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 «Поведінкові патерни проєктування» у закладах вищої освіти»

Виконав(ла)

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

Олена КЛИМЕНКО _____
(ім'я, прізвище) (підпис)

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

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

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

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

Харків – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В. Н. Каразіна

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

ЗАТВЕРДЖУЮ

Завідувач кафедри
(підпис) Інна НЕФЬОДОВА (ім'я, прізвище)

«08» жовтня 2024 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЄКТ)

Клименко Олена Іванівна
(прізвище, ім'я, по батькові здобувача)

1. Тема роботи Професійна підготовка фахівців з цифрових технологій для викладання освітнього модулю «Поведінкові патерни проєктування» у закладах вищої освіти
керівник роботи Чикунов Павло Олександрович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «08» жовтня 2024 року № 5101-5/3232

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

3. Перелік питань, які потрібно розробити: Актуальність професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Поведінкові 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 «Поведінкові 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 «Поведінкові 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 «Поведінкові 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) _____ Олена КЛИМЕНКО
(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 модулю «Поведінкові патерни проєктування» у закладах вищої освіти.

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

Предметом дослідження роботи є методика професійної підготовки фахівців з цифрових технологій до розробки комплексу цифрових освітніх ресурсів для викладання освітнього модулю «Поведінкові патерни проєктування» у закладах вищої освіти.

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

Виконано аналіз характеристик об'єктів галузі проєктування програмного забезпечення. Виконана постановка 2 нових лабораторних робіт. Виконано аналіз вимог до кадрового забезпечення об'єкту ІТ-галузі в умовах цифрової трансформації суспільства.

Розроблено дидактичний проект консультативного заняття з теми «Поведінкові патерни проєктування».

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.).

Робота складається із вступу, чотирьох розділів, висновків, бібліографічного списку з 30 джерел, два додатка, 5 таблиць, 20 рисунків.

ПРОФЕСІЙНА ПІДГОТОВКА, ЛАБОРАТОРНИЙ ПРАКТИКУМ, ПРИНЦИПИ SOLID, ПАТЕРНИ ПРОЄКТУВАННЯ, СТАН, СТРАТЕГІЯ, C#, КАДРОВЕ ЗАБЕЗПЕЧЕННЯ, МЕТОДИЧНА РОЗРОБКА

ABSTRACT

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

The object of the research is the process of professional training of specialists in digital technologies for teaching the educational module "Behavioral Design Patterns" in higher education institutions.

The subject of the research is the methodology of professional training for specialists in digital technologies for the development of a set of digital educational resources to teach the educational module "Behavioral Design Patterns" in higher education institutions.

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

An analysis of the characteristics of objects in the field of software design was carried out. Two new laboratory assignments were developed. An analysis of the requirements for staffing the IT sector in the conditions of the digital transformation of society was conducted.

A didactic project for a consultation session on the topic "Behavioral Design Patterns" was developed.

The main results of the research were presented in the form of a thesis publication at the VIII International Scientific and Practical Conference "Students and Youth – for the Future of the Country" (Bakhmut-Kharkiv, November 14-15, 2024).

The paper consists of an introduction, four chapters, conclusions, a bibliography of 30 sources, two appendices, five tables, and 20 figures.

PROFESSIONAL TRAINING, LABORATORY PRACTICUM, SOLID PRINCIPLES, DESIGN PATTERNS, STATE, STRATEGY, C#, STAFFING, METHODOLOGICAL DEVELOPMENT

ЗМІСТ

ВСТУП

Розширення міжнародних контактів у сфері господарської діяльності, спільні підприємства та виробництво, інтегровані в економіку різних національних держав, дав багатьом фахівцям завдання вивчення цифрових технологій як інструменту професійної діяльності, що в сучасних умовах є важливим показником їх рівня компетентності, і в той же час вагомим чинником професійної та соціальної кар'єри.

Прагнення України увійти у світовий економічний простір надає цьому аспекту теоретичного переосмислення багатьох питань, що пов’язано з професійною підготовкою майбутніх інженерів-педагогів з цифрових технологій.

Компетентності майбутніх інженерів-педагогів необхідно вдосконалювати у зв'язку зі змінами змісту дисциплін, що викликано інтеграційними економічними процесами і підвищеною увагою до використання досягнень світового досвіду у виробничій діяльності. Відповіддю педагогічної практики на зростаючий попит на фахівців з цифрових технологій стала поява різних систем підготовки фахівців з комп’ютерних технологій у своїй галузі. Необхідність зміни до професійної підготовки обумовлена тим, що інженери-педагоги з базовою спеціальною підготовкою часто не компетентні в роботі з програмним забезпеченням. Дотримуючись традиційної логіки, інженер-педагог має стати фахівцем у всіх сферах, де використовується його робота. Неефективність такого підходу стає все більш очевидною. Більш організаційним і економічним є забезпечення якісного навчання фахівців у сфері професійної діяльності, як опосередковано, так і шляхом виконання функцій викладача спеціальних дисциплін, а саме з цифрових технологій.

На сьогоднішній день у цій сфері професійної підготовки склалася ситуація, коли подальшому зростанню якості освіти перешкоджає відсутність єдиних підходів до професійної підготовки фахівців з цифрових технологій;

відсутність наукових досліджень, пов’язаних з професійною підготовкою фахівців з урахуванням соціально-психологічної та психологічної спеціалізації, педагогічних засад у їх зв’язку зі специфікою сучасного соціокультурного та науково-технічного контексту, з новими цінностями спільноти та освіти.

У сучасному контексті професійна підготовка стає основою для широкоформатного навчання в контексті безперервного навчання, що вимагає нових підходів до розробки системи підготовки тренерів з цифрових технологій у професійній сфері. Відсутність наукових знань про специфіку підготовки фахівців означає, що неможливо реалізувати ідею непорушного особистісного і професійного розвитку у вигляді дидактичних принципів і прийомів.

Проблема суб’єктогенезу в контексті підготовки фахівців з цифрових технологій для професійно-технічних та закладів вищої освіти залишається недостатньо розвиненою. По-перше, через свою багатовимірність вона вимагає всебічного вивчення, в якому відправною точкою є розгляд змісту освіти як моделі соціального порядку, що відображає конкретний досвід реалізації зв'язку між знаннями і практикою. По-друге, аналіз актуальної педагогічної практики закладів освіти для здійснення професійної підготовки фахівців свідчить про необхідність виявлення її ролі в актуалізації індивідуальних ресурсів особистісно-професійного зростання фахівця, його предметоутворення, механізмів мотивації, орієнтації на трансформацію його освітньої діяльності та самого себе в ній.

Становлення і розвиток цифрового середовища є вкрай необхідним.

Отже, Україні потрібні компетентні фахівці з цифрових технологій для здійснення професійної діяльності. Отже, ми зосереджуємося на фахівцях зі спеціальності 015.39 Професійна освіта (Цифрові технології).

Науковці, такі як В. Хоменко, М. Лазарєв досліджували вплив цифрових технологій на педагогічний процес.

Проблеми професійної компетентності фахівців розглянуто у працях

Н.Брюханової, О.Коваленко, В.Кулешової, В.Мальованої, В.Ягупова та ін.

Дослідження літератури засвідчило, що проблема професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Поведінкові патерни проєктування») у закладах вищої освіти, а також вирішення низки суперечностей, а саме між:

– вимоги до фахівця на ринку праці обумовлені входженням України у світовий економічний простір, об'єктивною потребою в підготовці фахівців з цифрових технологій й урахуванням вікових, соціальних і професійних особливостей, а також відсутністю розробки наукових основ здійснення такої підготовки;

– орієнтація традиційної системи навчання на репродуктивні методи та необхідність розробки шляхів професійного розвитку фахівця, що забезпечується реалізацією предметного підходу;

– сформоване уявлення про функціональний характер інженерно-педагогічної підготовки фахівців з цифрових технологій у професійній сфері та недостатній розвиток педагогічних засобів, умов, психологічних механізмів необхідних змін фахівця;

– важливість вирішення зазначених суперечностей і зумовили вибір теми дослідження.

Отже, актуальність, об’єктивна потреба, недостатня розробленість проблеми зумовили вибір теми дослідження: «Професійна підготовка фахівців з цифрових технологій для викладання освітнього модуля «Поведінкові патерни проєктування» у закладах вищої освіти».

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

Завдання дослідження:

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

2. Виконати аналіз характеристик об'єктів галузі, зокрема проаналізувати стан і стратегії розвитку.

3. Дослідити вимоги до кадрового забезпечення об'єкту галузі

4. Теоретично обґрунтувати, розробити перевірити методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Поведінкові патерни проєктування» у закладах вищої освіти.

Об'єкт дослідження: процес професійної підготовки здобувачів освіти.

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

Методи дослідження:

– загальнонаукові (аналіз, синтез, систематизація) з метою виявлення основних напрямів професійної підготовки фахівців з цифрових технологій.

– емпіричні (опитування);

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

Наукова новизна одержаних результатів дослідження:

вперше:

– теоретично обґрунтовано та експериментально перевірено методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Поведінкові патерни проєктування» у закладах вищої освіти засобами інноваційних технік; обґрунтовано специфічну складову змісту освіти, що орієнтовано на знання інтерес до індивідуалізованих методів набуття знань, умінь і навичок і методів саморегуляції пізнавальної діяльності, що дозволяє актуалізувати здобувача освіти як суб’єкта освітньої діяльності і, у сукупності, що надає здобувачеві освіти можливість управляти власною навчальною діяльністю;

– подальшого розвитку набули зміст професійної підготовки інженера-педагога з цифрових технологій.

Теоретичне та практичне значення одержаних результатів полягає в обґрунтуванні методики професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Поведінкові патерни проєктування» у закладах вищої освіти засобами інноваційних технік; обґрунтовано специфічну складову змісту освіти.

Матеріали дослідження використовувались при підготовці здобувачів вищої освіти Бахмутського Навчально-наукового професійно-педагогічного інституту Харківського національного університету імені В.Н.Каразіна зі спеціальності 015.39 Професійна освіта (Цифрові технології); удосконалено зміст дисципліни «Методика професійного навчання»).

Матеріали дослідження використовувались при підготовці здобувачів вищої освіти Навчально-наукового професійно-педагогічного інституту УІПА (м. Бахмут) зі спеціальності 015 Професійна освіта (Цифрові технології).

Апробація результатів дослідження: 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 ПРОФЕСІЙНА ПІДГОТОВКИ ФАХІВЦІВ З ЦИФРОВИХ ТЕХНОЛОГІЙ

Необхідність багатопрофільної та професійної підготовки фахівців обумовлена процесами інтеграції України у світовий економічний та освітній простір. Це зумовлює необхідність отримання фахівцями таких знань і навичок, які можуть передаватися з однієї сфери діяльності в іншу і виступати інтелектуальною основою і ресурсом для особистісного і професійного розвитку за умови інновації фахівця.

Значні перспективи впровадження концепції безперервної освіти відкриваються в процесі комплексного вивчення людини як суб'єкта діяльності і, перш за все, мотивації до безперервного професійного зростання та саморозвитку. При цьому як найважливіша умова ефективності безперервної професійної освіти розглядається розвиток творчості як механізму перетворення суб'єктом власної діяльності. Суб'єктність розглядається в сучасній вітчизняній педагогіці та психології як найважливіша характеристика людини, що відкриває їй можливості для особистісної та професійної самореалізації в динамічному соціальному, інформаційному та культурному середовищі [30, с. 12].

Аналіз досліджень з професійної підготовки, зміст мотивів навчальної діяльності здобувачів освіти принципово відрізняється від мотивів навчальної діяльності у попередні вікові періоди. Ця відмінність полягає не тільки в тому, що одержувані знання, вміння та навички безпосередньо «тут і тепер» можуть бути застосовані на практиці, але, перш за все, в тому, що навчання тут служить чинником і механізмом перетворення професійної діяльності. Професіонал постає як суб'єкт перетворення власної професійної діяльності. Переживання індивідом повноти своєї суб'єктності виступає найважливішим мотивом здобувача освіти. У цьому сенсі мотиви безперервної освіти може бути інтерпретовані як особливі новоутворення дорослості. Питання це, проте, досліджений у вітчизняній педагогіці явно недостатньо [29, с. 123].

Однією з проблем, що істотно впливають на становлення системи підготовки фахівців, є проблема професіоналізації. У широкому сенсі, типовому для психології, професіоналізація розглядається як аспект соціалізації на трудовій стадії життя індивіда [29, с. 12].

У вузькому значенні, характерному переважно педагогічних досліджень, професіоналізація – це засвоєння індивідом певного кола знань, умінь і навиків, притаманних конкретної професії [26, с. 74].

Актуальність проблеми професіоналізації пов'язана з тим, що сам собою цей процес, як показано в дослідженнях[25, с. 124], виявляється вкрай суперечливим. Одним з найбільш істотних суперечностей професіоналізації є одночасний позитивний і негативний вплив на суб'єкта: формування стійких позитивних мотивів, соціально значущих і професійно важливих якостей, готовності до професійного зростання і в той же час – стандартизація особистості, нівелювання індивідуальності та підпорядкування професійним зразкам поведінки; професійні деформації різного рівня глибини. Діалектика позитивних і негативних аспектів професіоналізації являє собою самостійну і поки неповно розроблену проблему в контексті підготовки фахівців з цифрових технологій. Негативні аспекти професіоналізації можуть суттєво впливати на мотивацію освітньої діяльності майбутніх інженерів-педагогів з цифрових технологій. Отже, спроєктовані системи підготовки повинні враховувати такі впливи та знижувати їх інтенсивність.

Важлива проблема, пов'язана з організацією підготовки фахівців з цифрових технологій, полягає у пошуку способів управління діяльністю фахівців. Дослідження у цій галузі виходять із те, що визначальну бік розвитку індивіда у процесі навчання становить ускладнення знань і способів діяльності. Більше того, соціально організована та стимульована діяльність є основною, засобом та умовою професійного розвитку [24, с. 129]. Однак здобувач освіти, є активною дійовою особистістю: її внутрішня активність, інтереси, воля, здібності, установки є внутрішніми чинниками, що опосередковують всю систему педагогічного впливу. Отже, будь-який вплив,

а тим більше спеціально організоване навчання, може призвести до бажаного результату, тільки якщо цей вплив переломлюється через внутрішні чинники.

Незважаючи на те, що ідея управління навчальною діяльністю на основі психологічних особливостей висловлюється вже давно, її реалізація явно відстає від потреб практики. Дослідження показують, що організація змісту освіти з урахуванням типологічних особливостей учнів (функціональна асиметрія мозку, когнітивні стилі, тип сприйняття, зберігання та переробки інформації та ін.) дає істотне підвищення ефективності навчання [26, с. 173].

Вважаємо, що організація навчального процесу, що спирається на активізацію рефлексії, усвідомлення переживань і сприяння у розвитку прагнень, має бути в основі навально-виховного процесу у ЗВО. Неважко помітити, що кожен з механізмів суб'єктогенезу робить свій внесок у підвищення ефективності професійної підготовки фахівців з цифрових технологій, виступаючи як механізм вирішення тієї чи іншої суперечності за рахунок перетворення його боків: активізація рефлексії стилю і способів навчальної діяльності, як психологічний механізм саморозвитку суб'єкта вивчення дисциплін комп’ютерного профіля (спеціальні дисципліни), як механізм подолання стереотипів навчальної та предметної діяльності, спілкування та самосприйняття; як психологічний механізм подолання особистісних обмежень, що знижують активність індивіда в освітньому просторі; від фахівців прагнень до дослідження власних можливостей як механізм безперервного особистісного та професійного саморозвитку.

Таким чином, основу професійної підготовки фахівців з цифрових технологій складають психологічні механізми активізації внутрішніх сил суб'єктів освітнього процесу.

Аналіз концептуальних засад програмованого та модульного навчання, концепцій професійного становлення особистості та андрогогіки дозволив нам розробити концепцію системи професійної підготовки фахівців з цифрових технологій.

До педагогічної науки і практики міцно увійшли такі поняття як «суб'єкт-суб'єктна взаємодія», «суб'єкт навчальної діяльності», «суб'єкт самовиховання» та ін. Дослідження педагогів дозволяють виявити суб'єктний підхід до підготовки фахівців [30, с. 47]. Перш за все, необхідно окреслити коло чинників, що впливають на становлення суб'єктності: усвідомлення себе суб'єктом, що самостійно вирішує завдання власної освіти; усвідомлення значущості своєї інтелектуальної праці іншим людям; здатність знаходити, перетворювати і використовувати інформацію для досягнення власних цілей; потреба в рефлексії як усвідомленої умови регулювання своєї поведінки.

Сутнісною ознакою суб'єкта, є активність, яка характеризується як інтегративна, тобто передбачає активну позицію людини у всіх проявах, починаючи від усвідомленого цілепокладання, діалектичного оперування, конструктивного коригування способів діяльності у всіх ситуаціях [30, с. 167]. Активність суб'єкта забезпечує нарощування можливості діяти за межами власної обмеженості. Наявність фіксованих форм поведінки вирішує обмеженість індивіда їх рамками. Наприклад, стратегії навчальної діяльності, що сформувалися на більш ранніх етапах вікового розвитку, можуть виявитися неефективними в умовах професійної підготовки, обмежуючи можливості засвоєння навчального матеріалу. Розвиток прагнень, усвідомлення переживань, що супроводжують навчання, рефлексія навчальної діяльності призводять до нарощування суб'єктності, тобто підвищують можливості виходу суб'єктом за рамки власної обмеженості. Подолаючи сформовану на більш ранніх етапах онтогенезу обмеженість, суб'єкт знаходить здатність до особистісного та професійного саморозвитку, результуючим ефектом якого є багатовимірне психологічне новоутворення – авторство в особистій і професійній долі. Авторство свого життя означає вільне і відповідальне дію суб'єкта у бік актуалізації свого потенціалу. Феномен авторства в особистій і професійній долі розкривається як самоактуалізація в динамічному мінливому соціальному і технічному середовищі, переживання професійної діяльності як особистісно значущої активності, й спонукається тільки мотивами

саморозвитку наднормативна активність і повна відповідальність за її результати і наслідки.

Рівень активності фахівців, які проходять професійну підготовку з цифрових технологій, залежить від ступеня взаємозв'язку навчального матеріалу та змісту професійної діяльності, а також визначається можливістю інтеграції навчальної інформації в особистий та професійний досвід індивіда.

Зміст професійної підготовки фахівців з цифрових технологій, перш за все, передбачає розробку організаційно-методичного забезпечення процесу підготовки і включає в себе такі складові:

– опис предметного аспекту змісту професійної підготовки (тобто того, чого вчити);

– опис операційного аспекту змісту професійної підготовки майбутніх інженерів-педагогів з цифрових технологій (систему методів роботи з учнями);

– опис форм реалізації методів, передбачених змістом підготовки;

– послідовність реалізації навчальних курсів;

– критерії розвитку пізнавальної активності та критерії ефективності професійної підготовки;

– комплекс діагностичних засобів, спрямованих на оцінку ефективності.

Розробка змісту професійної підготовки майбутніх інженерів-педагогів з цифрових технологій на основі підвищення пізнавальної активності здобувачів освіти спирається, з одного боку, на аналіз теоретико-методологічних аспектів проблеми, а з іншого боку, на співвідношення пізнавальної активності та специфічних умов професійної підготовки здобувачів освіти. Крім цього, обов'язково враховується: суб'єкт-суб'єктний характер взаємодії учасників освітнього процесу; орієнтованість на зміст професійної діяльності; актуалізація позитивних емоційних станів у процесі навчання; можливість реалізації навчальної діяльності у різних формах; відтворення у процесі взаємодії форм поведінки, специфічних для конкретної професійної групи; наявність багаторівневого зворотного зв'язку,

найважливішими каналами в якій є міжособистісне спілкування та педагогічна діагностика; діалогічний характер спілкування; проблемно-орієнтоване подання інформації.

Опанування фахівцями іншомовними комунікативними вміннями та навичками передбачає певну організацію їхньої навчальної діяльності. Мова йде про те, щоб викладач міг вільно інтегрувати у свою діяльність різні форми та методи навчання (використовувати індивідуальні та групові, аудиторні та позааудиторні, алгоритмізовані та інші методи роботи зі майбутніми фахівцями з цифрових технологій). При цьому незмінність завдання та загальної позиції, спрямованої на розвиток здобувача освіти як суб'єкта навчальної діяльності, зберігається.

Важливою умовою ефективності професійної підготовки фахівців з цифрових технологій є постійний педагогічний моніторинг, що вимагає:

– визначення критеріїв розвитку активності та критеріїв ефективності професійної підготовки;

– розробки комплексу діагностичних засобів, спрямованих на оцінку розвиваючих ефектів. Моніторинг повинен включати два аспекти: об'єктивні дані, одержувані за допомогою методів експертної оцінки, і суб'єктивні дані, одержувані у вигляді звітів учасників експерименту, самооцінки їх діяльності та ін.).

Ступінь ефективності реалізації системи підготовки фахівців визначається досягненням запланованих результатів, відповідних до освітньо-професійної програми для здобувачів освіти 015.39 Професійна освіта (Цифрові технололгії). Результативний компонент розкривається в особливостях досягнутих результатів на основі виявлених або спеціально розроблених показників. Результативність відображає ті зміни, які відбуваються у тих, хто навчається як суб'єктів навчальної діяльності, і ті зміни, які відбуваються у викладачів як суб'єктів освітньої діяльності.

Отже, результативний компонент запропонованої нами системи фахівців з цифрових технологій, повинен відображати зміни, що відбуваються

за багатьма параметрами, а саме: здобувачі освіти отримують можливість удосконалювати свою навчальну діяльність, зростають їх інтелектуальні та комунікативні можливості, у них формуються та вдосконалюються комунікативні вміння та навички; викладачі освоюють додаткові методи діагностики індивідуальних особливостей здобувачів, розширюють свій арсенал педагогічних методів та прийомів роботи. Зміст освіти також змінюється, оскільки до нього виявляються включеними нові методи (методи перетворення навчальної діяльності).

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

2.1 Огляд принципів об'єктно-орієнтованого проєктування класів SOLID

SOLID – це набір принципів, що стосується об'єктно-орієнтованого програмування, сформульований Робертом Мартіном у 1995 році. Його основна ідея полягає в зменшенні залежностей між компонентами коду, оскільки велика кількість залежностей ускладнює підтримку програмного забезпечення, що часто називають «спагеті-кодом». Основні проблеми такого підходу включають:

– жорсткість (Rigidity): зміна одного компонента призводить до необхідності змін в інших частинах коду;

– крихкість (Fragility): внесення змін в одну частину програми може викликати помилки в інших;

– нерухомість (Immobility): труднощі з повторним використанням коду поза поточним контекстом.

1. Принцип єдиної відповідальності (Single Responsibility Principle – SRP): не повинно існувати більше однієї причини для зміни даного класу.

Цей принцип говорить, що клас повинен мати лише одну причину для змін. Клас має виконувати лише одну задачу, і вся логіка, пов'язана з цією задачею, повинна бути зосереджена в одному місці. Поєднання різних завдань в одному класі вважається помилковим підходом у проектуванні.

Приклад 1. Порушення принципу єдиної відповідальності.

```
class Calculator
{
    public void Add(int x, int y)
    {
        Console.Write(x + y);
    }
    public void Sub(int x, int y)
    {
        Console.Write(x - y);
    }
}
```

Найпростіший спосіб вирішення проблеми – розділити клас Calculator на два класи.

Приклад 2. Дотримання принципу єдиної відповідальності:

```
class Calculator
{
    public int Add(int x, int y)
    {
        return x + y;
    }
    public int Sub(int x, int y)
    {
        return x - y;
    }
    class Printer
    {
        public static void Print(int value)
        {
            Console.WriteLine(value);
        }
    }
}
```

2. Принцип відкриття-закриття (Open-Closed Principle – OCP): Програмні сутності, такі як класи або модулі, повинні бути відкритими для розширення, але закритими для модифікацій. Це означає, що базова структура коду не повинна змінюватися при зміні вимог, а повинна розширюватися через додавання нового функціоналу.

Якщо необхідно розширити функціональність для обробки додаткових типів фігур, код не повинен потребувати зміни, а тільки доповнення новими класами (рис. 2.1).

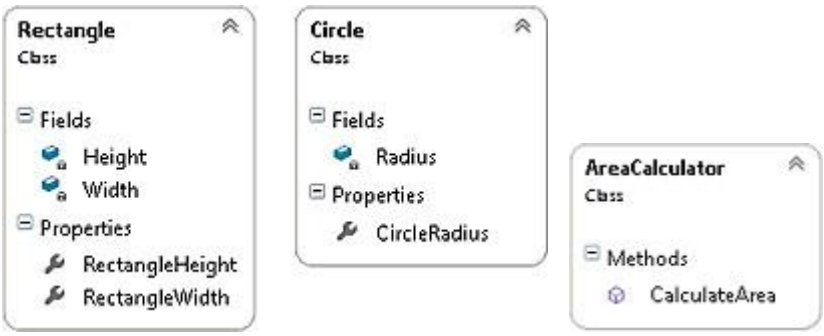

Рисунок 2.1 – Порушення принципу відкриття-закриття

Приклад 3. У цьому прикладі необхідно обчислювати сумарну площу множини прямокутників (Rectangle) та кіл (Circle), використовуючи метод CalculateArea класу AreaCalculator. Його вихідний код мовою C# може бути таким:

```
public double CalculateArea2(object[] shapes)
{
    double area = 0;
    foreach (object shape in shapes)
    {
        if (shape is Rectangle)
        {
            Rectangle rectangle = (Rectangle)shape;
            area += rectangle.RectangleWidth * rectangle.RectangleHeight;
        }
        else
        {
            Circle circle = (Circle)shape;
            area += circle.CircleRadius * circle.CircleRadius * Math.PI;
        }
    }
    return area;
}
```

У цьому коді перед обчисленням площі геометричної фігури необхідно перевіряти її тип (прямокутник чи ні). Якщо вимоги до програми зміняться так, щоб можна було додати ще декілька конкретних фігур та обчислювати їхню площу, код методу прийдеться також змінити. Чим більше типів геометричних фігур буде використовуватися, тим більше потрібно змін. Принцип відкриття-закриття дозволяє зробити, щоб код методу CalculateArea не залежав від цього (рис. 2.2).

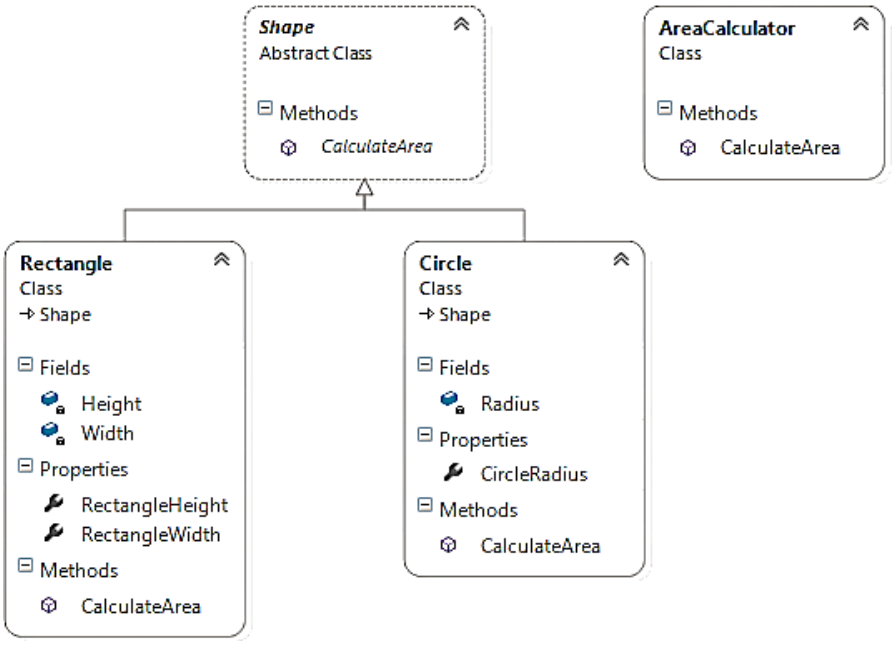

Рисунок 2.2 – Дотримання принципу відкриття-закриття

Тут у кожному класі, що описує деяку геометричну фігуру, є певна реалізація методу CalculateArea, яка перевизначає абстрактний метод

CalculateArea абстрактного класу Shape. Тоді маємо таку реалізацію цього методу в класі AreaCalculator:

```
public double CalculateArea(Shape[] shapes)
{
    double area = 0;
    foreach (var shape in shapes)
    {
        area += shape.CalculateArea();
    }
    return area;
}
```

3. Принцип заміщення Барбари Лісков (Liskov Substitution Principle – LSP): методи, які використовують об'єкти базових класів, повинні мати можливість працювати з об'єктами похідних класів без порушення логіки програми. Барбара Лісков – американська дослідниця в галузі інформатики, лауреатка премії Тюрінга 2008 року.

Розглянемо спочатку діаграму класів на рис. 2.3.

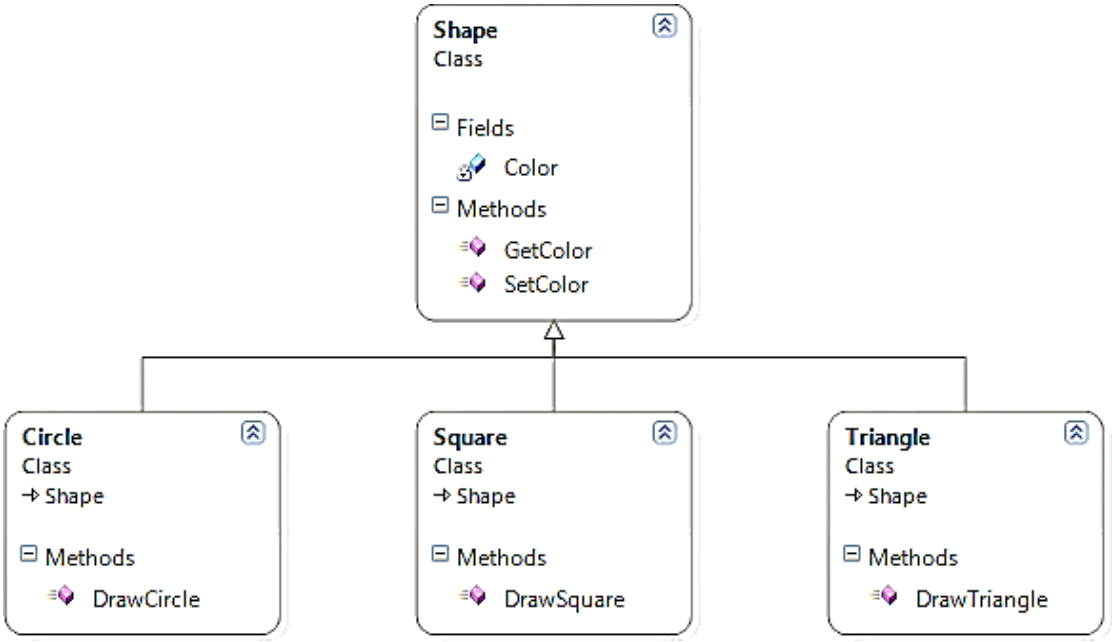

Рисунок 2.3 – Порушення принципу заміщення Барбари Лісков

Вона відбиває структуру успадкування класів, необхідних для розроблення програми, що "малює" геометричні фігури різних типів: кола (Circle), квадрати (Square) та трикутники (Triangle). Кожний з цих класів має свій метод "малювання": DrawCircle, DrawSquare та DrawTriangle.

Вихідний код метод Main такої програми має вигляд:

```
static void Main(string[] args)
{
    Shape[] ShapeList = new Shape[3];
    Circle C = new Circle();
    C.SetColor("Red");
    Square S = new Square();
    S.SetColor("Green");
    Triangle T = new Triangle();
    T.SetColor("Yellow");
    ShapeList[0] = C;
    ShapeList[1] = S;
    ShapeList[2] = T;
    foreach (Shape s in ShapeList)
    {
        if (s is Square) ((Square)s).DrawSquare();
        else if (s is Circle) ((Circle)s).DrawCircle();
        else if (s is Triangle) ((Triangle)s).DrawTriangle();
    }
}
```

У цьому прикладі є численні явні приведення типів, через це зі збільшенням кількості типів фігур код ускладнюється. Кожний клас, що описує конкретну геометричну фігуру, має певну реалізацію методу "малювання" draw, яка перевизначає абстрактний метод draw абстрактного класу Shape.

Для подолання цієї проблеми треба змінити попередню діаграму класів (рис. 2.4). У методі Main використовуються поліморфні виклики цього методу:

```
static void Main(string[] args)
{
    Shape[] ShapeList = new Shape[3];
    Circle C = new Circle();
    C.SetColor("Red");
    Square S = new Square();
    S.SetColor("Green");
    Triangle T = new Triangle();
    T.SetColor("Yellow");
    ShapeList[0] = C;
    ShapeList[1] = S;
    ShapeList[2] = T;
    foreach (Shape s in ShapeList)
    {
        s.draw();
    }
}
```

4. Принцип ізоляції інтерфейсу (Interface Segregation Principle – ISP): Клієнти не повинні бути змушені залежати від методів, які вони не використовують. Інтерфейси мають бути вузькоспеціалізованими, щоб уникати включення надмірної функціональності.

Приклад порушення принципу показаний на рис. 2.5. Інтерфейс Animal має три абстрактних варіанти поведінки: гавкати (Bark), літати (Fly) та бігати (Run). Тому класи-нащадки Bird (птиця), Dog (собака), Cat (кішка) мають

реалізувати всі методи цього інтерфейсу. Але не всі ці варіанти поведінки притаманні зазначеним тваринам. Інтерфейс повинен бути розділений на декілька менших інтерфейсів, щоб кожен клас реалізовував лише ті методи, які потрібні.

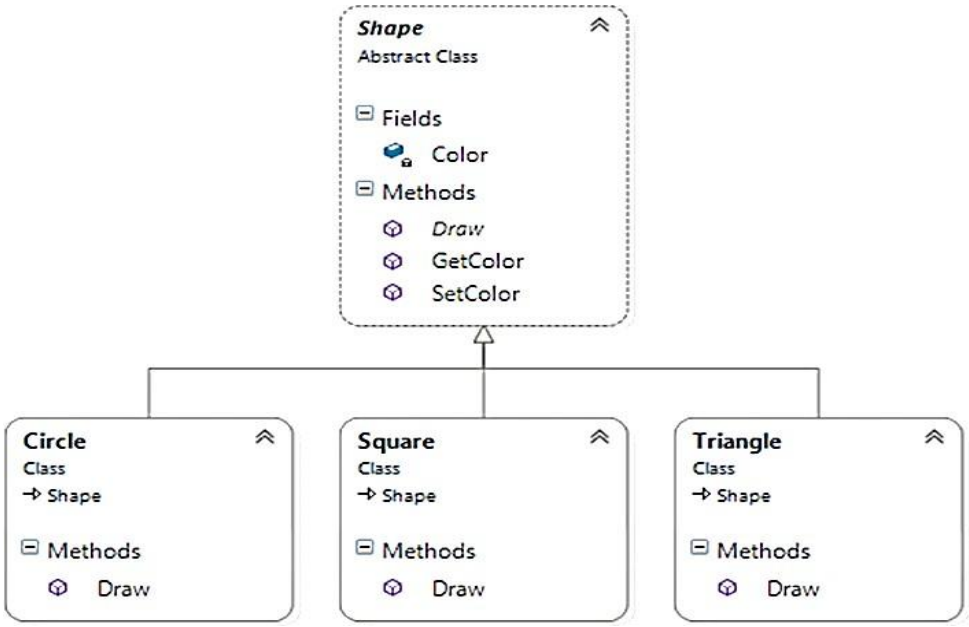

Рисунок 2.4 – Дотримання принципу заміщення Барбари Лісков

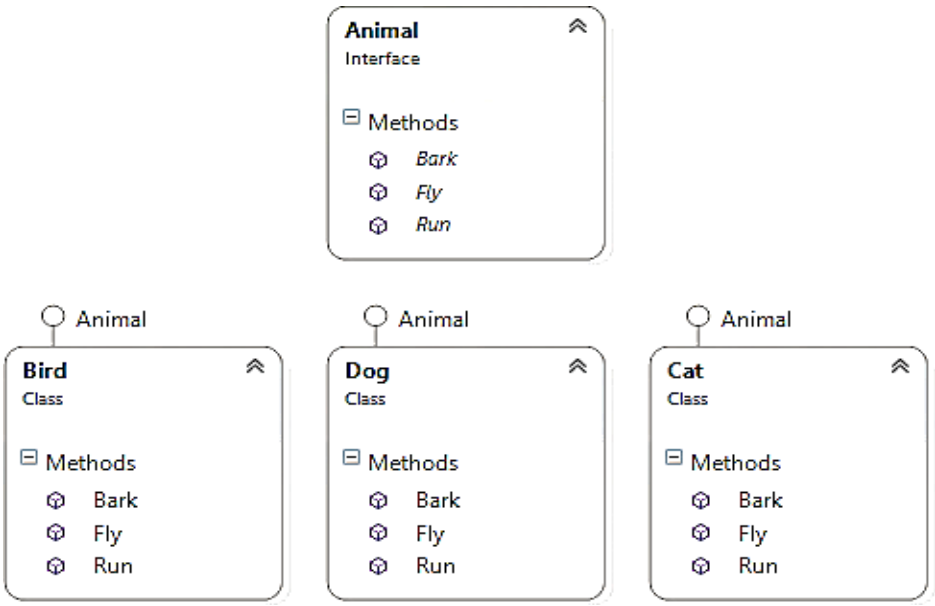

Рисунок 2.5 – Порушення принципу ізоляції інтерфейсу

Для подолання проблеми необхідно замість інтерфейсу Animal використати три інтерфейси (Barkable, Flyable та Runnable), в кожному з яких визначено тільки один варіант поведінки тварини (рис. 2.6).

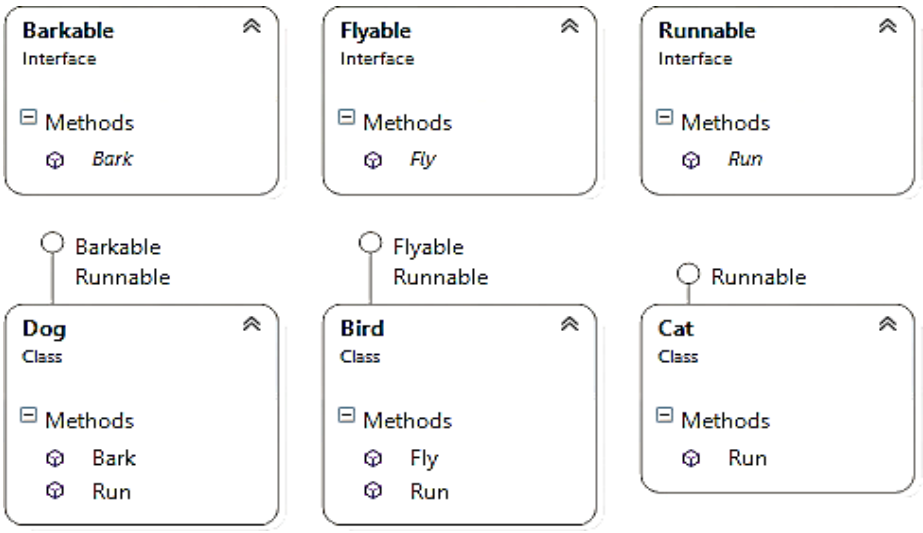

Рисунок 2.6 – Дотримання принципу ізоляції інтерфейсу

5. Принцип інверсії залежностей (Dependency Inversion Principle – DIP): модулі верхнього рівня не повинні залежати від модулів нижнього рівня. Уявімо собі програму, яка копіює введені символи з клавіатури на принтер. Якщо реалізація класу Copy залежить безпосередньо від конкретних класів KeyboardReader і PrinterWriter, то зі зміною вимог до вводу/виводу доведеться змінювати весь клас.

Всі модулі повинні бути прив'язані до абстракцій, а не до конкретних реалізацій, що робить систему гнучкішою для змін та розширень.

Приклад 4. Порушення принципу інверсії залежностей.

```
class Copy
{
    private KeyboardReader reader;
    private PrinterWriter writer;
    public Copy()
    {
        this.reader = new KeyboardReader();
        this.writer = new PrinterWriter();
    }
    public void DoWork()
    {
        writer.Write(this.reader.Read());
    }
}
```

Програма копіює символи, що вводяться з клавіатури, на принтер. Клас Copy безпосередньо використовує класи KeyboardReader і PrinterWriter, тому зі зміною вимог до вводу – виводу клас Copy доведеться змінити. Усі модулі повинні залежати від абстракцій.

Приклад 5. Дотримання принципу інверсії залежностей.

```
class Copy
{
    private IReader reader;
    private IWriter writer;
    public Copy(IReader reader, IWriter writer)
    {
        this.reader = reader);
        this.writer = reader;
    }
    public void DoWork()
    {
        writer.Write(reader.Read());
    }
}
```

2.2 Огляд поведінкових патернів проєктування «Стан» та «Стратегія»

2.2.1 Огляд поведінкового патерна проєктування «Стан»

Патерн «Стан» є поведінковим патерном проєктування, що дозволяє об’єктам змінювати свою поведінку в залежності від їхнього поточного стану. Зовнішньо це створює враження, що змінився клас об’єкта. Патерн «Стан» неможливо розглядати окремо від концепції машини станів, також відомої як стейт-машина або скінченний автомат.

Основна ідея полягає в тому, що програма може перебувати в одному з кількох станів, які постійно змінюються один за одним. Набір цих станів, а також переходів між ними, визначений наперед і є скінченним. Перебуваючи в різних станах, програма може по-різному реагувати на ті ж самі події, що з нею відбуваються.

Цей підхід можна застосувати й до окремих об’єктів. Наприклад, об’єкт «Документ» може перебувати в трьох станах: «Чернетка», «Модерація» або «Опублікований». У кожному з цих станів метод «опублікувати» працюватиме по-різному (рис. 2.7):

– у стані «Чернетка» – надсилає документ на модерацію;

– у стані «Модерація» – відправляє документ у публікацію, але лише за умови, що це зробив адміністратор;

– у стані «Опублікований» метод не виконує жодних дій.

Як приклад з реального життя можна навести ситуацію зі смартфоном, який змінює свою поведінку залежно від поточного стану:

– якщо телефон розблоковано, натискання кнопок виконує певні дії;

– якщо телефон заблоковано, натискання кнопок викликає екран розблокування;

– якщо телефон розряджений, натискання кнопок викликає екран зарядки.

Рисунок 2.7 – Можливі стани документу та переходи між ними

На рис. 2.8 зображено типову структуру ієрархії класів відповідно до вимог патерну «Стан».

1. «Контекст» зберігає посилання на об’єкт стану та делегує йому частину функцій, що залежать від стану. Він працює з цим об’єктом через загальний інтерфейс станів. Контекст повинен мати метод для присвоєння нового об’єкта-стану.

2. «Стан» описує загальний інтерфейс для всіх конкретних станів.

3. «Конкретні стани» реалізують поведінку, пов’язану з певним станом контексту. Іноді доводиться створювати ієрархії класів станів, щоб уникнути дублювання коду. Стан може мати зворотне посилання на об’єкт контексту, через який зручно отримувати інформацію та змінювати стан.

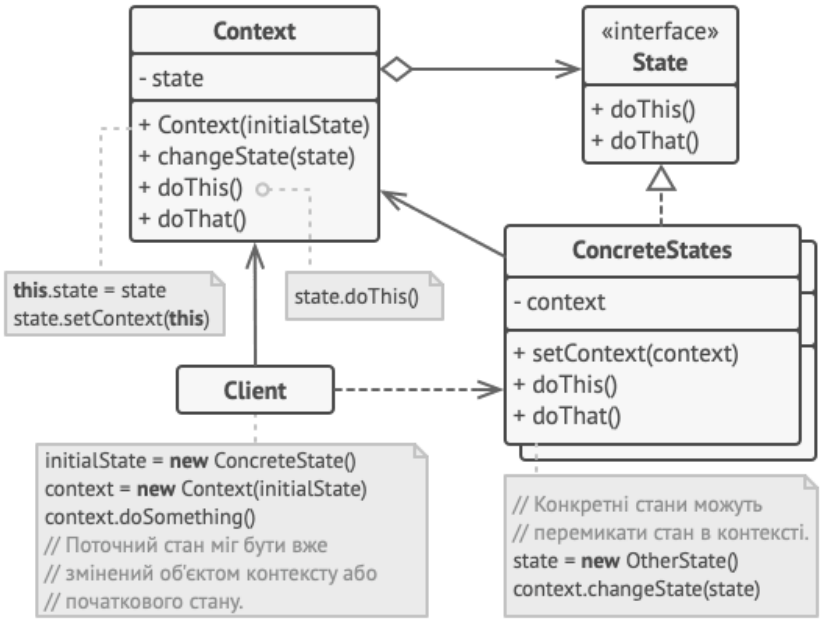

Рисунок 2.8 – Ієрархія класів, що відповідає патерну «Стан»

Використання патерну «Стан» доцільне, якщо об’єкт змінює свою поведінку залежно від внутрішнього стану, коли кількість станів значна, а їхній код часто змінюється. Патерн рекомендує виділити в окремі класи всі поля та методи, пов’язані з визначеним станом. Початковий об’єкт завжди посилатиметься на один з об’єктів-станів, делегуючи йому частину своєї функціональності. Для зміни стану достатньо підставити інший об’єкт-стан у контекст.

Для вдалого рефакторингу коду відповідно до патерну «Стан» необхідно виконати такі кроки.

1. Визначте клас, що гратиме роль контексту. Це може бути або існуючий клас з залежностями від стану, або новий клас, якщо логіка станів розподілена по кількох класах.

2. Створіть загальний інтерфейс станів, що описуватиме методи для всіх станів, що існують у контексті.

3. Для кожного конкретного стану реалізуйте клас, що імплементує інтерфейс стану. Перенесіть код, що відповідає за кожен конкретний стан, до відповідних класів.

4. Створіть у контексті поле для зберігання об’єктів-станів і метод для зміни стану.

5. Замініть старі методи контексту, що залежали від стану, на виклики відповідних методів об’єкта-стану.

Основні переваги патерну «Стан» такі:

– зменшує кількість умовних операторів у машині станів;

– концентрує в одному місці код, пов’язаний з певним станом;

– спрощує код контексту.

Недоліки патерну «Стан»: може ускладнити код, якщо кількість станів невелика і вони рідко змінюються.

2.2.2 Огляд поведінкового патерна проєктування «Стратегія»

Стратегія – це шаблон проєктування, що визначає групу подібних алгоритмів і розміщує кожен із них у окремому класі. Це дозволяє змінювати алгоритми під час роботи програми, замінюючи один на інший.

Уявіть, що потрібно створити програму-навігатор для подорожей. Вона повинна мати зручну карту для орієнтації в незнайомому місті. Однією з ключових функцій є пошук і побудова маршрутів. Користувач задає початкову точку і пункт призначення, а навігатор прокладає оптимальний шлях.

Перший варіант навігатора підтримував лише автомобільні маршрути, що було зручно для автомобілістів. Але не всі мандрують на авто, тому додали можливість прокладати піші маршрути.

Згодом виявилося, що частина туристів використовує громадський транспорт, тому додали і цю опцію. У майбутньому планується додати маршрути для велосипедів і маршрути до визначних місць.

Попри популярність навігатора, технічні труднощі збільшувалися. З кожним новим алгоритмом код головного класу зростав, що ускладнювало його розуміння і підтримку. Будь-які зміни або виправлення алгоритмів зачіпали основний клас, що збільшувало ризик помилок.

Командна робота також ускладнювалася: зміни в коді часто конфліктували, вимагаючи додаткового часу на їхнє вирішення.

У цьому прикладі кожен алгоритм пошуку маршруту буде перенесений у свій окремий клас. У кожному з цих класів буде реалізовано один метод, який приймає координати початку та кінця маршруту і повертає масив точок маршруту. Незалежно від того, як саме кожен клас прокладає маршрут, для навігатора це не має значення, адже його головна функція – це відображення маршруту. Навігатору достатньо передати стратегії координати початку та кінця, щоб отримати масив точок маршруту у заданому форматі.

Клас навігатора матиме метод для зміни стратегії, що дозволить змінювати алгоритм пошуку маршруту «на льоту». Це буде корисно для клієнтського коду, наприклад, для кнопок-перемикачів типів маршрутів у користувацькому інтерфейсі.

Патерн «Стратегія» пропонує розділити подібні алгоритми на окремі класи-стратегії (рис. 2.9). Основний клас (контекст) делегує виконання алгоритму одній із стратегій, дозволяючи легко змінювати їх. Головне, щоб усі стратегії мали єдиний інтерфейс. Це робить контекст незалежним від конкретних алгоритмів і дозволяє додавати нові стратегії без змін у його коді.

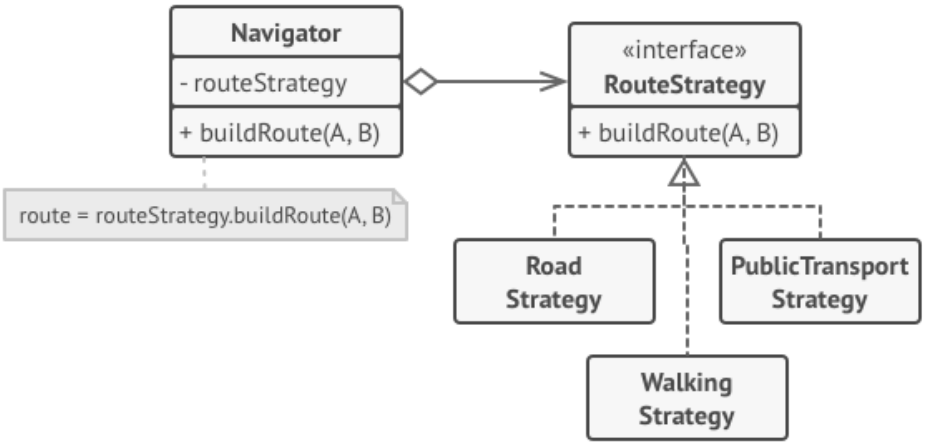

Рисунок 2.9 – Приклад ієрархії «Навігатор» відповідно патерна «Стратегія»

На рис. зображено типову структуру ієрархії класів відповідно до вимог патерну «Стратегія». До основних елементів ієрархії можна віднести таке.

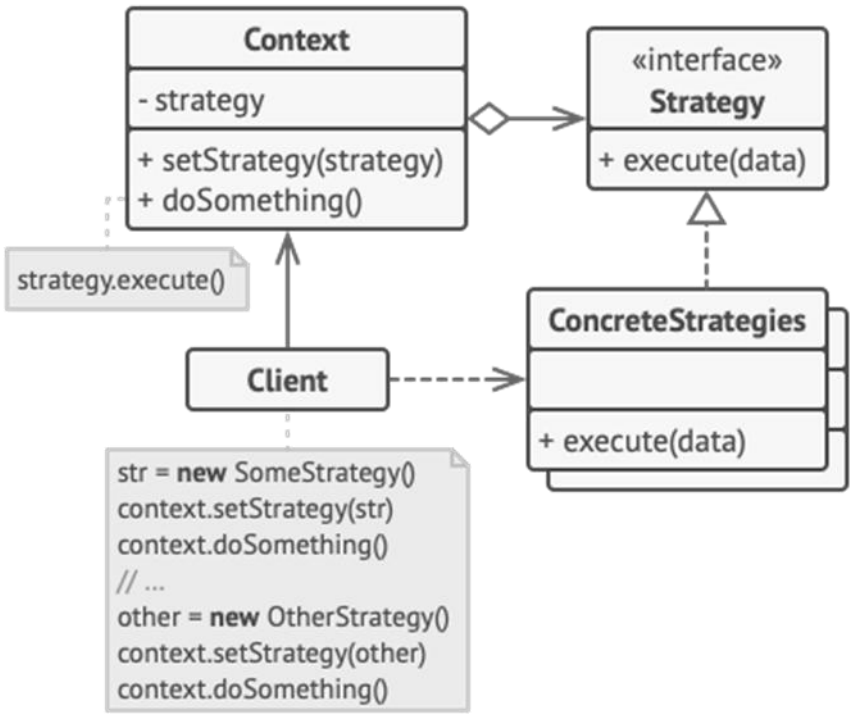

Рисунок 2.10 – Ієрархія класів, що відповідає патерну «Стратегія»

1. Контекст утримує посилання на конкретний об’єкт стратегії, взаємодіючі з ним через загальний інтерфейс стратегій.

2. Стратегія визначає інтерфейс, який є спільним для всіх варіантів алгоритму. Контекст використовує цей інтерфейс для виклику потрібного алгоритму. Для контексту не важливо, яку саме версію алгоритму обрано, адже всі вони реалізують один і той же інтерфейс.

3. Конкретні стратегії реалізують різні варіації алгоритму.

4. Під час виконання програми контекст отримує запити від клієнта і передає їх об’єкту конкретної стратегії.

5. Клієнт має створити об’єкт конкретної стратегії та передати його конструктору контексту. Також клієнт повинен мати можливість змінювати стратегію в процесі роботи, використовуючи сетер поля стратегії. Це дає змогу контексту не знати, яка саме стратегія обрана в даний момент.

Використання патерна «Стратегія» є доцільним, коли потрібно застосовувати різні варіації алгоритму в межах одного об’єкта. Стратегія дозволяє змінювати поведінку об’єкта під час виконання програми,

підставляючи різні об’єкти-поведінки (наприклад, з різним балансом швидкості та споживання ресурсів).

Якщо є багато схожих класів, що відрізняються лише окремими аспектами поведінки, патерн «Стратегія» дозволяє виокремити цю різницю в окрему ієрархію класів, зводячи початкові класи до одного, налаштовуючи його поведінку за допомогою стратегій.

Якщо не хочете оголювати деталі реалізації алгоритмів для інших класів, патерн дозволяє ізолювати код, дані й залежності алгоритмів від інших об’єктів, приховуючи ці деталі всередині класів-стратегій.

Коли різні варіації алгоритмів реалізуються за допомогою великого умовного оператора, кожна гілка цього оператора може стати окремим класом-стратегією. Потім контекст отримує певний об’єкт-стратегію від клієнта і делегує йому виконання роботи. Якщо виникне потреба змінити алгоритм, контекст може прийняти іншу стратегію.

Для ефективного рефакторингу коду відповідно до патерну «Стратегія» потрібно виконати наступні кроки:

1. Визначте алгоритм, що підлягає частим змінам, або той, що має кілька варіацій, які обираються під час виконання програми.

2. Створіть інтерфейс стратегій, що описує цей алгоритм, спільний для всіх його варіантів.

3. Розмістить варіації алгоритму в окремих класах, які реалізують цей інтерфейс.

4. У класі контексту створіть поле для зберігання посилання на поточний об’єкт-стратегію та метод для її зміни. Переконайтеся, що контекст працює з цим об’єктом лише через загальний інтерфейс стратегій.

5. Клієнти контексту мають передавати відповідний об’єкт-стратегію, коли хочуть, щоб контекст діяв певним чином.

До переваг та недоліків патерна «Стратегія» можна віднести таке:

– можливість «гарячої» заміна алгоритмів у процесі роботи;

– ізоляція коду та даних алгоритмів від інших класів;

– заміна наслідування делегуванням;

– реалізація принципу відкритості/закритості;

– ускладнення програми через наявність додаткових класів;

– клієнт повинен розуміти різницю між стратегіями.

2.3 Аналіз силабусів освітніх компонент

Для досягнення мети дослідження виконано аналіз освітніх компонент (ОК) вітчизняних закладів вищої освіти, присвячених вивченню процесів проєктування програмного забезпечення, з метою встановлення напрямків подальших досліджень.

Навчальна програма дисципліни «Конструювання програмного забезпечення» Міжрегіональної Академії управління персоналом [14] визначає головною метою вивчення студентами напрямку «Комп’ютерні науки» головних принципів підбору персоналу, організації роботи, розподілу функцій та написання технічних завдань для створення сучасного та конкурентоздатного програмного продукту.

Програма розрахована на спеціалістів з напрямку «Комп’ютерні науки», які володіють, в рамках відповідних навчальних курсів, знаннями з дисциплін: «Основи програмування та алгоритмічні мови», «Об’єктно-орієнтовне програмування та організація баз даних та знань».

До головних завдання віднесено:

– сформувати знання та отримати практичні навички про основи конструювання програмного забезпечення;

– отримати уяву про засоби зборки програмного забезпечення;

– набути навичок та отримати досвід по створенню та використанню програмних продуктів.

Змістовим модулем «Проєктування програмного забезпечення» передбачено вивчення таких тем:

– рефакторинг;

– рівні рефакторингу;

– безпечний рефакторинг;

– стратегії рефакторингу.

Зокрема, опис теми «стратегії рефакторингу» визначає необхідність опанування студентами таких навичок:

– виконувати рефакторинг при створенні нових методів та нових класів;

– виконувати рефакторинг при виправленні дефектів;

– виконувати рефакторинг модулів, в яких велика ймовірність виникнення помилок;

– виконувати рефакторинг складних модулів.

Силабус ОК «Проєктування програмного забезпечення» розроблено для освітньо-професійної програми «Інженерія програмного забезпечення» підготовки здобувачів вищої освіти ступеня бакалавра на факультеті кібербезпеки та інформаційних технологій у Національному університеті «Одеська юридична академія» з галузі знань 12 «Інформаційні технології» за спеціальністю 121 «Інженерія програмного забезпечення» [15].

Проєктування програмного забезпечення (ПЗ) описує детальне створення робочої програмної системи за допомогою комбінації процесів кодування, верифікації, модульного тестування, інтеграційного тестування та налагодження. Проєктування займає важливе місце в життєвому циклі програмного забезпечення та пов'язане з проєктуванням та тестуванням ПЗ. Навчальна дисципліна формує необхідний теоретичний і практичний базис знань сучасних методів та засобів проєктування ПЗ у фахівців з інженерії програмного забезпечення.

Мета навчальної дисципліни – ознайомлення здобувачів освіти з етапами процесу проєктування ПЗ, з етапами життєвого циклу ПЗ. Завдання навчальної дисципліни – опанування навичок проєктування програмних продуктів із застосуванням мов проєктування, програмних методів й інструментальних систем.

Технологія навчання дисципліни: аудиторне навчання із можливістю застосування електронного навчання і дистанційних освітніх технологій. Методи, які використовуються під час навчання: лекція, лабораторні роботи, консультація, самостійна робота, бесіди і дискусії.

Передбачено вивчення таких тем.

1. Фундаментальні складові процесу проєктування ПЗ: мінімізація складності, очікування змін, проєктування з можливістю перевірки, використання зовнішніх та внутрішніх стандартів проєктування. Життєвий цикл ПЗ.

2. Керування проєктуванням. Стандарти та моделі проєктування. Планування проєктування. Зміни в конструюванні. Блочне тестування, інтеграційне тестування і налагодження власного коду. Надійність та якість ПЗ. Системна інтеграція ПЗ. Методики сумісної розробки ПЗ.

3. Типові способи вирішення поширених проблем при конструюванні ПЗ. Породжуючі, Поведінкові та поведінкові патерни проєктування.

Аналіз запропонованих у силабусах підходів до вивчення процесів проєктування ПЗ показав, що у першу чергу необхідно враховувати досвід відомих дослідників у галузі ІТ, саме тому актуальним у підготовці майбутніх фахівців з цифрових технологій є вивчення патернів проєктування.

У подальшому матеріалі наведено етапи модернізації нового лабораторного практикуму «Поведінкові патерни проєктування», який акцентує увагу здобувачів на особливостях принципах проєктування SOLID та реалізації таких патернів, як «Стан» та «Стратегія».

2.4 Постановка лабораторної роботи «Проєктування предметної області з використанням поведінкового патерна «Стан»

Постановка завдання.

1. Опрацюйте теоретичний матеріал. Ознайомтесь з специфікою поведінкового патерна проектування «Стан».

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

3. Згенерувати програмний код реалізації системи згідно розглянутого шаблону проектування. Обов'язково повинен бути блок Main(), в якому створюються всі необхідні об'єкти. Код повинен бути працездатним та не містити помилки компіляції. Добавити коментарі та нотації.

4. Звіт до лабораторної роботи повинен містити:

– опис предметної області;

– UML-діаграму ієрархії класів (необхідний мінімум класів);

– код, згенерований по ієрархії класів, на мові C#, або C++, Java;

– скріншоти результатів роботи;

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

Приклад №1 виконання лабораторної роботи. За допомогою патерна «Стан» реалізувати систему моніторингу міського транспорту. На рис. 2.11 наведено UML-діаграму класів, що відповідає патерну «Стан».

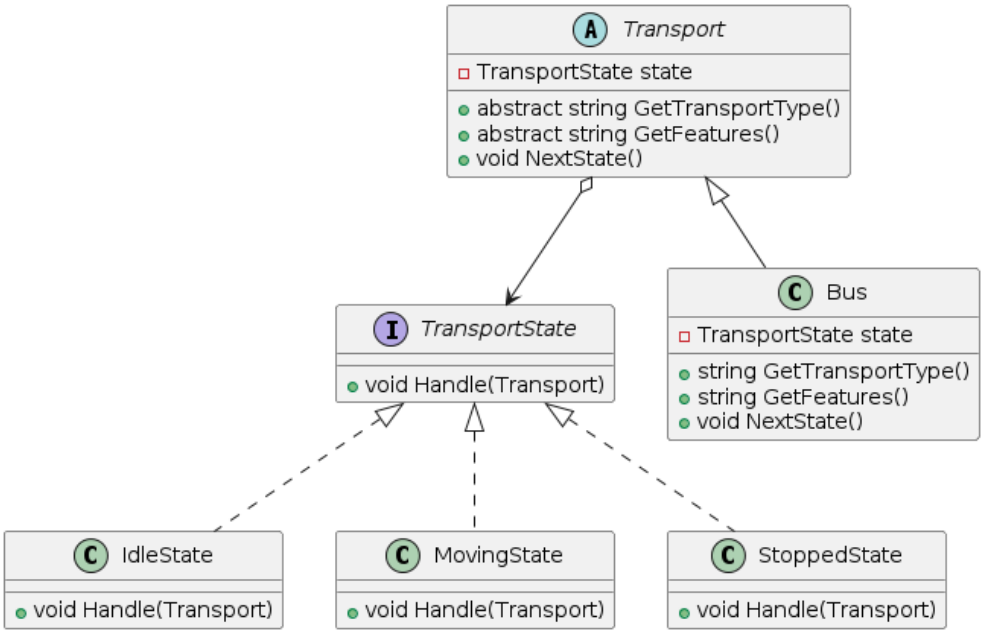

Рисунок 2.11 – UML-діаграма ієрархії класів

Програмний код реалізації патерна «Стан» для предметної області:

```
/*
 Клас Transport є базовим абстрактним класом, який наслідуватимуть класи окремих
видів транспорту. Він має абстрактний метод GetTransportType, який повертає тип
транспорту та абстрактний метод GetFeatures, який повертає перелік функціоналу
транспорту. Він також має властивість state, яка визначає стан транспорту та метод
NextState, який змінює стан транспорту.
*/
public abstract class Transport
{
  public TransportState state { get; set; }

  public Transport(TransportState initialState)
  {
    state = initialState;
  }
  public abstract string GetTransportType();
  public abstract string GetFeatures();
  public void NextState()
  {
    state.Handle(this);
  }
}

/*
 Інтерфейс TransportState визначає метод Handle, який викликається при зміні стану
транспорту.
*/
public interface TransportState
{
  void Handle(Transport transport);
}

/*
 Клас IdleState реалізує інтерфейс TransportState та визначає стан транспорту, коли
він не рухається.
*/
public class IdleState : TransportState
{
  public void Handle(Transport transport)
  {
    Console.WriteLine($"{transport.GetTransportType()} is now idle.");
    transport.state = new MovingState();
  }
}

/*
 Клас MovingState реалізує інтерфейс TransportState та визначає стан транспорту,
коли він рухається.
 */
public class MovingState : TransportState
{
  public void Handle(Transport transport)
  {
    Console.WriteLine($"{transport.GetTransportType()} is now moving.");
    transport.state = new StoppedState();
  }
}

/*
 Клас StoppedState реалізує інтерфейс TransportState та визначає стан транспорту,
коли він зупинився.
 */
public class StoppedState : TransportState
```

```
{
  public void Handle(Transport transport)
  {
    Console.WriteLine($"{transport.GetTransportType()} has stopped.");
    transport.state = new IdleState();
  }
}

/*
 Клас Bus є конкретним класом, який наслідує клас Transport і визначає автобус
(маршрутку) як тип транспорту.
 Він реалізує метод GetTransportType, який повертає тип транспорту
 та метод GetFeatures, який повертає перелік функціоналу транспорту.
*/
public class Bus : Transport
{
  public Bus() : base(new IdleState()) { }

  public override string GetTransportType()
  {
    return "Bus";
  }

  public override string GetFeatures()
  {
    return "Standard features";
  }
}

class Program
{
  static void Main(string[] args)
  {
    Transport bus = new Bus();
    Console.WriteLine($"{bus.GetTransportType()}: {bus.GetFeatures()}");

    // Виклики методу NextState змінює стан транспорту.
    bus.NextState(); // Тепер автобус рухається.
    bus.NextState(); // Тепер автобус зупиняється.
    bus.NextState(); // Тепер автобус не рухається.
  }
}
```

```
C:\Users\Nicolas\Desktop\Стаціонар (121)\Конструювання>dotnet run
Bus: Standard features
Bus is now idle.
Bus is now moving.
Bus has stopped.

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

Рисунок 2.12 – Приклад виконання програми

Приклад №2 виконання лабораторної роботи. За допомогою патерна «Стан» реалізувати систему бронювання та продажу квитків на проїзд автобусним транспортом. На рис. 2.13 наведено UML-діаграму класів, що відповідає патерну «Стан».

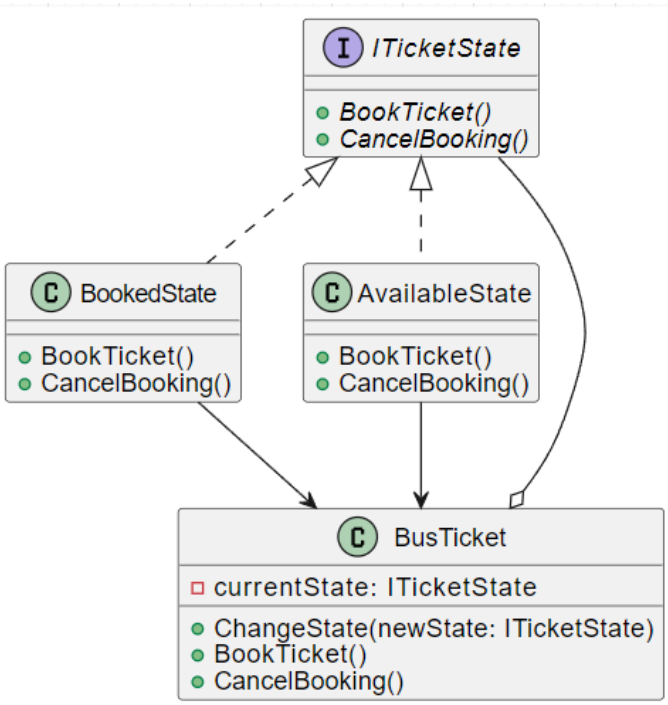

Рисунок 2.13 – UML-діаграма ієрархії класів

Програмний код реалізації патерна «Стан» для предметної області:

```
namespace TicketBookingSystem
{
  // Інтерфейс для шаблону стану
  public interface ITicketState
  {
    void BookTicket();
    void CancelBooking();
  }

  // Конкретне впровадження стану «Заброньовано».
  public class BookedState : ITicketState
  {
    public void BookTicket()
    {
      Console.WriteLine("Квиток уже заброньовано.");
    }
    public void CancelBooking()
    {
      Console.WriteLine("Скасування бронювання квитків...");
      Console.WriteLine("Бронювання квитків скасовано.");
    }
  }
  // Конкретна реалізація стану «Доступний».
  public class AvailableState : ITicketState
  {
    public void BookTicket()
    {
      Console.WriteLine("Бронювання квитка...");
      Console.WriteLine("Квиток успішно заброньовано.");
    }
    public void CancelBooking()
    {
      Console.WriteLine("Немає бронювання для скасування.");
    }
```

```
  }

  // Клас контексту, що представляє автобусний квиток
  public class BusTicket
  {
    private ITicketState currentState;

    // За замовчуванням конструктор ініціалізує стан квитка як «Доступний».
    public BusTicket()
    {
      currentState = new AvailableState();
    }

    // Метод зміни стану квитка
    public void ChangeState(ITicketState newState)
    {
      currentState = newState;
    }

    // Метод бронювання квитка
    public void BookTicket()
    {
      currentState.BookTicket();
      // Змінiти стан на "Забронiйовано" після бронювання
      ChangeState(new BookedState());
    }

    // Метод скасування бронювання
    public void CancelBooking()
    {
      currentState.CancelBooking();
      // Змінити стан назад на "Доступний" після скасування
      ChangeState(new AvailableState());
    }
  }

  // Клієнтський клас для тестування системи бронювання квитків
  public class Client
  {
    public static void Main(string[] args)
    {
      // Встановлення кодування консолі на UTF-8
      Console.OutputEncoding = Encoding.UTF8;

      // Створення нового автобусного квитка
      BusTicket ticket = new BusTicket();

      // Відображення початкового стану
      Console.WriteLine("Початковий стан квитка: доступний");

      // Спроба забронювати квиток
      Console.WriteLine("Спроба забронювати квиток...");
      ticket.BookTicket();

      // Спроба скасувати бронювання
      Console.WriteLine("Спроба скасувати бронювання...");
      ticket.CancelBooking();

      // Повторна спроба скасувати бронювання
      Console.WriteLine("Повторна спроба скасувати бронювання...");
      ticket.CancelBooking();
      Console.ReadKey();
    }
  }
}
```

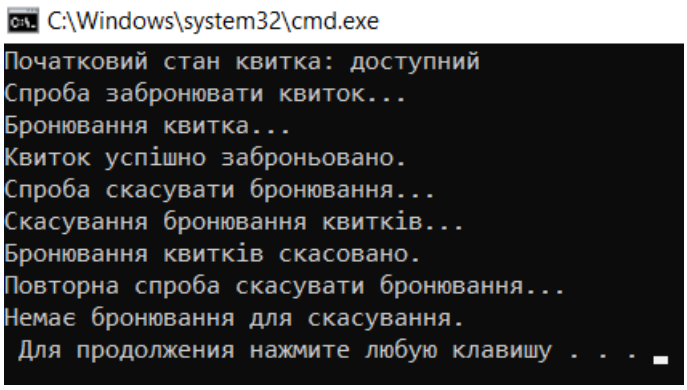

Рисунок 2.14 – Приклад виконання програми

2.5 Постановка лабораторної роботи «Проєктування предметної області з використанням поведінкового патерна «Стратегія»

Постановка завдання.

1. Опрацюйте теоретичний матеріал. Ознайомтесь з специфікою патерна проектування «Стратегія».

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

3. Згенерувати програмний код реалізації системи згідно розглянутого шаблону проектування. Обов’язково повинен бути блок Main(), в якому створюються всі необхідні об’єкти. Код повинен бути працездатним та не містити помилки компіляції. Добавити коментарі та нотації.

4. Звіт до лабораторної роботи повинен містити:

– опис предметної області;

– UML-діаграма ієрархії класів;

– код, згенерований по діаграмі класів, на мові C#, або C++, Java;

– скріншоти результатів роботи;

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

Приклад №1 виконання лабораторної роботи. За допомогою патерна «Стратегія» реалізувати систему моніторингу міського транспорту. На рис. 2.15 наведено UML-діаграму класів, що відповідає патерну «Стратегія».

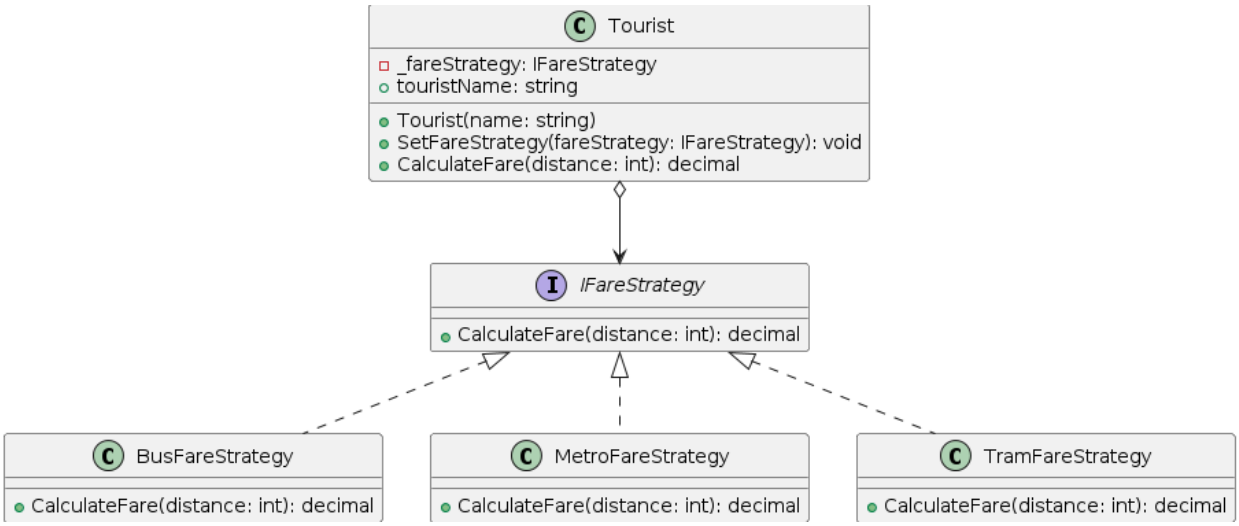

Рисунок 2.15 – UML-діаграма ієрархії класів

Програмний код реалізації патерна «Стратегія» для предметної області:

```
// Інтерфейс стратегії
interface IFareStrategy
{
  decimal CalculateFare(int distance);
}

// Конкретні стратегії
class BusFareStrategy : IFareStrategy
{
  public decimal CalculateFare(int distance)
  {
    return 1.5m + 0.1m * distance;
  }
}

class MetroFareStrategy : IFareStrategy
{
  public decimal CalculateFare(int distance)
  {
    return 2.0m + 0.15m * distance;
  }
}

class TramFareStrategy : IFareStrategy
{
  public decimal CalculateFare(int distance)
  {
    return 1.8m + 0.12m * distance;
  }
}

class Tourist
{
  private IFareStrategy _fareStrategy;
```

```
  public string touristName;

  public PublicTransport(string name)
  {
    touristName = name;
  }

  public void SetFareStrategy(IFareStrategy fareStrategy)
  {
    _fareStrategy = fareStrategy;
  }

  public decimal CalculateFare(int distance)
  {
    return _fareStrategy.CalculateFare(distance);
  }
}

class Program
{
  static void Main(string[] args)
  {
    var distance = 10;

    var travellingTourist = new Tourist("Danny");
    travellingTourist.setFareStrategy(new BusFareStrategy());
    Console.WriteLine($"Bus fare for {distance} km: " +
      $"{busTransport.CalculateFare(distance):C}");
    travellingTourist.setFareStrategy(new MetroFareStrategy());
    Console.WriteLine($"Metro fare for {distance} km: " +
      $"{metroTransport.CalculateFare(distance):C}");
    travellingTourist.setFareSteategy(new TramFareStrategy());
    Console.WriteLine($"Tram fare for {distance} km: " +
      $"{tramTransport.CalculateFare(distance):C}");
  }
}
```

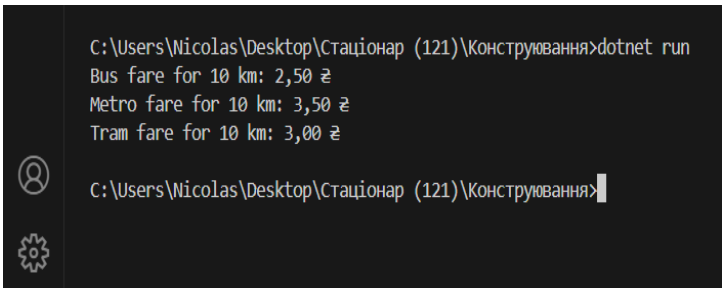

Рисунок 2.16 – Приклад виконання програми

Приклад №2 виконання лабораторної роботи. За допомогою патерна «Стратегія» реалізувати систему бронювання та продажу квитків на проїзд автобусним транспортом. На рис. 2.17 наведено UML-діаграму класів, що відповідає патерну «Стратегія».

Програмний код реалізації патерна «Стратегія» для предметної області:

```
namespace BusTicketBooking
{
  // Клас для обробки різних стратегій розрахунку цін на квитки
  public interface ITicketPriceStrategy
```

```
    {
        double CalculatePrice(double basePrice);
    }

    // Конкретна стратегія для розрахунку цін на квитки для дорослих
    public class AdultTicketPriceStrategy : ITicketPriceStrategy
    {
        public double CalculatePrice(double basePrice)
        {
            return basePrice; // Просто базова ціна для дорослих
        }
    }

    // Конкретна стратегія для розрахунку цін на квитки для дітей
    public class ChildTicketPriceStrategy : ITicketPriceStrategy
    {
        public double CalculatePrice(double basePrice)
        {
            return basePrice * 0.5; // Діти платять половину від базової ціни
        }
    }

    // Клас для представлення квитків
    public class BusTicket
    {
        private ITicketPriceStrategy _priceStrategy;
        private double _basePrice;

        public BusTicket(ITicketPriceStrategy priceStrategy, double basePrice)
        {
            _priceStrategy = priceStrategy;
            _basePrice = basePrice;
        }

        // Метод для зміни стратегії розрахунку ціни
        public void SetPriceCalculationStrategy(ITicketPriceStrategy priceStrategy)
        {
            _priceStrategy = priceStrategy;
        }

        // Властивість для отримання загальної ціни квитка з урахуванням стратегії
        public double TotalPrice
        {
            get { return _priceStrategy.CalculatePrice(_basePrice); }
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            // Створення квитків для дорослих та дітей з використанням об'єктів стратегій
            var adultTicket = new BusTicket(new AdultTicketPriceStrategy(), 50);
            var childTicket = new BusTicket(new ChildTicketPriceStrategy(), 50);
            // Вивід загальних цін квитків
            Console.WriteLine($"Total price for adult ticket: {adultTicket.TotalPrice}");
            Console.WriteLine($"Total price for child ticket: {childTicket.TotalPrice}");

            Console.ReadKey();
        }
    }
}
```

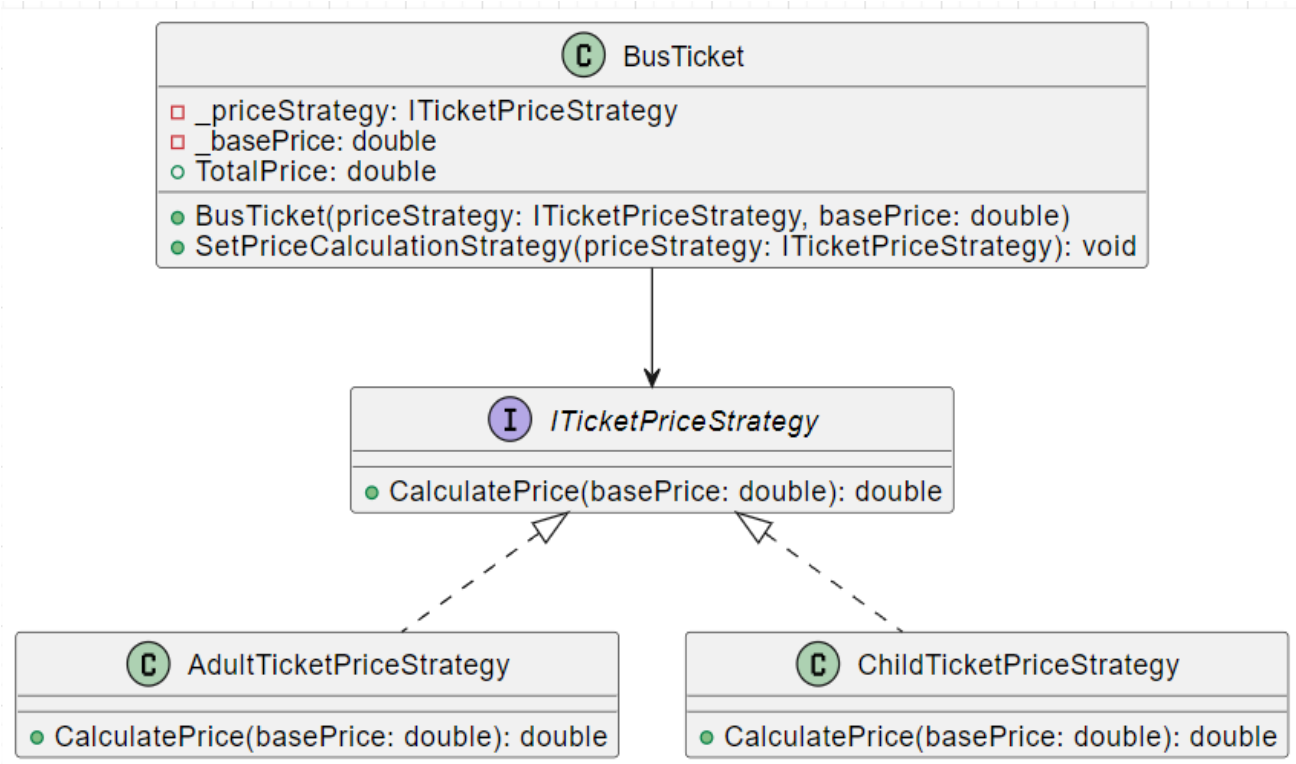

Рисунок 2.17 – UML-діаграма ієрархії класів

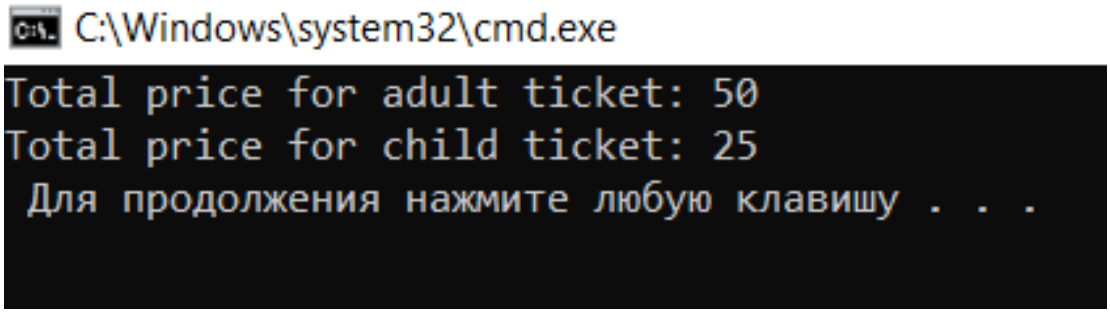

Рисунок 2.18 – Приклад виконання програми

Приклад №3 виконання лабораторної роботи. За допомогою патерна проєктування «Стратегія» реалізувати систему продажу у аптеці ліки різних виробників. Всі вони мають назву, групу ліків, до якої вони належать (антибіотики, протизапальні, шлункові тощо), ціну, термін зберігання. Розробити структуру класів, які можна використовувати для комп'ютерної обробки даних про ліки (як вітчизняні, так й імпортні) в аптеці.

На рис. 2.19 наведено UML-діаграму класів, що відповідає патерну «Стратегія».

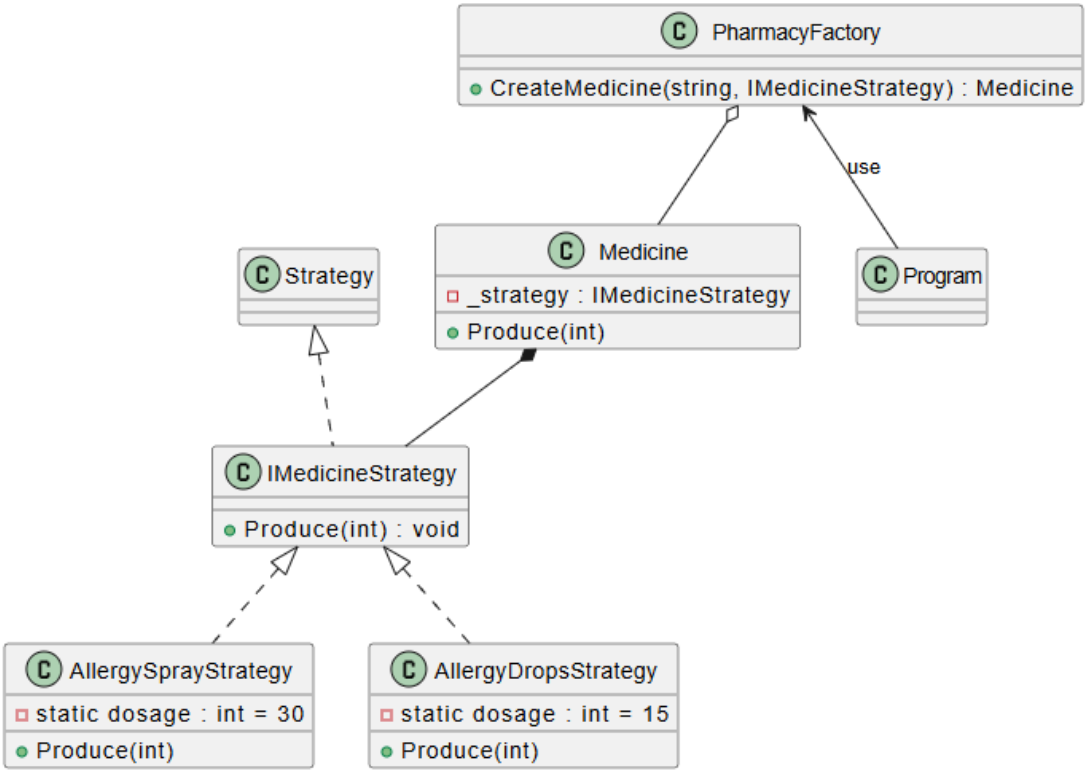

Рисунок 2.19 – UML-діаграма ієрархії класів

Далі наведено опис класів та їх атрибутів.

1. Інтерфейс IMedicineStrategy јписує загальний інтерфейс для стратегій виготовлення медикаментів. Кожна стратегія визначає, як буде виконуватись процес виготовлення. Абстрактний метод Produce() описує конкретні стратегії для виробництва медикаментів.

2. Клас AllergySprayStrategy реалізує конкретну стратегію для виробництва спрею від алергії. Статичний атрибут: dosage визначає дозування для спрею, фіксоване значення – 30 мг. Метод Produce() реалізує виготовлення спрею, виводячи на консоль інформацію про назву, дозування та номер партії.

3. Клас AllergyDropsStrategy реалізує конкретну стратегію для виробництва крапель від алергії. Статичний атрибут dosage визначає дозування для крапель, фіксоване значення – 15 мг. Метод Produce() реалізує виготовлення крапель, виводячи на консоль інформацію про назву, дозування та номер партії.

4. Клас Medicine представляє об'єкт медикаменту, для якого визначено стратегію виробництва. Приватний атрибут _strategy містить стратегію

виготовлення медикаменту (або спрею, або крапель). Метод Produce() викликає метод Produce обраної стратегії _strategy, передаючи номер партії.

5. Клас PharmacyFactory є фабрика для створення об'єктів Medicine з переданою стратегією виробництва. Метод CreateMedicine() створює об'єкт Medicine, використовуючи надану стратегію.

Програмний код реалізації патерна «Стратегія» для предметної області:

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

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

    int batch = 1;

    PharmacyFactory pharmacy = new PharmacyFactory();

    // Виготовлення трьох партій спрею з використанням стратегії
    for (int i = 0; i < 3; i++)
    {
      var allergySpray = pharmacy.CreateMedicine("Спрей від алергії 'Віста'",
                new AllergySprayStrategy());
      allergySpray.Produce(batch);
      batch++;
    }

    // Виготовлення двох партій крапель з використанням стратегії
    for (int i = 0; i < 2; i++)
    {
      var allergyDrops = pharmacy.CreateMedicine("Краплі від алергії 'Лютик'",
                new AllergyDropsStrategy());
      allergyDrops.Produce(batch);
      batch++;
    }
  }
}

// Інтерфейс стратегії для виготовлення медикаментів
interface IMedicineStrategy
{
  void Produce(int batch);
}

// Стратегія для спрею від алергії
class AllergySprayStrategy : IMedicineStrategy
{
  private static readonly int dosage = 30;

  public void Produce(int batch)
  {
    Console.WriteLine($"Виготовлено спрей від алергії 'Віста' дозуванням {dosage}мг
з партії під №{batch}.");
  }
}
```

```
// Стратегія для крапель від алергії
class AllergyDropsStrategy : IMedicineStrategy
{
  private static readonly int dosage = 15;
  public void Produce(int batch)
  {
    Console.WriteLine($"Виготовлено краплі від алергії 'Лютик' дозуванням {dosage}мг
з партії під №{batch}.");
  }
}

// Клас для представлення медикаменту з визначеною стратегією
class Medicine
{
  private readonly IMedicineStrategy _strategy;

  public Medicine(IMedicineStrategy strategy)
  {
    _strategy = strategy;
  }

  public void Produce(int batch)
  {
    _strategy.Produce(batch);
  }
}

// Фабрика для створення медикаментів з передачею стратегії
class PharmacyFactory
{
  public Medicine CreateMedicine(string key, IMedicineStrategy strategy)
  {
    return new Medicine(strategy);
  }
}
```

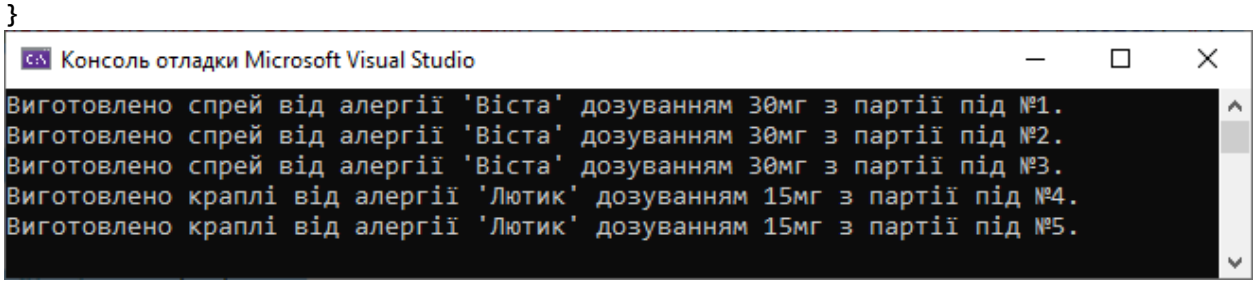

Рисунок 2.13 – Приклад виконання програми

Висновки до розділу 2

У розділі виконано аналіз характеристик об'єктів у галузі проєктування програмного забезпечення з використанням патернів. Зокрема, виконано огляд принципів об'єктно-орієнтованого проєктування класів SOLID та поведінкових патернів проєктування «Стан» та «Стратегія». Виконано огляд силабусів освітньої компоненти «Проєктування програмного забезпечення».

Виконана постановка нових лабораторних робіт:

1. проєктування предметної області з використанням поведінкового патерна «Стан»;

2. проєктування предметної області з використанням поведінкового патерна «Стратегія».

Опанування майбутніми фахівцями з цифрових технологій 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

В останній час, через карантинні обмеження та воєнний стан, спостерігається зниження якості кадрів. Зростає попит на фахівців рівня Middle і вище, які добре володіють сучасними інструментами та IT-технологіями. Формування національного IT-середовища ускладнюють стандарти інформаційної безпеки, які потребують високої кваліфікації та укомплектованості IT-персоналу. Бізнес почав більше інвестувати в кіберстійкість: збільшуються витрати як на створення внутрішніх команд, так і на аутсорсинг послуг для захисту від кіберзагроз. Зростання обсягу робіт змушує компанії конкурувати за спеціалістів та підвищувати їхні зарплати. Це спричиняє перехід досвідчених кадрів до великих корпорацій, залишаючи малий бізнес і громадський сектор з дефіцитом. Недостатня кількість кваліфікованих спеціалістів може призвести до зниження середньої якості працівників на ринку, і ця ситуація, ймовірно, погіршиться в найближчі три-п'ять років.

Технології та IT швидко розвиваються, і навчальні програми не встигають за цим темпом. Для вирішення цієї проблеми потрібна активна співпраця між навчальними закладами та бізнесом. Важливо, щоб держава створювала сприятливі умови для такої взаємодії та стимулювала її розвиток. Підвищення рівня освіти має підтримуватися реальним сектором, що задаватиме запити на спеціалістів з відповідними навичками та знаннями.

Вимоги до кадрового забезпечення підготовки фахівців з цифрових технологій для викладання освітнього модуля «Поведінкові патерни проєктування» є надзвичайно важливими для формування висококваліфікованих спеціалістів, які зможуть забезпечити потреби сучасного ринку праці. Сучасний світ стрімко розвивається завдяки таким технологіям, як паралельні обчислення та хмарні сервіси, що стали невід'ємною частиною життя.

Професія розробника – одна з найбільш популярних та перспективних. Цей фахівець створює програмне забезпечення, яке працює на платформах віртуалізації та хмарних сервісах. Ці програми відповідають за існування та безперебійну роботу нових функцій та можливостей, а також за виправлення помилок та покращення продуктивності системи. Розробник повинен добре володіти мовами програмування, а також мати досвід роботи з API-інтерфейсами та інструментами створення різноманітних додатків.

Використання розробниками ПЗ поведінкових патернів є основою для створення ефективних та легко підтримуваних програмних рішень. Вони дозволяють розробникам структурувати код таким чином, щоб він був гнучким до змін та легко масштабувався. Патерни, такі як «Цепочка відповідальності», «Команда», «Спостерігач» та «Стратегія», забезпечують можливість створення модульних рішень, які легко адаптуються до нових вимог бізнесу та змін на ринку. Використання патернів допомагає розробникам уникати повторень коду та спрощує підтримку існуючих систем, що особливо важливо в умовах швидкого розвитку технологій.

Для досягнення мети дослідження необхідно виконати аналіз професійних компетенцій майбутніх фахівців у галузі проєктування програмного забезпечення.

Для розробників програмного забезпечення є надзвичайно важливим володіння поведінковими патернами, оскільки вони дозволяють створювати гнучкі, адаптовані до змін та легкі в підтримці програмні рішення. Патерни, такі як «Стратегія», «Спостерігач» або «Стан», допомагають структурувати код таким чином, щоб він залишався зрозумілим та масштабованим. Зокрема, сучасному розробнику ПЗ необхідно мати:

– глибокі знання мов програмування (Java, Python, C++, та інші) у поєднанні з досвідом використання патернів є основою ефективної роботи;

– досвід роботи з API-інтерфейсами та інструментами для створення додатків, що функціонують на платформах віртуалізації та хмарних сервісах.

Системні адміністратори повинні:

– вміти встановлювати, налаштовувати та підтримувати хмарну інфраструктуру, що є основою для забезпечення стабільної роботи сервісів;

– знати мережеві протоколи, принципи безпеки даних та застосування поведінкових патернів для автоматизації завдань та оптимізації системних процесів;

– досвід роботи з системами моніторингу та оптимізації продуктивності, що допомагає мінімізувати ризики простою серверів.

Архітектори хмарних систем повинні:

– мати глибоке розуміння поведінкових патернів, таких як «Цепочка відповідальності» та «Команда», які дозволяють впроваджувати складні сценарії взаємодії між компонентами хмарних рішень;

– мати досвід проектування масштабованих інфраструктур та розробка стратегій міграції додатків у хмару з урахуванням сучасних тенденцій безпеки;

– мати глибокі знання у сфері управління великими обсягами даних та використання різних платформ хмарних послуг (AWS, Azure, Google Cloud).

Менеджери хмарних рішень повинні мати:

– навички планування та організації роботи в хмарі, що включає ефективне управління ресурсами та координацію команди фахівців;

– використання поведінкових патернів для впровадження механізмів прийняття рішень, підвищення якості командної роботи та оптимізації робочих процесів;

– знання принципів хмарних технологій та softskills командного лідерства.

Інженери хмарних технологій повинні мати:

– досвід розробки та налаштування серверів, а також забезпечення безпеки системи;

– використання поведінкових патернів для створення інтегрованих рішень, які забезпечують стабільність роботи сервісів та їх масштабованість.

Аналітики даних повинні мати:

– глибокі знання статистики, алгоритмів машинного навчання та програмування;

– застосування поведінкових патернів, таких як «Ітератор» та «Шаблонний метод», для оптимізації процесів обробки великих обсягів даних.

Для досягнення мети дослідження також необхідно виконати аналіз кваліфікаційних вимог до викладацького складу при підготовці майбутніх фахівців у галузі проєктування програмного забезпечення.

Досвід викладання: наявність досвіду викладання дисциплін, пов'язаних з цифровими технологіями, хмарними обчисленнями, проєктуванням програмного забезпечення та використанням патернів проєктування.

Наукова кваліфікація: ступінь кандидата або доктора наук за спеціальностями, такими як «Інженерія програмного забезпечення», «Комп'ютерні науки», «Комп'ютерна інженерія» або «Інформаційні системи та технології».

Також до викладачів ставляться вимоги до постійного підвищення кваліфікації:

1. регулярне проходження викладачами курсів з новітніх технологій, включаючи сертифіковані онлайн-курси з проєктування програмного забезпечення;

2. співпраця з представниками реального сектору ІТ та участь у проєктах, які вимагають знань і досвіду роботи з проєктування ПЗ;

3. розробка програм стажувань, які передбачають роботу з сучасними технологіями проєктування та розробки ПЗ.

Для реалізації та постійного розвитку освітніх програм адміністрація закладів вищої освіти має активно залучати фахівців з бізнесу до участі у викладанні та розробці освітніх програм для забезпечення актуальності навчання.

Для оновлення змісту освітніх компонент гаранти освітніх програм мають враховувати актуальні тенденції та появу нових професій у ІТ-галузі:

– підготовка фахівців до роботи з нейромережами та штучним інтелектом, що створює нові виклики та можливості для розробників ПЗ;

– розширення навчальних програм з акцентом на освоєння нових професій, таких як інженер-промт, тренер ШІ, сценарист чат-ботів.

Ці вимоги допоможуть забезпечити високий рівень підготовки фахівців з цифрових технологій для викладання освітнього модуля «Поведінкові патерни проєктування», що відповідає сучасним тенденціям розвитку ІТ-сфери та потребам бізнесу.

Висновки до розділу 3

У розділі виконано аналіз вимог до кадрового забезпечення об'єкту ІТ-галузі в умовах цифрової трансформації суспільства. Встановлено, що Використання розробниками ПЗ поведінкових патернів є основою для створення ефективних та легко підтримуваних програмних рішень.

Для досягнення мети дослідження виконано аналіз професійних компетенцій майбутніх фахівців у галузі проєктування програмного забезпечення, а також аналіз кваліфікаційних вимог до викладацького складу.

Ці вимоги допоможуть забезпечити високий рівень підготовки фахівців з цифрових технологій для викладання освітнього модуля «Поведінкові патерни проєктування», що відповідає сучасним тенденціям розвитку ІТ-сфери та потребам бізнесу.

РОЗДІЛ 4 МЕТОДИКА ПРОФЕСІЙНОЇ ПІДГОТОВКИ ФАХІВЦІВ З ЦИФРОВИХ ТЕХНОЛОГІЙ. ДИДАКТИЧНИЙ ПРОЄКТ КОНСУЛЬТАТИВНОГО ЗАНЯТТЯ З ТЕМИ «ПОВЕДІНКОВІ ПАТЕРНИ ПРОЄКТУВАННЯ» ДИСЦИПЛІНИ «КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ» ДЛЯ ЗДОБУВАЧІВ ОСВІТИ СПЕЦІАЛЬНОСТІ ПРОФЕСІЙНА ОСВІТА(ЦИФРОВІ ТЕХНОЛОГІЇ)

Вихідні дані:

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

Галузь знань: 01 Освіта /Педагогіка;

спеціальність: Професійна освіта (Цифрові технології);

освітній рівень: перший бакалаврський;

Форма навчання: заочна;

назва навчальної дисципліни і теми, з якої проводиться консультативне заняття: «Поведінкові патерни проєктування» дисципліни «Конструювання програмного забезпечення».

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

кількість кредитів – 4;

модулів – 1;

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

Форма контролю: іспит, залік.

Великий обсяг навчального матеріалу, обширні, складні цілі навчання та великий відсоток часу, що відведено на самостійну роботу, обумовлюють необхідність в проведенні консультативних занять для уточнення та пояснення навчального матеріалу з дисципліни «Конструювання програмного забезпечення».

Проектування цілей консультативного заняття представлені у табл. 4.1 [29].

Таблиця 4.1

Цілі консультативного заняття

Цілі заняття	Цілі формування різних рівнів засвоєння навчального матеріалу	Умови досягнення	Результат у вигляді дій студентів
1	З переліку визначень впізнавати основні поняття теми «Поведінкові патерни проєктування» такі, як «алгоритм», «об’єкти», «класи», «обєкти-колеги», «посередник», «патерн-спостерігач», «патерн-стратегія».	Знати визначення понять «алгоритм», «об’єкти», «класи», «об’єкти-колеги», «посередник», «патерн-спостерігач», «патерн-стратегія».	Правильно названі з переліку основні поняття теми «Поведінкові патерни проєктування» такі, як «алгоритм», «об’єкти», «класи», «об’єкти-колеги», «посередник», «патерн- спостерігач», «патерн-стратегія»; вміло названо сутність «алгоритмів», «об’єктів», «класів».
2	Уміти розрізняти відмінності між поняттями «алгоритм», «об’єкти», «класи», «обєкти-колеги», «посередник», «патерн-спостерігач», «патерн-стратегія».	Виконання дій першого рівня: правильно названі з переліку основні поняття теми «Поведінкові патерни проєктування» такі, як «алгоритм», «об’єкти», «класи», «обєкти-колеги», «посередник», «патерн-спостерігач», «патерн-стратегія», «патерн-команда», «патерн - стан», «патерн-відвідувач»	Вміло розпізнано відмінності між між поняттями такими як «алгоритм», «об’єкти», «класи», «обєкти-колеги», «посередник», «патерн-спостерігач», «патерн-стратегія», «патерн-команда», «патерн - стан», «патерн-відвідувач». Здійснено сутнісну характеристику понять.
3	Уміти аналізувати поняття. Характеризувати зміст патернів.	Виконання дій першого і другого рівнів: вміло розпізнано відмінності між поняттями «алгоритм», «об’єкти», «класи», «обєкти-колеги», «посередник», «патерн-спостерігач», «патерн-стратегія», «патерн-команда», «патерн - стан», «патерн-відвідувач».	Правильно проаналізовано сутнісні характеристики понять.
4	Уміти застосовувати шаблонний метод при вирішенні завдань.	Виконання дій першого, другого і третього рівнів.	Правильно досліджено патерни поведінки.

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

Рекомендована література:

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

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

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

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

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

Інформаційні ресурси

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

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

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

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

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

6.http://www.computer.org/portal/site/ieeecs/menuitem.c5efb9b8ade9096b8a9ca0108bcd45f3/index.jsp?&pName=ieeecs_level1&path=ieeecs/content&file=ethics.xml&xsl=generic.xsl& – IEEE-CS/ACM Software Engineering Ethics and Professional Practices.

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

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

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

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

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

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

Визначимо найбільш складних для розуміння та засвоєння питань (табл. 4.2) [25].

Таблиця 4.2

Обрання питань для консультування та формулювання відповідей на можливі питання

Теми (або тема) дисципліни	Зміст програми за кожною темою	Найбільш складні питання за темами (темою)	Відповіді на питання
1	2	3	4
Поведінкові патерни проєктування	1.Породжуючі, структурні та поведінкові патерни проєктування**.**	1. Для чого використовується спадкування у патернах поведінки?	1. У патернах поведінки рівня класу використовується спадкування - щоб розподілити поведінку між різними класами. З них більш простим і поширеним є шаблонний метод, який являє собою абстрактне визначення алгоритму. Алгоритм тут визначається покроково. На кожному кроці викликається або примітивна або абстрактна операція. Алгоритм «обростає м'ясом» за рахунок підкласів, де визначено абстрактні операції. Інший патерн поведінки рівня класу – інтерпретатор, який представляє граматику мови у вигляді ієрархії класів та реалізує інтерпретатор як послідовність операцій над екземплярами цих класів.

Продовження табл. 4.2

1	2	3	4
		2.Схарактеризуйте інші патерни поведінки.	2. Інші патерни поведінки пов'язані з інкапсуляцією поведінки в об'єкті та делегування йому запитів. Паттерн стратегія інкапсулює алгоритм об'єкта, спрощуючи його специфікацію та заміну. Паттерн команда інкапсулює запит у вигляді об'єкта, який можна передавати як параметр, зберігати у списку історії або використати якось інакше. Паттерн стан інкапсулює стан об'єкта таким чином, що при зміні стану об'єкт може змінювати поведінку**.** Паттерн відвідувач інкапсулює поведінку, яку інакше довелося б розподіляти між класами, а патерн ітератор абстрагує спосіб доступу та обходу об'єктів з деякого агрегату.

Оберемо методи активізації навчальної діяльності студентів на консультації (табл. 4.3) [26].

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

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

Таблиця 4.3

Методи активізації навчальної діяльності студентів на консультації

Дидактичні методи	Реалізація методів при проведенні консультаційного заняття
Методи підвищення наочності	Використання інтерактивної дошки для демонстрації слайдів з теми «Поведінкові патерни проєктування»
Мотиваційні методи	Для реалізації мотивації використаємо: тип: внутрішня мотивація; вид: вступна мотивація; метод: мотивуючий вступ; прийом: віднесення до особистості. Повідомлення важливості вивчення даної теми: «Тема «Поведінкові патерни проєктування».
Проблемні методи	Використання проблемного питання. Проблемні питання: «Чому у патернах поведінки рівня класу використовується спадкування? Обґрунтуйте доцільність. Наведіть приклади
Комунікативні методи	Імітація ситуацій з реального життя. Де Ви можете використати патерни поведінки? В яких професійних ситуаціях?

Таблиця 4.4

Варіанти організації консультативного заняття [24]

№ варіанта	Етапи організації заняття	Характеристика варіанта
1	2	3
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 студентами.
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.

Продовження табл. 4.2

1	2	3
4	- повідомлення теми, - консультування декількома фахівцями в певній області науки і техніка з актуальних питань науки і нової техніки	Цей варіант лекції-консультації проводиться, як правило, зі спеціальних дисциплін, іноді для цієї мети використовуються наукові семінари. Такі заняття дають можливість зіставити думки різних учених на одну і ту ж проблему і є чудовою школою ведення дискусії.

Наводимо розробку сценарію проведення консультативного заняття у відповідності до обраного варіанту його організації (табл. 4.5) [25]. Контурний конспект заняття з теми «Поведінкові патерни проєктування» представлено у додатку Б.

Таблиця 4.5

Сценарій консультативного заняття

Етапи проведення консультативного заняття	Дії викладача	Дії учнів (студентів)
1	2	3
Організаційний момент	Викладач вітає студентів, робить перекличку, пропонує студентам розпочати роботу на консультації.	Студенти вітають викладача, беруть участь у перекличці, налаштовуються на роботу на консультації.
Повідомлення теми і мети уроку	Повідомлення теми заняття «Поведінкові патерни проєктування», сутність понять «алгоритм», «об’єкти», «класи», «обєкти-колеги», «посередник», «патерн-спостерігач», «патерн-стратегія».	Фіксація теми.
Мотивація мети	Повідомлення важливості вивчення даної теми: «Поведінкові патерни проєктування».	Усвідомлення важливості вивчення теми.
Актуалізація знань	Викладач проводить фронтальне усне опитування з метою перевірки базових знань: 1. Як ви розумієте значення процесу розробки програмного забезпечення? 2. Схарактеризуйте суть «патерн поведіки»?	Здобувачі освіти беруть участь у опитуванні та відповідають на поставлені питання

Продовження табл. 4.5

1	2	3
Формування ООД	Викладач проводить консультацію згідно плану, за допомогою методу пояснення: План 1. Суть патерн поведінки. 2. Реалізація алгоритму у патернах поведінки.	Слухають пояснення, конспектують.
Визначення проблемних моментів під час вивчення питань теми та формування ВД	Викладач запитує здобувачів освіти про недоречності, які виникли у них під час самостійного вивчення теми. Викладач відповідає на поставлені запитання. • Найбільш простим і поширеним є шаблонний метод, який являє собою абстрактне визначення алгоритму. Алгоритм тут визначається покроково. На кожному кроці викликається або примітивна або абстрактна операція. Алгоритм «обростає м'ясом» за рахунок підкласів, де визначено абстрактні операції. Інший патерн поведінки рівня класу – інтерпретатор, який представляє граматику мови у вигляді ієрархії класів та реалізує інтерпретатор як послідовність операцій над екземплярами цих класів. • Інші патерни поведінки пов'язані з інкапсуляцією поведінки в об'єкті та делегування йому запитів. Паттерн стратегія інкапсулює алгоритм об'єкта, спрощуючи його специфікацію та заміну. Паттерн команда інкапсулює запит у вигляді об'єкта, який можна передавати як параметр, зберігати у списку історії або використати якось інакше. Паттерн стан інкапсулює стан об'єкта таким чином, що при зміні стану об'єкт може змінювати поведінку. Паттерн відвідувач інкапсулює поведінку, яку інакше довелося б розподіляти між класами, а патерн ітератор абстрагує спосіб доступу та обходу об'єктів з деякого агрегату	Студенти запитують: 1. У чому суть шаблонного методу»? 2. Схарактеризуйте інші патерни поведінки
Підведення підсумків	Підсумки консультації: «Чи розкрили ми всі питання.	Здобувачі освіти слухають, відповідають.

ВИСНОВКИ

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

Автором теоретично обґрунтовано та експериментально перевірено методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Поведінкові патерни проєктування» у закладах вищої освіти засобами інноваційних технік; обґрунтовано специфічну складову змісту освіти, що орієнтовано на знання інтерес до індивідуалізованих методів набуття знань, умінь і навичок і методів саморегуляції пізнавальної діяльності, що дозволяє актуалізувати здобувача освіти як суб’єкта освітньої діяльності і, у сукупності, що надає здобувачеві освіти можливість управляти власною навчальною діяльністю; подальшого розвитку набули зміст та засоби професійної підготовки інженера-педагога з цифрових технологій.

Автором переконливо обґрунтовано, що рівень активності фахівців, які проходять професійну підготовку з цифрових технологій, залежить від ступеня взаємозв'язку навчального матеріалу та змісту професійної діяльності, а також визначається можливістю інтеграції навчальної інформації в особистий та професійний досвід індивіда.

Виконано аналіз характеристик об'єктів у галузі проєктування програмного забезпечення з використанням патернів. Зокрема, виконано огляд принципів об'єктно-орієнтованого проєктування класів SOLID та поведінкових патернів проєктування «Стан» та «Стратегія». Виконано огляд силабусів освітньої компоненти «Проєктування програмного забезпечення».

Виконана постановка нових лабораторних робіт:

1. Проєктування предметної області з використанням поведінкового патерна «Стан»;

2. Проєктування предметної області з використанням поведінкового

патерна «Стратегія».

Опанування майбутніми фахівцями з цифрових технологій запропонованого лабораторного практикуму забезпечить отримання програмних результатів навчання, які сприяють широкому діапазону їх професійної діяльності та високій конкурентоспроможності на ринку праці.

У розділі виконано аналіз вимог до кадрового забезпечення об'єкту ІТ-галузі в умовах цифрової трансформації суспільства. Встановлено, що використання розробниками ПЗ поведінкових патернів є основою для створення ефективних та легко підтримуваних програмних рішень.

Розроблено дидактичний проєкт консультативного заняття з теми «Поведінкові патерни проєктування» дисципліни «Конструювання програмного забезпечення» для здобувачів інженерно-педагогічної спеціальності «Професійна освіта (Цифрові технології)».

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

Апробація результатів дослідження виконана під час виконання лабораторного практикуму з дисципліни «Програмна інженерія» бакалаврами спеціальності 015 Професійна освіта (Цифрові технології) ННППІ УІПА у жовтні 2024 р.

За основними результатами дослідження виконана публікація тез доповідей на конференції «Студенти та молодь – для майбутнього країни» (м. Харків, 14-15 листопада 2024 р.).

СПИСОК ВИКОРИСТОВАНИХ ДЖЕРЕЛ

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 проєктування: 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.

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

8. 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.

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

10. 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.

11. 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.

12. Чикунов П.О. Застосування патернів проєктування 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.

13. 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.

14. Дуднік А.С. Навчальна програма дисципліни “Конструювання програмного забезпечення” (для освітньо-кваліфікаційного рівня «бакалавр»). К.: МАУП, 2019 – С. 17.

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

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

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

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

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

20. The Complete Guide to Becoming a Software Architect – Онлайн-курс

Udemy. URL: https://www.udemy.com/course/the-complete-guide-to-becoming-a-software-architect/.

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

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

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

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

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

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

27. 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.).

28. Viktoriia Kuleshova, Viktoriia Malovana Methodological approaches to the formation of psychological and pedagogical competence of teachers of technical disciplines // SCHOLA 2019 R&E-SOURCE https://journal.ph-noe.ac.at Online

Journal for Research and Education Special Issue 17, Dec. 2019, ISSN 2313-1640 – P.196 -206.

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

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

ДОДАТКИ

ДОДАТОК А КОНТУРНИЙ КОНСПЕКТ «ПАТЕРНИ ПРОЄКТУВАННЯ»

Формалізм опису патернів дозволив зібрати великий каталог патернів, додатково перевіривши кожен патерн на дієвість.

Ви можете цілком успішно працювати, не знаючи жодного патерна. Більше того, ви могли вже не раз реалізувати який-небудь з патернів, навіть не підозрюючи про це.

Але якраз свідоме володіння інструментом відрізняє професіонала від аматора. Ви можете забити цвях молотком, а можете й дрилем, якщо дуже сильно постараєтесь. Але професіонал знає, що головна фішка дриля зовсім не в цьому. Отже, навіщо ж знати патерни?

- Перевірені рішення. Ви витрачаєте менше часу, використовуючи готові рішення, замість повторного винаходу велосипеда. До деяких рішень ви могли б дійти й самотужки, але багато які з них стануть для вас відкриттям.
- Стандартизація коду. Ви робите менше прорахунків при проектуванні, використовуючи типові уніфіковані рішення, оскільки всі приховані в них проблеми вже давно знайдено.

Загальний словник програмістів. Ви вимовляєте назву патерна, замість того, щоб годину пояснювати іншим програмістам, який крутий дизайн ви придумали і які класи для цього потрібні.

Вперше патерни проектування були систематично викладені в книзі "Патерни проектування: Елементи повторно використовуваного об'єктно-орієнтованого програмного забезпечення" ("Design Patterns: Elements of Reusable Object-Oriented Software", "Приемы объектно-ориентированного проектирования. Паттерны проектирования") авторів Е. Гамма Р. Хелм Р. Джонсон Дж. Вліссідес (http://www.uml.org.cn/c++/pdf/DesignPatterns.pdf (англ) http://www.sugardas.lt/~p2d/books/Priemioop.pdf, рос.). Ця книга вийшла англійською мовою в 1995 р у видавництві Addison Wesley Longman, Inc.

Подальший розвиток патернів відображено в книзі "Застосування UML і шаблонів проектування" автора Крега Лармана.

Ідентифікація патерну передбачає визначення таких елементів:

– ім'я; присвоювання шаблонами імен дозволяє проектувати на більш високому рівні абстракції;

– задача; опис того, коли слід застосовувати;

– рішення; опис елементів проектного рішення, відносин між ними, функцій кожного елемента;

– результати; наслідки застосування шаблону і можливі компроміси

У розв'язання задач проектування за допомогою патернів проектування входять такі етапи, як пошук відповідних об'єктів, визначення ступеня деталізації об'єкту, специфікування інтерфейсів об'єкта, специфікування реалізації об'єкта.

Для того, щоб скористатися патерном проектування, необхідно прочитати його опис, переконатися в розумінні згаданих класів і об'єктів, переглянути розділ "Приклад коду", вибрати для учасників відповідні імена, визначити класи, визначити імена операцій, реалізувати операції, які виконують обов'язки і відповідають за відносини, визначені в патерні проектування.

Патерни відрізняються за рівнем складності, деталізації та охоплення проектованої системи. Проводячи аналогію з будівництвом, ви можете підвищити безпеку на перехресті, встановивши світлофор, а можете замінити перехрестя цілою автомобільною розв’язкою з підземними переходами.

Найбільш низькорівневі та прості патерни – ідіоми. Вони не дуже універсальні, позаяк мають сенс лише в рамках однієї мови програмування.

Можна виділити такі групи шаблонів проектування:

- породжувальні патерни (твірні патерни, creational design patterns)
- структурні патерни;
- поведінкові патерни

Паттерни поведінки пов'язані з алгоритмами та розподілом обов'язків між об'єктами. Річ в них йде не лише про самі об'єкти та класи, а й про типові способи взаємодії. Паттерни поведінки характеризують помилковий потік управління, який важко простежити під час виконання програми. Увага акцентована не на потоці управління як такому, а на зв'язках між об'єктами.

У патернах поведінки рівня класу використовується спадкування - щоб розподілити поведінку між різними класами. З них більш простим і поширеним є шаблонний метод, який являє собою абстрактне визначення алгоритму. Алгоритм тут визначається покроково. На кожному кроці викликається або примітивна або абстрактна операція. Алгоритм «обростає м'ясом» за рахунок підкласів, де визначено абстрактні операції. Інший патерн поведінки рівня класу – інтерпретатор, який представляє граматику мови у вигляді ієрархії класів та реалізує інтерпретатор як послідовність операцій над екземплярами цих класів.

У патернах поведінки рівня об'єктів використовується не наслідування, а композиція. Деякі з них описують, як за допомогою кооперації безліч рівноправних об'єктів справляється із завданням, яке жодному з них не під силу. Важливо тут те, як об'єкти одержують інформацію про існування один одного. Об'єкти-колеги можуть зберігати посилання один на одного, але це збільшить ступінь зв'язаності системи. При максимальному ступені пов'язаності кожному об'єкту доведеться мати інформацію про всіх інших. Цю проблему вирішує посередник. Посередник, що між об'єктами-колегами, забезпечує опосередкованість посилань, необхідну для розривання зайвих зв'язків.

Паттерн ланцюжок обов'язків дозволяє й надалі зменшувати ступінь зв'язаності. Він дає можливість надсилати запити об'єкту не безпосередньо, а ланцюжком «об'єктів-кандидатів». Запит може виконати будь-який кандидат, якщо це допустимо в поточному стані виконання програми. Кількість кандидатів наперед не визначена, а підбирати учасників можна під час виконання.

Паттерн спостерігач визначає та відповідає за залежності між об'єктами.

Інші патерни поведінки пов'язані з інкапсуляцією поведінки в об'єкті та делегування йому запитів. Паттерн стратегія інкапсулює алгоритм об'єкта, спрощуючи його специфікацію та заміну. Паттерн команда інкапсулює запит у вигляді об'єкта, який можна передавати як параметр, зберігати у списку історії або використати якось інакше. Паттерн стан інкапсулює стан об'єкта таким чином, що при зміні стану об'єкт може змінювати поведінку. Паттерн відвідувач інкапсулює поведінку, яку інакше довелося б розподіляти між класами, а патерн ітератор абстрагує спосіб доступу.

ДОДАТОК Б ПУБЛІКАЦІЇ ЗА РЕЗУЛЬТАТАМИ ДОСЛІДЖЕННЯ

ЛАБОРАТОРНИЙ ПРАКТИКУМ «ПОВЕДІНКОВІ ПАТЕРНИ ПРОЄКТУВАННЯ»

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

В останній час, через карантинні обмеження та воєнний стан, спостерігається зниження якості кадрів. Зростає попит на фахівців рівня Middle і вище, які добре володіють сучасними інструментами та IT-технологіями. Зростання обсягу робіт змушує компанії конкурувати за спеціалістів та підвищувати їхні зарплати. Недостатня кількість кваліфікованих спеціалістів може призвести до зниження середньої якості працівників на ринку, і ця ситуація, ймовірно, погіршиться в найближчі три-п'ять років. Технології та IT швидко розвиваються, і навчальні програми не встигають за цим темпом. Для вирішення цієї проблеми потрібна активна співпраця між навчальними закладами та бізнесом. Підвищення рівня освіти має підтримуватися реальним сектором, що задаватиме запити на спеціалістів з відповідними навичками.

Для розробників програмного забезпечення full-stack є надзвичайно важливим володіння патернами проєктування, оскільки вони дозволяють створювати адаптовані до змін та легкі в підтримці програмні рішення [1].

Автором виконано аналіз процесів проєктування програмного забезпечення з використанням патернів. Зокрема, виконано огляд принципів об'єктно-орієнтованого проєктування класів SOLID та поведінкових патернів проєктування «Стан» та «Стратегія». Виконано огляд силабусів освітньої компоненти «Проєктування програмного забезпечення».

Виконана постановка нових лабораторних робіт:

1. проєктування предметної області з використанням поведінкового патерна «Стан»;
2. проєктування предметної області з використанням поведінкового патерна «Стратегія».

Встановлено, що використання розробниками поведінкових патернів є основою для створення ефективних та легко підтримуваних програмних рішень. Засвоєння майбутніми фахівцями з цифрових технологій нового практикуму забезпечить досягнення програмних результатів навчання, що сприятимуть розширенню їх професійних можливостей та підвищенню конкурентоспроможності на ринку праці.

Список використаних джерел

1. 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. : матеріали Міжнар. наук.-практ. конф. (м. Одеса, 19 травня 2023 р.). – Одеса : Вид. «Юридика», 2023. Т. 1. – С. 606-608.