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)
«02» 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 «Реалізація поліморфізму мовою C#» у закладах вищої освіти»

Виконав(ла)

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

Данило НАСТАРЧУК _____
(ім'я, прізвище) (підпис)

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

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

Консультант
к.пед.н., доц. Юлія БОБРИКОВА _____
(науковий ступінь, вчене звання, ім'я, прізвище) (підпис)

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

Харків – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В. Н. Каразіна

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

ЗАТВЕРДЖУЮ

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЄКТ)

Настарчук Данило Вадимович
(прізвище, ім'я, по батькові здобувача)

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

затверджені наказом по університету від «08» жовтня 2024 року № 5101-5/3236

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

3. Перелік питань, які потрібно розробити: Актуальність професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Реалізація поліморфізму мовою C#» 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 «Реалізація поліморфізму мовою C#» 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 «Реалізація поліморфізму мовою C#» 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 «Реалізація поліморфізму мовою C#» 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) _____ Данило НАСТАРЧУК
(підпис) (ім'я, прізвище)

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

РЕФЕРАТ

Мета дослідження – теоретично обґрунтувати та частково перевірити методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація поліморфізму мовою C#» у закладах вищої освіти.

Об'єктом дослідження роботи є процес професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація поліморфізму мовою C#» у закладах вищої освіти.

Предметом дослідження роботи є методика професійної підготовки фахівців з цифрових технологій до розробки комплексу освітніх ресурсів.

Охарактеризовано систему професійної підготовки фахівців для викладання освітнього модулю «Реалізація поліморфізму мовою C#».

Виконана постановка лабораторних робіт: приховування полів, властивостей та методів у класах; віртуальні методи та поліморфізм; абстрактні, статичні та запечатані класи; перевизначення операцій.

Виконано аналіз вимог до кадрового забезпечення об'єкту ІТ-галузі.

Розроблено дидактичний проект консультативного заняття.

За основними результатами дослідження виконана публікація тези доповіді на VII Всеукраїнської науково-практичної інтернет-конференції «Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні» (м. Харків, 05-06 грудня 2024 р.).

Обсяг дипломної роботи становить: пояснювальна записка, презентація доповіді. Пояснювальна записка складається з вступу, чотирьох розділів, висновків, списку використаних джерел, додатків. Загальний обсяг роботи 79 сторінок, з яких 46 сторінок основного тексту. Список використаних джерел становить 50 найменувань, 7 таблиць, 8 рисунків.

ПРОФЕСІЙНА ПІДГОТОВКА, ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, ПРИХОВУВАННЯ, ВІРТУАЛЬНІСТЬ, ПОЛІМОРФІЗМ, КАДРОВЕ ЗАБЕЗПЕЧЕННЯ, МЕТОДИЧНА РОЗРОБКА

ABSTRACT

The aim of the research is to theoretically substantiate and partially validate a methodology for the professional training of specialists in digital technologies to teach the educational module "Implementation of Polymorphism in C#" in higher education institutions.

The object of the research is the process of professional training of specialists in digital technologies to teach the educational module "Implementation of Polymorphism in C#" in higher education institutions.

The subject of the research is the methodology for the professional training of specialists in digital technologies to develop a comprehensive set of resources.

The study characterizes the system of professional training for specialists to teach the educational module "Implementation of Polymorphism in C#". Laboratory assignments have been designed, covering topics such as field, property, and method encapsulation in classes; virtual methods and polymorphism; abstract, static, and sealed classes; and operator overloading.

An analysis of staffing requirements for IT industry entities was conducted.

A didactic project for a consultative session was developed.

The key findings of the research were presented in a publication at the VII All-Ukrainian Scientific and Practical Internet Conference "Modern Technologies in Energy, Electromechanics, Control Systems, and Mechanical Engineering" (Kharkiv, December 5–6, 2024).

The diploma work consists of an explanatory note and a presentation for the report. The explanatory note includes an introduction, four chapters, conclusions, a list of references, and appendices. The total length of the work is 79 pages, of which 46 pages are the main text. The reference list includes 50 sources, 7 tables, and 8 figures.

PROFESSIONAL TRAINING, OBJECT-ORIENTED PROGRAMMING, ENCAPSULATION, VIRTUALITY, POLYMORPHISM, STAFFING, METHODOLOGICAL DEVELOPMENT.

ЗМІСТ

ВСТУП

Розвиток постіндустріального, інформаційного суспільства, глобалізація й інтенсифікація світового ринку цифрових та інформаційних технологій та ринку праці висувають нові вимоги до підготовки фахівців з цифрових технологій, які мають бути здатними не тільки створювати програмні продукти, працювати з інформацією, а й успішно адаптуватися до швидкоплинних умов їх професійної діяльності, навчатися впродовж життя, професійно спілкуватися і працювати в команді, розробляти комплексні інформаційні рішення, керувати проектами тощо.

Професійна підготовка фахівців з цифрових технологій вимагає суттєвого коригування змісту, форм і методів освітнього процесу, переходу від знаннєвої до компетентнісної парадигми, орієнтації на формування цілісної компетентності майбутніх фахівців у єдності її загальних та фахових складових.

Проблема професійної підготовки фахівців з цифрових технологій у вітчизняній науці знайшла відображення при вирішенні широкого кола теоретичних і прикладних питань, спрямованих переважно на дослідження: загальних проблем підготовки майбутніх спеціалістів в умовах закладу вищої освіти (А. М. Алексюк, В. А. Гладуш, Л. І. Воротняк, Г. О. Головченко, Н. С. Дворнікова, Т. М. Калюжна, А. П. Конох, А. І. Кузьмінський, О. Я. Кучерук та ін.); системного підходу в професійній освіті (А. М. Алексюк, Є. С. Барбіна, В. А. Гаманюк, І. О. Давидова, В. І. Євдокимов, Л. В. Зданевич, Н. М. Колісніченко, О. В. Лобова, В. В. Сагарда, М. А. Семенов, Н. О. Ткачова та ін.); теоретико-методологічних засад професійної освіти (О. Е. Коваленко, Н. Г. Ничкало, С. О. Сисоєва); компетентнісного підходу в освітньому процесі вищої школи (О. І. Гура, О. В. Касаткіна, Н. А. Побірченко, Ю. М. Рашкевич та ін.); засобів інформаційного забезпечення професійної підготовки фахівців (В. Ю. Биков, Л. М. Калініна, Р. В. Клопов, М. С. Львов, О. В. Співаковський та ін.).

Актуальність визначеної проблеми та вивчення педагогічного досвіду показали, що існують суперечності між:

– високим рівнем професійного замовлення на кваліфікованих спеціалістів у галузі програмування та неготовністю сучасної системи вищої освіти України задовольнити потреби у фахівцях, які відповідають сучасним вимогам ІТ-індустрії;

– утвердженням компетентнісного підходу у вищій освіті в цілому й ІТ-освіті зокрема та невизначеністю сутності професійної компетентності майбутнього інженера-педагога, відсутністю теоретико-методологічного підґрунтя системи її формування;

– актуальною потребою економіки та суспільства у відповідальних, творчих та критично мислячих фахівцях у сфері програмування, здатних самостійно й оперативно приймати системні професійні рішення, і переважанням традиційних, знаннєвих форм та методів професійної підготовки у вищій школі.

Необхідність подолання виявлених суперечностей й обумовила тему дослідження «Професійна підготовка фахівців з цифрових технологій для викладання освітнього модуля «Реалізація поліморфізму мовою C#» у закладах вищої освіти».

Мета дослідження – теоретично обґрунтувати та частково перевірити методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація поліморфізму мовою C#» у закладах вищої освіти».

У зв'язку з поставленою метою, в роботі вирішуються такі завдання.

1. Схарактеризувати особливості професійної підготовки майбутніх фахівців з цифрових технологій у закладах освіти.

2. Теоретично обґрунтувати та розробити методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація поліморфізму мовою C#» у закладах вищої освіти у Бахмутському навчально-науковому професійно-педагогічному інституті Харківського

національного університету імені В.Н. Каразіна.

Об'єктом дослідження роботи є процес професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація поліморфізму мовою C#» у закладах вищої освіти.

Предметом дослідження роботи є методика професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація поліморфізму мовою C#» у закладах вищої освіти.

В ході написання роботи використані такі методи дослідження:

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

– емпіричні: анкетування, опитування, тестування, спостереження, бесіда, діалог, самооцінювання для діагностування з метою вивчення стану сформованості професійних компетентностей у здобувачів вищої освіти зі спеціальності 015 Професійна освіта (Цифрові технології);

– педагогічний експеримент (констатувальний, пошуковий, формувальний) здійснювався з метою перевірки ефективності розробленої методичної системи професійної підготовки майбутніх фахівців з цифрових технологій.

Наукова новизна одержаних результатів дослідження:

– уточнено сутність поняття «професійна підготовка» фахівців з цифрових технологій для викладання освітнього модуля «Реалізація поліморфізму мовою C#» у Бахмутському навчально-науковому професійно-педагогічному інституті Харківського національного університету імені В.Н. Каразіна здобувачів вищої освіти зі спеціальності 015 Професійна освіта (Цифрові технології), яка розуміється як процес, спрямований на набуття фахівцями професійних компетентностей з удосконалення здатності до постійного оновлення знань та самоактуалізації за програмами післядипломної професійної освіти, що дозволяє їм освоїти новий вид

професійної діяльності;

– подальшого розвитку набули принципи навчання (органічної єдності й наступності базової та післядипломної освіти; гуманізму; демократизму; іманентності розвитку; вільного розвитку фахівця; свободи вибору); положення щодо післядипломної освіти як системи, формування особистості інженера-педагога комп'ютерного профілю та управління навчальним процесом у цій освітній галузі.

Теоретичне та практичне значення одержаних результатів полягає в розробці методики професійної підготовки фахівців з цифрових технологій, обґрунтуванні змісту, форм та методів організації процесу професійної підготовки фахівців з цифрових технологій.

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 VII Всеукраїнській науково-практичній інтернет-конференції «Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні» (м. Харків, 05-06 грудня 2024 р.).

РОЗДІЛ 1 АКТУАЛЬНІСТЬ ПРОФЕСІЙНОЇ ПІДГОТОВКИ ФАХІВЦІВ З ЦИФРОВИХ ТЕХНОЛОГІЙ ДЛЯ ВИКЛАДАННЯ ОСВІТНЬОГО МОДУЛЯ «РЕАЛІЗАЦІЯ ПОЛІМОРФІЗМУ МОВОЮ C#» У ЗАКЛАДАХ ВИЩОЇ ОСВІТИ

Сучасні світові тенденції вимагають від спеціалістів нових компетенцій, які неможливо опанувати без постійного професійного розвитку. Динамічні соціально-економічні зміни зумовлюють потребу у формуванні нового типу фахівця – особистості з високим інтелектуальним, творчим і професійно-кваліфікаційним потенціалом, що постійно вдосконалюється в умовах безперервної освіти, орієнтованої на навчання впродовж усього життя.

Сучасна Україна, як держава, що прагне інтеграції до європейської спільноти, стикається з гострою необхідністю у підвищенні професійної компетентності управлінських кадрів. Успішна модернізація країни та ефективне впровадження реформ залежать від здатності фахівців приймати зважені рішення та брати на себе відповідальність. Тому реформування та модернізація системи підготовки, перепідготовки та підвищення кваліфікації фахівців набуває першочергового значення.

Майбутнє та сьогодення Української держави вимагають від вітчизняних управлінців більшої професійної компетентності, здатності приймати найбільш ефективні рішення та спроможності брати на себе відповідальність за модернізацію своєї країни та реалізацію розпочатих реформ на шляху її інтеграції до європейського співтовариства [35]. Сьогодні нашій країні, як ніколи, потрібні високопрофесійні фахівці на всіх рівнях державної влади. Адже саме від компетентності та професіоналізму фахівців напряму залежить становлення України як правової, соціальної, демократичної держави. Саме тому питання реформування та вдосконалення діючої нині в нашій країні системи підготовки, перепідготовки та підвищення кваліфікації фахівців, перетворення її на сучасну функціонально-ефективну і

злагоджену структуру, здатну виховувати висококласних фахівців набуває особливого значення [41].

Нові нормативні вимоги до системи вищої освіти, окреслені у законодавчих документах, активізували такі інноваційні тенденції, явища, процеси як, комп'ютеризація навчання та інтеграція академічна доброчесність, моніторинг якості освіти,; синтез науки та освіти і т.п. Реалізація зазначених новацій ставить перед науковцями й освітянами нові вимоги до впровадження інноваційних підходів до змісту і організації освітнього процесу у сучасних закладах освіти, зокрема, оновлення змісту навчання шляхом модернізації професійних освітніх стандартів, форм, методів і засобів навчання. Таким чином, виникає потреба у нових сучасних підходах навчання в системі вищої освіти для забезпечення якісної технічної підготовки майбутніх фахівців з цифрових технологій.

Зміни в освітньому законодавстві України активізували інноваційні тенденції, такі як комп'ютеризація навчання, інтеграція принципів академічної доброчесності, моніторинг якості освіти, а також поєднання науки й освіти. Це ставить перед освітянами завдання оновлення змісту навчання, модернізації професійних стандартів, методів і засобів викладання, що особливо актуально для підготовки фахівців у сфері цифрових технологій.

У зв'язку з глобалізацією та зростаючими викликами цифровізації, підготовка конкурентоспроможних фахівців стає критично важливою. Така підготовка має орієнтуватися не лише на національні, а й на глобальні стандарти, забезпечуючи майбутнім спеціалістам знання й навички, необхідні для успішної професійної діяльності на міжнародному рівні. Зважаючи на це, дедалі гостріше постає питання у необхідності професійної підготовки фахівців з цифрових технологій у системі вищої освіти [34].

У педагогічній теорії та практиці подано досить широкий спектр досліджень, у яких висвітлено різні погляди на удосконалення освітнього процесу фахівців цифрових технологій. Сучасний фахівець повинен володіти сформованими професійними уміннями і навичками, інформаційними

знаннями. Освітній процес фахівців цифрових технологій повинен враховувати вимоги ринку праці, компетентнісний та творчий підходи, спонукати до використання засобів креативного підходу для вирішення професійних ситуацій. У цьому контексті постає питання про нове бачення професійної підготовки фахівців цифрових технологій – це пояснюється тим, що для забезпечення професійної діяльності на високому професійному рівні майбутні фахівці повинні володіти сучасними інформаційно-комунікаційними технологіями [35].

Ґрунтовною основою вивчення та розв'язання проблеми дослідження теоретичних та методичних засад підготовки фахівців з цифрових та інформаційних технологій стали результати напрацювань відомих науковців з різних напрямів освіти: проблем філософії освіти (В. Андрущенко, І. Зязюн, В. Кремень, В. Лутай); системного підходу до організації освітнього процесу (В. Кузьмін, Є. Юдін та ін.); психології освіти (Г. Балл, Л. Виготський, О. Леонтьєв, Ю. Самарін, В. Семиченко, Н. Тализіна та ін.); педагогіки професійної освіти (В. Безрукова, Р. Гуревич, О. Дубасенюк, О. Дубинчук, Н. Кузьміна, Л. Лук'янова, В. Мадзігон, Н. Ничкало, Л. Оршанський, Л. Сидорчук, В. Тименко, А. Цина); структурування знань у змісті освіти (Б. Гершунський, В. Гінецинський, В. Лєдньов, О. Щербак, та ін.); інтеграції технологій в освітній процес (О. Білик, М. Корець, Д. Корчевський, М. Піддячий та ін.); застосування в освіті цифрових та інформаційних технологій (О. Авраменко, В. Биков, Т. Бодненко, Т. Вакалюк, І. Войтович, А. Гедзик, Ю. Горошко, А. Гуржій, М. Жалдак, Л. Карташова, В. Лапінський, Л. Макаренко, Н. Морзе, Ю. Рамський, С. Семеріков, О. Спірін, Г. Ткачук, Ю. Триус, В. Франчук, С. Яшанов та ін.); рівневої підготовки майбутніх інженерів-програмістів (Т. Бодненко, Р. Горбатюк, З. Сейдаметова, А. Стрюк та ін.).

Освітній процес у системі підготовки фахівців цифрових технологій має:

– враховувати вимоги ринку праці та інтегрувати компетентнісний і творчий підходи;

– забезпечувати розвиток професійних умінь, використання сучасних інформаційно-комунікаційних технологій;

– сприяти формуванню здатності до самонавчання, креативного вирішення професійних завдань, а також самоактуалізації.

Наукові дослідження підтверджують важливість системного підходу до організації освітнього процесу та інтеграції технологій у викладання. Проте комплексне дослідження педагогічної системи професійної підготовки фахівців цифрових технологій, її концептуальних засад і методологічних підходів ще не було повною мірою реалізоване.

Пріоритетним завданням вищої школи є не тільки дати знання здобувачам освіти, але й розбудити особистісний мотив до навчання, потяг до самовдосконалення, тобто навчити здобувачів вчитися. Особливої уваги потребує в цьому плані професійна підготовка майбутніх фахівців в галузі цифрових та інформаційних технологій у закладах вищої освіти. Сучасні цифрові та інформаційні технології надають нові перспективи для підвищення якості навчального процесу. Оскільки суть освіти докорінно змінюється, то методи активного пізнання та самоосвіти знаходять належну увагу у вищій школі [42].

Особливого значення на сьогодні набувають створення й реалізація цілісної системи підготовки фахівців з цифрових технологій до професійної діяльності, що здатна своєчасно, оперативно, гнучко реагувати на зміни в науці та промисловості, вимоги ринку праці, забезпечуючи високу якість результатів.

Професійна підготовка фахівців зі спеціальності 015 Професійна освіта (Цифрові технології) розглядається як процес, спрямований на набуття фахівців з цифрових технологій професійних компетентностей з удосконалення здатності до постійного оновлення знань та самоактуалізації за програмами післядипломної безперервної освіти, що дозволяє їм освоїти новий вид професійної діяльності.

Особливої уваги заслуговує проблема формування професійної компетентності для викладання інноваційних освітніх модулів, наприклад, таких як «Реалізація поліморфізму мовою C#». Це вимагає створення спеціалізованого освітнього середовища, що базується на принципах безперервної освіти. Такий підхід дозволяє забезпечити якісну підготовку майбутніх фахівців, сприяти їхньому саморозвитку, а також задовольнити вимоги ринку праці.

В даний час теоретичні та методологічні засади впровадження ідей модернізації у системі професійної підготовки фахівців з цифрових технологій розроблені недостатньо і, по суті, є малодослідженою проблемою.

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

Цілеспрямована професійна підготовка фахівців з цифрових технологій дозволяє створити умови для безперервного розвитку особистості, як суб'єкта навчання, що дасть можливість отримати гарантовані результати готовності фахівців з цифрових технологій до професійної діяльності та безперервної освіти, самоактуалізації та самовдосконалення.

Освітній процес у системі професійної підготовки фахівців з цифрових технологій покликаний сприяти розширенню пізнавальної діяльності, активізації їх інтелекту та культурних та моральних цінностей та норм поведінки, становленню особистості сучасного фахівця. Метою освіти є не лише передача здобувачам вищої освіти сукупності знань, умінь та навичок у певній сфері, а й розвиток кругозору, міждисциплінарного чуття, здатності до індивідуальних креативних рішень, самонавчання, а також формування гуманістичних цінностей [40].

Сучасний етап розвитку системи професійної підготовки є етапом аналізу та узагальнення, наукового переосмислення, використання нових методологічних підходів до дослідження проблем післядипломної освіти [37].

Загалом, удосконалення системи підготовки фахівців із цифрових технологій має включати не лише теоретичні та методичні аспекти, але й враховувати практичні виклики сучасного ринку праці, інтегруючи інноваційні підходи до викладання та навчання.

У ході дослідження уточнено та систематизовано основні поняття, що характеризують сутність та структуру процесу професійної підготовки фахівців з цифрових технологій та виділено пріоритетні напрямки удосконалення професійних компетентностей.

Отже, професійна підготовка фахівців з цифрових технологій для викладання освітнього модуля «Реалізація поліморфізму мовою C#» у закладах вищої освіти потребує вирішення проблеми, яку ми вбачаємо через теоретичне обґрунтування методики професійної підготовки.

Висновки до розділу

У кваліфікаційній роботі відповідно до мети і завдань розкрито стан наукової проблеми. У ході дослідження уточнено та систематизовано основні поняття, що характеризують сутність та структуру процесу професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація поліморфізму мовою C#» у закладах вищої освіти та виділено пріоритетні напрямки удосконалення професійних компетентностей.

З’ясовано, що професійна підготовка фахівців з цифрових технологій для викладання освітнього модуля «Реалізація поліморфізму мовою C#» у закладах вищої освіти є актуальною у професійній педагогіці.

Проаналізовано ступінь актуальності проблеми професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація поліморфізму мовою C#» у закладах вищої освіти.

Охарактеризовано систему професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація поліморфізму мовою C#» у закладах вищої освіти.

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

2.1 Аналіз силабусів освітніх компонент

На початку дослідження автором виконано аналіз освітніх компонент (ОК) вітчизняних закладів вищої освіти, присвячених вивченню основ об'єктно-орієнтованого програмування, з метою встановлення напрямків подальших досліджень.

Силабус ОК «Об'єктно-орієнтоване програмування» Київського політехнічного інституту імені І. Сікорського, під авторством Алещенко О.В. [23], визначає дисципліну як нормативну та спрямовану на професійний розвиток бакалаврів. Основна мета курсу – формування у студентів базових знань та вмінь в об’єктно-орієнтованому програмуванні (ООП). Програма охоплює вивчення методів і засобів програмування для вирішення завдань різної складності за допомогою ООП.

У лабораторному практикумі студенти повинні опанувати навички вибору мови програмування, середовища розробки, створення структури класів, написання та відлагодження коду, рефакторингу, а також оформлення документації відповідно до стандартів. Серед лабораторних робіт варто виокремити: класи, відношення між класами, наслідування, поліморфізм та роботу з колекціями на прикладі мови програмування C#.

Силабус ОК «Об'єктно-орієнтоване програмування» Харківського національного економічного університету імені Семена Кузнеця, розроблений проф. Щербаковим О.В. [24], також відносить дисципліну до нормативних. Її мета – розвинути компетентності студентів у сфері аналізу та проектування з використанням ООП у процесах комп'ютерного проектування. Програма включає засвоєння основ ООП, опанування практичних навичок розробки програм із використанням принципів ООП. Лабораторний практикум охоплює

ключові аспекти мов C# та Java, зокрема класи, повторне використання коду, реалізацію поліморфізму та шаблонів проектування.

Силабус ОК «Об'єктно-орієнтоване програмування» Черкаського державного технологічного університету під авторством Рудницького С.В. [25], орієнтований на засвоєння сучасних технологій розробки складних програмних продуктів, включаючи мобільні додатки та веб-сайти. Метою курсу є підготовка студентів до застосування інструментів і методів для розв'язання прикладних задач, забезпечення глибокого розуміння технологій програмування. Лабораторні роботи охоплюють основи ООП, класи, об’єкти, конструктори, абстрактні класи й інтерфейси.

Силабус ОК «Об'єктно-орієнтоване програмування» Львівського національного університету імені І. Франка, створений доц. Шевчук І.Б. [26], також визначає дисципліну як нормативну. Її мета – навчити студентів основам ООП та їх застосуванню в мові Java. Основні завдання: вивчення принципів проектування ПЗ, освоєння методик роботи з візуальними середовищами, розробка та тестування програм для сучасних операційних систем. Лабораторні роботи включають концепції ООП, класи й об’єкти, модифікатори доступу, інкапсуляцію, наслідування, типи та внутрішні класи.

Аналіз силабусів свідчить, що здобувачам важливо засвоїти теоретичні та практичні аспекти таких тем:

1. Класи, об'єкти, поля, властивості, модифікатори доступу.

2. Методи класів та їх використання.

3. Робота з конструкторами й деструкторами.

4. Реалізація успадкування в задачах програмування.

5. Використання поліморфізму для створення гнучкого і масштабованого програмного забезпечення.

6. Практика використання колекцій для роботи з набором даних.

7. Інкапсуляція та організація даних у програмах для забезпечення їх безпеки та ефективності.

8. Розробка програм із урахуванням принципів абстракції та модульності, що забезпечує легкість у підтримці та розширенні ПЗ.

У подальшому матеріалі наведено етапи розробки нового лабораторного практикуму «Реалізація поліморфізму мовою C#», який акцентує увагу здобувачів на прийомах приховування полів та методів у похідному класі, реалізації поліморфізму, роботі з віртуальними методами та абстрактними класами.

2.2 Постановка лабораторної роботи «Приховування полів, властивостей та методів у класах»

Мета виконання роботи: отримання навичок модернізації та налаштування об’єктно-орієнтованих ієрархій класів з використанням операції приховання елементів даних та функціональних елементів класів.

Постановка завдання лабораторної роботи.

1. Згідно індивідуальному варіанту обрати предметну область. Визначити базовий клас у відповідності з індивідуальним варіантом опису предметної області. Додати до класу хоча б три поля, визначити їх області видимості як приватні, передбачити ініціалізацію значень, якщо це можливо, створити публічні властивості Set і Get для доступу до приватних полів класу (замість приватних полів можна створити автореалізовані властивості.). Додати до базового класу хоча б два методи для демонстрації можливостей класу. Додати до класу конструктор за замовчуванням, конструктор хоча б з одним параметром, та деструктор. Створити та додати до ієрархії всі необхідні класи, що є похідними від базового класу. Визначити поля, властивості та методи похідних класів, конструктори і деструктори.

2. Додати до ієрархії хоча б один похідний клас з власними полями та методами.

3. Виконати приховування у похідному класі публічних елементів базового класу за допомогою ключового слова «new»: одного поля, однієї

властивості та одного методу. У реалізації прихованого методу звернеться до базового методу за допомогою ключового слова «base».

4. У головній точці входу «main()» виконайте створення об'єктів базового та похідного класів. Звернеться до публічного поля базового класу та до її прихованої версії. Звернеться до публічної властивості базового класу та до її прихованої версії. Продемонструйте виклик методу базового класу та прихованого методу похідного класу. Виведіть результати в консоль, щоб продемонструвати особливості механізму приховання елементів класу.

5. Створити UML-діаграму класу засобами онлайн-сервісу «Draw.io» з використанням мови PlantUML. Клієнтський код (клас Program та метод static void Main) на діаграмі не показувати.

6. Підготувати звіт до захисту лабораторної роботи, що містить:

– опис предметної області;

– UML-діаграму ієрархії класів;

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

– програмний код мовою C#;

– скриншот з результатами роботи програми.

Приклад виконання лабораторної роботи. Опис предметної області: виконати програмну реалізацію ієрархії геометричних фігур, а саме кругів та секторів. Круг представляється через радіус та координати центра, а сектор є частиною круга, визначеною центральним кутом. Основними операціями є обчислення площ цих фігур. На рис. 2.1 наведено UML-діаграму ієрархії класів предметної області.

Опис класів та їх атрибутів наведено далі.

1. Клас Circle (Круг) є базовим для інших фігур круга і містить такі атрибути. Публічне поле type – константний рядок, який визначає тип фігури як "circle". Властивості Radius, X, Y: радіус круга та координати центра круга. Конструктор за замовчуванням встановлює значення радіуса та координат центра на 0.0. Конструктор з параметрами приймає значення для радіуса та

координат центра. Метод CalcArea(): обчислює площу круга. Деструктор виконує дії при завершенні життєвого циклу об’єкта класу.

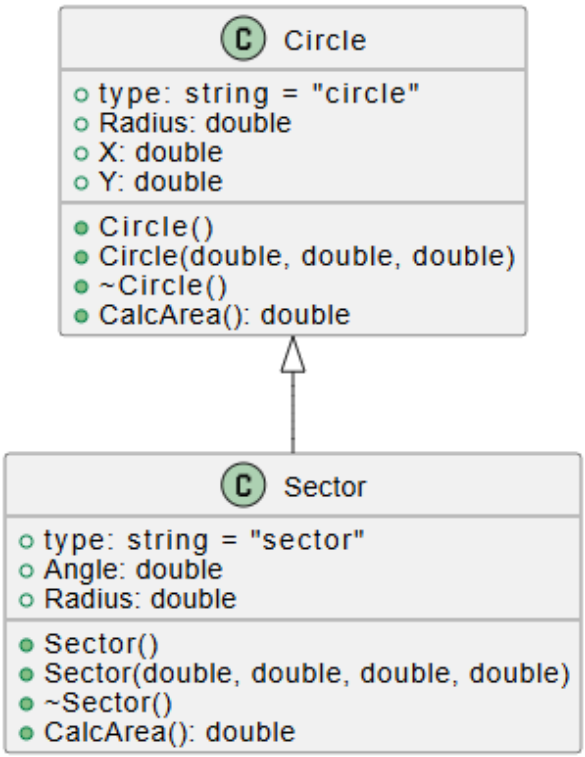

Рисунок 2.1 – UML-діаграма ієрархії класів

2. Клас Sector (Сектор) успадковується від Circle, використовуючи поліморфізм, та додає функціональність для роботи із сектором круга. Атрибути класу: приховане поле type містить константний рядок, який визначає тип фігури як "sector". Властивість Radius приховує успадковану властивість базового класу, але зі зміненим доступом – додає контроль допустимості значень (від 0 до 1000). Нова властивість Angle визначає кут сектора у градусах. Конструктор за замовчуванням встановлює значення для кута сектора на 0.0 та ініціалізує значення за замовчуванням з базового класу. Конструктор з параметрами приймає значення для радіуса, кута, та координат центра. Метод CalcArea() приховує успадкований метод базового класу та обчислює площу сектора Деструктор виконує дії при завершенні життєвого циклу об’єкта класу.

3. В головному класі Program створюються об'єкти класів Circle і Sector та демонструється робота з ними.

Програмний код програми на мові C#:

```
namespace Polymorph
{
    // базовий клас Circle
    class Circle
    {
        // публічне поле – тип фігури
        public readonly string type = "circle";

        // Автореалізовані властивості
        public double Radius { get; set; } // радіус
        public double X { get; set; }      // координата x центру
        public double Y { get; set; }      // координата y центру

        // Конструктор за замовчуванням
        public Circle()
        {
            Radius = 0.0d; // ініціалізація радіусу
            X = 0.0d;      // ініціалізація координати x
            Y = 0.0d;      // ініціалізація координати y
            Console.WriteLine("Конструктор за замовчуванням викликано");
        }

        // Конструктор зі всіма параметрами
        public Circle(double radius, double x, double y)
        {
            Radius = radius;
            X = x;
            Y = y;
            Console.WriteLine("Конструктор з параметрами 'radius', " +
                "'x', 'y' викликано");
        }

        ~Circle()  // Деструктор
        {
            Console.WriteLine("Деструктор викликано");
        }

        // Метод обчислення площі кругу
        public double CalcArea()
        {
            return Math.PI * Radius * Radius;
        }
    }

    class Sector : Circle
    {
        // приховування поля type
        public new readonly string type = "sector";

        // приховування властивості Radius
        public new double Radius
        {
            get
            {
                return base.Radius;
            }
            set
            {
                // контроль коректності значення
                if (value < 0 || value > 1000)
                {
                    base.Radius = 0;
                }
```

```
                else
                {
                    base.Radius = value;
                }
            }
        }

        // Нова автореалізована властивість, величина кута сектора в градусах
        public double Angle { get; set; }

        // Конструктор за замовчуванням
        public Sector() : base()
        {
            Angle = 0.0;
            Console.WriteLine("Конструктор за замовчуванням для " +
                "Sector викликано");
        }

        // Конструктор зі всіма параметрами
        public Sector(double radius, double angle, double x, double y)
        {
            X = x;
            Y = y;
            Angle = angle;
            // звернення до властивості з контролем коректності
            Radius = radius;
            Console.WriteLine("Конструктор з параметрами 'radius', 'angle', " +
                "'x', 'y' для Sector викликано");
        }

        // Приховування методу обчислення площі сектора
        public new double CalcArea()
        {
            return base.CalcArea() * (Angle / 360.0);
        }

        ~Sector()  // Деструктор
        {
            Console.WriteLine("Деструктор викликано");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            // підтримка українських літер
            Console.OutputEncoding = Encoding.UTF8;

            // створюємо об'єкт класу Circle з некоректним значенням радіусу
            Circle circle1 = new Circle(-20, 15, 25);
            Console.WriteLine("Об'єкт circle2: площа = {0:f3}, радіус = {1:f3}, " +
                "X = {2:f3}, Y = {3:f3}, тип = {4}", circle1.CalcArea(),
                circle1.Radius, circle1.X, circle1.Y, circle1.type);

            // створюємо об'єкт класу Sector з некоректним значенням радіусу
            Sector sector1 = new Sector(-20, 90, 100, 200);
            Console.WriteLine("Об'єкт sector1: площа = {0:f3}, радіус = {1:f3}, " +
                "X = {2:f3}, Y = {3:f3}, кут = {4:f3}, тип = {5}",
                sector1.CalcArea(), sector1.Radius, sector1.X, sector1.Y,
                sector1.Angle, sector1.type);

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

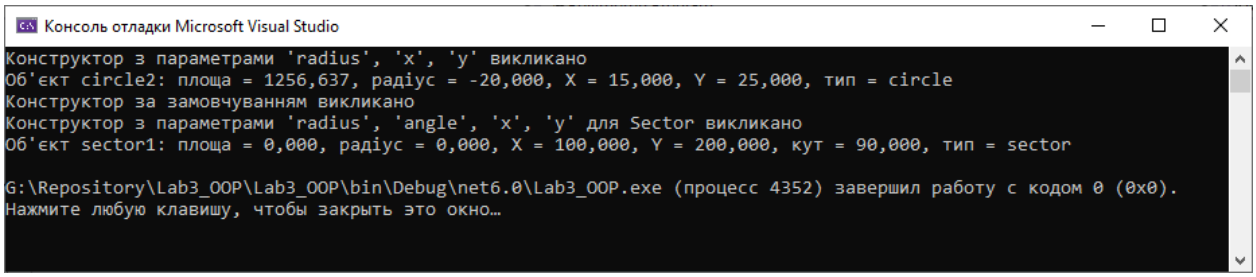

Рисунок 2.2 – Результати роботи програми

2.3 Постановка лабораторної роботи «Віртуальні методи та поліморфізм у класах»

Мета виконання роботи: отримання навичок модернізації та налаштування об’єктно-орієнтованих ієрархій класів з реалізацією поліморфізму класів за допомогою віртуальних методів та їх перевизначення у похідних класах.

Постановка завдання лабораторної роботи.

1. Виконати модернізацію програмної коду з лабораторної роботи «Приховування полів, властивостей та методів у класах» таким чином. У базовому класі один з методів визначити, як віртуальний за допомогою ключового слова «virtual». Для кожного класу-нащадку виконати перевизначення цього методу за допомогою ключового слова «override». Функціонал перевизначених методів повинен відрізнятися від базового методу. У реалізації перевизначеного методу необхідно звернутися до віртуального методу за допомогою ключового слова «base».

2. Застосувати два рівні доступності класів: загальний (public) та внутрішній (internal).

3. Дослідити модифікатори доступу елементів класу, такі як public, protected, private, і застосувати їх для демонстрації різних рівнів доступу.

4. У головній точці входу «main()» виконайте створення об'єктів базового та похідного класів. Виконайте виклик віртуального методу базового класу та перевизначеного методу похідного класу.

5. Створити UML-діаграму класу засобами онлайн-сервісу «Draw.io» з використанням мови PlantUML. Клієнтський код (клас Program та метод static void Main) на діаграмі не показувати.

6. Підготувати звіт до захисту лабораторної роботи, що містить:

– опис предметної області;

– UML-діаграму ієрархії класів;

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

– програмний код мовою C#;

– скриншот з результатами роботи програми.

Приклад виконання лабораторної роботи. Опис предметної області: виконати програмну реалізацію ієрархії класів тварин, зокрема хижаків, та їхньої поведінки з механізмами як наслідування, поліморфізму та доступу до методів і атрибутів у базових і похідних класах. Реалізувати базовий клас «Хижак» та похідний клас «Собака», де кожен з них має певний набір властивостей і методів, що визначають їхню поведінку.

На рис. 2.3 наведено UML-діаграму ієрархії класів предметної області.

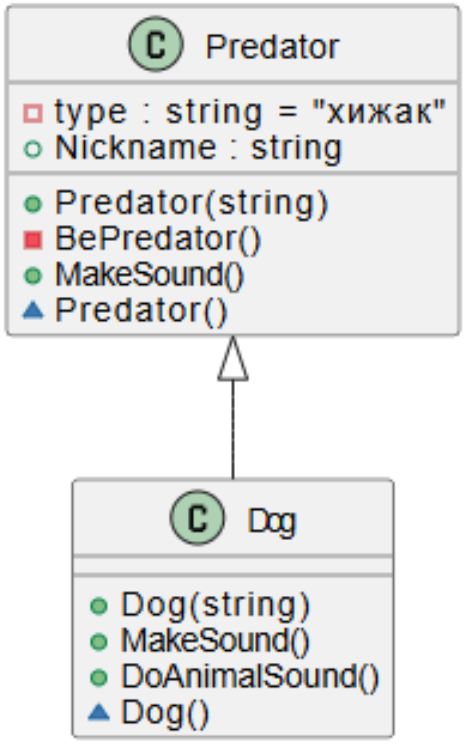

Рисунок 2.3 – UML-діаграма ієрархії класів

Опис класів та їх атрибутів наведено далі.

1. Клас Predator є базовим для представлення характеристик загального хижого об'єкта. Він має такі атрибути: поле type доступне у класі та його нащадках, за замовчуванням має значення «хижак», Nickname – властивість для отримання та встановлення прізвиська тварини, конструктор Predator() приймає прізвисько, встановлює його у властивість Nickname та виводить повідомлення, що об'єкт створено, ~Predator() – деструктор класу, який викликається під час знищення об'єкта та виводить повідомлення про видалення хижака, BePredator() – приватний метод, який недоступний у похідних класах і виводить повідомлення, що тільки цей хижак може жити в лісі, MakeSound() – віртуальний метод, який може бути перевизначений у похідних класах, виводить звук, який видає хижак, і викликає метод BePredator().

2. Клас Dog клас наслідує Predator і представляє більш конкретний вид тварини – «Собака». Він має такі атрибути: використовує Nickname та type з базового класу Predator, Dog(string nickname) – конструктор, який викликає конструктор базового класу Predator, встановлює тип тварини як "домашня тварина" і виводить повідомлення про створення собаки, ~Dog() – деструктор, який виводить повідомлення про знищення об'єкта Dog, MakeSound() – перевизначений метод, який видає звук, властивий собаці, DoAnimalSound() – метод, який викликає базовий метод MakeSound за допомогою ключового слова base та виводить додаткове повідомлення.

3. Клас Program є точкою входу в програму і використовується для створення об'єктів класів Predator і Dog та виклику їх методів у межах Main. Метод Main встановлює кодування консолі для підтримки українських літер, створює об'єкт класу Predator з прізвиськом і викликає його метод MakeSound, створює об'єкт класу Dog і викликає методи MakeSound та DoAnimalSound.

Програмний код програми на мові C#:

```
using System;
using System.Text;

// Базовий клас "Хижак"
```

```
public class Predator
{
    protected string type = "хижак";
   // прізвисько
    public string Nickname
    {
        get;
        set;
    }

    // конструктор класу
    public Predator(string nickname)
    {
        Nickname = nickname;
        Console.WriteLine($"{Nickname} - {type}!");
    }

    // Приватний метод не доступний у похідному класі,
    // доступний лише власним методам
    private void BePredator()
    {
        Console.WriteLine($"Лише {Nickname} може жити у лісі");
    }

    // Віртуальний метод "Голос"
    public virtual void MakeSound()
    {
        Console.WriteLine($"{Nickname} говорить: гррррр!");
        BePredator();
    }

    // деструктор класу за замовчуванням
    ~Predator()
    {
        Console.WriteLine("Хижака знищено");
    }
}

// Клас "Собака" наслідує базовий клас "Тварина"
internal class Dog : Predator
{

    public Dog(string nickname) : base(nickname)
    {
        // доступ до поля protected із похідного класу
        Nickname=nickname;
        type = "домашня тварина";
        Console.WriteLine($"{Nickname} - {type}!");
    }

    ~Dog()
    {
        Console.WriteLine("Собаку знищено");
    }

    // Перевизначення методу "Голос" для собаки
    public override void MakeSound()
    {
        Console.WriteLine($"{Nickname} гавкає: Гав-гав!");
    }

    // Метод, який викликає віртуальний метод базового класу
    public void DoAnimalSound()
    {
        Console.WriteLine($"{Nickname} говорить: я нащадок хижаків");
```

```
        // Виклик віртуального методу базового класу за допомогою "base"
        base.MakeSound();
    }
}

class Program
{
    static void Main(string[] args)
    {
        Console.OutputEncoding = UTF8Encoding.UTF8; // підтримка укр. літер

        // Створення об'єкта класу "Predator"
        Predator predator1 = new Predator("Вбивця");
        predator1.MakeSound();

        // прямий доступ до поля protected заборонено
        // predator1.type = "домашня тварина";

        // Створення об'єкта класу "Собака"
        Dog dog1 = new Dog("Барсик");

        // Виклик методу "MakeSound" класу "Собака"
        dog1.MakeSound();

        // Виклик базового методу "DoAnimalSound"
        dog1.DoAnimalSound();

        // прямий доступ до поля private заборонено
        // dog1.BePredator();
    }
}
```

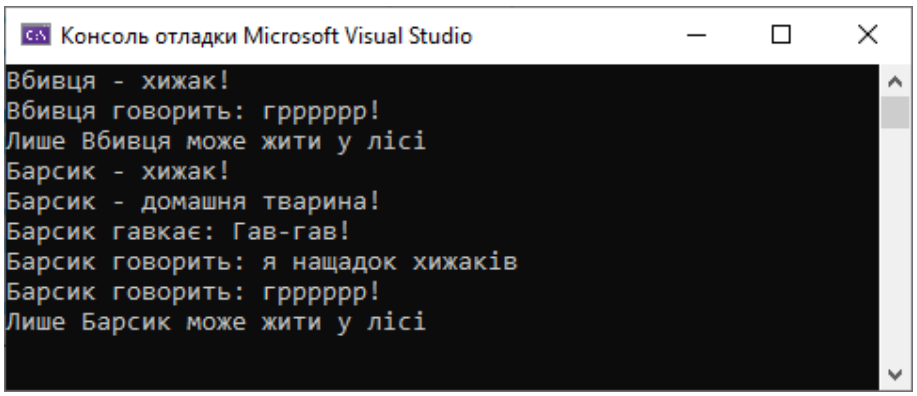

Рисунок 2.4 – Результати роботи програми

2.4 Постановка лабораторної роботи «Абстрактні, статичні та запечатані класи»

Мета виконання роботи: отримання навичок модернізації та налаштування об’єктно-орієнтованих ієрархій класів з реалізацією абстрактних класів та елементів класів, статичних класів та елементів класів, запечатаних класів.

Постановка завдання лабораторної роботи.

1. Визначити базовий абстрактний клас у відповідності з індивідуальним варіантом опису предметної області. Додати до класу неабстрактні підтримуюче поле та властивість, абстрактні властивість та метод, неабстрактні метод та конструктор.

2. Створити запечатаний клас-нащадок абстрактного класу, реалізувати в ньому абстрактні властивість та метод.

3. Додати до ієрархії класів статичний клас зі статичними полем, властивістю та методом.

5. У головній точці входу «main()» продемонструвати створення об’єктів базового та похідного класів, виклик базового та перевизначеного методів, звернення до статичного поля, виклик статичного методу.

6. Створити UML-діаграму класу засобами онлайн-сервісу «Draw.io» з використанням мови PlantUML. Клієнтський код (клас Program та метод static void Main) на діаграмі не показувати.

7. Підготувати звіт до захисту лабораторної роботи, що містить:

– опис предметної області;

– UML-діаграму ієрархії класів;

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

– програмний код мовою C#;

– скриншот з результатами роботи програми.

Теоретичні данні.

1. У мові програмування C# абстрактні класи оголошують шляхом розміщення ключового слова «abstract» перед ім’ям класу. Не можна створити екземпляр абстрактного класу викликом конструктора чи іншим шляхом. Він існує тільки для того, щоб слугувати базою для інших класів.

Абстрактні класи можуть містити абстрактні методи та властивості. Абстрактні класи можуть також містити звичайні методи та властивості з програмною реалізацією. Ці функціональні елементи можуть бути успадковані

та використані похідними класами без змін.

Абстрактні методи оголошуються з використанням ключового слова «abstract» і вони не мають тіла, тобто програмної реалізації. Вони лише оголошують сигнатуру методу, яку похідні класи мають реалізувати.

Похідний клас, який успадковує абстрактний клас, зобов'язаний надати реалізацію для всіх абстрактних методів і властивостей базового класу, інакше цей клас компілятор буде вважати теж абстрактним.

Абстрактні класи можуть також містити поля і конструктори.

2. У мові програмування C# статичними можуть бути: класи, методи, поля, властивості. Щоб клас або елемент класу був статичним, перед його оголошенням ставиться ключове слово «static».

Неможна створювати об'єкти статичного класу викликом конструктора чи іншим шляхом. Статичний клас повинен містити тільки статичні члени (поля, властивості, методи).

У будь-якому нестатичному класі може бути оголошено як статичні методи, так і нестатичні. Відмінність між викликом статичного та нестатичного методу класу полягає в наступному: щоб викликати нестатичний метод класу, потрібно створити екземпляр цього класу. Статичний метод викликається без створення об'єкту класу – перед іменем методу вказується ім'я класу, в якому цей статичний метод оголошений.

3. У мові програмування C# запечатані класи оголошують шляхом розміщення ключового слова «sealed» перед ім'ям класу. Запечатаний клас не може використовуватися як базовий клас. Тому він також може бути абстрактним класом. Запечатані класи запобігають наслідуванню. Оскільки їх не можна використовувати як базові класи, певна оптимізація під час виконання дозволяє дещо прискорити виклик членів запечатаних класів.

Приклад виконання лабораторної роботи. Опис предметної області: виконати програмну реалізацію ієрархії класів тварин, зокрема хижаків, та їхньої поведінки з механізмами як наслідування, поліморфізму та доступу до методів і атрибутів у базових і похідних класах.

Код UML-діаграми ієрархії класів на мові PlantUML:

```
@startuml

abstract class Animal
{
    - type : string = "Тварина"
    + Type : string { get; set }
    + {abstract} NickName: string { get; set }
    + Animal(string)
    + {abstract} MakeSound()
    + BePredator()
}

static class God
{
    + {static} name: string = "Бог"
    + {static} Gender: string { get }
    + {static} BeGod()
}


class Kitten << final >>
{
    + numOfPaws : int
    - nickName : string = "кіт"
    + NickName : string
    + Kitten(string, string, int)
    + MakeSound()
}

Animal <|-- Kitten

@enduml
```

На рис. 2.5 наведено UML-діаграму ієрархії класів предметної області.

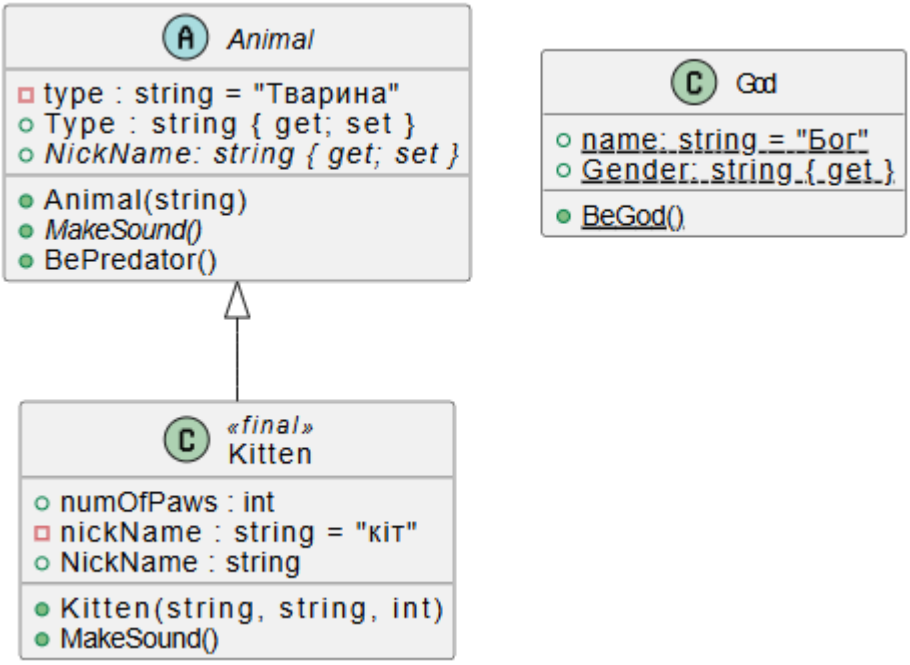

Рисунок 2.5 – UML-діаграма ієрархії класів

Опис класів та їх атрибутів наведено далі.

1. Абстрактний клас Animal є базовим класом у ієрархії. Атрибути класу: type – це поле, яке містить тип тварини, Type – неабстрактна властивість для отримання та встановлення значення поля type, NickName – абстрактна властивість для отримання та встановлення прізвиська тварини. Конструктор Animal(type : string), який ініціалізує значення властивості Type. Абстрактний метод BePredator(), який має бути реалізований у похідних класах для відтворення звуку тварини.

2. Статичний клас God має статичне поле name, яке містить ім'я Бога, статична властивість Gender повертає гендер Бога. Статичний метод BeGod() виводить повідомлення про любов Бога до всіх.

3. Запечатаний клас (не можна успадковувати) Kitten має поле numOfPaws, яке містить кількість лап кошеня, поле nickName, яке містить прізвисько кошеня. Реалізація абстрактної властивості NickName контролює коректність отриманого значення. Конструктор Kitten(string, string, int) ініціалізує значення полів numOfPaws і nickName, а також виводить інформацію про кошеня. Реалізація абстрактного методу MakeSound() виводить звук, що муркоче кошеня.

Програмний код програми на мові C#:

```
using System;
using System.Text;

// Абстрактний клас "Тварина"
internal abstract class Animal
{
    // не-абстрактне поле
    private string type = "Тварина";

    // спроба оголосити абстрактне поле викликає помилку компілятора
    // abstract private string nickname;

    // не-абстрактна властивість
    public string Type
    {
        get
        {
            return type;
        }

        set
        {
            type = value;
```

```
        }
    }

    // абстрактна властивість "призвісько"
     abstract public string NickName
    {
        get;
        set;
    }

    // не-абстрактний конструктор
    public Animal(string type)
    {
        Type = type;
    }

    // Абстрактний метод
    public abstract void MakeSound();

    // Не-абстрактний метод
    public void BePredator()
    {
        Console.WriteLine("Хижаки завжди харчуються м'ясом!");
    }
}

// Статичний клас "Бог"
internal static class God
{
    // статичне поле
    public static string name = "Бог";

    // статична властивість
    public static string Gender
    {
        get
        {
            return "невизначений";
        }
    }

    // статичний метод
    public static void BeGod()
    {
        Console.WriteLine($"Бог нас всіх любить!");
    }
}

// Запечатаний клас "Kitten"
internal sealed class Kitten : Animal
{
    // власне поле - кількість лап
    public int numOfPaws;
    // власне поле - прізвисько
    public string nickName="кіт";
    // реалізація базової абстрактної властивості
    public override string NickName
    {
        get
        {
            return nickName;
        }
        set
        {
            // контроль коректності отриманого значення
```

```
            if (value == "Васька" || value == "Аліса")
            {
                nickName = "Барсік";
            }
            else
            {
                nickName = value;
            }
        }
    }
    // конструктор класа Kitten
    public Kitten(string type, string nickname, int numofpaws)
        : base(type) // виклик базового неабстрактного конструктора
    {
        // ініціалізація власного поля
        numOfPaws = numofpaws;
        NickName = nickname;
        Console.WriteLine($"{NickName} - це маленьке {type}, " +
            $"у нього {numOfPaws} лапи та немає нащадків!");
    }

    // Перевизначення абстрактного методу "MakeSound" для кошеня
    public override void MakeSound()
    {
        Console.WriteLine($"{NickName} муркоче: Мяу-мяу!");
    }
}

class Program
{
    static void Main(string[] args)
    {
        Console.OutputEncoding = UTF8Encoding.UTF8; // підтримка укр. літер

        // Створення об'єкта класу "Kitten" з некоректним значенням "Аліса"
        // для властивості NickName
        Kitten myKitten = new Kitten("Кошеня", "Аліса", 4);

        // Виклик методу "MakeSound" класу "Кошеня"
        myKitten.MakeSound();

        // Виклик успадкованого не-абстрактного методу "BePredator"
        Console.Write($"{myKitten.NickName} - нащадок хижаків, тому ... ");
        myKitten.BePredator();

        // Звернення до статичного поля
        Console.WriteLine($"Верховна духовна Сутність - це {God.name}.");
        Console.WriteLine($"Його гендер - {God.Gender}.");

        // Звернення до статичного методу
        God.BeGod();
    }
}
```

```
Select Консоль отладки Microsoft Visual Studio
Барсік - це маленьке Кошеня, у нього 4 лапи та немає нащадків!
Барсік муркоче: Мяу-мяу!
Барсік - нащадок хижаків, тому ... Хижаки завжди харчуються м'ясом!
Верховна духовна Сутність - це Бог.
Його гендер - невизначений.
Бог нас всіх любить!
```

Рисунок 2.6 – Результати роботи програми

2.5 Постановка лабораторної роботи «Перевизначення операцій для класів»

Мета виконання роботи: отримання навичок модернізації та налаштування об’єктно-орієнтованих ієрархій класів з реалізацією абстрактних класів та елементів класів, статичних класів та елементів класів, запечатаних класів.

Постановка завдання лабораторної роботи.

1. У базовому класі індивідуального варіанту перевизначити операції «true, false, &, !, +, –».

2. Створити запечатаний клас-нащадок абстрактного класу, реалізувати в ньому абстрактні властивість та метод.

3. У головній точці входу «main()» продемонструвати створення об’єктів базового класу та використання перевизначених операцій.

4. Створити UML-діаграму класу засобами онлайн-сервісу «Draw.io» з використанням мови PlantUML. Клієнтський код (клас Program та метод static void Main) на діаграмі не показувати.

5. Підготувати звіт до захисту лабораторної роботи, що містить:

– опис предметної області;

– UML-діаграму ієрархії класів;

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

– програмний код мовою C#;

– скриншот з результатами роботи програми.

Теоретичні данні. Перевизначення операцій в C# дозволяє змінювати поведінку стандартних операторів для конкретних класів або структур. Це корисно, коли потрібно виконувати операції над об'єктами користувацьких типів даних, наприклад, додавання двох векторів або порівняння двох об'єктів.

Щоб перевизначити оператор, потрібно:

1. використати ключове слово public static;

2. вказати тип повернення оператора;

3. використати ключове слово operator з оператором, який потрібно перевизначити (наприклад, +, –, ==, != тощо);

4. визначити вхідні параметри та реалізацію оператора.

Не всі операції можуть бути перевантажені. Існують певні правила та обмеження на перевантаження операцій, які показано у табл. 2.1

Перевантаження дозволяє інтуїтивно використовувати звичні оператори, застосовуючи їх до об'єктів класів, і при цьому розширювати логіку цих об'єктів безпосередньо через оператори, роблячи код більш зрозумілим і зручним для користування.

Таблиця 2.1

Перелік операцій, які підлягають або не підлягають перевантаженню

Операція	Можливості та особливості перевантаження
+, –, !, ~, + +, – –, true, false	Унарні символи операцій допускають навантаження. Значення true та false також є операціями
+, –, *, /, %, &, |, ^, <<, >>	Бінарні символи операцій, що допускають перевантаження
==, !=, <, >, <=, >=	Операції порівняння перевантажуються
&&, ||	Умовні логічні операції моделюються з допомогою раніше перевизначених операцій «&» та «|»
[]	Операції доступу до елементів масивів моделюються за рахунок індексаторів
+=, –=, *=, /=, %=, &=, =, ^=, <<=, >>=	Операції не перевантажуються через неможливість перевантаження операції присвоєння
=, ., ?:, –>, new, is, sizeof, typeof	Операції, що не підлягають перевантаженню

Приклад виконання лабораторної роботи. Опис предметної області: виконати програмну реалізацію ієрархії класів тварин та їхньої поведінки з механізмами як наслідування, поліморфізму та доступу до методів і атрибутів у базових і похідних класах.

На рис. 2.7 наведено UML-діаграму ієрархії класів предметної області.

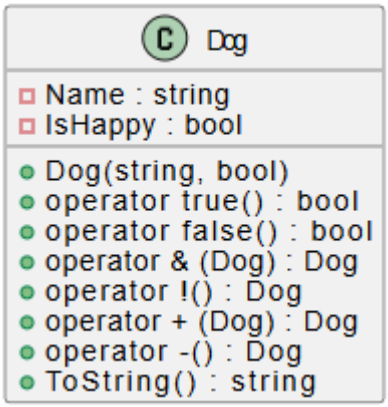

Рисунок 2.7 – UML-діаграма ієрархії класів

Опис класів та їх атрибутів наведено далі.

Клас Dog: модель собаки, яка має атрибути: Name – ім'я собаки, IsHappy – стан щастя собаки (щаслива чи ні). Методи класу включають перевантаження операторів true, false, &, !, +, і -. Оператор «+» об’єднує імена собак і щастя, задаючи новий стан, якщо хоча б одна собака щаслива. Оператор «–» змінює стан щастя собаки на нещасливий і додає префікс до імені. Оператор «!» інвертує стан щастя і змінює ім'я собаки.

Клас Program містить метод Main, що є точкою входу для тестування класу Dog.

Програмний код програми на мові C#:

```
using System;
using System.Text;

class Dog
{
    public string Name { get; set; }
    public bool IsHappy { get; set; }

    public Dog(string name, bool isHappy)
    {
        Name = name;
        IsHappy = isHappy;
    }

    // Перевизначення операції true: повертає true, якщо собака щаслива
    public static bool operator true(Dog dog)
    {
        return dog.IsHappy;
    }

    // Перевизначення операції false: повертає true, якщо собака нещаслива
    public static bool operator false(Dog dog)
    {
        return !dog.IsHappy;
```

```
    }

    // Перевизначення операції &: об'єднує двох собак з іменами та щастям
    public static Dog operator &(Dog dog1, Dog dog2)
    {
        bool result = dog1.IsHappy && dog2.IsHappy;
        return new Dog($"Об'єднаний_{dog1.Name}_{dog2.Name}", result);
    }

    // Перевизначення операції !: створює нову собаку з протилежним станом щастя
    public static Dog operator !(Dog dog)
    {
        return new Dog($"Не_{dog.Name}", !dog.IsHappy);
    }

    // Перевантаження операції +: об'єднує імена собак
    // і зберігає щасливий стан, якщо хоча б одна щаслива
    public static Dog operator +(Dog dog1, Dog dog2)
    {
        return new Dog($"{dog1.Name}_{dog2.Name}", dog1.IsHappy || dog2.IsHappy);
    }

    // Перевантаження операції -: створює нову собаку з ім'ям,
    // що містить префікс Unhappy_, і встановлює нещасливий стан
    public static Dog operator -(Dog dog)
    {
        return new Dog($"Нещасний_{dog.Name}", false);
    }

    // Метод ToString для зручного виведення стану щастя собаки у текстовому форматі
    public override string ToString()
    {
        return IsHappy ? "Щасливий" : "Нещасний";
    }
}

class Program
{
    static void Main()
    {
        Dog dog1 = new Dog("Бадді", true);
        Dog dog2 = new Dog("Макс", false);

        Console.OutputEncoding = UTF8Encoding.UTF8; // підтримка укр. літер
        // Використання перевантажених операцій
        Console.WriteLine($"{dog1.Name} є щасливим: {dog1}");
        Console.WriteLine($"{dog2.Name} є щасливим: {dog2}");

        Dog combinedDog = dog1 & dog2;
        Console.WriteLine($"Об'єднаний пес: {combinedDog.Name} " +
            $"є щасливим: {combinedDog}");

        Dog notHappyDog = !dog1;
        Console.WriteLine($"Не {dog1.Name}: {notHappyDog.Name} " +
            $"є щасливим: {notHappyDog}");

        Dog addedDogs = dog1 + dog2;
        Console.WriteLine($"Додані пси: {addedDogs.Name} " +
            $"є щасливим: {addedDogs}");

        Dog unhappyDog = -dog1;
        Console.WriteLine($"Нещасний {dog1.Name}: {unhappyDog.Name} " +
            $"є щасливим: {unhappyDog}");
    }
}
```

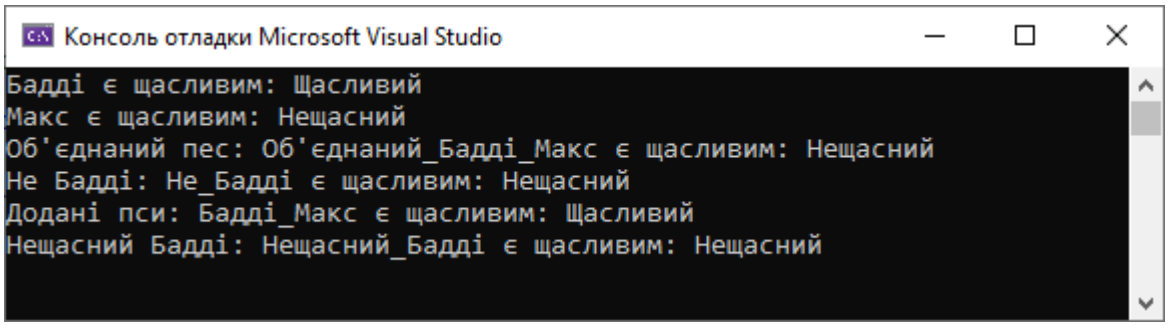

Рисунок 2.8 – Результати роботи програми

Висновки до розділу 2

У розділі виконано аналіз освітніх компонент вітчизняних закладів вищої освіти, присвячених вивченню основ об'єктно-орієнтованого програмування, з метою встановлення напрямків подальших досліджень.

За результатами виконаного дослідження виконана постановка завдань нового лабораторного практикуму «Реалізація поліморфізму мовою C#», який акцентує увагу здобувачів на побудові об’єктно-орієнтованої ієрархії класів з використанням операцій приховання елементів класів, реалізації поліморфізму за допомогою віртуальних методів, використанню різних видів класів:

1. приховування полів, властивостей та методів у класах;
2. віртуальні методи та поліморфізм у класах;
3. абстрактні, статичні та запечатані класи;
4. перевизначення операцій для класів.

Розроблено завдання лабораторних робіт, приклади виконання робіт та оформлення звітів, що містять опис предметної області, UML-діаграми ієрархії класів, опис класів та їх атрибутів, програмний код на мові C#, приклад виконання програми у середовищі Visual Studio.

РОЗДІЛ 3 ВИМОГИ ДО КАДРОВОГО ЗАБЕЗПЕЧЕННЯ ОБ'ЄКТУ ГАЛУЗІ

Об'єктно-орієнтоване програмування (ООП) стало невід'ємною частиною сучасного світу інформаційних технологій. Забезпечити високоякісну підготовку викладачів з ООП стає ключовим завданням в освітньому процесі. Освіта у цьому напрямку вимагає не лише глибоких знань від викладачів, але й певних особливостей, які відображають вимоги сучасного інформаційного суспільства [9, 13-14].

ООП революціонізувало спосіб написання програмного забезпечення. Від великомасштабних корпоративних додатків до невеликих хобі-проектів, ООП забезпечує потужний і гнучкий підхід до розробки програмного забезпечення. Це дозволяє розробникам мислити в термінах об’єктів, класів і методів, роблячи код більш інтуїтивно зрозумілим і простішим в обслуговуванні. ООП допомагає розробникам створювати ефективний і багаторазово використовуваний код, скорочуючи час і вартість розробки. Крім того, ООП дозволяє розробникам додавати нові функції та функції до існуючого коду, що робить його ідеальним вибором для гнучкої розробки. Завдяки своїм потужним функціям і гнучкості ООП став важливим інструментом для сучасної розробки програмного забезпечення [1].

Забезпечення кадрового підготовки фахівців з цифрових технологій для викладання освітнього модулю «Об'єктно-орієнтоване програмування» в закладах фахової передвищої освіти передбачає виконання ряду вимог і кроків.

Підготовка викладачів з об'єктно-орієнтованого програмування (ООП) є ключовою для забезпечення ефективного викладання цієї складної області програмування. Викладачі повинні володіти не тільки концепціями ООП, але і знати мови програмування, які активно використовують об'єктно-орієнтований підхід, такі як Java, Python, C++, C#, або Kotlin. Викладачі

повинні розумітися на сучасних тенденцій у світі програмування та цифрових технологій [12].

Викладачі з ООП повинні володіти не лише теоретичними аспектами програмування, а й активно використовувати практичні вправи для закріплення знань. Однією з особливостей є необхідність не тільки розуміти основні концепції ООП, такі як спадкування чи інкапсуляція, але і бути здатними пояснити їх студентам простим та доступним мовою способом.

Важливо враховувати, що викладання ООП повинно бути орієнтоване на практичне застосування. Це означає, що викладачі повинні мати практичний досвід у розробці програмного забезпечення, що використовує принципи ООП. Цей досвід дозволяє їм краще розуміти вимоги ринку праці та враховувати їх у навчальних програмах. Викладачам необхідний досвід роботи на практиці у сфері розробки програмного забезпечення, особливо з використанням ООП [17].

При розробці нормативно-методичної документації необхідна актуалізація змісту освітніх програм підготовки, зокрема включення актуальних тем і завдань з області ООП, забезпечення реальних прикладів та проектів, що відображають сучасні вимоги ринку праці тощо.

Викладачі мають розробляти лабораторні заняття та проекти, які дозволяють здобувачам вищої освіти отримати практичні навички в ООП.

Застосування інтерактивних методів викладання стає ще однією важливою особливістю. Використання онлайн-ресурсів, візуалізації та інтерактивних завдань допомагає студентам краще засвоювати матеріал та розвивати практичні навички. Такий підхід також відповідає сучасній тенденції активного залучення студентів до навчання [27].

Під час організації навчального процесу підготовки фахівців з цифрових технологій необхідно використання сучасних засобів навчання, зокрема інтерактивних, а також необхідно впровадження в інструкційний процес інтерактивних та візуалізаційних інструментів для полегшення засвоєння матеріалу. Необхідно застосування різноманітних методів навчання,

включаючи лекції, практичні заняття, лабораторні роботи та проекти, застосування графічних та візуалізаційних засобів для пояснення основних концепцій ООП [31].

У сучасному світі важко переоцінити роль викладачів як наставників і менторів. Вони повинні не лише передавати знання, але й сприяти розвитку креативності та критичного мислення. Взаємодія зі студентами в рамках лабораторних робіт, проектів та групових завдань сприяє створенню сприятливого навчального середовища [28].

Для розвитку самопізнання необхідно залучати здобувачів до роботи з відкритим програмним забезпеченням та розвитку власних проектів, для відслідкування за останніми тенденціями у світі ООП та цифрових технологій.

Впродовж життя необхідний постійний професійний розвиток викладачів: участь у тренінгах та конференціях, де викладачі можуть обмінюватися досвідом та найкращими практиками, постійне підвищення кваліфікації через участь у тренінгах, семінарах та конференціях з об'єктно-орієнтованого програмування.

Окрім технічних знань, важливою є участь викладачів у професійних спільнотах та мережах. Це дозволяє залишатися в курсі сучасних тенденцій, обмінюватися досвідом і найкращими практиками з колегами, а також підтримувати мережі для подальшого розвитку та підвищення кваліфікації.

Об’єктно-орієнтовані мови програмування є потужним інструментом, який дозволяє розробникам створювати додатки за моделлю об’єктів реального світу. Використовуючи об’єкти, класи та структури даних, розробники можуть створювати надійні, ефективні практичні програми. Розробники можуть створювати багаторазові та настроювані програми, використовуючи найкращі об’єктно-орієнтовані мови програмування, такі як Java, C++ і Python. Використовуючи ці мови, розробники можуть створювати більш чутливі та інтерактивні додатки, що дозволяє користувачам мати більш приємний і продуктивний досвід. Іншими популярними мовами ООП є Ruby, C# і PHP [22].

Адміністрація закладів фахової передвищої освіти повинна залучати викладачів до професійних спільнот та мереж для обміну досвідом та найкращими практиками, відповідно до конкретних умов та вимог навчального закладу.

Викладачі, які володіють такими характеристиками, зможуть більш ефективно викладати об'єктно-орієнтоване програмування та надавати студентам необхідні знання та навички для успішної роботи в сфері розробки програмного забезпечення.

Підготовка фахівців у сфері цифрових технологій, зокрема викладачів для освітнього модуля «Реалізація поліморфізму мовою C#» в закладах вищої освіти, є важливим завданням, яке потребує виконання низки критичних кроків. Викладачі повинні володіти не лише теоретичними знаннями про ООП, але й практичним досвідом у мовах програмування, які реалізують об'єктно-орієнтований підхід, таких як Java, Python, C++, C# і Kotlin. Вони також мають бути обізнані з сучасними тенденціями в світі програмування та цифрових технологій, щоб адекватно реагувати на зміни в галузі. Це означає, що викладачі повинні постійно вдосконалювати свої знання і навички, слідкувати за новинами в IT-індустрії та бути готовими до впровадження інновацій у навчальний процес.

Однією з основних складових підготовки викладачів є поєднання теоретичних знань з практичними вправами, що дозволяє закріпити матеріал на практиці. Викладачі повинні не тільки розуміти основні концепції ООП, такі як спадкування, поліморфізм, абстракція та інкапсуляція, але й уміти пояснювати їх студентам простими й зрозумілими словами. Це вимагає від викладачів не лише експертизи в програмуванні, а й уміння комунікувати складні ідеї доступно та зрозуміло .

Значну роль у підготовці викладачів відіграє практичне застосування знань під час навчання. Викладачі, які мають досвід у розробці реальних проектів із використанням ООП, можуть надати студентам цінну інформацію про вимоги ринку праці та впроваджувати їх у навчальні програми. Серед

таких вимог є актуалізація змісту освітніх програм, включення сучасних тем і завдань у сфері ООП, а також створення реальних проектів, що відображають потреби роботодавців. Важливо, щоб студенти могли працювати над проектами, які відповідають вимогам сучасного ринку, оскільки це дозволяє їм отримати практичний досвід, який буде корисним у їхній майбутній кар'єрі.

Окремо слід зазначити важливість інтерактивних методів навчання. Використання онлайн-ресурсів, візуалізації даних та інтерактивних завдань дозволяє студентам краще засвоювати матеріал та розвивати практичні навички. Цей підхід відповідає сучасним тенденціям активного залучення студентів у навчальний процес, що, у свою чергу, сприяє їхньому глибшому розумінню предмету. Інтерактивні методи викладання можуть включати використання симуляцій, ігор, а також колективної роботи над проектами, що стимулює критичне мислення і креативність.

Для ефективного навчального процесу необхідно використовувати сучасні навчальні засоби, такі як інтерактивні платформи та візуалізаційні інструменти. Комбінація різних методів навчання, включаючи лекції, практичні заняття та проекти, сприяє поглибленню знань студентів. Викладачі можуть використовувати графічні засоби для пояснення складних концепцій, що дозволяє краще засвоювати інформацію. Наприклад, візуалізація даних може допомогти студентам краще зрозуміти, як працюють різні структури даних та алгоритми, використані в ООП.

Сьогодні роль викладачів виходить за рамки простої передачі знань. Вони повинні бути менторами, заохочувати розвиток креативності та критичного мислення. Взаємодія з учнями під час лабораторних робіт, проектів і групових завдань сприяє створенню сприятливого навчального середовища, в якому студенти можуть вільно висловлювати свої думки і ідеї. Створення атмосфери довіри та відкритості допомагає студентам почуватися впевненіше під час навчання та експериментів.

Для підвищення професійних навичок викладачі повинні регулярно брати участь у тренінгах, конференціях і семінарах. Ці заходи сприяють

обміну досвідом та найкращими практиками, що є важливим аспектом їх професійного розвитку. Крім того, важливо, щоб викладачі мали можливість обговорювати проблеми і виклики, з якими вони стикаються у своїй практиці, та отримувати підтримку від колег. Це дозволяє їм зберігати актуальність своїх знань та впроваджувати інновації у викладанні.

Окрім технічних знань, участь у професійних спільнотах та мережах дозволяє викладачам бути в курсі новітніх тенденцій, обмінюватися досвідом з колегами, а також підтримувати мережі для подальшого розвитку. Це може включати участь у онлайн-форумах, групах у соціальних мережах або професійних асоціаціях, які зосереджуються на ООП. Викладачі можуть ділитися своїми напрацюваннями, отримувати відгуки від колег і працювати над спільними проектами.

Об’єктно-орієнтовані мови програмування, такі як Java, C++, Python, Ruby, C# і PHP, надають розробникам потужні інструменти для створення ефективних програм. Використовуючи об’єктно-орієнтований підхід, розробники можуть створювати інтерактивні додатки, що забезпечують користувачам приємний досвід. Завдяки можливості створення об'єктів, які можуть мати свої властивості та методи, програмісти можуть моделювати реальний світ у своїх додатках, що робить їх більш зрозумілими для кінцевих користувачів.

Для успішної розробки програм із застосуванням ООП важливо дотримуватися ряду рекомендацій, таких як: прагнення до простоти та ефективності, дотримання принципів SOLID для забезпечення гнучкості й легкості обслуговування коду, чітке визначення класів і їхніх зв’язків.

Висновки до розділу 3

У розділі виявлено, що ООП є невід'ємною складовою сучасної розробки програмного забезпечення, надаючи розробникам можливість створювати складні системи, що легко підтримуються та модифікуються. Його принципи,

такі як спадкування, інкапсуляція та поліморфізм, сприяють створенню чіткого та зрозумілого коду.

Кваліфіковані викладачі з ООП є критично важливими для успішного навчання студентів. Вони повинні мати глибокі теоретичні знання та практичний досвід, що дозволяє їм пояснювати складні концепції доступною мовою. Підготовка фахівців у галузі ООП є складним і багатогранним процесом, який вимагає інтеграції теоретичних знань, практичних навичок та інноваційних підходів до навчання. Це дозволяє створити конкурентоспроможних спеціалістів, здатних ефективно вирішувати складні завдання у сфері інформаційних технологій.

Таким чином, комплексний підхід до підготовки викладачів з об'єктно-орієнтованого програмування є необхідним для розвитку кваліфікованих кадрів у цій важливій та динамічній сфері.

РОЗДІЛ 4 МЕТОДИКА ПРОФЕСІЙНОЇ ПІДГОТОВКИ ФАХІВЦІВ З ЦИФРОВИХ ТЕХНОЛОГІЙ ДЛЯ ВИКЛАДАННЯ ОСВІТНЬОГО МОДУЛЯ «РЕАЛІЗАЦІЯ ПОЛІМОРФІЗМУ МОВОЮ C#» У ЗАКЛАДАХ ВИЩОЇ ОСВІТИ. ДИДАКТИЧНИЙ ПРОЕКТ КОНСУЛЬТАТИВНОГО ЗАНЯТТЯ З ТЕМИ «СПАДКУВАННЯ КЛАСІВ. ПРИХОВУВАННЯ ЕЛЕМЕНТІВ БАЗОВОГО КЛАСУ. ПОЛІМОРФІЗМ. ВІРТУАЛЬНІ МЕТОДИ ТА ЇХ ПЕРЕВИЗНАЧЕННЯ. МОДИФІКАТОРИ ДОСТУПУ КЛАСІВ. АБСТРАКТНІ КЛАСИ ТА ЕЛЕМЕНТИ. ЗАПЕЧАТАНІ КЛАСИ. СТАТИЧНІ КЛАСИ» ДИСЦИПЛІНИ «ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ» ДЛЯ ЗДОБУВАЧІВ ВИЩОЇ ОСВІТИ СПЕЦІАЛЬНОСТІ 015 ПРОФЕСІЙНА ОСВІТА (ЦИФРОВІ ТЕХНОЛОГІЇ)

4.1 Вихідні дані до проєкту

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

галузь знань: 01 Освіта / Педагогіка;

спеціальність: 015 Професійна освіта (Цифрові технології);

рівень вищої освіти: перший (бакалаврський);

освітній ступінь: бакалавр;

дисципліна: «Об'єктно-орієнтоване програмування»;

тема: «Спадкування класів. Приховування елементів базового класу. Поліморфізм. Віртуальні методи та їх перевизначення. Модифікатори доступу класів. Абстрактні класи та елементи. Запечатані класи. Статичні класи».

Дисципліна містить такі характеристики, як:

кількість кредитів – 8.

Загальна кількість годин для вивчення дисципліни – 240 навчальних години з яких 150 годин самостійної роботи (30 годин курсова робота та 120

годин підготовка до аудиторних занять, контрольних заходів) та 90 годин аудиторної (30 години лекційних занять, 60 годин лабораторної роботи) для денної форми навчання;

Загальна кількість годин для вивчення дисципліни – 240 навчальних години з яких 216 годин самостійної роботи (30 годин курсова робота та 186 годин підготовка до аудиторних занять, контрольних заходів) та 24 годин аудиторної (12 годин лекційних занять та 12 годин лабораторної роботи) для заочної форми навчання;

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

- для денної та заочної форми навчання – 90/150;
- для заочної форми навчання – 24/216.

Дисципліна «Об’єктно-орієнтоване програмування» читається у 3-му та 4-му семестрі 2-го року професійної підготовки бакалаврів для денної та заочної форм навчання.

Форми контролю: залік, екзамен.

Великий обсяг навчального матеріалу, обширні, складні цілі навчання та великий відсоток часу, що відведено на самостійну роботу, обумовлюють необхідність у проведенні консультативних занять для уточнення та пояснення навчального матеріалу з дисципліни «Об’єктно-орієнтоване програмування».

4.2 Проектування цілей консультативного заняття

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

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

Проектування цілей консультативного заняття представлені у табл. 4.1 [43].

Таблиця 4.1

Цілі консультативного заняття

Цілі консультативного заняття	Цілі формування різних рівнів засвоєння навчального матеріалу	Умови досягнення	Результат у вигляді дій здобувачів освіти
1	2	3	4
1	З переліку визначень впізнавати основні поняття теми «Спадкування класів. Приховування елементів базового класу. Поліморфізм. Віртуальні методи та їх перевизначення. Модифікатори доступу класів. Абстрактні класи та елементи. Запечатані класи. Статичні класи» такі, як клас, елементи класу, модифікатори, додаток, набір файлів, проект, головний програмний файл, уміти називати способи роботи з класами, та типовими програмами та групою взаємодіючих між собою об'єктів.	Знати визначення понять «Клас» та «Елементи класів», «Модифікатори доступу» «Файл» та «Файли опису форм», знання сутності поняття «Файли програмних модулів ».	Правильно названі з переліку основні поняття теми «Спадкування класів. Приховування елементів базового класу. Поліморфізм. Віртуальні методи та їх перевизначення. Модифікатори доступу класів. Абстрактні класи та елементи. Запечатані класи. Статичні класи» такі, як клас, елементи класу, модифікатори, додаток, набір файлів, проект, головний програмний файл, уміти називати способи роботи з класами, та типовими програмами С в мовах програмування та групою взаємодіючих між собою об'єктів.

Продовження табл. 4.1

1	2	3	4
2	Уміти характеризувати режими класів, використовувати елементи класів які встановлюють характер використання класів та характеризувати програмами С і записувати їх у файл.	Виконання дій першого рівня: правильно названі з переліку основні поняття теми «Спадкування класів. Приховування елементів базового класу. Поліморфізм. Віртуальні методи та їх перевизначення. Модифікатори доступу класів. Абстрактні класи та елементи. Запечатані класи. Статичні класи» такі, як клас, елементи класу, модифікатори, додаток, набір файлів, проект, головний програмний файл, уміти називати способи роботи з класами, та типовими програмами С в мовах програмування та групою взаємодіючих між собою об'єктів.	Правильно охарактеризовано режими використання елементів класу, які встановлюють характер використання класів та охарактеризовано основні типи даних в програмі С та групою взаємодіючих між собою об'єктів.
3	Уміти робити аналіз помилки та недоліки в програмі та робити висновки щодо можливості виправлення помилок.	Виконання дій першого і другого рівнів: правильно охарактеризовано використання елементів класу, які встановлюють характер використання класів та охарактеризовано основні типи даних в програмі С та групою взаємодіючих між собою об'єктів.	Правильно зроблений аналіз помилок та недоліків в програмі та зроблені висновки щодо можливості виправлення помилок.

Продовження табл. 4.1

1	2	3	4
4	Уміти розробляти клас, що використовується для опису об’єкта і визначає його характеристики та дії.	Виконання дій першого, другого і третього рівнів: правильно зроблений аналіз помилок та недоліків в програмі та зроблені висновки щодо можливості виправлення помилок.	Правильно розроблено клас, що використовується для опису об’єкта і визначає його характеристики та дії.

Таким чином, нами були розроблені цілі консультативного заняття з теми «Спадкування класів. Приховування елементів базового класу. Поліморфізм. Віртуальні методи та їх перевизначення. Модифікатори доступу класів. Абстрактні класи та елементи. Запечатані класи. Статичні класи» дисципліни «Об’єктно-орієнтоване програмування» для здобувачів вищої освіти спеціальності 015 Професійна освіта (Цифрові технології).

4.3 Перелік джерел інформації

Майбутній фахівець має вміти самостійно користуватися джерелами інформації.

Наведемо перелік джерел інформації для підготовки здобувачів вищої освіти до консультації згідно з робочою програмою дисципліни «Об’єктно-орієнтоване програмування». Нижче представлено перелік основної та допоміжної літератури, а також інформаційні ресурси для вивчення дисципліни та підготовки до консультативного заняття:

Рекомендована література:

Основна (базова) література

1. Об’єктно-орієнтоване програмування: електрон. метод. посіб. до лек. та лаб. занять для студентів ф-ту математики, фізики та інформ. технол.

першого (бакалавр.) рівня освіти / уклад.: А. Л. Рачинська, О. А. Недєва, К. С. Палій. – Одеса : Одес. нац. ун-т ім. І. І. Мечникова, 2024. – 338 с. https://library.megu.edu.ua:9443/jspui/handle/123456789/4132

2. Коноваленко І. В., Марущак П. О. Платформа .NET та мова програмування C# 8.0 : навч. посібник. Тернопіль : ФОП Паляниця В. А., 2020. 320 с. http://elartu.tntu.edu.ua/handle/lib/32825

3. Куліков В.М., Рябцев В.В., Паршуков С.С. К85 Об'єктно-орієнтоване програмування для фахівців з кібербезпеки: навч. посіб. / ІСЗЗІ КПІ ім. Ігоря Сікорського. Київ: КПІ ім. Ігоря Сікорського, 2023. 365 с. https://files.znu.edu.ua/files/Bibliobooks/Inshi78/0058602.pdf

4. Муляр В. П. Об'єктно-орієнтоване програмування: лабораторний практикум. Луцьк : Вежа-Друк, 2022. 112 с. https://evnuir.vnu.edu.ua/handle/123456789/21291

5. Основи об'єктно-орієнтованого програмування: навч. посібник / Гришанович Т. О., Глинчук Л. Я.; ВНУ імені Лесі Українки. Луцьк : ВНУ імені Лесі Українки, 2022. 120 с. https://evnuir.vnu.edu.ua/handle/123456789/20320

Додаткова (допоміжна) література

1. Chykunov P. Services for creating UML class diagrams // P. Chykunov, I. Chykunov/ Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні: Матеріали VI Всеукраїнської науково-практичної інтернет-конференції (м. Харків, 06-07 грудня 2023 р.). – Харків: ННППІ УІПА, 2023. С 42-43.

2. Prokop Y.V., Trofymenko O.G., Kapustin M.M. A study of software development tools that are required in the job market in Ukraine and the world. Proceedings of the O.S. Popov ONAT. 2018. Vol. 2, pp. 101-108. DOI 10.33243/2518-7139-2018-1-2-101-108.

3. Ланевич Т. Принципи SOLID у розробці програмного забезпечення. Матеріали VII Міжнародної студентської науково-технічної конференції «Природничі та гуманітарні науки. Актуальні питання», 2024, 89-90.

4. Омельчук Л.Л. Об’єктно-орієнтоване програмування. Лабораторний практикум: навчальний посібник. – Київ, 2021. 265 с.

5. Прокоп Ю. В., Трофименко О. Г., Толокнов А. А., Дубовой Я. В. Комплексний аналіз підходів до викладання курсів CS1 і CS2 в університетах світу та України. Вісник Черкаського державного технологічного університету. Технічні науки. 2021. № 2. С. 18-30. DOI: 10.24025/2306-4412.1.2021.229502. URL: http://vtn.chdtu.edu.ua/article/view/229502.

6. Сурков К., Книшук А., Сорокун С. Інноваційні методи навчання при вивченні об’єктно орієнтованих мов програмування для майбутніх ІТ-фахівців. Перспективи та інновації науки, 2024, 4 (38).

7. Трофименко О.Г., Савєльєва О.С., Прокоп Ю.В., Логінова Н.І., Дика А.І. Soft skills для розробників програмного забезпечення. Кібербезпека: освіта, наука, техніка. 2023. № 3(19). С. 6-19. URL: https://csecurity.kubg.edu.ua/index.php/journal/article/view/435/350

8. Чикунов П.О. Модернізація лабораторного практикуму з дисципліни «Об’єктно-орієнтоване програмування» / П.О. Чикунов, І.П. Чикунов // Актуальні тенденції розвитку освіти, науки та технологій : Матеріали VII Міжнародної науково-практичної конференції (м. Бахмут, м. Харків, 15 травня 2024 р.). : у 3-х ч. Бахмут – Харків: ННПІІ УІПА, 2024. Ч 2. С. 97-99.

9. Чикунов П.О. Розробка лабораторного практикуму «Об’єктно-орієнтоване програмування» / П.О. Чикунов, В.С. Лизунов // Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні: Матеріали VI Всеукраїнської НПІК. – Харків: ННПІІ УІПА, 2023. С. 38-39.

Інформаційні ресурси

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

2. C# Basics for Beginners: Learn C# Fundamentals by Coding – Онлайн-курс Udemy. URL: https://www.udemy.com/course/csharp-tutorial-for-beginners/

3. Learn Parallel Programming with C# and .NET – Онлайн-курс Udemy. URL: https://www.udemy.com/course/parallel-dotnet/

4. C# Intermediate: Classes, Interfaces and OOP – Онлайн-курс Udemy. URL: https://www.udemy.com/course/csharp-intermediate-classes-interfaces-and-oop/

5. Overview of classes, structs, and records in C# – Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/object-oriented/

6. Основи програмування на C# – Онлайн-курс Prometheus. URL: https://courses.prometheus.org.ua/courses/Microsoft/CS201/2016_T1/about

7. C++ Programming Essentials – Онлайн-курс edX. URL: https://www.edx.org/professional-certificate/ibm-c-programming-essentials

8. Object Oriented Development using C# – Онлайн-курс Coursera. URL: https://www.coursera.org/learn/oo-development-using-c-sharp

9. Object-Oriented programming (C#). URL: https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/tutorials/oop

10. C# OOP. URL: https://www.w3schools.com/cs/cs_oop.php

11. Learn Object-Oriented Programming with C#. URL: https://www.tutorialsteacher.com/csharp/oop

12. A Complete Guide To Object Oriented Programming In C#. URL: https://www.c-sharpcorner.com/UploadFile/84c85b/object-oriented-programming-using-C-Sharp-net/

13. Object Oriented Programming (OOPs) in C#. URL: https://dotnettutorials.net/lesson/object-oriented-programming-csharp/

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

Основним джерелом для підготовки здобувачів вищої освіти до консультації є конспект лекцій, розроблений викладачем, оскільки він є найбільш адаптованим до робочої програми дисципліни «Об’єктно-орієнтоване програмування».

4.4 Визначення найбільш складних для розуміння та засвоєння питань

Визначимо найбільш складні для розуміння та засвоєння здобувачами вищої освіти питання теми «Спадкування класів. Приховування елементів базового класу. Поліморфізм. Віртуальні методи та їх перевизначення. Модифікатори доступу класів. Абстрактні класи та елементи. Запечатані класи. Статичні класи» дисципліни «Об’єктно-орієнтоване програмування» та сформулюємо можливі відповіді на них (табл. 4.2) [42].

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
Основні поняття теми «Спадкування класів. Приховування елементів базового класу. Поліморфізм. Віртуальні методи та їх перевизначення. Модифікатори доступу класів. Абстрактні класи та елементи. Запечатані класи. Статичні класи»	1. Поняття клас та елементи класів. 2.Поняття інформаційної системи. 3. Типова програма на C#.	1. Що представляє собою клас?	1. Клас представляє структуру, яка крім даних може містити виконавчий код. Складові частини класу називають елементами класу.
		2.Дайте відповідь поняттю «клас»?	2. *Клас* – це тип даних користувача, що використовується для опису об’єкта і визначає його характеристики та дії. У класі поєднують сукупність пов'язаних даних та функцій, які в програмі представляють об'єкт реального світу. Проаналізувавши суттєві характеристики та поведінку модельованого об'єкта, програміст повинен спроектувати клас, який їх відображуватиме.

Отже, на даному етапі ми визначили найбільш складні для розуміння та засвоєння питання з теми «Спадкування класів. Приховування елементів

базового класу. Поліморфізм.» для здобувачів вищої освіти спеціальності 015 Професійна освіта (Цифрові технології).

4.5 Вибір дидактичних методів активізації навчальної діяльності студентів на консультації

Оберемо методи активізації навчальної діяльності здобувачів вищої освіти на консультації (табл. 4.3) [44].

Таблиця 4.3

Методи активізації навчальної діяльності здобувачів вищої освіти на консультації

Дидактичні методи	Реалізація методів при проведенні консультаційного заняття
1	2
Методи підвищення наочності	Використання інтерактивної дошки та мультимедійного проектора для демонстрації слайдів з теми «Спадкування класів. Приховування елементів базового класу. Поліморфізм. Віртуальні методи та їх перевизначення. Модифікатори доступу класів. Абстрактні класи та елементи. Запечатані класи. Статичні класи», та використання плакатів для демонстрації програми «Використання елементів класу».
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 «Спадкування класів. Приховування елементів базового класу. Поліморфізм. Віртуальні методи та їх перевизначення. Модифікатори доступу класів.» є дуже важливою, оскільки на підприємствах існує безліч ситуацій, в яких фахівців з цифрових технологій має виявити власну компетентність. Тому для вас, як майбутніх фахівців, ця тема є достатньо актуальною. Відповідно до вашої майбутньої професійної діяльності знання цього навчального матеріалу знадобляться вам для програмування на підприємстві, а саме це дозволить ефективно використовувати засоби сучасних інформаційних систем. Отримання знань з області розробки алгоритмів та програмування. Оволодіння такими знаннями дозволить реалізовувати задачі автоматизації обробки інформації, автоматизації керування об'єктами, зокрема радіоелектронними, за допомогою комп'ютерної техніки. Такі знання майбутній спеціаліст зможе застосовувати як при подальшому навчанні, так і після отримання вищої освіти у своїй професійній діяльності. Від того, наскільки успішно ви справитеся з даним завданням, вже працюючи на підприємстві, залежить його благополуччя, а вам дозволить рости у якості високоякісного фахівця в області програмування та стверджуватимуть вас як висококваліфікованого фахівця».

Продовження табл. 4.3

1	2
Проблемні методи	Використання проблемного питання. 1. Проблемне питання: «Якими символами обрамляються коментарі в C#? 2. Які Ви знаєте компоненти C#, що підтримують функції множинного введення-виведення? Назвіть їхні властивості.
Комунікативні методи	Імітація ситуацій з реального життя. Відшуковувати, обробляти, аналізувати та оцінювати інформацію, що стосується професійної діяльності, користуватися спеціалізованим програмним забезпеченням та сучасними засобами зберігання та обробки інформації. Розв'язувати типові спеціалізовані задачі, пов'язані з вибором алгоритмів, технологій та методів програмної інженерії й виконувати необхідні розрахунки, кодування, тестування й оновлення програмного забезпечення різних рівнів чи елементів інформаційних й комп'ютерних технологій.

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

4.6 Вибір способів організації консультативного заняття

Вибір способів організації консультативного заняття здійснюється з урахуванням даних, наведених в таблиці 4.4 [42].

Таблиця 4 4

Варіанти організації консультативного заняття

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

Продовження табл. 4 4

1	2	3
3	- видача завдань на самостійне вивчення матеріалу теми. - підготовка питань лектору. - відповіді і їх обговорення	В цьому випадку консультування грає функцію додаткового інформування зі складних питань і пояснення незрозумілого навчального матеріалу.
4	- повідомлення теми, - консультування декількома фахівцями в певній області науки і техніка з актуальних питань науки і нової техніки	Цей варіант 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.

Згідно з представленою таблицею обираємо 1 варіант організації консультативного заняття, на якому викладач пояснює питання, які здалися незрозумілими здобувачам вищої освіти**.**

4.7 Розробка сценарію проведення консультативного заняття

Наведемо розробку сценарію проведення консультативного заняття у відповідності до обраного варіанту його організації (табл. 4.5) [44].

Таблиця 4.5

Сценарій консультативного заняття

Етапи проведення консультативного заняття	Дії викладача	Дії здобувачів освіти
1	2	3
Організаційний момент	Викладач вітає здобувачів освіти, робить перекличку, пропонує студентам розпочати роботу на консультації.	Здобувачі освіти вітають викладача, беруть участь у перекличці, налаштовуються на роботу на консультації.
Повідомлення теми і мети уроку	Повідомлення теми заняття «Класи: опис об'єктів. Модифікатори доступу. Використання елементів класу», формулювання його цілей: Сформувати знання про основні елементи класу.	Фіксація теми, сприйняття цілей, представлення результатів засвоєння матеріалу теми даного заняття.

Продовження табл. 4.5

1	2	3
Мотивація мети	Повідомлення важливості вивчення даної теми: Тема «Класи: опис об'єктів. Модифікатори доступу. Використання елементів класу» є дуже важливою, оскільки на підприємствах існує безліч ситуацій, в яких фахівець з інформаційних технологій має виявити власну компетентність. Тому для вас, як майбутніх фахівців, ця тема є достатньо актуальною. Відповідно до вашої майбутньої професійної діяльності знання цього навчального матеріалу знадобляться вам для програмування на підприємстві, а саме це дозволить ефективно використовувати засоби сучасних інформаційних систем та техноологій. Класи: опис об'єктів. Модифікатори доступу. Використання елементів класу є одним з основних завдань вашої майбутньої діяльності. Від того, наскільки успішно ви справитеся з даним завданням, вже працюючи на підприємстві, залежить його благополуччя, а вам дозволить рости у якості високоякісного програміста в області програмування та стверджуватимуть вас як висококваліфікованого фахівця».	Сприйняття важливості і актуальності вивчення теми, прояв інтересу до неї.
Актуалізація знань	Викладач проводить фронтальне усне опитування з метою перевірки базових знань: 1. Як ви розумієте термін «Клас»? 2.Поясніть загальну схему використання елементів класу? 3. Що означує оголошення класу?	Здобувачі освіти беруть участь у опитуванні та відповідають на поставлені питання. Передбачувані відповіді: 1. *Клас* – це тип даних користувача, що використовується для опису об’єкта і визначає його характеристики та дії. У класі поєднують сукупність пов'язаних даних та функцій, які в програмі представляють об'єкт реального світу. Проаналізувавши суттєві

Продовження табл. 4.5

1	2	3
		характеристики та поведінку модельованого об'єкта, програміст повинен спроектувати клас, який їх відображуватиме. 2. Елементи класу, які зберігають дані про об'єкт, називають *елементами даних*. Вони моделюють характеристики реального об'єкта, якого представляє клас. До елементів даних відносять поля та константи класу. Елементи класу, які містять виконавчий код, називають *функціональними елементами*. Вони моделюють поведінку реального об'єкта. До функціональних елементів класу відносять методи (включаючи конструктори та деструктори), властивості, оператори, індексатори та події. 3. Оголошення класу означує його елементи даних та функціональні елементи. При цьому не створюється екземпляр класу (об'єкт). Клас, як інші тип, задає шаблон, за яким можна створювати екземпляри цього класу.
Формування ООД	Викладач проводить консультацію згідно плану, за допомогою методу - пояснення: 1. Поняття клас та елементи класів. 2.Поняття інформаційної системи. 3. Типова програма на C#.	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. 1. Оголошення класу починають ключовим словом class, після якого вказують назву класу. Далі між фігурними дужками розміщують оголошення елементів класу. Цю частину оголошення класу називають	Здобувачі освіти запитують: 1. Яким ключовим словом починають оголошення класу? 2. Що є найважливішими елементами класу?

Продовження табл. 4.5

1	2	3
	тілом класу. Елементи класу можна оголошувати у тілі класу в будь-якій послідовності. 2. Найважливішими елементами класу є поля та методи. Поля є елементами даних, а методи – функціональними елементами. *Поля* – це змінні, які належать класові. Вони можуть бути довільного типу (як попередньо визначеного C#, так і користувацького). Подібно до звичайних змінних, поля зберігають дані.	
Підведення підсумків	Викладач підводить підсумки проведення консультації: «Сьогодні ми розглянули незрозумілі вам питання теми. Зараз перевіримо як ви засвоїли незрозумілий вам матеріал. Скажіть, що ж таке «Клас» та «Приватні елементи класу» ?	Здобувачі освіти слухають, відповідають: «*Клас* – це тип даних користувача, що використовується для опису об’єкта і визначає його характеристики та дії. У класі поєднують сукупність пов'язаних даних та функцій, які в програмі представляють об'єкт реального світу. Проаналізувавши суттєві характеристики та поведінку модельованого об'єкта, програміст повинен спроектувати клас, який їх відображуватиме. Приватні елементи класу (оголошені з модифікатором private) доступні тільки всередині класу, у якому вони оголошені. Інші класи не можуть ні "бачити" їх, ні працювати з ними. Приватний рівень доступу забезпечується за замовчуванням, тому, коли при оголошенні елемента класу модифікатор доступу не вказують, то елемент буде приватним». Здобувачі прощаються.

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

На заключному етапі представлено контурний конспект з теми

«Спадкування класів. Приховування елементів базового класу. Поліморфізм. Віртуальні методи та їх перевизначення. Модифікатори доступу класів. Абстрактні класи та елементи. Запечатані класи. Статичні класи» дисципліни «Об’єктно-орієнтоване програмування» для здобувачів вищої освіти спеціальності 015 Професійна освіта (Цифрові технології).

Конспект – сукупність взаємозв'язаних опорних сигналів теми.

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

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

Контурний конспект заняття з теми «Спадкування класів. Приховування елементів базового класу. Поліморфізм. Віртуальні методи та їх перевизначення. Модифікатори доступу класів. Абстрактні класи та елементи. Запечатані класи. Статичні класи» дисципліни «Об’єктно-орієнтоване програмування» для здобувачів вищої освіти спеціальності 015 Професійна освіта. (Цифрові технології) представлено у Додатку.

Висновки до розділу

У четвертому розділі розроблено дидактичний проект консультативного заняття з теми «Спадкування класів. Приховування елементів базового класу. Поліморфізм. Віртуальні методи та їх перевизначення. Модифікатори доступу класів. Абстрактні класи та елементи. Запечатані класи. Статичні класи» дисципліни «Об’єктно-орієнтоване програмування» для здобувачів вищої освіти спеціальності 015 Професійна освіта. (Цифрові технології).

Сформульовано цілі консультативного заняття. Обрано методи активізації навчальної діяльності здобувачів освіти на консультації. Здійснено вибір способів організації консультативного заняття.

Розроблено сценарій проведення консультативного заняття у відповідності до обраного варіанту його організації.

Проаналізовано джерела інформації для підготовки здобувачів освіти до консультації згідно з робочою програмою дисципліни «Об’єктно-орієнтоване програмування». Подано список використаних джерел та відповідні посилання.

ВИСНОВКИ

У кваліфікаційній роботі теоретично обґрунтована та перевірена методика професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Реалізація поліморфізму мовою C#» у закладах вищої освіти та виділено пріоритетні напрямки удосконалення професійних компетентностей.

З’ясовано, що професійна підготовка фахівців з цифрових технологій для викладання освітнього модулю «Реалізація поліморфізму мовою C#» у закладах вищої освіти є актуальною у професійній педагогіці.

Проаналізовано ступінь актуальності проблеми професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Реалізація поліморфізму мовою C#».

У розділі виконано аналіз освітніх компонент вітчизняних закладів вищої освіти, присвячених вивченню основ об'єктно-орієнтованого програмування, з метою встановлення напрямків подальших досліджень.

За результатами виконаного дослідження виконана постановка завдань нового лабораторного практикуму «Реалізація поліморфізму мовою C#», який акцентує увагу здобувачів на побудові об’єктно-орієнтованої ієрархії класів з використанням операцій приховання елементів класів, реалізації поліморфізму за допомогою віртуальних методів, використанню різних видів класів.

Розроблено завдання лабораторних робіт, приклади виконання робіт та оформлення звітів, що містять опис предметної області, UML-діаграми ієрархії класів, опис класів та їх атрибутів, програмний код на мові C#, приклад виконання програми у середовищі Visual Studio.

Встановлено, що кваліфіковані викладачі з ООП є критично важливими для успішного навчання студентів. Вони повинні мати глибокі теоретичні знання та практичний досвід, що дозволяє їм пояснювати складні концепції доступною мовою. Підготовка фахівців у галузі ООП є складним і

багатогранним процесом, який вимагає інтеграції теоретичних знань, практичних навичок та інноваційних підходів до навчання. Це дозволяє створити конкурентоспроможних спеціалістів, здатних ефективно вирішувати складні завдання у сфері інформаційних технологій.

Розроблено дидактичний проект консультативного заняття з теми «Спадкування класів. Приховування елементів базового класу. Поліморфізм. Віртуальні методи та їх перевизначення. Модифікатори доступу класів. Абстрактні класи та елементи. Запечатані класи. Статичні класи» дисципліни «Об’єктно-орієнтоване програмування» для здобувачів вищої освіти спеціальності 015 Професійна освіта. (Цифрові технології).

Сформульовано цілі консультативного заняття. Обрано методи активізації навчальної діяльності здобувачів освіти на консультації. Здійснено вибір способів організації консультативного заняття.

Розроблено сценарій проведення консультативного заняття у відповідності до обраного варіанту його організації.

Проаналізовано джерела інформації для підготовки здобувачів освіти до консультації згідно з робочою програмою дисципліни «Об’єктно-орієнтоване програмування». Подано список використаних джерел та відповідні посилання.

За основними результатами дослідження виконана публікація тези доповіді на конференції:

– Чикунов П.О. Постановка лабораторного практикуму «Реалізація поліморфізму мовою C#» / П.О. Чикунов, Д.В. Настарчук // Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні: Матеріали VII Всеукраїнської науково-практичної інтернет-конференції (м. Харків, 05-06 грудня 2024 р.). – Харків: БННППІ УІПА ХНУ ім. Каразіна, 2024.

СПИСОК ВИКОРИСТОВАНИХ ДЖЕРЕЛ

1. A Complete Guide To Object Oriented Programming In C#. URL: https://www.c-sharpcorner.com/UploadFile/84c85b/object-oriented-programming-using-C-Sharp-net/

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

3. C# Basics for Beginners: Learn C# Fundamentals by Coding – Онлайн-курс Udemy. URL: https://www.udemy.com/course/csharp-tutorial-for-beginners/

4. 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.

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

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

7. Коноваленко І.В., Марущак П.О. Платформа .NET та мова програмування C# : навч. посіб. Тернопіль: ФОП Паляниця В. А., 2020. 320 с.

8. Куліков В.М., Рябцев В.В., Паршуков С.С. Об’єктно-орієнтоване програмування для фахівців з кібербезпеки: навч. посіб. Київ: КПІ ім. Ігоря Сікорського, 2023. 365 с.

9. Ланевич Т. Принципи SOLID у розробці програмного забезпечення. Природничі та гуманітарні науки. Актуальні питання: матер. VII міжнар. студ. наук.-техн. конф. 2024. С. 89-90.

10. Муляр В. П. Об’єктно-орієнтоване програмування: лабораторний практикум. Луцьк: Вежа-Друк, 2022. 112 с.

11. Об’єктно-орієнтоване програмування: електрон. метод. посіб. / уклад.: А. Л. Рачинська, О. А. Недєва, К. С. Палій. Одеса : Одес. нац. ун-т ім.

І. І. Мечникова, 2024. 338 с.

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

13. Омельчук Л.Л. Об’єктно-орієнтоване програмування. Лабораторний практикум: навчальний посібник. Київ, 2021. 265 с.

14. Основи об’єктно-орієнтованого програмування: навч. посіб. / Гришанович Т. О., Глинчук Л. Я. Луцьк : ВНУ імені Лесі Українки, 2022. 120 с. https://evnuir.vnu.edu.ua/handle/123456789/20320

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

16. Сурков К., Книшук А., Сорокун С. Інноваційні методи навчання при вивченні об’єктно орієнтованих мов програмування для майбутніх ІТ-фахівців. Перспективи та інновації науки. 2024. № 4 (38).

17. Трофименко О.Г., Савєльєва О.С., Прокоп Ю.В., Логінова Н.І., Дика А.І. Soft skills для розробників програмного забезпечення. Кібербезпека: освіта, наука, техніка. 2023. № 3(19). С. 6-19.

18. Чикунов П.О., Лизунов В.С. Розробка лабораторного практикуму «Об’єктно-орієнтоване програмування». Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні: матер. VI Всеукр. наук.-практ. інтернет-конф. (м. Харків, 06-07 грудня 2023 р.). Харків: ННПІ УІПА, 2023. С. 38-39.

19. Чикунов П.О., Чикунов І.П. Модернізація лабораторного практикуму з дисципліни «Об’єктно-орієнтоване програмування». Актуальні тенденції розвитку освіти, науки та технологій : матер. VII Міжнар. наук.-практ. конф. (Бахмут - Харків, 15 травня 2024 р.). : у 3-х ч. Ч 2. С. 97-99.

20. Щербаков О. В., Парфьонов Ю. Е., Федорченко В. М. Основи об'єктно-орієнтованого програмування: навч. посіб. Харків: ХНЕУ ім. С. Кузнеця, 2019. 237 с.

21. Поліщук А.М., Ніколюк П.К. Технологія програмування та основні

етапи її розвитку. Комп’ютерні технології обробки даних, 2022, 89-91.

22. Бородкіна І. Інженерія програмного забезпечення: навчальний посібник. Центр учбової літератури, 2021. – 204 с.

23. Алещенко О.В. Силабус навчальної дисципліни «Об’єктно-орієнтоване програмування». URL: https://comsys.kpi.ua/upload/Силабус - Програмування-2. Об'єктно-орієнтоване програмування.pdf

24. Щербаков О.В. Силабус навчальної дисципліни «Об’єктно-орієнтоване програмування». URL: https://www.hneu.edu.ua/wp-content/uploads/2020/10/Ob-yektno-oriyentovane-programuvannya-Sylabus-121-Bakalavr.pdf

25. Рудницький С.В. Силабус навчальної дисципліни «Об’єктно-орієнтоване програмування». URL: https://er.chdtu.edu.ua/bitstream/ChSTU/1051/1/Силабус_ООП_19.pdf

26. Шевчук І.Б. Силабус навчальної дисципліни «Об’єктно-орієнтоване програмування». URL: https://financial.lnu.edu.ua/wp-content/uploads/2015/10/sylabus_Obiektno-oriientovane-prohramuvannia.pdf

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

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

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

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

31. Методика формування пошуково-дослідницьких умінь майбутніх інженерів-педагогів у процесі професійної підготовки: колективна монографія / В.В.Кулешова, В.В.Мальована. – Артемівськ: ННППІ УІПА, 2012. – 264 с. (власний внесок: Р1; Р2 с.86-92; Р3; 9,8 д.а.).

32. 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.).

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

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

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

36. Олійник В. В. Відкрита післядипломна педагогічна освіта: нові моделі та форми професійного розвитку / Освіта дорослих у перспективі змін: інновації, технології, прогнози: колективна монографія / За ред.. А. Василюк, А. Стоговського. – Ніжин: Видавець ПП Лисенко М.М., 2017. – 248 с. С. 93–108.

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

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

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

40. Сисоєва С.О. Теорія і практика вищої освіти: навч. посібник / С.О.

Сисоєва, І.В. Соколова. – К., 2016. – 338 с. – Режим доступу: http://lib.iitta.gov.ua/711948/1/Sysoieva_Socolova_2016_pos..pdf

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

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

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

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

45. Коваленко О. Е., Брюханова Н. О., Корольова Н.В. Методика професійного навчання: дидактичне проектування: Підручник для студентів інженерно-педагогічних спеціальностей. – Харків: УІПА, 2019. – 204 с.

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

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

2017. - 100 с.

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

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

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

ДОДАТОК А КОНТУРНИЙ КОНСПЕКТ

Спадкування – один із трьох фундаментальних принципів об'єктно-орієнтованого програмування. Саме завдяки йому можливе створення ієрархічних груп класів. Спадкування класів дозволяє оголосити новий клас, який містить весь функціонал іншого класу і розширює або модифікує його. Вже існуючий клас, який називають *базовим класом*, стає основою для нового класу, який називають *похідним*. Похідний клас містить всі елементи базового класу і може мати свої власні елементи.

У C# кожен клас може мати тільки один базовий клас. Це називають *одиничним спадкуванням*. Інші мови (наприклад, C++, Java) підтримують *множинне спадкування*, при якому специфікація бази класу може містити більше одного класу. Але у C# для підвищення надійності коду від цього відмовилися.

На рис. 1 показано приклад ієрархії класів. Стрілки показують напрям від базового класу до похідного.

```
class Class01
{ ... }

class Class02 : Class01
{ ... }

class Class03 : Class02
{ ... }
```

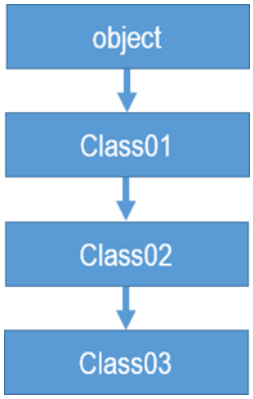

Рис. 1. Кілька пов'язаних через спадкування класів та їх дерево ієрархії

У похідному класі не можна видалити жодного успадкованого елемента. Але можна приховати елемент базового класу новим елементом з такою ж назвою. Коли клас має успадкований метод, який не підходить для виконання його специфічного завдання, ми можемо оголосити інший метод з такою ж назвою і реалізувати в ньому потрібну поведінку.

Щоб приховати елемент даних, слід оголосити новий елемент того ж

типу і з такою ж назвою, але іншого типу (або з іншим значенням). Щоб приховати функціональний елемент класу (наприклад, метод), потрібно оголосити новий елемент з такою самою сигнатурою (назва елемента та послідовність і типи параметрів). Статичні елементи класу також можна приховувати.

Оголошуючи елемент класу, який приховує успадкований елемент, використовують модифікатор new. Він дозволяє явно вказати, що відбувається навмисне приховування. Без цього модифікатора компіляція програми буде також успішною, але компілятор згенерує попередження про неявне приховування успадкованого елемента.

Для доступу до прихованого елемента базового класу з коду похідного класу використовують ключове слово base, після якого через крапку вказують назву потрібного елемента базового класу.

Зокрема, звернутись до поля базового класу sideA з методу похідного класу можна так:

```
base.sideA = 300;
```

Аналогічно можна викликати метод базового класу:

```
base.PrintBasicInfo();
```

Віртуальні методи дозволяють використовувати посилання на базовий клас для звертання до елементів похідних класів. При виклику віртуальних методів використовують не оголошений на час компіляції клас змінної, а фактичний клас об'єкта під час виконання.

Саме за допомогою віртуальних методів та їх заміщення забезпечується реалізація поліморфізму – одного з трьох головних принципів об'єктно-орієнтованого програмування. Поліморфізм забезпечує "багатоформенність" об'єкта, імітуючи зміну його "форми" (типу) залежно від умов виконання.

Щоб оголосити віртуальний та заміщуючий методи у базовому та похідному класах, потрібно забезпечити виконання таких умов:

► метод похідного класу та метод базового класу повинні мати однакову сигнатуру та тип;

► метод базового класу слід позначити як virtual (віртуальний метод);

► метод похідного класу потрібно позначити як override.

Наступний приклад демонструє оголошення віртуального методу у базовому класі та його заміщення в похідному класі за допомогою модифікаторів virtual та override:

```
class TheBaseClass // Базовий клас
{
    virtual public void Print() // Віртуальний метод
    ...
}
class TheDerivedClass : TheBaseClass // Похідний клас
{
    override public void Print() // Заміщуючий метод
    ...
}
```

Абстрактні класи – це класи, призначені виключно для спадкування їх іншими класами. Створити екземпляр такого класу неможливо. В абстрактному класі описують (але не реалізують) найзагальніші характеристики об'єкта, які повинні бути уточнені шляхом конкретної їх реалізації у похідному класі.

```
abstract class TheClass
{
      ...
}
```

Абстрактний клас може містити звичайні та абстрактні елементи. Він може бути похідними від іншого абстрактного класу. У класі, похідному від абстрактного, слід реалізувати всі успадковані абстрактні елементи, позначивши їх ключовим словом override (якщо тільки клас сам не є абстрактним).

Абстрактний елемент класу – це елемент, призначений для заміщення у похідному класі. Абстрактними можуть бути методи та інші функціональні елементи (властивості, події, індексатори). Елементи даних класу не можуть бути абстрактними. Абстрактний елемент позначають модифікатором abstract. Він не повинен мати реалізації: його код представляють однією порожньою інструкцією (крапкою з комою):

```
abstract class ClassX
{
    abstract public void TheMethod();  // оголошення абстрактного методу
```

```
    abstract public int TheProperty    // оголошення абстрактної властивості
    {
        // оголошення методів доступу до властивості
        get; // імплементація методу відсутня
        set; // імплементація методу відсутня
    }
}
```

Абстрактні елементи класу можна оголошувати тільки в абстрактному класі. Хоча вони і мають бути заміщені в похідному класі з використанням директиви override, але при їх оголошенні не можна використовувати модифікатор virtual. Віртуальні елементи, на відміну від абстрактних, повинні мати код реалізації. У похідному класі віртуальні елементи можуть бути заміщеними, а абстрактні – повинні бути заміщеними.

Лістинг 1. Абстрактний клас з абстрактним методом

```
abstract class TheAbstractClass // Абстрактний клас
{
    public void PrintIdentify() // Не-абстрактний метод
    {
        Console.WriteLine("Надруковано з абстрактного класу");
    }
    abstract public void AbstractMethod(); // Абстрактний метод
}

class TheClass : TheAbstractClass  // Похідний клас від абстрактного
{
    public override void AbstractMethod()  // Замішення абстрактного методу
    {
        Console.WriteLine("Надруковано з похідного класу");
    }
}

class Program
{
    static void Main(string[] args)
    {
        TheClass tc = new TheClass();
        tc.PrintIdentify();
        tc.AbstractMethod();
    }
}
```

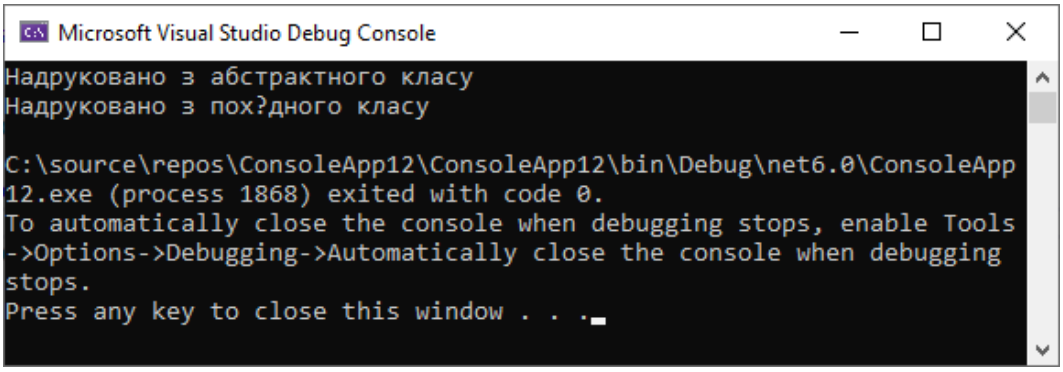

Рис. 2. Результат виконання програми з лістингу

У лістингу приведено приклад, який містить оголошення абстрактного класу TheAbstractClass (рядки 1…8). Він має два методи: звичайний метод PrintIdentify (рядки 3…6) та абстрактний метод AbstractMethod (рядок 7). Далі оголошено похідний клас (рядки 10…16), в якому заміщено та реалізовано абстрактний метод (рядки 12…15).

В методі Main створюється екземпляр класу TheClass та викликаються два його методи: один успадкований з абстрактного базового класу, а другий – з похідного класу, реалізований для заміщення абстрактного. При виконанні ця програма виведе у консольне вікно дві фрази, які позначають виконання обох методів.

Кожен елемент класу може мати свій власний модифікатор доступу, який задаватиме рівень його видимості в системі. За замовчуванням, якщо не вказати ніякого модифікатора доступу, використовується рівень private.

Для елементів класу можна задати такі модифікатори доступу:

- public;
- private;
- protected;
- internal.

У таблиці 1 зведено інформацію про призначення модифікаторів доступу елементів типу. Рівні доступу в таблиці приведено в порядку зменшення обмежень: від найобмеженішого private до найвільнішого public.

Таблиця 1. Модифікатори доступу елементів класу

Модифікатор	Опис доступності елемента класу
private	Доступний тільки у класі
internal	Доступний для всіх класів збірки
protected	Доступний для всіх похідних класів
protected internal	Доступний для всіх похідних класів та всіх класів збірки
public	Доступний для всіх

Запечатані класи забороняють успадковувати їх. Вони є протилежністю до абстрактних. Якщо з абстрактного класу неможливо створити екземпляр, і його перед використанням обов'язково слід успадкувати, то запечатаний клас призначений виняткова для створення екземплярів, його неможливо успадкувати. Спроба використати запечатаний клас в ролі базового для іншого класу спричинить помилку компіляції.

```
sealed class TheClass
{
    ...
}
```

Статичним називають клас, у якого всі елементи статичні. Такі класи використовують для групування даних та функцій, які не зв'язані з екземпляром. Наприклад, статичний клас доречно використати для проектування математичної бібліотеки, яка містить набір математичних значень та функцій.

Статичний клас при оголошенні позначають модифікатором static. *Всі його елементи також слід оголосити статичними.* Такий клас може мати статичний конструктор, але не може мати жодного конструктора екземпляра, оскільки створити екземпляр такого класу неможливо. Статичний клас за замовчуванням запечатаний. Його не можуть спадкувати інші класи. До елементів статичного класу можна звертатися через його назву.

У лістингу 2 приведено приклад оголошення та використання статичного класу IntelectualMath, який містить кілька методів для математичних операцій. У класі оголошено два статичних поля типу double (рядки 3…4): одне з них представляє число π, а інше – число Ейлера. Крім цього, клас містить два статичних методи. Перший з них, метод XPi (рядки 6…9), повертає результат множення числа π на заданий аргумент, а другий метод (XE) робить те саме з числом Ейлера.

В методі Main продемонстровано звертання до всіх чотирьох статичних елементів класу IntelectualMath: значення полів та результати виконання методів виводяться у консольне вікно (рис. 3).

Лістинг 2 Приклад статичного класу

```
static class IntelectualMath
{
    public static double Pi = 3.14159265;
    public static double E = 2.71828183;

    public static double XPi(double x)
    {
        return x * Pi;
    }

    public static double XE(double x)
    {
        return x * E;
    }
}

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Pi = {0}", IntelectualMath.Pi);
        Console.WriteLine("E = {0}", IntelectualMath.E);
        Console.WriteLine("12*Pi = {0:0.00000}",
        IntelectualMath.XPi(12));
        Console.WriteLine("12*e = {0:0.00000}",
        IntelectualMath.XE(12));
    }
}
```

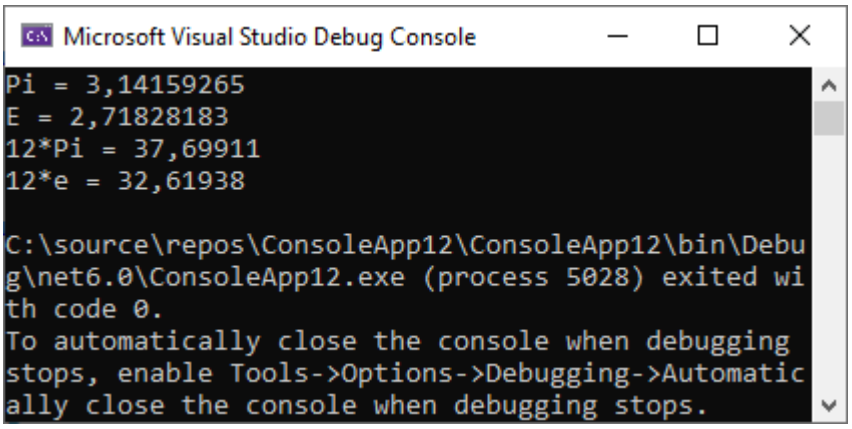

Рис. 3. Результат виконання програми з лістингу

ДОДАТОК Б ПУБЛІКАЦІЇ ЗА РЕЗУЛЬТАТАМИ ДОСЛІДЖЕННЯ

ПОСТАНОВКА ЛАБОРАТОРНОГО ПРАКТИКУМУ «РЕАЛІЗАЦІЯ ПОЛІМОРФІЗМУ МОВОЮ C#»

Чикунов П.О., к.т.н., доц.
Настарчук Д.В., здобувач вищої освіти другого рівня
Бахмутський навчально-науковий професійно-педагогічний інститут ХНУ імені В. Н. Каразіна

Методологія об'єктно-орієнтованого підходу (ООП) до розробки програмного забезпечення складається з об'єктно-орієнтованого аналізу, проектування та програмування. Головні поняття ООП – це клас, який визначає абстрактний тип даних, структура якого обумовлена проблематикою розв'язуваної задачі, та об'єкт, що є екземпляром класу. До головних принципів ООП відносяться абстракція, інкапсуляція, наслідування та поліморфізм.

Поліморфізм полягає у використанні однакових підходів для роботи з різними за функціональністю об'єктами. Слово «поліморфізм» походить з грецької і означає багатоформність, тобто здатність одного й того ж імені методу виконувати різні дії в залежності від класу, до якого належить об'єкт. Існують два види поліморфізму:

1. поліморфізм під час компіляції (статичний поліморфізм) реалізується через перевантаження методів або перевантаження операторів;
2. поліморфізм під час виконання (динамічний поліморфізм) реалізується за допомогою абстрактних класів або інтерфейсів.

Метою виконання теоретично обґрунтувати та частково перевірити методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація поліморфізму мовою C#» у закладах вищої освіти.

Під час виконання дослідження виконано аналіз освітніх компонент закладів вищої освіти, присвячених вивченню основ ООП, з метою встановлення напрямків подальших досліджень. Встановлено, що найбільш популярними ООП-мовами є C++, Java, C#, Python. За результатами аналізу рейтингу мов програмування за 2024 рік встановлено, що мова C# займає перше місце у категорії розробки «FullStack». Вибір мови C# для лабораторного практикуму з реалізації поліморфізму також є доцільним з кількох причин:

1. C# має чітко визначену підтримку ООП, яка включає всі основні принципи: інкапсуляцію, спадкування, поліморфізм і абстракцію;
2. C# пропонує зручний синтаксис для реалізації поліморфізму через ключові слова virtual, override та abstract;
3. C# має чіткі та зрозумілі модифікатори доступу, що дозволяє зручно приховувати елементи класу та контролювати їх доступність;
4. C# дозволяє реалізовувати інтерфейси й абстрактні класи, що дає змогу студентам використовувати різні типи поліморфізму;
5. C# є строго типізованою мовою, що допомагає студентам уникнути багатьох помилок на етапі компіляції, які можуть бути пов’язані з неправильним використанням типів у реалізації класів і методів.

Отже, C# є одним із найкращих виборів для реалізації лабораторного

практикуму з поліморфізму, оскільки він має зрозумілі конструкції, які сприяють вивченню та застосуванню об'єктно-орієнтованих підходів, полегшуючи студентам побудову ієрархій класів та реалізацію поліморфізму.

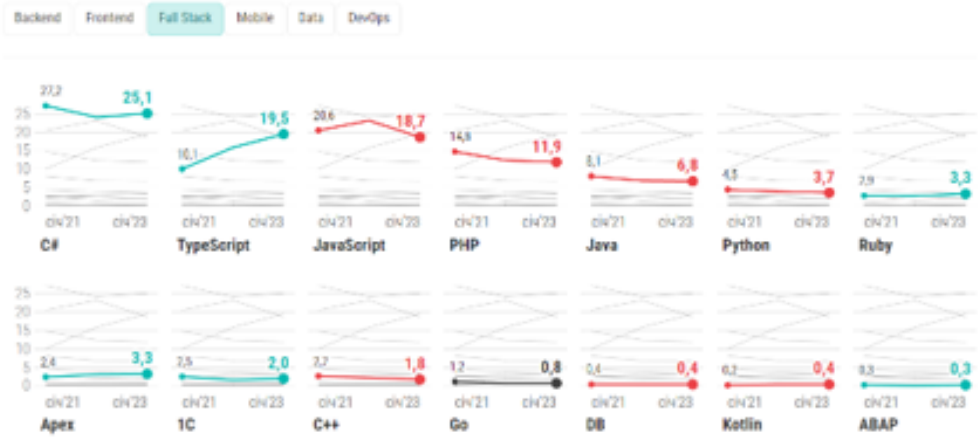

Рисунок 1 – Популярність мов програмування за рейтингом випускників

За результатами виконаного дослідження виконана постановка завдань лабораторного практикуму «Реалізація поліморфізму мовою C#», який акцентує увагу здобувачів на побудові об'єктно-орієнтованої ієрархії класів з використанням операцій приховання елементів класів, реалізації поліморфізму за допомогою віртуальних методів, використанню різних видів класів:

1. приховування полів, властивостей та методів у класах;
2. віртуальні методи та поліморфізм у класах;
3. абстрактні, статичні та запечатані класи;
4. перевизначення операцій для класів.

Розроблено завдання лабораторних робіт, приклади виконання робіт та оформлення звітів, що містять опис предметної області, UML-діаграми ієрархії класів, опис класів та їх атрибутів, програмний код на мові C#, приклад виконання програми у середовищі Visual Studio.

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

Список використаних джерел

1. Алещенко О.В. Силабус навчальної дисципліни «Об'єктно-орієнтоване програмування». URL: https://comsys.kpi.ua/upload/Силабус - Програмування-2. Об'єктно-орієнтоване програмування.pdf. (дата звернення 13.11.2024).

2. Рудницький С.В. Силабус навчальної дисципліни «Об'єктно-орієнтоване програмування». URL: https://er.chdtu.edu.ua/bitstream/ChSTU/1051/1/Силабус_ООП_19.pdf. (дата звернення 13.11.2024).

3. Чикунов П.О. Силабус навчальної дисципліни «Об'єктно-орієнтоване програмування». URL: http://surl.li/iaezef. (дата звернення 13.11.2024).

4. Спільнота програмістів DOU: Рейтинг мов програмування 2024. TypeScript в трійці лідерів, Python з'являється у всіх нішах, а Rust – улюблена мова. UML: https://dou.ua/lenta/articles/language-rating-2024/. (дата звернення 13.11.2024).