolleybusFactory, FunicularFactory) реалізує інтерфейс ITransportFactory і відповідає за створення конкретного типу транспорту.

4. Головний клас програми Program демонструє використання фабрик для створення об'єктів транспорту і їх виведення на консоль.

Програмний код реалізації патерна «Абстрактна фабрика» для предметної області:

```
using System;

/**
 * Інтерфейс ITransport представляє набір функціональності,
 * яка повинна бути реалізована у кожному конкретному транспорті.
 * Метод GetTransportType повинен повертати тип транспорту у вигляді рядка.
 */
public interface ITransport
{
    string GetTransportType();
}

/**
 * Інтерфейс ITransportFactory представляє набір функціональності,
 * яка повинна бути реалізована кожною фабрикою транспорту.
 * Метод CreateTransport повинен створювати екземпляр конкретного транспорту.
 */
public interface ITransportFactory
{
    ITransport CreateTransport();
}

/**
 * Клас Bus є конкретним класом, який наслідує інтерфейс ITransport
 * і визначає автобус (маршрутку) як тип транспорту.
 * Він реалізує метод GetTransportType, який повертає тип транспорту.
 */
public class Bus : ITransport
{
    public string GetTransportType()
    {
        return "Bus";
    }
}

/**
 * Клас Tram є конкретним класом, який наслідує інтерфейс ITransport
 * і визначає трамвай як тип транспорту.
 * Він реалізує метод GetTransportType, який повертає тип транспорту.
```

```
 */
public class Tram : ITransport
{
    public string GetTransportType()
    {
        return "Tram";
    }
}

/**
 * Клас Trolleybus є конкретним класом, який наслідує інтерфейс ITransport
 * і визначає тролейбус як тип транспорту.
 * Він реалізує метод GetTransportType, який повертає тип транспорту.
 */
public class Trolleybus : ITransport
{
    public string GetTransportType()
    {
        return "Trolleybus";
    }
}

/**
 * Клас Funicular є конкретним класом, який наслідує інтерфейс ITransport
 * і визначає фунікулер як тип транспорту.
 * Він реалізує метод GetTransportType, який повертає тип транспорту.
 */
public class Funicular : ITransport
{
    public string GetTransportType()
    {
        return "Funicular";
    }
}

/**
 * Клас BusFactory реалізує інтерфейс ITransportFactory і є фабрикою
 * для створення об'єктів класу Bus.
 * Метод CreateTransport створює новий екземпляр транспорту типу Bus.
 */
public class BusFactory : ITransportFactory
{
    public ITransport CreateTransport()
    {
        return new Bus();
    }
}

/**
 * Клас TramFactory реалізує інтерфейс ITransportFactory і є фабрикою
 * для створення об'єктів класу Tram.
 * Метод CreateTransport створює новий екземпляр транспорту типу Tram.
 */
public class TramFactory : ITransportFactory
{
    public ITransport CreateTransport()
    {
        return new Tram();
    }
}

/**
 * Клас TrolleybusFactory реалізує інтерфейс ITransportFactory і є фабрикою
 * для створення об'єктів класу Trolleybus.
 * Метод CreateTransport створює новий екземпляр транспорту типу Trolleybus.
```

```
 */
public class TrolleybusFactory : ITransportFactory
{
    public ITransport CreateTransport()
    {
        return new Trolleybus();
    }
}

/**
 * Клас FunicularFactory реалізує інтерфейс ITransportFactory і є фабрикою
 * для створення об'єктів класу Funicular.
 * Метод CreateTransport створює новий екземпляр транспорту типу Funicular.
 */
public class FunicularFactory : ITransportFactory
{
    public ITransport CreateTransport()
    {
        return new Funicular();
    }
}

/**
 * Головний клас програми, який демонструє використання фабрик
 * для створення об'єктів транспорту
*/
class Program
{
    static void Main(string[] args)
    {
        // Створення фабрики для автобуса та отримання об'єкта Bus
        ITransportFactory busFactory = new BusFactory();
        ITransport bus = busFactory.CreateTransport();
        Console.WriteLine(bus.GetTransportType());

        // Створення фабрики для трамвая та отримання об'єкта Tram
        ITransportFactory tramFactory = new TramFactory();
        ITransport tram = tramFactory.CreateTransport();
        Console.WriteLine(tram.GetTransportType());

        // Створення фабрики для тролейбуса та отримання об'єкта Trolleybus
        ITransportFactory trolleybusFactory = new TrolleybusFactory();
        ITransport trolleybus = trolleybusFactory.CreateTransport();
        Console.WriteLine(trolleybus.GetTransportType());

        // Створення фабрики для фунікулера та отримання об'єкта Funicular
        ITransportFactory funicularFactory = new FunicularFactory();
        ITransport funicular = funicularFactory.CreateTransport();
        Console.WriteLine(funicular.GetTransportType());
    }
}
```

```
C:\Users\Nicolas\Desktop\Стаціонар (121)\Конструювання>dotnet run
Bus
Tram
Trolleybus
Funicular

C:\Users\Nicolas\Desktop\Стаціонар (121)\Конструювання>
```

Рисунок 2.9 – Приклад виконання програми

Приклад №2 виконання лабораторної роботи. За допомогою патерна проєктування «Абстрактна фабрика» реалізувати систему бронювання та продажу квитків на проїзд автобусним транспортом.

Цей код є реалізацією патерну «Абстрактна фабрика» в контексті системи бронювання квитків на автобус. Патерн Абстрактна фабрика» використовується для створення сімейств пов'язаних об'єктів без зазначення їх конкретних класів. У цій реалізації використовуються класи для створення різних типів квитків – економ-класу та бізнес-класу.

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

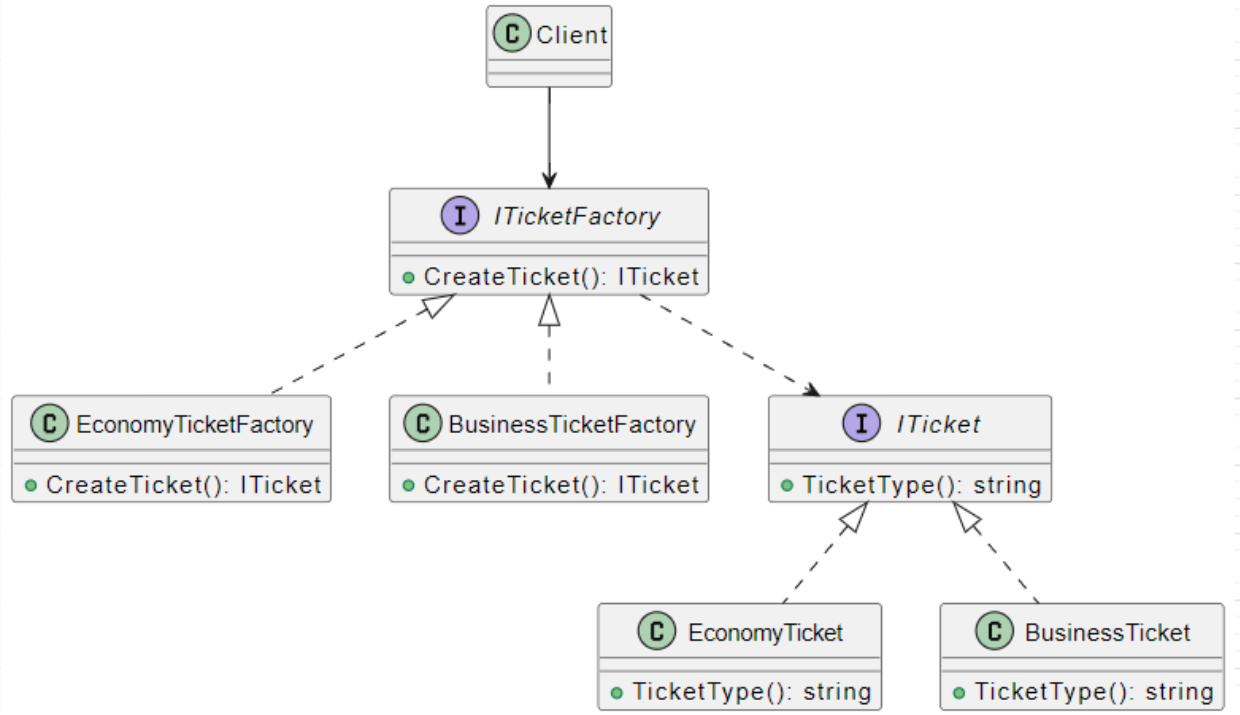

Рисунок 2.10 – UML-діаграма ієрархії класів

Система передбачає наявність інтерфейсу для створення квитків (ITicketFactory), що дозволяє клієнтському коду працювати з конкретними фабриками для створення квитків економ- або бізнес-класу, не знаючи деталей їх реалізації. Конкретні фабрики (EconomyTicketFactory та BusinessTicketFactory) реалізують методи створення об'єктів типу квитка.

Реалізація також включає два типи продуктів (EconomyTicket та BusinessTicket), які мають спільний інтерфейс (ITicket). Клієнтський код

використовує цей інтерфейс для отримання типу квитка, незалежно від його конкретного типу.

Клас Program у цьому коді відіграє роль точки входу в програму. Метод Main викликає інші методи або об'єкти, які виконують основну логіку програми. У даному випадку, він створює об'єкт класу Client і викликає його метод Main, який тестує роботу фабрики квитків.

Програмний код реалізації патерна «Абстрактна фабрика» для предметної області:

```
using System;

namespace BusTicketBookingSystem
{
    // Інтерфейс Абстрактної Фабрики
    public interface ITicketFactory
    {
        ITicket CreateTicket(); // Метод для створення квитка
    }

    // Конкретна Фабрика для Економ-класу
    class EconomyTicketFactory : ITicketFactory
    {
        public ITicket CreateTicket()
        {
            return new EconomyTicket();
            // Створення квитка для Економ-класу
        }
    }

    // Конкретна Фабрика для Бізнес-класу
    class BusinessTicketFactory : ITicketFactory
    {
        public ITicket CreateTicket()
        {
            return new BusinessTicket();
            // Створення квитка для Бізнес-класу
        }
    }

    // Інтерфейс Абстрактного Продукту
    public interface ITicket
    {
        string TicketType(); // Метод для отримання типу квитка
    }

    // Конкретний Продукт для квитка Економ-класу
    class EconomyTicket : ITicket
    {
        public string TicketType()
        {
            return "Квиток Економ-класу";
            // Повертає тип квитка для Економ-класу
        }
    }

    // Конкретний Продукт для квитка Бізнес-класу
    class BusinessTicket : ITicket
```

```
    {
        public string TicketType()
        {
            return "Квиток Бізнес-класу";
            // Повертає тип квитка для Бізнес-класу
        }
    }

    // Клієнт
    class Client
    {
        public void Main()
        {
            Console.WriteLine("Клієнт: Перевірка роботи з фабрикою " +
                "квитків Економ-класу...");
            ClientMethod(new EconomyTicketFactory());
            // Тестування фабрики квитків Економ-класу
            Console.WriteLine();

            Console.WriteLine("Клієнт: Перевірка роботи з фабрикою " +
                "квитків Бізнес-класу...");
            ClientMethod(new BusinessTicketFactory());
            // Тестування фабрики квитків Бізнес-класу
        }

        public void ClientMethod(ITicketFactory factory)
        {
            var ticket = factory.CreateTicket();
            // Створення квитка з використанням фабрики
            Console.WriteLine($"Тип квитка: " +
                $"{ticket.TicketType()}"); // Виведення типу створеного квитка
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Console.OutputEncoding = System.Text.Encoding.UTF8;
            // Встановлення кодування для коректного відображення символів в консолі
            new Client().Main();
            // Запуск головного методу клієнта
        }
    }
}
```

```
C:\Windows\system32\cmd.exe
Клієнт: Перевірка роботи з фабрикою квитків Економ-класу...
Тип квитка: Квиток Економ-класу

Клієнт: Перевірка роботи з фабрикою квитків Бізнес-класу...
Тип квитка: Квиток Бізнес-класу
Для продолжения нажмите любую клавишу . . .
```

Рисунок 2.11 – Приклад виконання програми

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

Постановка завдання.

1. Опрацюйте теоретичний матеріал. Ознайомтесь з специфікою породжувального патерна «Одинак (Singleton)».

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

3. Згенерувати програмний код реалізації спроектованої системи згідно розглянутого патерна проєктування.

4. Підготувати звіт до захисту лабораторної роботи.

Звіт до лабораторної роботи повинен містити:

– аналіз предметної області:

– UML-діаграма ієрархії класів (необхідний мінімум класів).

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

– код згенерований по UML-діаграмі, мова програмування C#, C++, TypeScript, або Java;

– в оформленні коду дотримуватись єдиного стилю та доповнювати код коментарями.

Приклад №1 виконання лабораторної роботи. За допомогою патерна проєктування «Одинак» реалізувати однопоточну систему бронювання та продажу квитків на проїзд автобусним транспортом.

Кодова база, представлена у цьому прикладі, описує транспортні засоби як об'єкти програмного забезпечення з використанням об'єктно-орієнтованого підходу. Класи, визначені в коді, використовують наслідування та патерн проєктування «Одинак» для управління об'єктами транспорту та їх спостереженням. Ця система дозволяє створювати різні види транспорту

(автобуси, трамваї, тролейбуси, фунікулери) та відстежувати їх стан у режимі реального часу.

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

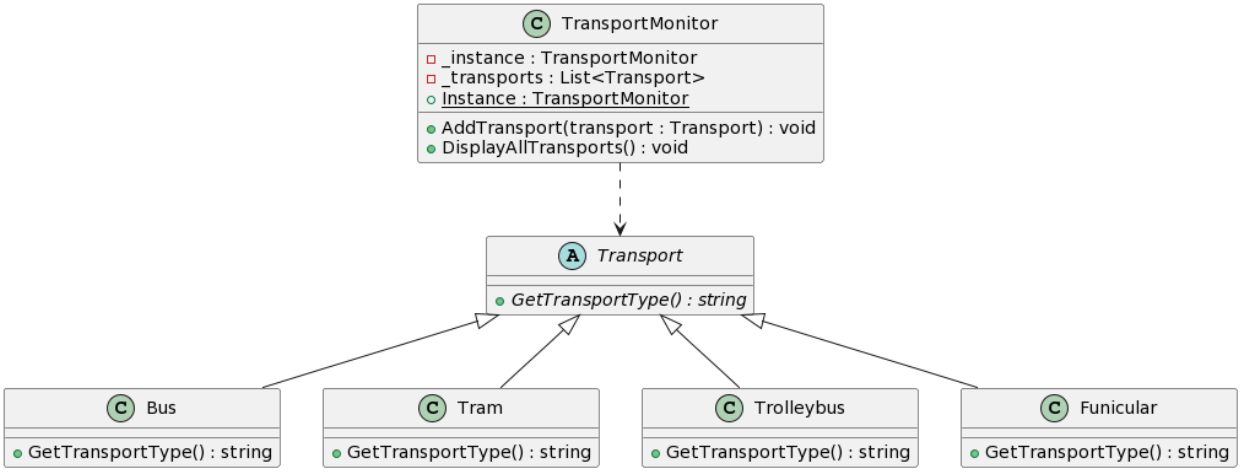

Рисунок 2.12 – UML-діаграма ієрархії класів

У наведеному далі коді на мові C# використовується концепція об'єктно-орієнтованого програмування для представлення транспортних засобів за допомогою базового абстрактного класу Transport, який є спільним для всіх типів транспорту. Конкретні транспортні засоби, такі як Bus, Tram, Trolleybus і Funicular, наслідують цей клас і реалізують метод для визначення свого типу.

Клас TransportMonitor реалізований відповідно патерну «Одинак», що дозволяє відстежувати всі додані транспортні засоби в єдиному екземплярі монітора. Цей підхід забезпечує централізоване управління інформацією про всі транспортні засоби та дозволяє уникнути створення зайвих екземплярів монітора.

Програмний код реалізації патерна «Одинак» для предметної області:

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

/**
 * Клас Transport є базовим абстрактним класом, який наслідуватимуть класи окремих
видів транспорту.
 * Він має абстрактний метод GetTransportType, який повертає тип транспорту.
 */
public abstract class Transport
{
    // Абстрактний метод для отримання типу транспорту.
```

```
    public abstract string GetTransportType();
}

/**
 * Клас Bus є конкретним класом, який наслідує клас Transport і визначає автобус
(маршрутку) як тип транспорту.
 * Він реалізує метод GetTransportType, який повертає тип транспорту.
 */
public class Bus : Transport
{
    // Реалізація методу для повернення типу транспорту: Bus.
    public override string GetTransportType()
    {
        return "Bus";
    }
}

/**
 * Клас Tram є конкретним класом, який наслідує клас Transport і визначає трамвай як
тип транспорту.
 * Він реалізує метод GetTransportType, який повертає тип транспорту.
 */
public class Tram : Transport
{
    // Реалізація методу для повернення типу транспорту: Tram.
    public override string GetTransportType()
    {
        return "Tram";
    }
}

/**
 * Клас Trolleybus є конкретним класом, який наслідує клас Transport
 * і визначає тролейбус як тип транспорту.
 * Він реалізує метод GetTransportType, який повертає тип транспорту.
 */
public class Trolleybus : Transport
{
    // Реалізація методу для повернення типу транспорту: Trolleybus.
    public override string GetTransportType()
    {
        return "Trolleybus";
    }
}

/**
 * Клас Funicular є конкретним класом, який наслідує клас Transport
 * і визначає фунікулер як тип транспорту.
 * Він реалізує метод GetTransportType, який повертає тип транспорту.
 */
public class Funicular : Transport
{
    // Реалізація методу для повернення типу транспорту: Funicular.
    public override string GetTransportType()
    {
        return "Funicular";
    }
}

/**
 * Клас TransportMonitor є синглтоном, який призначений для відстеження
 * інформації про всі види транспорту.  Він містить методи додавання
 * транспорту для відстеження та відображення інформації про весь відстежуваний
транспорт.
 */
```

```
public sealed class TransportMonitor
{
    // Статичне поле для зберігання єдиного екземпляра TransportMonitor.
    private static TransportMonitor? _instance;

    // Список для зберігання об'єктів транспорту.
    private List<Transport> _transports;

    // Приватний конструктор для запобігання створенню екземплярів класу за межами
класу.
    private TransportMonitor()
    {
        _transports = new List<Transport>();
    }

    // Властивість для отримання єдиного екземпляра TransportMonitor.
    public static TransportMonitor Instance
    {
        get
        {
            if (_instance == null)
            {
                _instance = new TransportMonitor();
            }
            return _instance;
        }
    }

    // Метод для додавання об'єкта транспорту до списку відстежуваних транспортів.
    public void AddTransport(Transport transport)
    {
        _transports.Add(transport);
    }

    // Метод для відображення інформації про всі відстежувані транспортні засоби.
    public void DisplayAllTransports()
    {
        foreach (var transport in _transports)
        {
            Console.WriteLine(transport.GetTransportType() + " is observed by
TransportMonitor at "
                + DateTime.Now);
        }
    }
}

/**
 * Головний клас програми, в якому здійснюється тестування функціональності класів.
 */
class Program
{
    static void Main(string[] args)
    {
        // Отримання єдиного екземпляра TransportMonitor.
        TransportMonitor monitor = TransportMonitor.Instance;

        // Додавання різних транспортних засобів до списку відстежуваних.
        monitor.AddTransport(new Bus());
        monitor.AddTransport(new Tram());
        monitor.AddTransport(new Trolleybus());
        monitor.AddTransport(new Funicular());

        // Відображення інформації про всі відстежувані транспортні засоби.
        monitor.DisplayAllTransports();
```

```
        // Отримання ще одного посилання на єдиний екземпляр TransportMonitor.
        TransportMonitor monitor2 = TransportMonitor.Instance;

        // Повторне відображення інформації про всі відстежувані транспортні засоби.
        monitor.DisplayAllTransports();

        // Перевірка, чи є обидва посилання на один і той самий екземпляр (патерн
синглтон).
        if (monitor == monitor2)
        {
            Console.WriteLine("TransportMonitor is a singleton.");
        }
        else
        {
            Console.WriteLine("TransportMonitor is not a singleton.");
        }
    }
}
```

Рисунок 2.13 – Приклад виконання програми

Приклад №2 виконання лабораторної роботи. За допомогою патерна проєктування «Одинак» реалізувати багатопотокову систему бронювання та продажу квитків на проїзд автобусним транспортом.

Ця програма моделює систему бронювання та продажу квитків на автобус, яка дозволяє користувачам резервувати квитки та купувати їх. Вона реалізує шаблон проектування «Одинак» для управління квитками, щоб гарантувати, що в системі існує лише один об'єкт, який відповідає за операції з квитками.

На рис. 2.14 наведено UML-діаграму класів, що відповідає патерну «Одинак».

У наведеному далі коді на мові C# використовується концепція об'єктно-орієнтованого програмування для представлення багатопотоковості у програмі для одночасного виконання операцій бронювання та продажу квитків на автобус. Програмний код складається з трьох основних частин.

1. Клас BusTicket: представляє квиток на автобус, що містить основну інформацію про подорож, таку як місце відправлення, місце призначення, час відправлення і ім'я пасажира.

2. Клас TicketManager: використовує шаблон проектування Singleton для керування операціями бронювання та продажу квитків. Цей клас гарантує, що в програмі існує тільки один екземпляр TicketManager, що дозволяє централізовано обробляти всі запити на бронювання і продаж. Він містить методи для бронювання (ReserveTicket) і продажу квитків (SellTicket).

3. Клас Program: це точка входу в програму, де здійснюється створення квитків, отримання екземпляру TicketManager, а також запуск процесів бронювання та продажу квитків в окремих потоках.

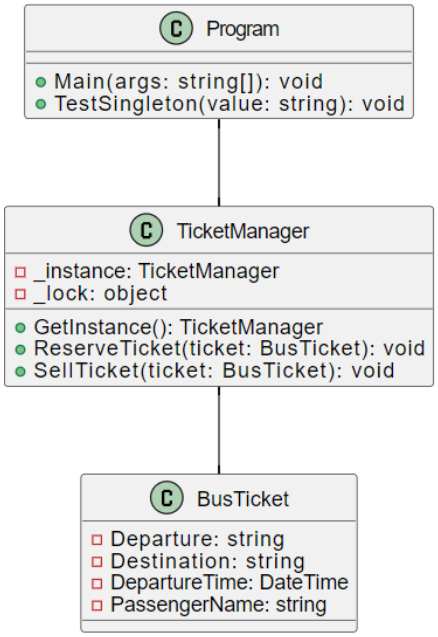

Рисунок 2.14 – UML-діаграма ієрархії класів

Програмний код реалізації патерна «Одинак» для предметної області:

```
using System;
using System.Threading;

namespace BusTicketBookingSystem
{
    // Клас для представлення квитка на проїзд автобусом
    class BusTicket
    {
        // Місце відправлення
        public string Departure { get; set; }

        // Місце призначення
        public string Destination { get; set; }
```

```
    // Час відправлення автобуса
    public DateTime DepartureTime { get; set; }

    // Ім'я пасажира
    public string PassengerName { get; set; }
}

// Клас для керування бронюванням та продажем квитків
class TicketManager
{
    // Статична змінна для збереження єдиного екземпляра TicketManager
    private static TicketManager _instance;

    // Об'єкт для блокування, синхронізація доступу до екземпляра
    private static readonly object _lock = new object();

    // Приватний конструктор для запобігання створенню екземплярів ззовні
    private TicketManager() { }

    // Метод для отримання єдиного екземпляру TicketManager
    public static TicketManager GetInstance()
    {
        // Перевірка наявності екземпляра
        if (_instance == null)
        {
            // Блокування для забезпечення потокобезпеки
            lock (_lock)
            {
                // Повторна перевірка всередині заблокованого блоку
                if (_instance == null)
                {
                    // Створення нового екземпляра TicketManager
                    _instance = new TicketManager();
                }
            }
        }
        // Повернення єдиного екземпляра
        return _instance;
    }

    // Метод для бронювання квитка
    public void ReserveTicket(BusTicket ticket)
    {
        Console.WriteLine($"Ticket reserved for {ticket.PassengerName} " +
            $"from {ticket.Departure} to {ticket.Destination} " +
            $"at {ticket.DepartureTime}");
    }

    // Метод для продажу квитка
    public void SellTicket(BusTicket ticket)
    {
        Console.WriteLine($"Ticket sold to {ticket.PassengerName} " +
            $"from {ticket.Departure} to {ticket.Destination} " +
            $"at {ticket.DepartureTime}");
    }
}

// Головний клас програми
class Program
{
    static void Main(string[] args)
    {
        // Створення квитків для бронювання та продажу
        BusTicket ticket1 = new BusTicket
```

```
            {
                PassengerName = "Alice",
                Departure = "City A",
                Destination = "City B",
                DepartureTime = DateTime.Now.AddHours(1)
            };

            BusTicket ticket2 = new BusTicket
            {
                PassengerName = "Bob",
                Departure = "City B",
                Destination = "City C",
                DepartureTime = DateTime.Now.AddHours(2)
            };

            // Отримання єдиного екземпляру TicketManager
            TicketManager ticketManager = TicketManager.GetInstance();

            // Бронювання та продаж квитків в окремих потоках
            Thread reserveThread = new Thread(() =>
            {
                ticketManager.ReserveTicket(ticket1);
            });

            Thread sellThread = new Thread(() =>
            {
                ticketManager.SellTicket(ticket2);
            });

            // Запуск потоків
            reserveThread.Start();
            sellThread.Start();

            // Очікування завершення потоків
            reserveThread.Join();
            sellThread.Join();
        }
    }
}
```

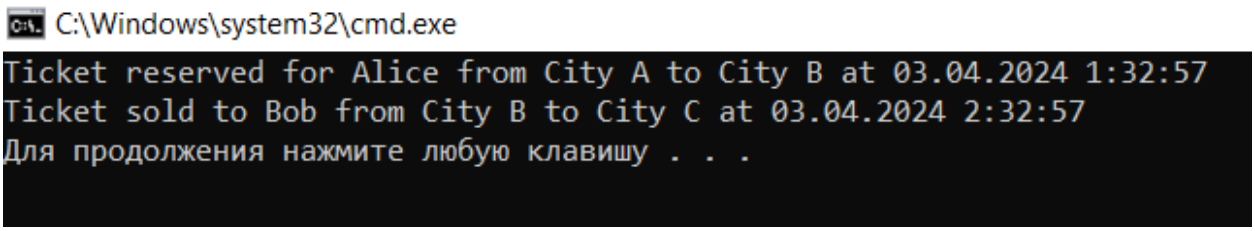

Рисунок 2.15 – Приклад виконання програми

Висновки до розділу

У розділі виконано аналіз характеристик об'єктів галузі проєктування об’єктно-орієнтованої ієрархії класів прикладних програмних систем. Зокрема, виконано аналіз породжувальних патернів проєктування «Фабричний метод», «Абстрактна фабрика», «Одинак».

Виконана постановка нових лабораторних робіт:

1. проєктування предметної області з використанням патерна “Фабричний метод”;

2. проєктування предметної області з використанням патерна “Абстрактна фабрика”;

3. проєктування предметної області з використанням патерна “Одинак”.

Опанування здобувачами вищої освіти запропонованого лабораторного практикуму забезпечить отримання програмних результатів навчання, які сприяють широкому діапазону їх професійної діяльності та високій конкурентоспроможності на ринку праці.

РОЗДІЛ 3 ВИМОГИ ДО КАДРОВОГО ЗАБЕЗПЕЧЕННЯ ОБ'ЄКТУ ГАЛУЗІ

Сьогодні вартість інформації та даних у бізнесі є надзвичайно високою. Компанії у всьому світі активно збирають, аналізують і використовують дані для прийняття стратегічних рішень. Проєктувальники програмного забезпечення розробляють спеціалізовані інструменти та програми для ефективної обробки цих даних, допомагаючи бізнесу отримувати важливу інформацію для прийняття рішень [7].

Проєктувальник програмного забезпечення (Software Architects або Software Designers) – це спеціаліст, який відповідає за архітектуру і дизайн програмних продуктів, розробку і підтримку програмного коду, а також забезпечення його відповідності вимогам проєкту. У сучасному бізнес-середовищі проєктувальники програмного забезпечення є ключовими фігурами, які допомагають компаніям автоматизувати процеси, впроваджувати нові технології і підвищувати ефективність діяльності.

Проєктувальники програмного забезпечення повинні володіти навичками роботи з базами даних, такими як Microsoft SQL Server, MySQL, PostgreSQL, та іншими засобами для зберігання і обробки даних. Їхня робота тісно пов'язана з забезпеченням безпеки даних, оптимізацією запитів і забезпеченням цілісності інформації. У сучасних умовах великі корпорації вимагають від розробників знання методів захисту даних, криптографії та протоколів безпеки [12].

Мова програмування C# – це одна з найпоширеніших мов програмування, яка розроблена компанією Microsoft і активно використовується для створення програмного забезпечення різного рівня складності. C# поєднує в собі простоту синтаксису мов високого рівня та потужні можливості для роботи з об'єктно-орієнтованим програмуванням. Вона пропонує великий набір бібліотек і фреймворків, що дозволяє значно скоротити час розробки та підвищити продуктивність команди розробників.

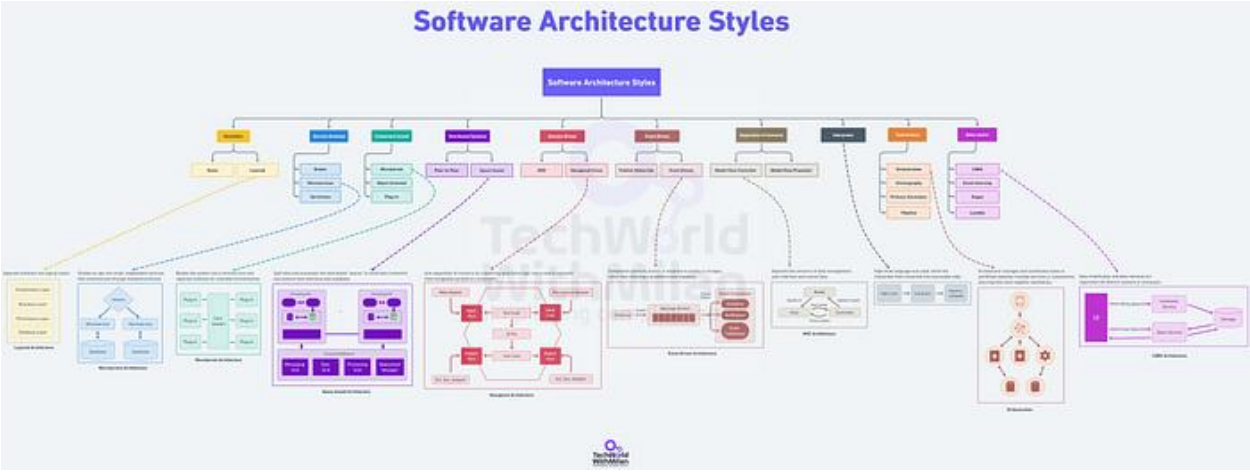

Рисунок 3.1 – Стилі архітектури програмного забезпечення

Хоча стандартні можливості C# досить великі, існує ще величезна кількість бібліотек і фреймворків, таких як .NET, ASP.NET, Entity Framework, які дозволяють вирішувати найрізноманітніші завдання – від розробки веб-додатків до складних корпоративних систем. Саме завдяки цим можливостям мова C# займає провідні позиції серед програмістів у світі.

Мова C# є однією з найбільш популярних для створення програмного забезпечення завдяки своїй інтеграції з Microsoft Visual Studio – однією з найпотужніших платформ для розробки, що підтримує безліч інструментів для дебагінгу, тестування та випуску програмних продуктів. Visual Studio дозволяє розробникам використовувати вбудовані засоби аналізу коду, інтеграцію з системами контролю версій і різні фреймворки для створення веб-додатків, мобільних застосунків і корпоративних систем.

Крім технічних навичок, проєктувальники програмного забезпечення повинні володіти розвинутими навичками управління проектами, комунікації та взаємодії з командою. Це включає вміння ефективно планувати роботу, встановлювати цілі, визначати пріоритети і досягати результатів у визначені терміни. Розробники часто працюють у складі великих команд, де важливо вміти знаходити спільну мову з колегами, менеджерами проектів та іншими стейкхолдерами.

Один з ключових елементів успіху проєктувальника програмного

забезпечення – це постійне вдосконалення своїх навичок та знань. Сфера інформаційних технологій розвивається надзвичайно швидко, і ті, хто прагне бути на передовій, повинні завжди слідкувати за новими трендами та технологіями. Професійний розвиток включає участь у спеціалізованих курсах, конференціях, майстер-класах, а також самостійне вивчення нових інструментів та платформ [28-29, 34].

У сучасному IT-світі існує безліч напрямків для розвитку кар'єри проєктувальника програмного забезпечення. Вони можуть перейти на позиції архітектора програмних рішень, технічного директора або навіть керівника IT-відділу компанії. Цей шлях вимагає не лише глибоких технічних знань, а й стратегічного мислення, вміння керувати командами та проєктами, а також досконалого розуміння бізнес-процесів [35].

На сьогоднішній день проєктування програмного забезпечення – це одна з найбільш затребуваних професій. Попит на кваліфікованих розробників перевищує пропозицію, що створює величезні можливості для тих, хто володіє необхідними навичками. Багато компаній готові інвестувати у навчання та розвиток своїх співробітників, аби лише залучити талановитих фахівців у свою команду.

Як правило, успішні проєктувальники програмного забезпечення володіють знаннями з об'єктно-орієнтованого програмування, розробки алгоритмів, роботи з різними форматами даних та візуалізації результатів. Їх робота включає проєктування архітектури систем, оптимізацію коду та розробку інтерфейсів користувача, які відповідають вимогам сучасних стандартів.

Проєктування програмного забезпечення стає все більш популярним напрямком через зростаючий попит на кваліфікованих фахівців у цій сфері. Багато університетів і навчальних закладів зараз активно викладають основи проєктування програмного забезпечення, і є безліч курсів для підвищення кваліфікації.

Проєктувальники програмного забезпечення поділяються на кілька

спеціалізацій, залежно від типу систем, з якими вони працюють, і від їхнього рівня залученості в різних аспектах розробки програмного забезпечення.

Розглянемо основні види проєктувальників програмного забезпечення.

1. Архітектор програмного забезпечення (Software Architect) відповідає за загальну архітектуру системи [23]. Він проєктує високорівневу структуру програми, визначає компоненти, їхні взаємодії та інтерфейси. Також він розробляє план реалізації системи, вибирає відповідні технології та визначає стандарти кодування. Може спеціалізуватися на конкретній галузі, наприклад, розподілених системах або хмарних технологіях.

2. Технічний архітектор (Technical Architect) зазвичай зосереджується на технічних аспектах проєкту та розробляє технічні рішення для конкретних завдань та контролює їх реалізацію. Він відповідає за якість коду, архітектурні шаблони та забезпечує відповідність технічних рішень загальним вимогам бізнесу [8]. Технічний архітектор часто взаємодіє з командою розробників, щоб забезпечити коректну реалізацію архітектури.

3. Архітектор корпоративних систем (Enterprise Architect) працює на рівні компанії, забезпечуючи узгодженість IT-інфраструктури з бізнес-стратегією організації [25]. Він розробляє масштабні IT-рішення, що охоплюють усі аспекти діяльності компанії, інтегрує різні програмні системи та забезпечує їхню взаємодію для оптимізації бізнес-процесів. Архітектор систем часто приймає рішення про закупівлю нових технологій та їх інтеграцію в існуючу інфраструктуру.

4. Розробник інтерфейсу користувача (UI/UX Designer) зосереджується на проєктуванні взаємодії користувача з програмним забезпеченням, створює інтерфейси, які є зручними, естетичними та інтуїтивно зрозумілими. Розробник відповідає за візуальні аспекти програми та користувацький досвід. Він Тісно співпрацює з командою розробників, щоб забезпечити відповідність дизайну технічним можливостям.

5. Архітектор баз даних (Database Architect) проєктує структуру баз даних, визначає моделі даних та їх взаємозв'язки, оптимізує бази даних для

ефективного зберігання та доступу до інформації. Архітектор БД в першу чергу забезпечує цілісність, безпеку та продуктивність баз даних. Він співпрацює з іншими командами, щоб забезпечити відповідність баз даних потребам додатків.

6. Архітектор рішень (Solution Architect) створює спеціалізовані рішення для конкретних бізнес-задач та визначає, які технології, платформи та інструменти будуть використані для реалізації проєкту. Він координує роботу між різними командами, щоб забезпечити успішну інтеграцію різних компонентів системи. Також він забезпечує, щоб розроблене рішення відповідало вимогам замовника та стандартам якості.

7. Архітектор безпеки (Security Architect) проєктує програмні системи та процедури, спрямовані на забезпечення безпеки програмного забезпечення та захист даних. Розробляє стратегії кібербезпеки та визначає вимоги до захисту інформації. Виявляє потенційні загрози, вразливості та розробляє методи їхнього усунення. Співпрацює з іншими розробниками для інтеграції безпеки на всіх етапах розробки програмного забезпечення.

Наведені спеціалізації проєктувальників програмного забезпечення допомагають забезпечити якість, надійність і ефективність програмних рішень, а також їх відповідність бізнес-цілям і потребам користувачів.

Експерти вважають, що ринок праці у сфері проєктування програмного забезпечення перегрітий через високий попит на кваліфікованих фахівців і обмежену кількість дійсно грамотних професіоналів [31-33]. Для того щоб залишатися затребуваним фахівцем, важливо постійно вдосконалювати свої навички, вивчати нові інструменти та технології, а також брати участь у проектах і навчальних програмах.

Проєктувальники програмного забезпечення часто співпрацюють з іншими розробниками, інженерами, менеджерами проєктів, що дозволяє створювати комплексні рішення для бізнесу. Їх робота має стратегічний характер і вимагає тісної взаємодії з усіма учасниками проєкту для досягнення спільних цілей.

Зростання і розвиток професії проєктувальника програмного забезпечення відкриває нові можливості для кар'єрного зростання. Фахівці можуть переходити на посади технічних лідерів, архітекторів програмного забезпечення або навіть керівників у великих компаніях.

Висновки до розділу

Аналіз вимог до кадрового забезпечення об'єкту ІТ-галузі вказує на високий попит на проєктувальників програмного забезпечення. Професія стала ключовою у процесі цифрової трансформації бізнесу та державних установ. Успішні фахівці у цій сфері повинні поєднувати технічні навички з розвиненими комунікативними здібностями та здатністю ефективно працювати у команді.

Ринок праці демонструє значне зростання попиту на проєктувальників програмного забезпечення, особливо в умовах швидкого розвитку інформаційних технологій і переходу до цифрової економіки. Підготовка спеціалістів має бути спрямована на формування не лише технічних знань, а й на розвиток стратегічного мислення, управління проектами і гнучких навичок, що дозволить адаптуватися до нових викликів у цій сфері.

РОЗДІЛ 4 МЕТОДИКА ПРОФЕСІЙНОЇ ПІДГОТОВКИ ФАХІВЦІВ З ЦИФРОВИХ ТЕХНОЛОГІЙ ДЛЯ ВИКЛАДАННЯ ОСВІТНЬОГО МОДУЛЯ «ПОРОДЖУВАЛЬНІ ПАТЕРНИ ПРОЄКТУВАННЯ» У ЗАКЛАДАХ ВИЩОЇ ОСВІТИ. ДИДАКТИЧНИЙ ПРОЕКТ КОНСУЛЬТАТИВНОГО ЗАНЯТТЯ З ТЕМИ «ПАТЕРНИ ПРОЕКТУВАННЯ «АБСТРАКТНА ФАБРИКА» ТА «ОДИНАК» ДИСЦИПЛІНИ «КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ» ДЛЯ ЗДОБУВАЧІВ ВИЩОЇ ОСВІТИ СПЕЦІАЛЬНОСТІ 015 ПРОФЕСІЙНА ОСВІТА (ЦИФРОВІ ТЕХНОЛОГІЇ)

4.1 Вихідні дані до проєкту

Навчальний заклад: Бахмутський навчально-науковий професійно-педагогічний інститут Харківського національного університету імені В.Н. Каразіна;

галузь знань: 01 Освіта / Педагогіка;

спеціальність: 015 Професійна освіта (Цифрові технології);

рівень вищої освіти: перший (бакалаврський);

освітній ступінь: бакалавр;

дисципліна: «Конструювання програмного забезпечення»;

тема: «Патерни проектування «Абстрактна фабрика» та «Одинак».

Дисципліна містить такі характеристики, як:

кількість кредитів – 4.

Загальну кількість годин для вивчення дисципліни – 120 навчальних годин з яких 78 годин самостійної роботи та 42 години аудиторної роботи (12 годин лекційних занять та 30 годин лабораторної роботи) для денної форми навчання;

загальну кількість годин для вивчення дисципліни – 120 навчальних годин з яких 108 годин самостійної роботи та 12 годин аудиторної роботи (4

години лекційних занять та 8 годин лабораторної роботи) для заочної форми навчання;

Співвідношення кількості годин аудиторних занять до самостійної і індивідуальної роботи становить:

- для денної форми навчання – 42/78;
- для заочної форми навчання – 12/108.

Дисципліна «Конструювання програмного забезпечення» викладається у 6-му семестрі 3-го року професійної підготовки бакалаврів для денної форми навчання та у 8-му семестрі 4-го року професійної підготовки бакалаврів для заочної форми навчання.

Форма контролю: Екзамен.

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».

4.2 Проектування цілей консультативного заняття

Визначення навчальних цілей заняття є одним з головних питань, від вирішення якого залежить методична організація заняття, вибір його форм, методів, засобів навчання та контролю. В цьому сенсі навчальні цілі є системоутворюючим фактором, бо, відображуючи кінцеві необхідні результати досягнень здобувачів вищої освіти, чітко детермінують принципи побудови системи навчання по всіх основних її параметрах.

Тому методична підготовка здобувачів вищої освіти передбачає перш за все оволодіння вміннями диференційовано визначати навчальні цілі, відповідно до певних рівнів професійної підготовки або рівнів засвоєння.

Проектування цілей консультативного заняття представлені у табл. 4.1 [2].

Таблиця 4.1

Цілі консультативного заняття

Цілі консультативного заняття	Цілі формування різних рівнів засвоєння навчального матеріалу	Умови досягнення	Результат у вигляді дій здобувачів освіти
1	З переліку визначень впізнавати основні поняття теми «Патерни проектування «Абстрактна фабрика» та «Одинак» такі, як фабричний метод, абстрактна фабрика, одинак, уміти називати розподіл функцій між викладачем і засобами навчання.	Знати визначення понять «Патерн Адаптер», «Патерн Спостерігач та «Патерн Стратегія», знання сутності поняття «Інформаційні освітні технології».	Правильно названі з переліку основні поняття теми «Патерни проектування «Абстрактна фабрика» та «Одинак» такі, як фабричний метод, абстрактна фабрика, одинак, уміти називати розподіл функцій між викладачем і засобами навчання.
2	Уміти характеризувати функціональні можливості комп'ютера в діяльності педагога, уміти класифікувати компоненти інформаційних технологій та інформаційні технології за видами діяльності викладачів.	Виконання дій першого рівня: правильно названі з переліку основні поняття теми «Патерни проектування «Абстрактна фабрика» та «Одинак» такі, як фабричний метод, абстрактна фабрика, одинак, уміти називати розподіл функцій між викладачем і засобами навчання.	Правильно охарактеризовані функціональні можливості комп'ютера в діяльності педагога, про класифіковані компоненти інформаційних технологій та інформаційні технології за видами діяльності викладачів.
3	Уміти робити еволюційний аналіз поколінь інформаційних технологій навчання та робити висновки, щодо розвитку інформаційних технологій.	Виконання дій першого і другого рівнів: правильно охарактеризовані функціональні можливості комп'ютера в діяльності педагога, про класифіковані компоненти ІТ за видами діяльності викладачів.	Правильно зроблений еволюційний аналіз поколінь інформаційних технологій навчання та висновки, щодо розвитку інформаційних технологій.
4	Уміти досліджувати інформаційні та комп'ютерні технології.	Виконання дій першого, другого і третього рівнів: правильно зроблений еволюційний аналіз поколінь інформаційних технологій навчання та висновки, щодо розвитку ІТ.	Правильно досліджені інформаційні та комп'ютерні технології.

Таким чином, нами були розроблені цілі консультативного заняття з теми «Патерни проектування «Абстрактна фабрика» та «Одинак» дисципліни «Конструювання програмного забезпечення» для здобувачів вищої освіти спеціальності 015 Професійна освіта (Цифрові технології).

4.3 Перелік джерел інформації

Майбутній фахівець має вміти самостійно користуватися джерелами інформації.

Наведемо перелік джерел інформації для підготовки здобувачів вищої освіти 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:

Основна (базова) 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.

Додаткова (допоміжна) література

1. Chykunov P., Chykunov I. Services for creating UML class diagrams. Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні: матер. VI Всеукр. наук.-практ. інтернет-конф. (м. Харків, 06-07 грудня 2023 р.). Харків: ННППІ УІПА, 2023. С 42-43.

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

3. 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.

4. Трофименко О. Г., Соколов А. В., Чикунов П. О., Ахмаметьєва Г. В., Атанасевич А. О. Аналіз ролі фахівців із тестування та забезпечення якості. *Збірник наукових праць Національного університету кораблебудування імені адмірала Макарова*. 2024. № 3 (496). С. 106-113. DOI: https://doi.org/10.15589/znp2024.3(496).16.

5. Чикунов П.О. Застосування патернів проектування у процесі конструювання програмного забезпечення. *Європейські орієнтири розвитку України в умовах війни та глобальних викликів XXI століття: синергія наукових, освітніх та технологічних рішень* : у 2 т. : матер. Міжнар. наук.-практ. конф. (м. Одеса, 19 травня 2023 р.). Одеса : Видавництво «Юридика», 2023. Т. 1. С. 606-608.

6. 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.

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.

9. http://surl.li/ksmxr – The top three clean code principles to follow in 2023

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

Основним джерелом для підготовки здобувачів вищої освіти до консультації є конспект лекцій, розроблений викладачем, оскільки він є найбільш адаптованим до робочої програми дисципліни «Конструювання програмного забезпечення».

4.4 Визначення найбільш складних для розуміння та засвоєння питань

На даному етапі визначимо найбільш складні для розуміння та засвоєння питання (табл. 4.2) [2].

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
«Патерни проектування «Абстрактна фабрика» та «Одинак».	1. Фабричний метод. 2. Абстрактна фабрика. 3. Одинак	1. Назвіть та охарактеризуйте, будь ласка, які переваги патернів проектування? 2. Що є недоліком патерну? 3. В чому призначення патерну «Абстрактна фабрика»?	1. Переваги використання патернів такі: • Підвищення якості коду. • Поліпшення читабельності коду. • Підвищення продуктивності. • Прискорення розробки. 2. Недоліком є те, що підтримувати новий вид продуктів важко. Розширення абстрактної фабрики для виготовлення нових видів продуктів – складний процес. 3. Призначення патерну «Абстрактна фабрика» в тому, що він надає інтерфейс для створення сімейств взаємопов'язаних або заємозалежних об'єктів не специфікуючи їх конкретних класів.

Отже, на даному етапі ми визначили найбільш складні для розуміння та засвоєння питання з теми «Патерни проектування «Абстрактна фабрика» та «Одинак» дисципліни «Конструювання програмного забезпечення» для здобувачів вищої освіти спеціальності 015 Професійна освіта (Цифрові технології).

4.5 Вибір дидактичних методів активізації

Здійснюємо вибір методів активізації навчальної діяльності здобувачів вищої освіти на консультації для спонукання до активної розумової та практичної діяльності в процесі оволодіння навчальним матеріалом (табл. 4.3) [3].

Таблиця 4.3

Методи активізації навчальної діяльності здобувачів освіти на консультації

Дидактичні методи	Реалізація методів при проведенні консультаційного заняття
1	2
Методи підвищення наочності	Використання інтерактивної дошки для демонстрації слайдів з теми «Патерни проектування «Абстрактна фабрика» та «Одинак» та мультимедійного проектора.
Мотиваційні методи	Для реалізації мотивації використаємо: тип: внутрішня мотивація; вид: вступна мотивація; метод: мотивуючий вступ; прийом: віднесення до особистості. Формування позитивного відношення до вивчення даної теми: «Тема «Патерни проектування «Абстрактна фабрика» та «Одинак» – одна з найбільш важливих тем курсу – вміти розробляти та аналізувати, вміти обирати методику конструювання, оцінювати ефективність та складність та самостійно опановувати нові методи та технології конструювання ПЗ. ».
Проблемні методи	Використання проблемного питання. Проблемне питання: «Коли використовується патерн «Абстрактна фабрика»?».
Комунікативні методи	Імітація ситуацій з реального життя. Усі наші знання необхідні для подальшої вашої професійної діяльності, тому що знання сутності інформаційних технологій дає змогу збільшити розуміння предмету. Отже, наприклад: розподіл функцій між викладачем і засобами навчання для

Продовження табл. 4.3

1	2
	кожного покоління ІТН. Перетин функцій в деяких поколіннях ІТО викликаний тим, що деякі функції одночасно виконувалися викладачем і засобами навчання. Наприклад, викладач міг демонструвати простий фізичний або хімічний досвід, використовуючи лабораторні установки або лабораторне обладнання, і одночасно показати навчальний фільм про ядерну реакцію синтезу речовин в колоні тощо.

Таким чином, ми обрали методи активізації навчальної діяльності здобувачів вищої освіти на консультації.

4.6 Вибір способів організації консультативного заняття

Здійснюємо вибір способів організації консультативного заняття, згідно даних, наведених в таблиці 4.4 [1].

Таблиця 4.4

Варіанти організації консультативного заняття

№ варіанта	Етапи організації заняття	Характеристика варіанта
1	- вступне слово лектора, - відповіді на питання студентів і обговорення їх, - заключне слово викладача	Недоліком цього варіанту проведення лекції-консультації є відсутність 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, - відповіді на питання, що поступили, - відповіді на додаткові питання, - обмін думками, - висновки	Цей варіант, на відміну від попереднього, дозволяє викладачу групувати відповіді, що сприяє кращому засвоєнню навчального матеріалу здобувачами освіти.
3	- видача завдань на самостійне вивчення матеріалу теми. - підготовка питань лектору. - відповіді і їх обговорення	В цьому випадку консультування грає функцію додаткового інформування зі складних питань і пояснення незрозумілого навчального матеріалу.
4	- повідомлення теми, - консультування декількома фахівцями в певній області науки і техніка з актуальних питань науки і нової техніки	Цей варіант проводиться 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.

Консультативне заняття будемо здійснювати згідно першого варіанту організації, на ньому викладач пояснює питання, які здалися складними здобувачами освіти.

4.7 Розробка сценарію проведення консультативного заняття

Проведення консультативного заняття пропонуємо здійснити згідно обраного варіанту його організації (табл. 4.5) [3].

Таблиця 4.5

Сценарій консультативного заняття

Етапи проведення консульта-тивного заняття	Дії викладача	Дії здобувачів освіти
1	2	3
Організа-ційний момент	Вітання, фіксація відсутніх, перевірка зовнішньої обстановки в аудиторії	Вітання викладача. Перевірка присутності, формування позитивної мотивації на здійснення навчальної діяльності.
Повідомлен-ня теми і мети заняття	Повідомлення теми заняття: «Патерни проектування «Абстрактна фабрика» та «Одинак». Формулювання його цілей: Сформувати уміння давати визначення поняттям: «Патерн Адаптер», «Патерн Спостерігач та «Патерн Стратегія».	Сприйняття теми заняття та її мети.
Мотивація мети	Повідомлення важливості вивчення даної теми: «Патерни проектування «Абстрактна фабрика» та «Одинак» є достатньо актуальним. Відповідно до вашої майбутньої професійної діяльності знання цього навчального матеріалу знадобляться вам для роботи на підприємстві, а саме це дозволить правильно використовувати інформаційні технології. Все це є одним з основних завдань вашої майбутньої діяльності. Від того, наскільки успішно ви справитеся з даним завданням, вже працюючи на підприємстві, залежить його благополуччя, а вам дозволить рости у якості високоякісного програміста та стверджуватимуть вас як висококваліфікованого фахівця».	Сприйняття важливості і актуальності вивчення теми, прояв інтересу до неї.

Продовження табл. 4.5

1	2	3
Актуалізація знань	Викладач проводить фронтальне усне опитування з метою перевірки базових знань: 1. Назвіть будь ласка, які дві теми актуальні для патернів, що породжують об’єкти?	Здобувачі освіти беруть участь у опитуванні та відповідають на поставлені питання. Передбачувані відповіді: 1. По-перше, ці патерни інкапсулюють знання про конкретні класи, які застосовуються в системі. По-друге, приховують деталі того, як ці класи створюються і взаємодіють. Єдина інформація про об'єкти, відома системі, - це їх інтерфейси, визначені за допомогою абстрактних класів. Отже, патерни, що породжують об’єкти, забезпечують більшу гнучкість при вирішенні питання про те, що створюється, хто це створює, як і коли. Можна зібрати систему з «готових» об'єктів з самої різною структурою і функціональністю статично (на етапі компіляції) або динамічно (під час виконання).
	2. Назвіть будь ласка, три основні типи патернів?	2. Існує три основні типи патернів: **Породжуючі патерни** (Creational Patterns). Вони забезпечують механізми створення об’єктів, даючи змогу зробити систему незалежною від способу їхнього створення, композиції та представлення. Приклади: Сінглтон (Singleton), Фабричний метод (Factory Method), Абстрактна фабрика (Abstract Factory).

Продовження табл. 4.5

1	2	3
	3. Скажіть, будь ласка, в чому полгяіє патернів проектування?	**Структурні патерни** (Structural Patterns). Визначають способи композиції класів або об’єктів для формування більших структур. Приклади таких: Адаптер (Adapter), Декоратор (Decorator), Фасад (Facade). **Поведінкові патерни** (Behavioral Patterns). Позначають алгоритми та способи взаємодії між об’єктами, забезпечуючи більш ефективну та гнучку взаємодію. Приклади: Спостерігач (Observer), Стратегія (Strategy), Команда (Command). 3. Їхня роль полягає в тому, що вони допомагають нам проектувати програми більш організовано, роблять код більш перевикористовуваним і спрощують підтримку системи.
Формування ООД	Викладач проводить консультацію згідно плану, за допомогою методу – пояснення: 1. Фабричний метод. 2. Абстрактна фабрика. 3. Одинак.	Слухають пояснення, конспектують.
Визначення проблемних моментів під час вивчення питань теми та формування ВД	Викладач запитує у здобувачів освіти про проблеми, які виникли у них під час самостійного вивчення теми. Викладач відповідає на поставлені запитання: 1. Цей патерн використовується коли: система не повинна залежати від того, як створюються, компонуються і представляються об'єкти, що входять до неї; взаємопов'язані об'єкти, що входять в сімейство, повинні використовуватися разом і вам необхідно забезпечити виконання цього обмеження; система повинна конфігуруватися одним з сімейств об'єктів, з яких вона складається; необхідно створювати групи або сімейства	Питання здобувачів освіти: 1. Коли використовується патерн «Абстрактна фабрика»?

Продовження табл. 4.5

1	2	3
	взаємопов'язаних об'єктів, виключаючи можливість одночасного використання різнорідних об'єктів в одному контексті; необхідно надати бібліотеку об'єктів, розкриваючи тільки їх інтерфейси, але не реалізацію. 2. Абстрактна фабрика – це породжувальний патерн проектування, що дає змогу створювати сімейства пов’язаних об’єктів, не прив’язуючись до конкретних класів створюваних об’єктів. 3. Фабричний метод – визначає інтерфейс для створення об’єкта, але залишає підкласам рішення про те, який клас інстанціювати. Застосування: коли заздалегідь невідомо, які об’єкти потрібно створювати, і коли система має бути незалежною від того, як саме створюються, комбінуються та відображаються ці об’єкти.	2. Що таке «Абстрактна фабрика»? 3. Що визначає «Фабричний метод» і коли він застосовується?
Підведення підсумків	Викладач підводить підсумки проведення консультації: «На консультації ми розглянули складні для вас питання теми для самостійного вивчення. Для перевірці засвоєння складного для вас матеріалу дайте мені, будь ласка, відповідь на запитання: «Що таке патерни проектування і для чого їх використовують?»	Здобувачі слухають, відповідають: Патерни проектування – це перевірені часом рішення, які допомагають розробникам створювати надійне, гнучке та масштабоване ПЗ. Вони являють собою шаблони, засновані на загальних принципах і архітектурних рішеннях, які можна використовувати в різних ситуаціях.. Здобувачі освіти прощаються.

Отже, на цьому етапі ми розробили сценарій проведення консультативного заняття у відповідності до обраного варіанту його організації.

На заключному етапі представлено контурний конспект з теми «Патерни проектування «Абстрактна фабрика» та «Одинак» дисципліни «Конструювання програмного забезпечення я» для здобувачів вищої освіти

спеціальності 015 Професійна освіта (Цифрові технології).

За обсягом інформації, що представляється, конспекти діляться на повні і контурні (опорні), а за способом подання інформації – на плани-конспекти і конспекти-схеми. План-конспект стисло представляє зміст кожного з пунктів плану. Конспект-схема – це ієрархія понять теми, впорядкованих згідно плану і доповнених основними відомостями.

У повному конспекті міститься, переважно, вся нова основна інформація. У контурному (опорному) ж конспекті містяться тільки ключові положення нової основної інформації, виражені за допомогою таблиць, графіків, абревіатури, різного роду позначень, акцентів.

Контурний конспект заняття з теми «Патерни проектування «Абстрактна фабрика» та «Одинак» дисципліни «Конструювання програмного забезпечення» для здобувачів вищої освіти спеціальності 015 Професійна освіта (Цифрові технології) представлено у Додатку.

Висновки до розділу

У четвертому розділі розроблено дидактичний проект консультативного заняття з теми «Патерни проектування «Абстрактна фабрика» та «Одинак» дисципліни «Конструювання програмного забезпечення» для здобувачів вищої освіти спеціальності 015 Професійна освіта. (Цифрові технології).

Сформульовано цілі консультативного заняття. Обрано методи активізації навчальної діяльності здобувачів освіти 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 здобувачів освіти до консультації згідно з робочою програмою дисципліни «Конструювання програмного забезпечення». Подано список використаних джерел та відповідні посилання.

ВИСНОВКИ

У ході дослідження уточнено та систематизовано основні поняття, що характеризують сутність та структуру процесу професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Породжувальні патерни проєктування» у закладах вищої освіти та виділено пріоритетні напрямки удосконалення професійних компетентностей.

З’ясовано, що професійна підготовка фахівців з цифрових технологій для викладання освітнього модуля «Породжувальні патерни проєктування» у закладах вищої освіти є актуальною у професійній педагогіці.

Проаналізовано ступінь актуальності проблеми професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Породжувальні патерни проєктування» у закладах вищої освіти.

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

Виконано аналіз характеристик об'єктів галузі проєктування об’єктно-орієнтованої ієрархії класів прикладних програмних систем. Зокрема, виконано аналіз породжувальних патернів проєктування «Фабричний метод», «Абстрактна фабрика», «Одинак».

Виконана постановка нових лабораторних робіт:

1. проєктування предметної області з використанням патерна «Фабричний метод»;

2. проєктування предметної області з використанням патерна «Абстрактна фабрика»;

3. проєктування предметної області з використанням патерна «Одинак».

Опанування здобувачами вищої освіти запропонованого лабораторного практикуму забезпечить отримання програмних результатів навчання, які сприяють широкому діапазону їх професійної діяльності та високій конкурентоспроможності на ринку праці.

Результати аналізу вимог до кадрового забезпечення об'єкту ІТ-галузі вказує на високий попит на проєктувальників програмного забезпечення. Професія стала ключовою у процесі цифрової трансформації бізнесу та державних установ. Успішні фахівці у цій сфері повинні поєднувати технічні навички з розвиненими комунікативними здібностями та здатністю ефективно працювати у команді. Ринок праці демонструє значне зростання попиту на проєктувальників програмного забезпечення, особливо в умовах швидкого розвитку інформаційних технологій і переходу до цифрової економіки.

Розроблено дидактичний проект консультативного заняття з теми «Патерни проектування «Абстрактна фабрика» та «Одинак» дисципліни «Конструювання програмного забезпечення» для здобувачів вищої освіти спеціальності 015 Професійна освіта. (Цифрові технології).

Сформульовано цілі консультативного заняття. Обрано методи активізації навчальної діяльності здобувачів освіти на консультації. Здійснено вибір способів організації консультативного заняття. Розроблено сценарій проведення консультативного заняття. Проаналізовано джерела інформації для підготовки здобувачів освіти до консультації згідно з робочою програмою дисципліни «Конструювання програмного забезпечення».

Апробація результатів дослідження виконана під час виконання лабораторного практикуму з дисципліни «Конструювання програмного забезпечення» бакалаврами спеціальності 122 «Комп’ютерні науки» Національного університету «Одеська юридична академія».

За основними результатами дослідження виконана публікація тез доповідей на конференції:

– Рябуха О.В. Лабораторний практикум «Породжувальні патерни проєктування» // Матеріали VIII міжнародної науково-практичної конференції здобувачів вищої освіти та молодих учених «Студенти та молодь – для майбутнього країни» (м. Харків, 14-15 листопада 2024 р.) [упоряд. Г. Г. Михальченко]. Харків, ХНУ ім. Каразіна, 2024.

СПИСОК ВИКОРИСТОВАНИХ ДЖЕРЕЛ

1. Баран С.В. Розробка програмного забезпечення з використанням патернів проектування: Навчальний посібник. – Кривий Ріг: Державний університет економіки і технологій, 2023. –203 с.

2. Порівняння фабрик. URL: https://refactoring.guru/uk/design-patterns/factory-comparison

3. Абстрактна фабрика. URL: https://refactoring.guru/uk/design-patterns/abstract-factory

4. Одинак. URL: https://refactoring.guru/uk/design-patterns/singleton

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

6. Most Common Software Architecture Styles. URL: https://medium.com/@techworldwithmilan/most-common-software-architecture-styles-86881d779683

7. Струк А. Формування здатності майбутніх інженерів-програмістів до проектування програмного забезпечення. *Journal of Information Technologies in Education (ITE)*, 2022, №50. Pp. 61-79.

8. Оберванюк Н.-П. Б. Методика забезпечення якості при проектуванні архітектури програмного забезпечення в Agile-проектах. Master's Thesis 2020.

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

10. Bukovčan M., et al. Clean architecture of client-side software development for smart furniture control. In: *2022 11th Mediterranean Conference on Embedded Computing (MECO)*. IEEE, 2022. Pp. 1-4.

11. 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.

12. Anagnostopoulos A. Hands-On Software Engineering with Golang: Move beyond basic programming to design and build reliable software with clean code. Packt Publishing Ltd, 2020.

13. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. *Addison-Wesley Professional*, 2004.

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

15. Hunt A., Thomas D. The Pragmatic Programmer: Your Journey to Mastery. Addison-Wesley Professional, 2019.

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

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

18. 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.

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

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

21. GeeksforGeeks. Software Engineering Tutorial. URL: https://www.geeksforgeeks.org/software-engineering/

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

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

24. Martin Fowler's Blog. URL: https://www.martinfowler.com/

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

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

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

28. DOU. 7 сертифікацій для того, щоб почати кар'єру в IT. URL: https://dou.ua/lenta/articles/certifications-to-start-career/

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

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

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

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. Про роботу, війну, новини й не тільки: добірка Telegram-каналів українських IT-фахівців. URL: https://dou.ua/lenta/articles/telegram-channels-new-list/

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

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

36. Антонюк Л.Л. Компетентністний підхід у вищій освіті: світовий досвід навч. пос. Київ : КНЕУ, 2016.66 с.

37. Професійна педагогіка : Підручник / Авт. : О. В. Грабовський, Л. В.Коломієць, О. С. Савельєва, А. В.Семенова, В. Ф.Яні; за заг. ред. А. В.Семенової. – Одеса : Бондаренко М. О., 2020. – 575 с.

38. Професійна педагогіка: навч. посібник для вищих навч. закладів/ В. І. Жигірь, О. Чернєга ; за ред. М. В. Вачевського. - Київ: К.: Кондор, 2016. – 336 с. 978-966-351-359-1. Код 233704.

39. Зайченко І. В. Теорія і методика професійного навчання: навч. посібник. 2-е вид., доповн. і переробл. К.: Видавництво Ліра-К, 2016. 580 с.

40. 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.).

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. Кулешова В.В., Мальована В.В. Особливості особистості викладача технічних дисциплін у вищих навчальних закладах. *Проблеми інженерно-педагогічної освіти*. Збірник наукових праць. №50-51 Харків: УІПА,– 2016 р., – С.322-329

43. Кулешова В.В., Мальована В.В. Формування професійних методичних умінь у майбутніх інженерів-педагогів економічного профілю / *Міжнародний науковий журнал «ІНТЕРНАУКА».* №7 (29) Київ:– 2017 р., – С.26-29

44. Кулешова В.В. Формування креативної компетентності майбутніх інженерів у процесі професійної підготовки / *Проблеми інженерно-педагогічної освіти.* Збірник наукових праць. № 58. Харків: УІПА,– 2018 р., – С.21-26

45. Коваленко А.В. Шляхи забезпечення формування безперервної освіти: школа – ПЗО – Фаховий коледж – Академія. *Роль закладів фахової передвищої та 194 професійної освіти в системі безперервної освіти*: матеріали VII наук.-практ. конф. Одеса, 2020. С.21-24.

46. Олійник В. В. Відкрита післядипломна педагогічна освіта: нові моделі та форми професійного розвитку / Освіта дорослих у перспективі змін: інновації, технології, прогнози: колективна монографія / За ред.. А. Василюк, А. Стоговського. – Ніжин: Видавець ПП Лисенко М.М., 2017. – 248 с.

47. Ортинський В. Л. Педагогіка вищої школи: навч. посіб. / Ортинський В. Л. – Центр учбової літератури, 2017. – 472 с

48. Професійна освіта України на шляху до євроінтеграції (1992–2017) / науков. ред. Н.Г. Ничкало; упорядники: Л.В. Горбань, В.П. Тименко. – К.: ДП «Інформ.-аналіт. агенство», 2018. – 358 с.

49. Сисоєва С.О. Теорія і практика вищої освіти: навч. посібник / С.О. Сисоєва, І.В. Соколова. – К., 2016. – 338 с. – Режим доступу: http://lib.iitta.gov.ua/711948/1/Sysoieva_Socolova_2016_pos.pdf

50. Теорія та методика викладання фахових дисциплін у ЗВО: навчально-методичний посібник / укладач І. В. Казанжи – Миколаїв : СПД Румянцева, 2018. – 154 с.

51. Теорія і практика вищої професійної освіти в Україні : навч. посіб. для магістрантів зі спеціальності 011 «Освітні, педагогічні науки» / [авт.-укл.: Т.О. Дороніна]. – Кривий Ріг : КДПУ, 2018. – 250 с.

52. Методика професійного навчання: метод. вказ. по виконанню курсової роботи для здобувачів освіти освітнього ступеня «бакалавр» денної та заочної форми навч. інженерно-педагогічних спеціальностей / ННППІ Укр. інж.-пед. акад. ; упоряд. : В. В. Кулешова, В.В. Мальована, Ю.С. Бобрикова ; за заг. ред. д-ра пед. наук, проф.В. В. Кулешової. – Бахмут, 2022. – 92 с.

53. Методика професійного навчання : конспект лекцій для здобувачів вищої освіти ОС «бакалавр» денної та заоч. форм здобуття освіти спец. 015 Проф. освіта (за спеціалізаціями). Ч. 1 / О. Е. Коваленко, Н. О. Брюханова, Н. В. Корольова; Укр. інж.-пед. акад., Каф. педагогіки, методики та менеджменту освіти. - Харків: УІПА, 2020. - 200 с.

54. Методика професійного навчання: конспект лекцій для здобувачів вищої освіти ОС «бакалавр» денної та заоч. форм здобуття освіти спец. 015 Проф. освіта (за спеціалізаціями). Ч. 2/ О. Е. Коваленко, Н. О. Брюханова, Н. В. Корольова; Укр. інж.-пед. акад., Каф. педагогіки, методики та менеджменту освіти. - Харків: УІПА, 2020. - 180 с.

55. Коваленко О. Е., Брюханова Н. О., Корольова Н.В. Методика

професійного навчання: дидактичне проектування: Підручник для студентів інженерно-педагогічних спеціальностей. – Харків: УІПА, 2019. – 204 с.

56. Коваленко О. Е., Брюханова Н. О., Корольова Н.В. Методика професійного навчання: основні технології навчання: Підручник для студентів інженерно-педагогічних спеціальностей. – Харків: УІПА, 2019. – 174 с.

57. Методика професійного навчання: дидактичне проектування: конспект лекцій для студ. денної та заоч. форм навч. спец. 015.06 "Проф. освіта. Електроніка, радіотехн. та телекомунікації" освітнього рівня "бакалавр"/ Н. Г. Кошелева; Укр. інж.-пед. акад. (Бахмут), ННППІ. - Бахмут, 2017. - 100 с.

58. Методика професійного навчання: основні технології навчання: конспект лекцій для студ. денної та заоч. форм навч. спец. 015.06 "Проф. освіта. Електроніка, радіотехн. та телекомунікації" освітнього рівня "бакалавр"/ Н. Г. Кошелева; Укр. інж.-пед. акад. (Бахмут), ННППІ. - Бахмут, 2017. - 101 с.

59. Теорія і методика професійного навчання : навч. посібник. – 2-е вид., доповн. і переробл. / І.В.Зайченко. – К.: Видавництво Ліра-К, 2016. – 580 с.

60. Методика професійного навчання: навч. посібник для вищих навч. закладів інж.-пед. спец. для традиційної та дистанційної форм навчання. Ч. 1: Дидактичне проектування/ О. Е. Коваленко, Н.О. Брюханова, Н.В. Корольова; Укр. інж.- пед. академія. - 2-ге вид., перероб. та доп.. - Х.: ФОП Шевченко С. О., 2010. - 264 с.

ДОДАТОК А КОНТУРНИЙ КОНСПЕКТ НА ТЕМУ «ВСТУП У ПОРОДЖУВАЛЬНІ ПАТЕРНИ ПРОЄКТУВАННЯ»

Патерни проектування «Абстрактна фабрика» та «Одинак»

Абстрактна фабрика – це породжувальний патерн проектування, що дає змогу створювати сімейства пов'язаних об'єктів, не прив'язуючись до конкретних класів створюваних об'єктів.

Постановка проблеми

Уявіть, що ви пишете симулятор меблевого магазину. Ваш код містить:

1. Сімейство залежних продуктів. Скажімо, Крісло + Диван + Столик.
2. Кілька варіацій цього сімейства. Наприклад, продукти Крісло, Диван та Столик представлені в трьох різних стилях: Ар-деко, Вікторіанському і Модерн.

Сімейства продуктів та їхніх варіацій

Вам потрібно створювати об'єкти продуктів у такий спосіб, щоб вони завжди пасували до інших продуктів того самого сімейства. Це дуже важливо, адже клієнти засмучуються, коли отримують меблі, що не можна поєднати між собою.

Клієнти засмучуються, якщо отримують продукти, що не поєднуються.

Крім того, ви не хочете вносити зміни в існуючий код під час додавання в програму нових продуктів або сімейств. Постачальники часто оновлюють свої каталоги, але ви б не хотіли змінювати вже написаний код кожен раз при надходженні нових моделей меблів.

Рішення

Для початку, патерн Абстрактна фабрика пропонує виділити загальні інтерфейси для окремих продуктів, що складають одне сімейство, і описати в них спільну для цих продуктів поведінку. Так, наприклад, усі варіації крісел отримають спільний інтерфейс Крісло, усі дивани реалізують інтерфейс Диван

тощо.

Всі варіації одного й того самого об’єкта мають жити в одній ієрархії класів.

Далі ви створюєте абстрактну фабрику – загальний інтерфейс, який містить методи створення всіх продуктів сімейства (наприклад, створитиКрісло, створитиДиван і створитиСтолик). Ці операції повинні повертати абстрактні типи продуктів, представлені інтерфейсами, які ми виділили раніше – Крісла, Дивани і Столики.

Конкретні фабрики відповідають певній варіації сімейства продуктів.

Як щодо варіацій продуктів? Для кожної варіації сімейства продуктів ми повинні створити свою власну фабрику, реалізувавши абстрактний інтерфейс. Фабрики створюють продукти однієї варіації. Наприклад, ФабрикаМодерн буде повертати тільки КріслаМодерн,ДиваниМодерн і СтоликиМодерн.

Клієнтський код повинен працювати як із фабриками, так і з продуктами тільки через їхні загальні інтерфейси. Це дозволить подавати у ваші класи будь-які типи фабрик і виробляти будь-які типи продуктів, без необхідності вносити зміни в існуючий код.

Наприклад, клієнтський код просить фабрику зробити стілець. Він не знає, якому типу відповідає ця фабрика. Він не знає, отримає вікторіанський або модерновий стілець. Для нього важливо, щоб на цьому стільці можна було сидіти та щоб цей стілець відмінно виглядав поруч із диваном тієї ж фабрики.

Залишилося прояснити останній момент: хто ж створює об’єкти конкретних фабрик, якщо клієнтський код працює лише із загальними інтерфейсами? Зазвичай програма створює конкретний об’єкт фабрики під час запуску, причому тип фабрики вибирається на підставі параметрів оточення або конфігурації.

Структура

1. AbstractProductA та AbstractProductA оголошують інтерфейси продуктів, що пов’язані один з одним за змістом, але виконують різні функції.
2. ConcreteProduct – великий набір класів, що належать до різних

абстрактних продуктів (крісло/столик), але мають одні й ті самі варіації (Вікторіанський/Модерн).

3. Інтерфейс AbstractFactory оголошує методи створення різних абстрактних продуктів (крісло/столик).

4. ConcreteFactory1 та ConcreteFactory1 належать до кожної з варіації продуктів (Вікторіанський/Модерн) і реалізують методи абстрактної фабрики, даючи змогу створювати всі продукти певної варіації.

5. Незважаючи на те, що конкретні фабрики породжують конкретні продукти, сигнатури їхніх методів мусять повертати відповідні абстрактні продукти. Це дозволить клієнтського коду, що використовує фабрику, не прив'язуватися до конкретних класів продуктів. Клієнт зможе працювати з будь-якими варіаціями продуктів через абстрактні інтерфейси.

Застосування

- Багато архітектур починаються із застосування Фабричного методу (простішого та більш розширюваного за допомогою підкласів) та еволюціонують у бік Абстрактної фабрики, Прототипу або Будівельника (гнучкіших, але й складніших).
- Будівельник концентрується на будівництві складних об'єктів крок за кроком. Абстрактна фабрика спеціалізується на створенні сімейств пов'язаних продуктів. Будівельник повертає продукт тільки після виконання всіх кроків, а Абстрактна фабрика повертає продукт одразу.
- Класи Абстрактної фабрики найчастіше реалізуються за допомогою Фабричного методу, хоча вони можуть бути побудовані і на основі Прототипу.
- Абстрактна фабрика може бути використана замість Фасаду для того, щоб приховати платформо-залежні класи.
- Абстрактна фабрика може працювати спільно з Мостом. Це особливо корисно, якщо у вас є абстракції, які можуть працювати тільки з деякими реалізаціями. В цьому випадку фабрика визначатиме типи створюваних абстракцій та реалізацій.

• Абстрактна фабрика, Будівельник та Прототип можуть реалізовуватися за допомогою Одинака.

Патерн проектування «Одинак» (Singleton)

Одинак – це породжуючий патерн, який гарантує існування тільки одного об’єкта певного класу, а також дозволяє дістатися цього об’єкта з будь-якого місця програми.

Одинак має такі ж переваги та недоліки, що і глобальні змінні. Його неймовірно зручно використовувати, але він порушує модульність вашого коду.

Ви не зможете просто взяти і використовувати клас, залежний від одинака, в іншій програмі. Для цього доведеться емулювати там присутність одинака. Найчастіше ця проблема проявляється при написанні юніт-тестів.

Постановки проблеми

Одинак вирішує відразу дві проблеми (порушуючи принцип єдиного обов’язку класу):

1. Гарантує наявність єдиного екземпляра класу. Найчастіше за все це корисно для доступу до якогось спільного ресурсу, наприклад, бази даних.

Уявіть собі, що ви створили об’єкт, а через деякий час намагаєтесь створити ще один. У цьому випадку хотілося б отримати старий об’єкт замість створення нового.

Таку поведінку неможливо реалізувати за допомогою звичайного конструктора, оскільки конструктор класу завжди повертає новий об’єкт.

Клієнти можуть не підозрювати, що працюють з одним і тим самим об’єктом

2. Надає глобальну точку доступу. Це не просто глобальна змінна, через яку можна дістатися до певного об’єкта. Глобальні змінні не захищені від запису, тому будь-який код може підмінити їхнє значення без вашого відома.

Проте, є ще одна особливість. Було б непогано й зберігати в одному місці код, який вирішує проблему №1, і мати до нього простий та доступний

інтерфейс.

Цікаво, що в наш час патерн став настільки відомим, що тепер люди називають «одинаками» навіть ті класи, які вирішують лише одну з проблем, перерахованих вище.

Рішення

Всі реалізації Одинака зводяться до того, аби приховати типовий конструктор та створити публічний статичний метод, який і контролюватиме життєвий цикл об’єкта-одинака.

Якщо у вас є доступ до класу одинака, отже, буде й доступ до цього статичного методу. З якої точки коду ви б його не викликали, він завжди віддаватиме один і той самий об’єкт.

Аналогія з життя

Уряд держави – вдалий приклад Одинака. У державі може бути тільки один офіційний уряд. Незалежно від того, хто конкретно засідає в уряді, він має глобальну точку доступу «Уряд країни N».

Структура

1. Одинак визначає статичний метод getInstance, який повертає один екземпляр свого класу.

Конструктор Одинака повинен бути прихований від клієнтів. Виклик методу getInstance повинен стати єдиним способом отримати об’єкт цього класу.

Застосування

Коли в програмі повинен бути єдиний екземпляр якого-небудь класу, доступний усім клієнтам (наприклад, спільний доступ до бази даних з різних частин програми).

Одинак приховує від клієнтів всі способи створення нового об’єкта, окрім спеціального методу. Цей метод або створює об’єкт, або віддає існуючий об’єкт, якщо він вже був створений.

Коли ви хочете мати більше контролю над глобальними змінними.

На відміну від глобальних змінних, Одинак гарантує, що жоден інший

код не замінить створений екземпляр класу, тому ви завжди впевнені в наявності лише одного об’єкта-одинака.

Тим не менше, будь-коли ви можете розширити це обмеження і дозволити будь-яку кількість об’єктів-одинаків, змінивши код в одному місці (метод getInstance).

Кроки реалізації

1. Додайте до класу приватне статичне поле, котре міститиме одиночний об’єкт.
2. Оголосіть статичний створюючий метод, що використовуватиметься для отримання Одинака.
3. Додайте «ліниву ініціалізацію» (створення об’єкта під час першого виклику методу) до створюючого методу одинака.
4. Зробіть конструктор класу приватним.
5. У клієнтському коді замініть прямі виклики конструктора одинака на виклики його створюючого методу.

Переваги та недоліки

- Гарантує наявність єдиного екземпляра класу.
- Надає глобальну точку доступу до нього.
- Реалізує відкладену ініціалізацію об’єкта-одинака.
- Порушує принцип єдиного обов’язку класу.
- Маскує поганий дизайн.
- Проблеми багатопоточності.

Відносини з іншими патернами

- Фасад можна зробити Одинаком, оскільки зазвичай потрібен тільки один об’єкт-фасад.
- Патерн Легковаговик може нагадувати Одинака, якщо для конкретного завдання ви змогли зменшити кількість об’єктів до одного.

ДОДАТОК Б ПУБЛІКАЦІЇ ЗА РЕЗУЛЬТАТАМИ ДОСЛІДЖЕННЯ

ЛАБОРАТОРНИЙ ПРАКТИКУМ «ПОРОДЖУВАЛЬНІ ПАТЕРНИ ПРОЄКТУВАННЯ»

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

Проєктувальник програмного забезпечення (ПЗ) – це спеціаліст, який відповідає за архітектуру і дизайн програмних продуктів, розробку і підтримку програмного коду, а також забезпечення його відповідності вимогам проєкту. У сучасному бізнес-середовищі проєктувальники ПЗ є ключовими фігурами, які допомагають компаніям автоматизувати процеси, впроваджувати нові технології і підвищувати ефективність діяльності. Результати аналізу вимог до кадрового забезпечення об'єкту ІТ-галузі вказують на високий попит на проєктувальників ПЗ. Професія стала ключовою у процесі цифрової трансформації бізнесу та державних установ.

Патерни проєктування представляють перевірені рішення для поширених проблем у розробці програмного забезпечення [1]. Володіючи ними, розробники можуть швидко застосовувати вже існуючі підходи замість того, щоб вигадувати рішення з нуля. Породжувальні патерни дозволяють забезпечувати гнучкість і підтримувати розширення програмного забезпечення шляхом виділення процесу створення об'єктів в окремі класи. Кожен з цих патернів вирішує певні проблеми проектування і може бути використаний залежно від конкретних потреб вашого проекту.

Виконана постановка трьох лабораторних робіт: проєктування предметної області з використанням патерна «Фабричний метод», патерна «Абстрактна фабрика» та патерна «Одинак». До кожної роботи виконана постановка завдання, вимоги до оформлення звіту, наведено приклади виконання роботи та оформлення звіту, що містить UML-діаграму ієрархії класів, описи класів та їх атрибутів, програмний код реалізації патерна, приклад виконання програми на мові C# у середовищі Visual Studio.

Опанування здобувачами вищої освіти запропонованого лабораторного практикуму забезпечить отримання програмних результатів навчання, які сприяють широкому діапазону їх професійної діяльності та високій конкурентоспроможності на ринку праці [2].

Список використаних джерел

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

2. Струк А.М. Формування здатності майбутніх інженерів-програмістів до проектування програмного забезпечення. Інформаційні технології в освіті, 2022. № 50 (1). – С. 39-58.