МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В. Н. Каразіна

Бахмутський навчально-науковий професійно-педагогічний інститут

Кафедра електромеханічних та комп'ютерних систем

До захисту допущено

Завідувач кафедри

_____ Інна НЕФЬОДОВА

(підпис) (ім'я, прізвище)

«07» грудня 2024 року

КВАЛІФІКАЦІЙНА РОБОТА (ПРОЄКТ)

рівень вищої освіти другий (магістерський)

спеціальність 015.39 Професійна освіта (Цифрові технології)

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

тема «Професійна підготовка фахівців з цифрових технологій для викладання освітнього модулю «Робота з інтерфейсами, колекціями та індексаторами на мові C#» у закладах вищої освіти»

Виконав(ла)

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

Олександр НИКИТСЬКИХ _____
(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
_____ Інна НЕФЬОДОВА
(підпис) (ім'я, прізвище)
«08» жовтня 2024 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЄКТ)

Никитських Олександр Сергійович
(прізвище, ім'я, по батькові здобувача)

1. Тема роботи Професійна підготовка фахівців з цифрових технологій для викладання освітнього модулю «Робота з інтерфейсами, колекціями та індексаторами на мові 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 «Робота з інтерфейсами, колекціями та індексаторами на мові 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 «Робота з інтерфейсами, колекціями та індексаторами на мові 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 «Робота з інтерфейсами, колекціями та індексаторами на мові 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 «Робота з інтерфейсами, колекціями та індексаторами на мові 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) _____ (підпис) Олександр НИКИТСЬКИХ (ім'я, прізвище)

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

РЕФЕРАТ

Мета дослідження – теоретично обґрунтувати та частково перевірити методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#» у закладах вищої освіти.

Об'єктом дослідження роботи є процес професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#».

Предметом дослідження роботи є методика професійної підготовки фахівців з цифрових технологій до розробки комплексу освітніх ресурсів. Охарактеризовано систему професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю у закладах вищої освіти.

Виконано аналіз характеристик об'єктів галузі аналіз та рефакторинг коду. Виконана постановка 3 нових лабораторних робіт. Виконано аналіз вимог до кадрового забезпечення об'єкту ІТ-галузі.

Розроблено дидактичний проект консультативного заняття з теми «Інтерфейси. Використання інтерфейсу IComparable. Інтерфейс як посилальний тип даних. Спадкування інтерфейсів» дисципліни «Програмна інженерія» для здобувачів вищої.

За основними результатами дослідження виконана публікація тези доповіді на VIII міжнародній НПК «Студенти та молодь – для майбутнього країни» (Бахмут-Харків, 14-15 листопада 2024 р.).

Обсяг дипломної роботи становить: пояснювальна записка, презентація доповіді. Пояснювальна записка складається з вступу, чотирьох розділів, висновків, списку використаних джерел, додатків. Загальний обсяг роботи 88 сторінок, з яких 62 сторінки основного тексту. Список використаних джерел становить 52 найменування, 5 таблиць, 15 рисунків.

ПРОФЕСІЙНА ПІДГОТОВКА, ЛАБОРАТОРНИЙ ПРАКТИКУМ, ПРОГРАМНА ІНЖЕНЕРІЯ, КАДРОВЕ ЗАБЕЗПЕЧЕННЯ, МЕТОДИЧНА РОЗРОБКА

ABSTRACT

The aim of the study is to theoretically substantiate and partially test the methodology for the professional training of digital technology specialists to teach the educational module "Working with Interfaces, Collections, and Indexers in C#" in higher education institutions.

The object of the study is the process of professional training of digital technology specialists for teaching the educational module "Working with Interfaces, Collections, and Indexers in C#."

The subject of the study is the methodology of professional training of digital technology specialists for the development of a set of educational resources.

The system of professional training for digital technology specialists to teach the educational module in higher education institutions has been characterized. An analysis of the characteristics of objects in the field, including code analysis and refactoring, has been conducted. Three new laboratory assignments have been developed. An analysis of the requirements for personnel provision in the IT sector under conditions of societal digital transformation has been carried out.

A didactic project for a consultation session on the topic "Interfaces. Using the IComparable Interface. Interface as a Reference Data Type. Inheritance of Interfaces" within the discipline "Software Engineering" for higher education students has been developed.

Based on the main research results, a thesis presentation was published at the 8th International Conference "Students and Youth – for the Future of the Country" (Bakhmut-Kharkiv, November 14-15, 2024).

The work consists of an introduction, four chapters, conclusions, a bibliography with 52 sources, two appendices, 5 tables, and 15 figures.

Keywords: PROFESSIONAL TRAINING, LABORATORY PRACTICE, SOFTWARE ENGINEERING, PERSONNEL PROVISION, METHODOLOGICAL DEVELOPMENT

ЗМІСТ

ВСТУП

Людство у XXI столітті переживає кардинальні, різнорівневі, глобальні, трансформаційні зміни свого буття, зумовлені швидкою еволюцією цифрових технологій, комп’ютерних мереж.

Використання сучасних технологій в освіті має відігравати ключову роль у створенні необхідних умов для саморозвитку всіх суб’єктів навчальної діяльності, активізації когнітивних та творчих процесів, формуванню необхідних компетентностей.

З позицій сьогодення затребуваними на ринку праці є компетентні, відповідальні, конкурентоздатні фахівці, котрі володіють критичним мисленням, є ініціативними, креативними, соціально активними й професійно мобільними, грамотно й свідомо використовують засоби інформаційно-комунікаційних технологій у професійній діяльності. Нині цифрові технології є інструментом, який активно використовується в освіті й сприяє підвищенню її якості.

Розвиток та вдосконалення системи професійної підготовки фахівців з цифрових технологій спрямовано на створення умов для забезпечення їх професійного зростання, а також здатності професійної підготовки фахівців з цифрових технологій, самостійно вирішувати професійні проблеми та формування спрямованості на досягнення вершин професійної діяльності.

Аналіз літератури показав, що питання пов'язані з теоретико-методологічним обґрунтуванням професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#» у закладах вищої освіти не отримали необхідного відображення в ній.

Необхідність дослідження й вирішення проблеми та вивчення педагогічного досвіду показали зумовлені наявні суперечності між:

– державними вимогами щодо підготовки фахівців з інформаційних технологій в педагогічних закладах вищої освіти та недостатньою

розробленістю цього питання у педагогічній теорії та практиці;

– стрімким розвитком електронних освітніх ресурсів і необхідністю дослідження та розроблення підходів і методів управління електронними освітніми ресурсами.

Теоретико-методологічний базис вітчизняної системи професійної освіти у різних сферах підготовки фахівців представлений у дослідженнях Е. Зеєра, Е. Клімова, Л. Мардахаєва, В. Сластеніна та ін; необхідність історичного аналізу проблеми професіоналізму спеціалістів різного профілю зумовлена роботами В. Жукова, В. Морозова, В. Ніколаєнко, Т. Харламової та ін; компетентнісний підхід у професійній освіті – В. Байденко, В. Лебедєв, Ю. Татур, Т. Шамова та ін; концепції акмеологічного підходу до професійної освіти - Б. Ананьєва, А. Бодальова, А. Деркача, Л. Лаптєва та ін; специфіка освіти дорослих людей – А. Вербицького, С. Вершловського, М. Громкової, С. Змєєва, В. Подобєда, Є. Тонконоговий та ін; моделі професійної освіти та її аксіологічні основи – Е. Артамонової, О. Воленко, Н. Кузьміної, А. Маркової, Н. Нікітіної, Н. Романенко та ін; професійно-творчий розвиток особистості – М. Віленського, Т. Іванюк, В. Кан-Каліка та ін; інженерна діяльність, з погляду етичної та соціальної відповідальності – А. Васенкіна, А. Хунінга та ін; порівняльний аналіз вітчизняної та зарубіжної професійної підготовки фахівців – А. Мітіної, Р. Хайрул-Ліна та ін; проблеми безперервної професійної освіти та освіти дорослих за кордоном – Л. Андерсона (L.Andersson), Ж. Бьорнавола (J. Bjornavol), Е. Холтона (E. Holton), М. Ноулза (M. Knowles), Р. Сво-Унсона (R. Swanson), Б. Менсфілда (B. Mansfield) та ін.

Отже, недостатня розробленість проблеми зумовили вибір теми дослідження: «Професійна підготовка фахівців з цифрових технологій для викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#» у закладах вищої освіти.

Мета дослідження - теоретично обґрунтувати та частково перевірити методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Робота з інтерфейсами, колекціями та

індексаторами на мові C#» у закладах вищої освіти.

Завдання дослідження:

1. Визначити ступінь актуальності проблеми професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#» у закладах вищої освіти.

2. Теоретично обґрунтувати, розробити методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#» у закладах вищої освіти.

Об'єкт дослідження – процес професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#» у закладах вищої освіти.

Предмет дослідження – методика професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#» у закладах вищої освіти з метою удосконалення їхнього професіоналізму у системі освіти.

Методи дослідження:

– загальнонаукові (аналіз, синтез, систематизація, зіставлення, узагальнення) з метою систематизації теоретичних ідей та узагальнення досвіду формування професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#» у закладах вищої освіти;

– емпіричні (тестування, опитування), обсерваційні (анкетування) для діагностування рівня сформованості професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#» у закладах вищої освіти.

Наукова новизна одержаних результатів дослідження полягає в:

– уточненні сутності поняття «професійна підготовка», наукові уявлення про сутність та структуру процесу професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#» у закладах вищої освіти;

– подальшого розвитку набули зміст, форми та методи організації процесу професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#» у закладах вищої освіти.

Теоретичне та практичне значення одержаних результатів полягає в розробці методики професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#» у закладах вищої освіти, обґрунтуванні змісту, форм та методів організації процесу професійної підготовки фахівців з цифрових технологій.

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

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

РОЗДІЛ 1 АКТУАЛЬНІСТЬ ПРОФЕСІЙНОЇ ПІДГОТОВКИ ФАХІВЦІВ З ЦИФРОВИХ ТЕХНОЛОГІЙ ДЛЯ ВИКЛАДАННЯ ОСВІТНЬОГО МОДУЛЯ «РОБОТА З ІНТЕРФЕЙСАМИ, КОЛЕКЦІЯМИ ТА ІНДЕКСАТОРАМИ НА МОВІ C#» У ЗАКЛАДАХ ВИЩОЇ ОСВІТИ

Сучасна освіта стає відкритою та безперервною, що відкриває нові можливості для розвитку як учнів, так і викладачів. Це стає можливим завдяки впровадженню нових технологій, методик і концепцій, які сприяють створенню умов для адаптації освіти до змінюваних соціально-економічних реалій. Нова філософія вищої школи вимагає надання студенту можливостей для формування власної освітньої траєкторії, що враховує його інтереси, здібності та майбутні професійні вимоги. Крім того, важливим є забезпечення діалогічної позиції учасників навчального процесу, адже успішне навчання можливе лише за умови активної взаємодії між студентами та викладачами. Важливо також змінити підхід до навчання: від інформаційно-репродуктивного до продуктивного та активно-творчого, де студенти не лише отримують знання, а й набувають практичних навичок, працюючи над реальними проектами та завданнями.

Враховуючи, що професійна складова займає значну частину у підготовці фахівців з цифрових технологій, виникає велика потреба в розробці методичної системи професійної підготовки, яка включає в себе не лише навчання теоретичних знань, але й активне застосування їх на практиці. Тут важливу роль відіграє наукове обґрунтування теоретичних і методичних принципів, способів практичної реалізації та конфігураційних рішень, адже ефективна система підготовки неможлива без чіткої теоретичної бази та системи методичних рекомендацій. У результаті цього процесу, майбутні фахівці отримують не лише конкретні знання і навички, але й здатність до адаптації та саморозвитку у професійній сфері, що є особливо важливим в умовах стрімкого розвитку технологій.

На сьогоднішній день теоретичні та методологічні засади модернізації професійної підготовки фахівців з цифрових технологій розроблені ще недостатньо, і ця проблема фактично є малодослідженою. Зокрема, немає достатньо досліджень, які б охоплювали всі аспекти цієї проблеми, включаючи застосування новітніх підходів до навчання та підготовки кадрів у цифровій сфері.

Ґрунтовною основою вивчення та розв'язання проблеми дослідження теоретичних та методичних засад підготовки фахівців з цифрових 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 цифрових технологій в освіті (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.); різні аспекти професійної підготовки майбутніх інженерів-педагогів (Е. Абільтарова, І. Васильєва, Н. Волкова, Н. Брюханова, Р. Горбатюк, О. Коваленко, В. Кулешова, В. Мальована, М. Лазарєв, С. Хоменко, Л. Штефан та ін.); процес формування комунікативної компетентності та її складових у здобувачів освіти закладів інженерної та інженерно-педагогічної освіти (Т. Калініченко, К. Ковальова, В. Кручек та ін.).

Аналіз наукових джерел показує, що окремі питання, пов'язані з професійною підготовкою фахівців з цифрових технологій, потребують

подальших досліджень. Зокрема, в науковій літературі відзначається, що недостатньо уваги приділяється теоретичним і практичним аспектам використання компетентнісного підходу в освіті [37]. Компетентнісний підхід передбачає формування не лише знань і навичок, а й умінь застосовувати їх у реальних умовах. Цей підхід ще не отримав достатньої реалізації в професійній підготовці фахівців з цифрових технологій, і тому є важливим напрямком для подальших наукових досліджень. Також важливою є проблема адаптації освітніх програм до вимог ринку праці, де технології постійно змінюються, а фахівці мають бути готовими швидко реагувати на ці зміни.

Це ускладнюється низкою суперечностей, які потребують вирішення. Наприклад, існує певний розрив між вимогами суспільства до якості професійної підготовки майбутніх фахівців з цифрових технологій та недостатнім рівнем забезпечення відповідними педагогічними умовами в закладах вищої освіти. Крім того, проблема полягає в необхідності поетапного формування професійної компетентності студентів в процесі їхнього навчання, а також в відсутності чітких методичних рекомендацій, які б дозволяли організувати цей процес відповідно до вимог сучасного виробництва та ринку праці.

Одним із головних ресурсів розвитку освіти в умовах модернізації є система професійної підготовки, головною метою якої є не лише підготовка до нового виду професійної діяльності, а й вироблення у фахівців здібностей до постійного саморозвитку. Через свою мобільність, гнучкість, здатність до оперативної модернізації система професійної підготовки є перспективним напрямом освіти. Ключовим моментом у її модернізації є підвищення якості професійної підготовки фахівців до нового виду професійної діяльності, при цьому основною функцією виступає функція перетворення освітнього процесу в реальний механізм розвитку особистості [35, 39].

Важливим у підготовці майбутніх фахівців з цифрових технологій є формування у них поруч із професійними компетентностями також і особистісних навичок [30, 34].

Важливою складовою підготовки майбутніх фахівців з цифрових технологій є формування у них не лише професійних компетентностей, але й особистісних навичок, таких як здатність до ефективної комунікації, прийняття рішень в умовах невизначеності та вміння працювати в команді. Проблема формування професійної компетентності фахівців з цифрових технологій у вищих навчальних закладах потребує детальнішого вивчення, зокрема, відсутність єдності термінології щодо методики формування цих компетентностей, не визначено критеріїв і показників для оцінки їх рівня, а також відсутність ефективного методичного забезпечення цього процесу.

Зважаючи на технічний аспект підготовки фахівців з цифрових технологій, важливо також розробити методичну систему педагогічної підготовки, що базується на наукових дослідженнях теоретичних і методичних засад її реалізації. Підготовка фахівців для викладання модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#» має забезпечити високий рівень підготовки конкурентоспроможних фахівців, які будуть готові до змін і здатні успішно працювати в умовах швидкого розвитку технологій [37].

Сучасний етап розвитку системи професійної підготовки є етапом аналізу та узагальнення, наукового переосмислення, використання нових методологічних підходів до дослідження проблем у системі освіти [39].

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

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 з цифрових технологій 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.

Проблема професійної підготовки кваліфікованих фахівців у сучасних соціально-економічних умовах пред'являє до системи професійної підготовки фахівців з цифрових технологій для організації та ведення ділової документації засобами комп’ютерних технологій низку вимог, зумовлених тим, що економічна діяльність у ринкових умовах стає дедалі сконцентрованою на людині, оскільки від володіння фахівцями технологіями роботи з людьми безпосередньо залежить її ефективність.

Тому роль процесу формування професійної компетентності фахівців для організації та ведення ділової документації засобами комп’ютерних технологій у процесі професійної підготовки фахівців з цифрових технологій, що виступає як складова професійної культури, надзвичайно висока, оскільки саме вона визначає соціальну орієнтацію особистості в контексті професійної діяльності, що відображає всі її компоненти: сукупність знань, умінь, сформовану особисту та професійну позицію, особисту та професійну самооцінку, що визначає активне, творче ставлення до діяльності; сукупність особистісних та професійно важливих якостей особистості, що визначають успішність професійної діяльності спеціаліста, його самореалізацію [41].

Для підвищення ефективності освітнього процесу в системі професійної підготовки і відповідності сучасним вимогам модернізації освіти, потрібно враховувати сучасні інтерпретації принципу неперервності з концептуальним осмисленням специфічної природи нового виду професійної діяльності.

Виходячи із зазначеного вище, з урахуванням сучасних тенденцій та процесів, пов'язаних з розвитком професійної підготовки фахівців з технологій, особистісної потреби у професійній підготовці фахівців, можна

виділити проблему: підвищення рівня ефективності професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#» у закладах вищої освіти.

Отже, ефективна професійна підготовка фахівців з цифрових технологій для викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#» у закладах вищої освіти полягає в окресленні, відповідно до вимог сучасного ринку праці, цілей та завдань, організаційних навчальних методів та засобів.

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 «Робота з інтерфейсами, колекціями та індексаторами на мові 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 «Робота з інтерфейсами, колекціями та індексаторами на мові C#» у закладах вищої освіти є актуальною у професійній педагогіці.

Проаналізовано ступінь актуальності проблеми професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#».

Охарактеризовано систему професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#» у закладах вищої освіти.

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

2.1 Огляд теоретичних відомостей

Інтерфейс в C# – це контракт, який визначає набір методів, властивостей, індексаторів та подій, які клас або структура має реалізувати. Інтерфейси не можуть містити реалізацію методів.

Клас, що реалізує інтерфейс, повинен реалізувати всі методи та властивості, визначені в цьому інтерфейсі. Інтерфейси можуть наслідувати інші інтерфейси. Клас, що реалізує похідний інтерфейс, повинен реалізувати всі члени як базового, так і похідного інтерфейсів.

Приклад оголошення інтерфейсу:

```
public interface IMyInterface
{
    void Method1();
    int Property1 { get; set; }
}
```

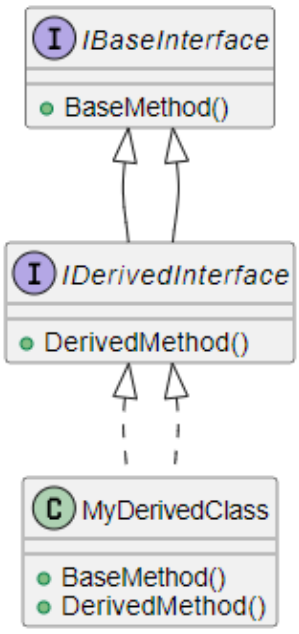

Рисунок 2.1 – Приклад UML-діаграми наслідування інтерфейсів

Узагальнені колекції дозволяють створювати типобезпечні структури даних, які забезпечують високопродуктивну роботу з даними. Наприклад, List<T>, Dictionary<TKey, TValue>, Queue<T>, Stack<T> та інші.

Ось деякі з найпоширеніших узагальнених колекцій, які надає .NET:

1. List<T>: Динамічний список, який можна змінювати за розміром. Використовується для зберігання елементів одного типу.

2. Dictionary<TKey, TValue>: Колекція пар ключ-значення, що забезпечує швидкий доступ до значень за їх ключами.

3. Queue<T>: Колекція, що реалізує принцип черги (FIFO — перший прийшов, перший пішов). Елементи додаються в кінець черги та вилучаються з початку.

4. Stack<T>: Колекція, що реалізує принцип стеку (LIFO — останній прийшов, перший пішов). Елементи додаються та вилучаються з верхівки стеку.

Індексатори є синтаксичною зручністю, що дозволяє створювати клас, структуру або інтерфейс, доступ до якого клієнтські програми отримують як до масиву. Найчастіше індексатори реалізуються для доступу до закритої внутрішньої колекції або закритого масиву. Разом з модифікаторами доступу індексатори реалізують механізм інкапсуляції для полів-масивів та є аналогами властивостей, що визначаються для звичайних полів.

Індексатори визначаються за допомогою ключового слова this. Вони можуть мати параметри, як методи, але викликаються за допомогою квадратних дужок, як масиви. Індексатори можуть бути як для читання, так і для запису.

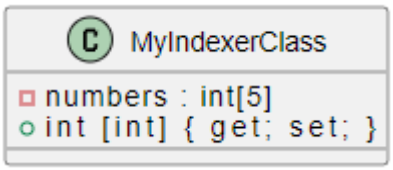

Рисунок 2.2 – Приклад UML-діаграми класу з індексатором

Приклад оголошення індексатора:

```
public class AClass1
{
    private int[] imyArray = new int[20];
    public int this[int ind1] //індексатор
    {
        get
```

```
        { return imyArray[ind1]; }
        set
        { imyArray[ind1] = value; }
    }
}
```

Параметризовані класи – класи, дозволяють визначити тип своїх аргументів при безпосередньому створенні об'єктів, але необхідно стежити, щоб операції, що використовуються для типу-параметра, були визначені для всіх типів або використовувати механізми перетворення типів.

```
public class AClass1<T>
{
    private T[] imyArray = new T[20];
}
public class M
{
    static voidMain(string[] args)
    {
        AClass1<string> K1 = new AClass1<string>();
        AClass1<int> K2 = new AClass1<int>();
    }
}
```

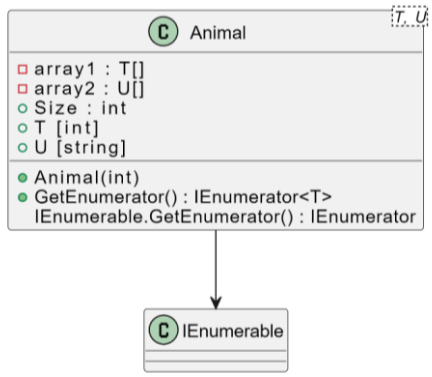

Рисунок 2.3 – UML-діаграма ієрархії класів

Ітератори дозволяють зручно перераховувати елементи колекції, використовуючи ключове слово yield. Вони значно спрощують реалізацію інтерфейсу IEnumerable або IEnumerable<T>.

Приклад використання ітератору:

```
public class MyCollection
{
    private int[] numbers = { 1, 2, 3, 4, 5 };

    public IEnumerator<int> GetEnumerator()
    {
        foreach (int number in numbers)
        {
            yield return number;
        }
    }
}
```

2.2 Постановка лабораторної роботи «Інтерфейси. Наслідування інтерфейсів»

Мета виконання роботи: ознайомитися з концепцією інтерфейсів в мові програмування C#. Навчитися використовувати інтерфейс IComparable для порівняння об'єктів. Розглянути інтерфейс як посилальний тип даних і особливості наслідування інтерфейсів.

Постановка завдання. На основі опису ієрархії класів індивідуального варіанту виконайте наступні дії.

1. Створіть інтерфейс з методами, що притаманні всім похідним класам.

2. Створіть класи-нащадки, які реалізують інтерфейс IComparable для можливості порівняння об'єктів за певними критеріями.

3. У клієнтському коді створіть список об’єктів одного з базових класів, відсортуйте його за допомогою методу Sort з використанням інтерфейсу IComparable та виведіть результат.

4. Створіть інтерфейс ILogger, який оголошує метод для виводу повідомлень. Створіть похідні від інтерфейсу класи ConsoleLogger та FileLogger, які реалізують методи виводу системних повідомлень у консоль. У клієнтському коді створіть об'єкти обох класів та продемонструйте їх використання як посилальних типів даних.

5. Створить новий інтерфейс, який спадкує структуру іншого інтерфейсу або декількох інтерфейсів. Створіть об'єкт цього класів та продемонструйте особливості процесу наслідування інтерфейсів.

6. Побудувати UML-діаграму ієрархії класів за допомогою сервісу Draw.io за синтаксисом PlantUML.

7. Проведіть тестування консольного додатку з фіксацією результатів скріншотами.

8. У звіті навести опис предметної області, UML-діаграму ієрархії класів, опис класів та їх атрибутів, програмний код на мові C# з коментарями, приклад виконання програми у IDE Visual Studio.

Приклад №1 виконання лабораторної роботи. Опис предметної області: створити ієрархію класів для збереження геометричних фігур та їх властивостей для реалізація обчислень площі, периметра фігур, а також сортування цих фігур за певними критеріями. Програма включає можливості логування, щоб відстежувати інформацію про дії та результати обчислень.

На рис. 2.4 наведено UML-діаграму класів, що відповідає завданню лабораторної роботи для предметної області.

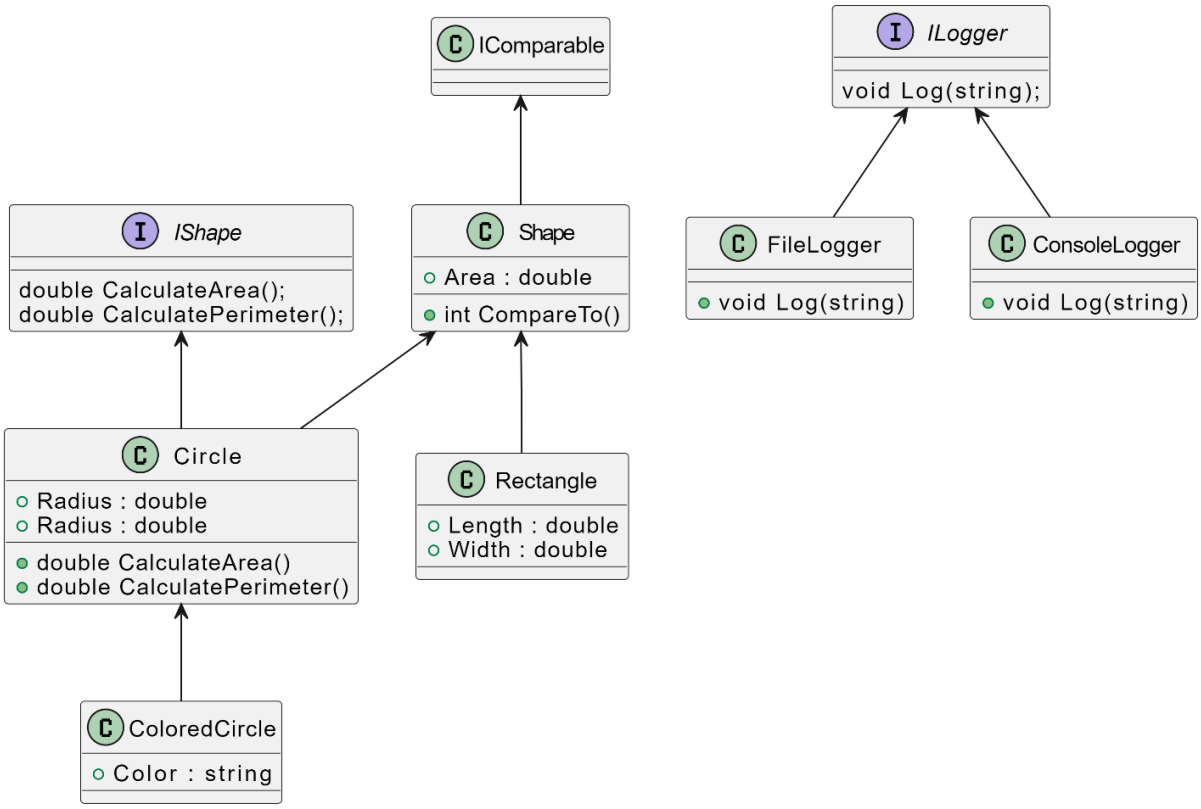

Рисунок 2.4 – UML-діаграма ієрархії класів

Далі наведено описи класів та їх атрибутів.

Інтерфейс IShape визначає загальні методи для обчислення площі та периметра фігур. Використовується для забезпечення уніфікованого способу роботи з різними типами фігур. Оголошує методи: CalculateArea()для обчислення площі фігури, CalculatePerimeter()для обчислення периметра фігури.

Інтерфейс ILogger для логування повідомлень. Використовується для

створення різних видів логерів, що можуть зберігати або виводити інформацію про події у програмі. Метод Log(string message) для запису повідомлення в лог.

Абстрактний базовий клас Shape представляє загальні характеристики для всіх геометричних фігур. Використовується для визначення спільних властивостей та методів, які будуть успадковуватися підкласами. Властивості класу: Area представляє площу фігури, Perimeter представляє периметр фігури. Метод CompareTo(Shape other) – порівняння двох фігур за площею, реалізує інтерфейс IComparable<Shape>.

Клас Circle представляє коло і успадковує базовий клас Shape. Він містить специфічні для кола атрибути та методи. Властивості класу: Radius: представляє радіус кола, Area повертає площу кола, Perimeter повертає довжину кола. Методи: CalculateArea() обчислює та повертає площу кола, CalculatePerimeter() обчислює та повертає периметр.

Клас ColoredCircle є розширенням класу Circle і додає додаткову характеристику кольору. Властивість Color представляє колір кола.

Клас Rectangle представляє прямокутник і успадковує базовий клас Shape. Властивості класу: Length довжина прямокутника, Width ширина прямокутника, Area площа прямокутника, обчислену за формулою, Perimeter повертає периметр прямокутника. Методи: CalculateArea() обчислює та повертає площу кола, CalculatePerimeter() обчислює та повертає периметр.

Клас FileLogger реалізує інтерфейс ILogger і забезпечує логування повідомлень у файл. Методи Log(message) виводить повідомлення, яке повинно було б бути записане у файл.

Клас ConsoleLogger реалізує інтерфейс ILogger і забезпечує логування повідомлень на консоль. Методи Log(string) виводить повідомлення на консоль.

Програмний код на мові C# для виконання завдань лабораторної роботи:

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

// Базовий клас Shape з реалізацією IComparable
public abstract class Shape : IComparable<Shape>
{
```

```
    public abstract double Area { get; }
    public abstract double Perimeter { get; }

    public int CompareTo(Shape other)
    {
        // Порівняння за площею
        return Area.CompareTo(other.Area);
    }
}

// Клас Circle, що успадковує Shape та реалізує IShape
public class Circle : Shape, IShape
{
    public double Radius { get; set; }

    public override double Area => Math.PI * Math.Pow(Radius, 2);
    public override double Perimeter => 2 * Math.PI * Radius;

    public double CalculateArea() => Area;
    public double CalculatePerimeter() => Perimeter;
}

// Клас ColoredCircle, що успадковує Circle
public class ColoredCircle : Circle
{
    public string Color { get; set; }
}

// Клас Rectangle, що успадковує Shape та реалізує IShape
public class Rectangle : Shape, IShape
{
    public double Length { get; set; }
    public double Width { get; set; }

    public override double Area => Length * Width;
    public override double Perimeter => 2 * (Length + Width);

    public double CalculateArea() => Area;
    public double CalculatePerimeter() => Perimeter;
}

// Інтерфейс IShape для обчислення площі та периметра
public interface IShape
{
    double CalculateArea();
    double CalculatePerimeter();
}

// Інтерфейс ILogger для логування
public interface ILogger
{
    void Log(string message);
}

// Клас FileLogger, що реалізує ILogger
public class FileLogger : ILogger
{
    public void Log(string message)
    {
        // Логування в файл (зараз виводиться на консоль для прикладу)
        Console.WriteLine($"Log to file: {message}");
    }
}

// Клас ConsoleLogger, що реалізує ILogger
```

```
public class ConsoleLogger : ILogger
{
    public void Log(string message)
    {
        // Логування на консоль
        Console.WriteLine($"Log to console: {message}");
    }
}

// Головна програма
class Program
{
    static void Main()
    {
        // Створення списку фігур
        List<Shape> shapes = new List<Shape>
        {
            new Circle { Radius = 5 },
            new Rectangle { Length = 4, Width = 6 },
            new ColoredCircle { Radius = 3, Color = "Red" }
        };

        // Сортування фігур за площею
        shapes.Sort();

        // Логування результатів за допомогою ConsoleLogger
        ILogger logger = new ConsoleLogger();

        // Виведення результату
        foreach (var shape in shapes)
        {
            logger.Log($"Shape Type: {shape.GetType().Name}, Area: {shape.Area},
Perimeter: {shape.Perimeter}");
        }
    }
}
```

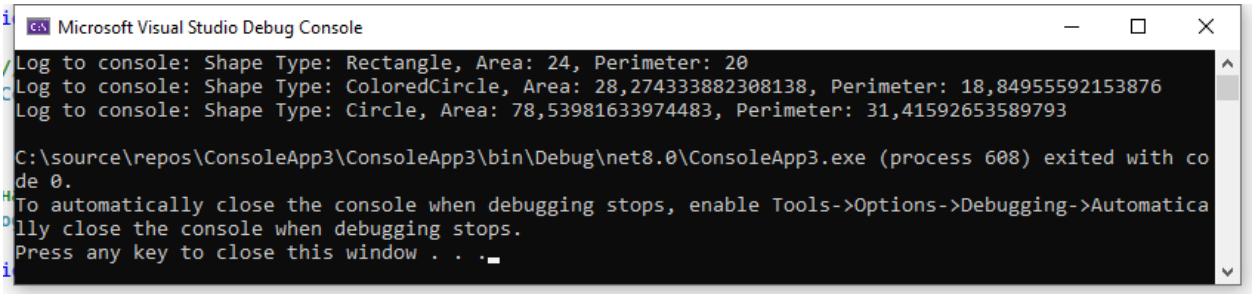

Рисунок 2.5 – Приклад виконання програми

Приклад №2 виконання лабораторної роботи. Опис предметної області: створити ієрархію класів для збереження інформації про працівників. Створити клас Person для збереження прізвища, ім’я, по батькові, дати народження, статі та ін. Створити похідний клас Emloyee з додатковими полями: табельний номер, оклад, стаж роботи, кількість відпрацьованих годин, зарплата за годину роботи. Розробити клас Payment, у закритій частині якого

розміщено список об’єктів з даними про співробітників, для яких розраховується зарплата.

На рис. 2.6 наведено UML-діаграму класів, що відповідає завданню лабораторної роботи для предметної області.

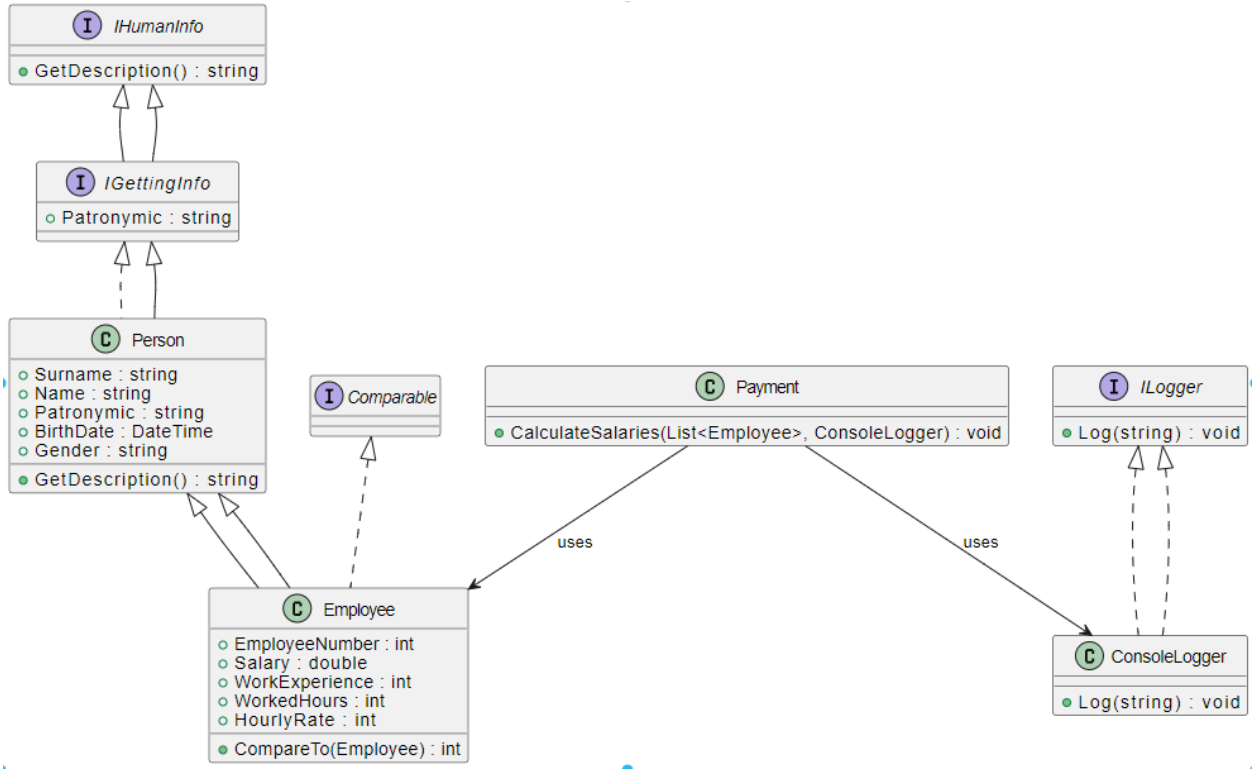

Рисунок 2.6 – UML-діаграма ієрархії класів

Далі наведено описи класів та їх атрибутів.

Інтерфейс IHumanInfo містить метод GetDescription(), який повертає опис людини.

Інтерфейс IGettingInfo наслідує IHumanInfo, містить Patronymic – властивість, що представляє по батькові людини.

Інтерфейс ILogger містить метод Log(string) для логування повідомлень.

Клас ConsoleLogger реалізує ILogger, містить методи Log(string), що виводить повідомлення на консоль

Клас Person реалізує інтерфейс IGettingInfo та містить такі атрибути: Surname – прізвище особи, Name – ім'я особи, Patronymic – по батькові особи, BirthDate – дата народження особи, Gender – стать особи. Метод GetDescription() повертає опис особи у вигляді рядка.

Клас Employee наслідує клас Person і реалізує IComparable<Employee>). Атрибути класу Employee такі: EmployeeNumber – номер працівника, Salary – заробітна плата працівника, WorkExperience – досвід роботи працівника, WorkedHours – кількість відпрацьованих годин, HourlyRate – погодинна ставка. Метод класу Employee: CompareTo(Employee) порівнює об'єкти Employee за погодинною ставкою.

Клас Payment містить метод CalculateSalaries(List<Employee> employees, ConsoleLogger logger), що обчислює зарплату співробітників та виводить інформацію про них за допомогою логування.

У класі Progfram міститься метод Main(string[] args) : void – основна точка входу програми, яка виконує ініціалізацію та обчислення зарплат працівників, а також сортує їх.

Програмний код на мові C# для виконання завдань лабораторної роботи:

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

// Інтерфейс IHumanInfo з методами, які притаманні всім нащадкам
public interface IHumanInfo
{
    string GetDescription();
}

// Створить новий інтерфейс, який спадкує структуру іншого інтерфейсу
public interface IGettingInfo : IHumanInfo
{
    public string Patronymic
    {
        get;
        set;
    }
}

// Створіть інтерфейс ILogger, який містить метод для логування повідомлень.
public interface ILogger
{
    void Log(string message);
}

/* Створіть клас ConsoleLogger, який також реалізує інтерфейс ILogger та виводить
повідомлення на консоль*/
public class ConsoleLogger : ILogger
{
    public void Log(string message)
    {
        // Логування на консоль
       Console.WriteLine($"\n{message} employee:");
    }
}
```

```
public class Person : IGettingInfo
{
    // Створюємо автоматичні властивості для класу Person
    public string Surname { get; set; }
    public string Name  { get; set; }
    public string Patronymic  { get; set; }
    public DateTime BirthDate  { get; set; }
    public string Gender  { get; set; }

    // Створюємо конструктор ініціалізації класу Person
    public Person(string surname, string name, string patronymic, DateTime
birthDate, string gender)
    {
        // Присвоюємо приватним полям їх значення, які передаються як параметри
        Surname = surname;
        Name = name;
        Patronymic = patronymic;
        BirthDate = birthDate;
        Gender = gender;
    }

    // Створюємо конструктор копіювання (робить дублікат того ж самого об'єкту)
    public Person(Person person)
    {
        Surname = person.Surname;
        Name = person.Name;
        Patronymic = person.Patronymic;
        BirthDate = person.BirthDate;
        Gender = person.Gender;
    }

    // Реалізація методу інтерфейсу, який успадкований.
    public string GetDescription()
    {
        return String.Format($"Employee: {this.Name} {this.Surname}");
    }
}

/* Похідний клас Employee від базового класу Person та інтерфейс
IComparable<Employee> для можливості порівняння об'єктів за певними критеріями*/
public class Employee : Person, IComparable<Employee>
{
    // Створюємо автоматичні властивості для класу Employee
    public int EmployeeNumber  { get; set; }
    public double Salary {
        get;
        set;
    }
    public int WorkExperience  { get; set; }
    public int WorkedHours  { get; set; }
    public int HourlyRate  { get; set; }

    /*Створюємо конструктор ініціалізації класу Employee. Ключове слово base передає
параметри у базовий клас Person*/
    public Employee(string surname, string name, string patronymic, DateTime
birthDate, string gender, int employeeNumber, double salary, int workExperience, int
workedHours, int hourlyRate) : base(surname, name, patronymic, birthDate, gender)
    {
        // Присвоюємо приватним полям їх значення, які передаються як параметри
        EmployeeNumber = employeeNumber;
        Salary = salary;
        WorkExperience = workExperience;
        WorkedHours = workedHours;
        HourlyRate = hourlyRate;
    }
```

```
    // Реалізація методу CompareTo для порівняння об'єктів за почасовою оплатою
    public int CompareTo(Employee obj)
    {
        return HourlyRate.CompareTo(obj.HourlyRate);
    }
}

public class Payment
{
    // Метод CalculateSalaries для обчислення зарплати співробітників
    public void CalculateSalaries(List<Employee> employees, ConsoleLogger logger)
    {
        // За допомогою циклу foreach робимо перебір кожного об'єкта класу Employee
        int counter = 0;
        foreach (Employee in employees)
        {
            counter++;
            // Вираховуємо зарплату співробітників
            double salary = employee.WorkedHours * employee.HourlyRate;
            // Використання методу Log для виведення інформації
            string employeeamount = counter.ToString();
            logger.Log(employeeamount);
            // Виводимо в консоль інформацію за допомогою методу інтерфейсу
            Console.WriteLine($"{employee.GetDescription()} | General salary:
{salary} | Hourly rate: {employee.HourlyRate}");
        }
    }
}

class Program
{
    static void Main(string[] args) // Вхід в головну функцію програми
    {
        // У клієнтському коді створіть список об’єктів базового класу
        List<Employee> employees = new List<Employee>
        {
            new Employee("Gratin", "Egor", "Evgenievich", DateTime.Now, "male",
20581, 2300, 3, 560, 23),
            new Employee("Tenova", "Elizabeth", "Valerievna", DateTime.Now,
"female", 18456, 2000, 2, 420, 20),
            new Employee("Nicolov", "Arthur", "Romanovich", DateTime.Now, "male",
89230, 2100, 4, 730, 18)
        };
        // Відсортуйте його за допомогою методу Sort з використанням IComparable
        employees.Sort();
        // У клієнтському коді створюємо об'єкт класу ConsoleLogger
        ConsoleLogger loggermess = new ConsoleLogger();
        // Вивід результату відсортування та даних робітників
        Payment payment = new Payment();
        payment.CalculateSalaries(employees, loggermess);
    }
}
```

2.3 Постановка лабораторної роботи «Узагальнені колекції та ітератори»

Мета роботи: ознайомитися з концепцією колекцій в мові C#; розглянути узагальнені колекцій та ітератори.

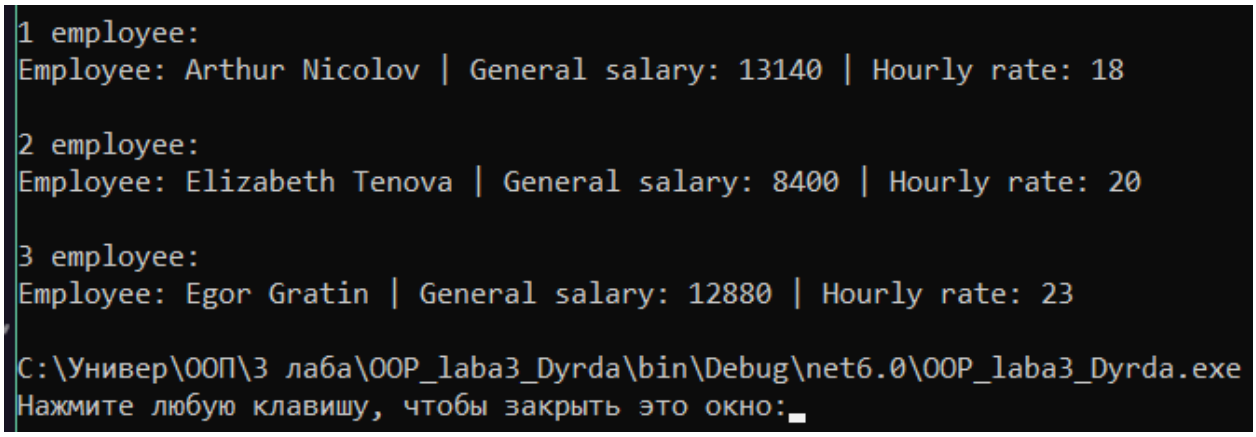

Рисунок 2.7 – Приклад виконання програми

Постановка завдання. На основі опису ієрархії класів індивідуального варіанту виконайте наступні дії.

1. Створіть консольний застосунок, який демонструє використання різних інтерфейсів узагальнених колекцій у C#:

– створіть список (List<T>) рядків і виведіть його вміст у консоль;

– створіть словник (Dictionary<TKey, TValue>) з ключами та значеннями типу int; додайте кілька елементів і виведіть їх у консоль;

– створіть чергу (Queue<T>) з об'єктами вашого вибору; додайте та видаліть елементи та виведіть змінений стан черги;

– створіть стек (Stack<T>) з об'єктами вашого вибору; додайте та видаліть елементи та виведіть змінений стан стеку.

2. Створіть окремий клас для керування об'єктами конкретного типу, використовуючи класи узагальнених колекцій:

– створіть базовий клас свого варіанту з необхідними властивостями;

– створіть клас узагальненої колекції, який буде містити колекцію об’єктів базового класу, додайте методи для додавання та видалення об’єктів.

– у клієнтському коді створіть екземпляр колекції, додайте декілька об’єктів та виведіть їхні дані у консоль з використанням ітератора foreach.

3. Побудувати UML-діаграму ієрархії класів за допомогою сервісу Draw.io за синтаксисом PlantUML.

4. Проведіть тестування консольного додатку з фіксацією результатів

скріншотами.

5. У звіті навести опис предметної області, UML-діаграму ієрархії класів, опис класів та їх атрибутів, програмний код на мові C# з коментарями, приклад виконання програми у IDE Visual Studio.

Приклад №1 виконання лабораторної роботи. На прикладі класів «Книга» та «Бібліотека» продемонструвати роботу з узагальненими колекціями (List<T>, Dictionary<TKey, TValue>, Queue<T>, Stack<T>), а потім з класами узагальнених колекцій (Book і Library).

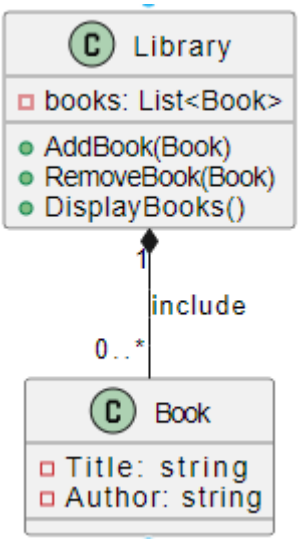

Рисунок 2.8 – UML-діаграма ієрархії класів

Далі наведено описи класів та їх атрибутів.

Клас Book представляє модель книги і містить інформацію про неї. Клас Book має два атрибути: Title (string) – назва книги, Author (string) – ім'я автора книги. Ці атрибути реалізовані як властивості з методами доступу get і set, що дозволяє отримувати та встановлювати значення назви та автора книги.

Клас Library представляє бібліотеку, яка містить колекцію книг і забезпечує методи для роботи з цією колекцією. Клас Library містить приватний атрибут books (List<Book>) – список книг у бібліотеці, що використовується для зберігання всіх книг, доданих до бібліотеки.

Цей клас також має три методи: AddBook(Book) додає книгу до бібліотеки, RemoveBook(Book) видаляє книгу з бібліотеки, DisplayBooks()

відображає список книг у бібліотеці.

Відношення Library і Book позначено як асоціацію типу "один-до-багатьох" (1 *-- 0..*), що вказує на те, що один об'єкт Library може містити багато об'єктів Book.

Клас Program є головним класом програми, який містить метод Main(), що відповідає за виконання програми. У цьому класі демонструється робота з узагальненими колекціями та взаємодія з бібліотекою книг таким чином:

– створює і заповнює дані в узагальнених колекціях: List<T>, Dictionary<TKey, TValue>, Queue<T>, Stack<T>;

– створює екземпляри книг і додає їх до бібліотеки;

– виводить список книг, наявних у бібліотеці;

– видаляє одну книгу з бібліотеки і повторно виводить оновлений список книг.

Програмний код на мові C# для виконання завдань лабораторної роботи:

```
class Book
{
    public string Title { get; set; }
    public string Author { get; set; }
}

class Library
{
    private List<Book> books = new List<Book>();

    public void AddBook(Book book)
    {
        books.Add(book);
    }

    public void RemoveBook(Book book)
    {
        books.Remove(book);
    }

    public void DisplayBooks()
    {
        Console.WriteLine("Книги в бібліотеці:");
        foreach (var book in books)
        {
            Console.WriteLine($"Назва: {book.Title}, Автор: {book.Author}");
        }
    }
}

class Program
{
    static void Main()
    {
```

```
// Підтримка українських літер
Console.OutputEncoding = Encoding.UTF8;
// Робота з узагальненими колекціями

// 1. List<T>
Console.WriteLine("Демонстрація колекції List");
List<string> stringList = new List<string> {"Яблуко", "Банан", "Апельсин" };
Console.WriteLine("Елементи списку:");
foreach (var item in stringList)
{
    Console.WriteLine(item);
}

// 2. Dictionary<TKey, TValue>
Console.WriteLine("\nДемонстрація колекції Dictionary");
Dictionary<int, string> intStringDict = new Dictionary<int, string>
{
    { 1, "Один" },
    { 2, "Два" },
    { 3, "Три" }
};
Console.WriteLine("Елементи словника:");
foreach (var kvp in intStringDict)
{
    Console.WriteLine($"{kvp.Key}: {kvp.Value}");
}

// 3. Queue<T>
Console.WriteLine("\nДемонстрація колекції Queue");
Queue<int> intQueue = new Queue<int>();
intQueue.Enqueue(10);
intQueue.Enqueue(20);
intQueue.Enqueue(30);
Console.WriteLine("Елементи черги:");
while (intQueue.Count > 0)
{
    Console.WriteLine(intQueue.Dequeue());
}

// 4. Stack<T>
Console.WriteLine("Демонстрація колекції Stack");
Stack<double> doubleStack = new Stack<double>();
doubleStack.Push(3.14);
doubleStack.Push(2.718);
Console.WriteLine("Елементи стеку:");
while (doubleStack.Count > 0)
{
    Console.WriteLine(doubleStack.Pop());
}

// Робота з класами узагальнених колекцій
// Створення екземплярів
Console.WriteLine("\nДемонстрація класів узагальнених колекцій");
Book book1 = new Book {Title = "Ловець у житі", Author = "Дж.Д. Селінджер"};
Book book2 = new Book {Title = "Убити пересмішника", Author = "Гарпер Лі" };

Library library = new Library();

// Додавання книг до бібліотеки
library.AddBook(book1);
library.AddBook(book2);

// Відображення книг у бібліотеці
Console.WriteLine("Книги в бібліотеці після додавання:");
library.DisplayBooks();
```

```
        // Видалення книги з бібліотеки
        library.RemoveBook(book1);

        // Відображення оновлених книг у бібліотеці
        Console.WriteLine("\nКниги в бібліотеці після видалення:");
        library.DisplayBooks();
    }
}
```

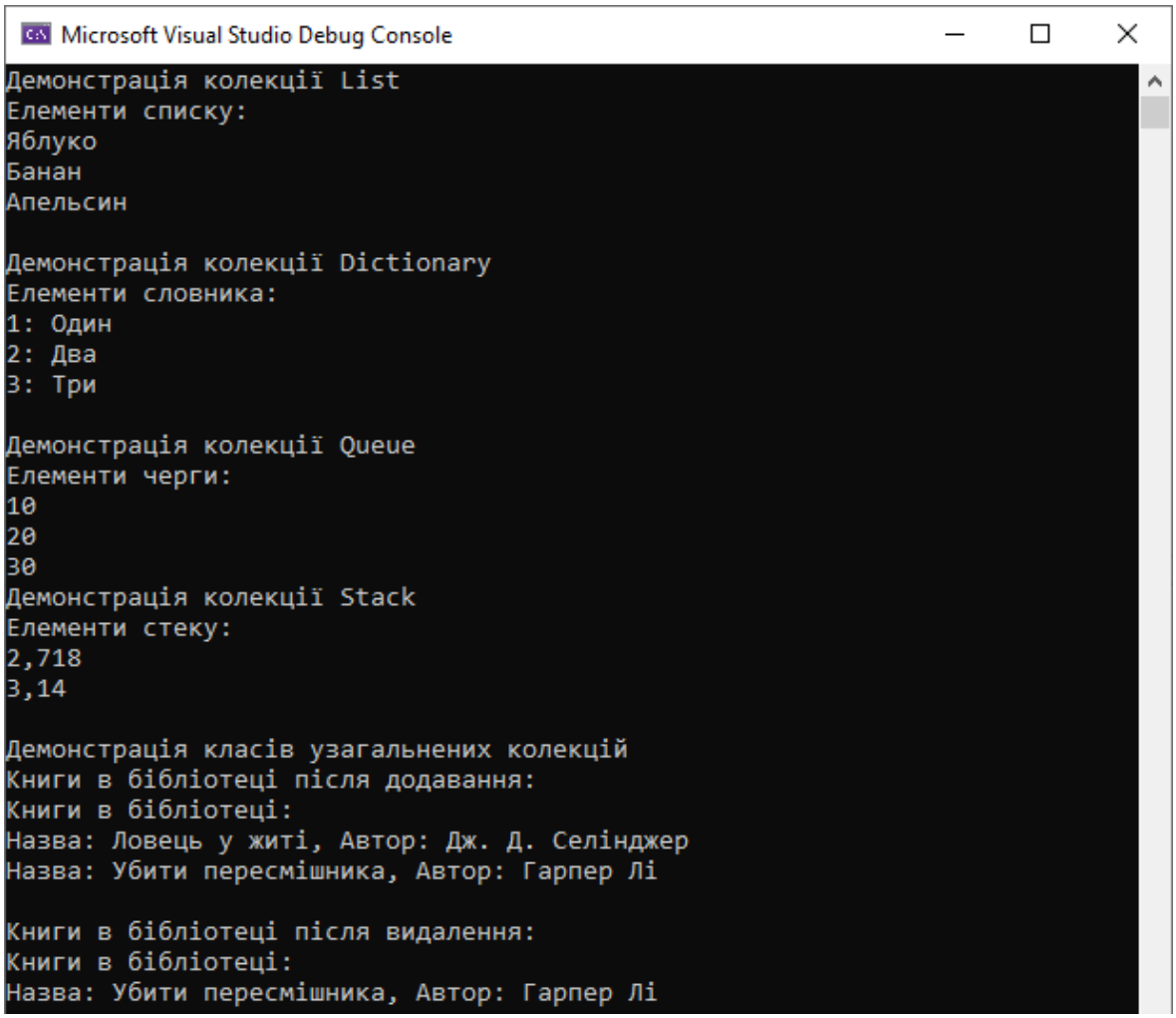

Рисунок 2.9 – Приклад виконання програми

Приклад №2 виконання лабораторної роботи. Опис предметної області: створити ієрархію класів для збереження інформації про працівників. Створити клас Person для збереження прізвища, ім’я, по батькові, дати народження, статі та ін. Створити похідний клас Emloyee з додатковими полями: табельний номер, оклад, стаж роботи, кількість відпрацьованих годин, зарплата за годину роботи. Розробити клас Payment, у закритій частині якого розміщено список об’єктів з даними про співробітників, для яких розраховується зарплата.

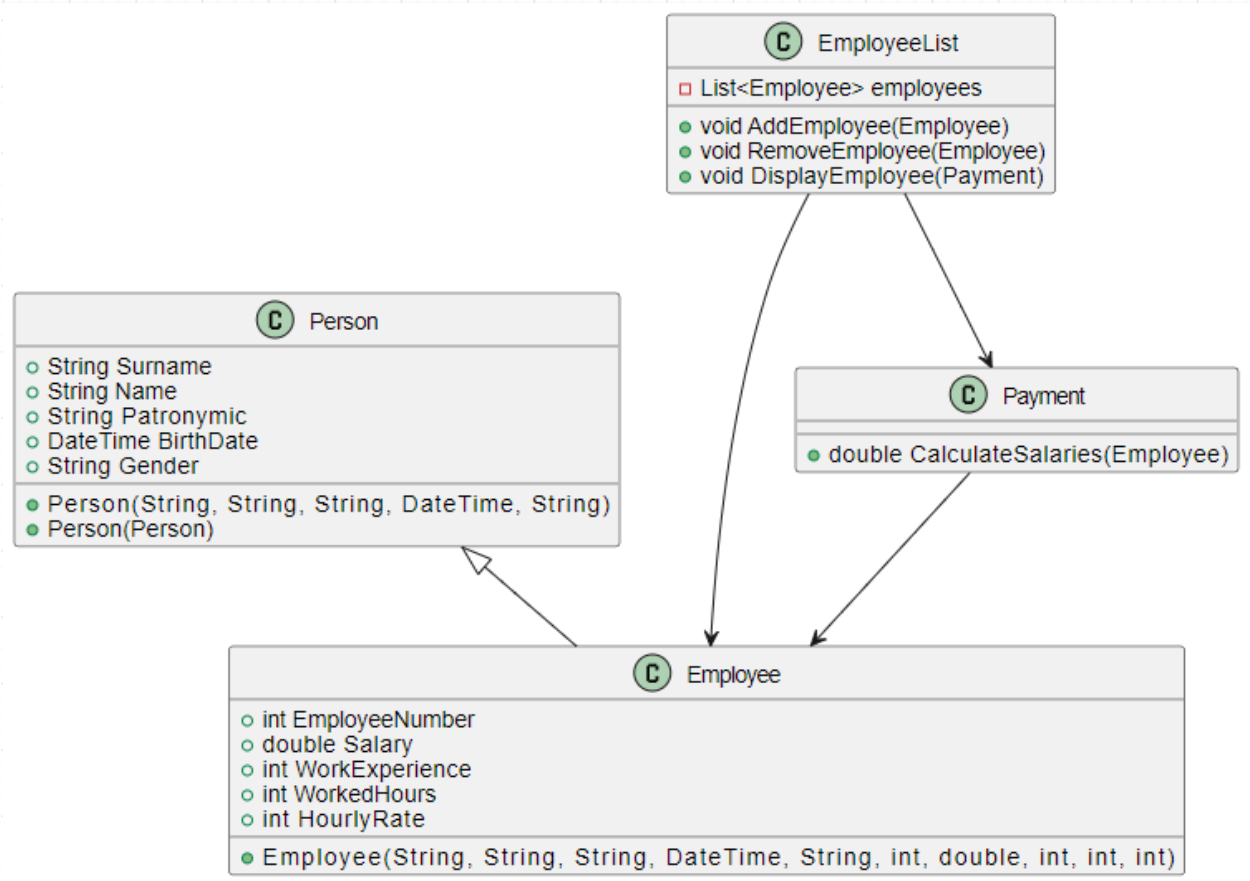

Рисунок 2.10 – UML-діаграма ієрархії класів

Далі наведено описи класів та їх атрибутів.

Клас Person є базовим класом, який представляє загальну інформацію про особу. Атрибути класу: Surname (String) – прізвище особи, Name (String) – ім'я особи, Patronymic (String) – по-батькові особи, BirthDate (DateTime) – дата народження особи, Gender (String) – стать особи. Методи класу: Person(String, String, String, DateTime, String) – конструктор для створення об'єкта з конкретними значеннями атрибутів, Person(Person) – конструктор копіювання, який створює новий об'єкт на основі існуючого.

Клас Employee є похідним від класу Person і представляє конкретного співробітника з додатковими атрибутами, які стосуються роботи. Атрибути класу: EmployeeNumber (int) – унікальний номер співробітника, Salary (double) – заробітна плата співробітника, WorkExperience (int) – кількість років робочого стажу, WorkedHours (int) – кількість відпрацьованих годин, HourlyRate (int) – оплата за годину роботи. Методи класу: Employee(String,

String, String, DateTime, String, int, double, int, int, int) – конструктор для ініціалізації об'єкта з інформацією про співробітника.

Клас Payment використовується для обчислення заробітної плати співробітників. Методи CalculateSalaries(Employee) обчислює заробітну плату співробітника на основі його відпрацьованих годин та погодинної ставки.

Клас EmployeeList є колекцією, яка управляє списком співробітників. Атрибут employees (List<Employee>) – список об'єктів співробітників. Методи класу: AddEmployee(Employee) – додає співробітника до списку, RemoveEmployee(Employee) – видаляє співробітника зі списку, DisplayEmployee(Payment) – виводить інформацію про всіх співробітників у списку, включаючи розраховану заробітну плату.

Зв'язки між класами:

1. клас Employee є спадкоємцем класу Person, що означає, що кожен співробітник є особою;

2. клас Payment використовує об'єкти Employee для розрахунку заробітної плати;

3. клас EmployeeList управляє колекцією об'єктів Employee;

4. клас EmployeeList взаємодіє з об'єктом Payment для обчислення та виведення заробітної плати співробітників.

Програмний код на мові C# для виконання завдань лабораторної роботи:

```
public class Person
{
    // Створюємо автоматичні властивості для класу Person
    public string Surname { get; set; }
    public string Name { get; set; }
    public string Patronymic { get; set; }
    public DateTime BirthDate { get; set; }
    public string Gender { get; set; }

    // Створюємо конструктор ініціалізації класу Person
    public Person(string surname, string name, string patronymic,
        DateTime birthDate, string gender)
    {
        // Присвоюємо приватним полям їх значення, які передаються як параметри
        Surname = surname;
        Name = name;
        Patronymic = patronymic;
        BirthDate = birthDate;
        Gender = gender;
    }
```

```
    // Створюємо конструктор копіювання (робить дублікат того ж самого об'єкту)
    public Person(Person person)
    {
        Surname = person.Surname;
        Name = person.Name;
        Patronymic = person.Patronymic;
        BirthDate = person.BirthDate;
        Gender = person.Gender;
    }
}

/*Похідний клас Employee від базового класу Person та інтерфейс
IComparable<Employee> для можливості порівняння об'єктів за певними критеріями*/
public class Employee : Person
{
    // Створюємо автоматичні властивості для класу Employee
    public int EmployeeNumber { get; set; }
    public double Salary { get; set; }
    public int WorkExperience { get; set; }
    public int WorkedHours { get; set; }
    public int HourlyRate { get; set; }

    /*Створюємо конструктор ініціалізації класу Employee.
     * Ключове слово base передає параметри у базовий клас Person*/
    public Employee(string surname, string name, string patronymic, DateTime
birthDate,
        string gender, int employeeNumber, double salary, int workExperience,
        int workedHours, int hourlyRate) : base(surname, name, patronymic,
birthDate, gender)
    {
        // Присвоюємо приватним полям їх значення, які передаються як параметри
        EmployeeNumber = employeeNumber;
        Salary = salary;
        WorkExperience = workExperience;
        WorkedHours = workedHours;
        HourlyRate = hourlyRate;
    }
}

public class Payment
{
    // Метод CalculateSalaries для обчислення зарплати співробітників
    public double CalculateSalaries(Employee employee)
    {
        // Вираховуємо зарплату співробітників
        double salary = employee.WorkedHours * employee.HourlyRate;
        // Повертаємо результат обчислень
        return salary;
    }
}

/*Створіть клас узагальненої колекції, який буде містити колекцію об’єктів
 * базового класу, додайте методи для додавання та видалення об’єктів*/
public class EmployeeList
{
    private List<Employee> employees = new List<Employee>();
    public void AddEmployee(Employee employee)
    {
        employees.Add(employee);
    }
    public void RemoveEmployee(Employee employee)
    {
        employees.Remove(employee);
    }
```

```
    // Додаємо параметр для методу DisplayEmployee
    public void DisplayEmployee(Payment payment)
    {
        Console.WriteLine("\nСписок працівників:");
        foreach (Employee emp in employees)
        {
            Console.WriteLine($"{emp.Surname} {emp.Name} {emp.Patronymic} є
працівником, " +
                $"{emp.Gender} | Номер співробітника: {emp.EmployeeNumber}, " +
                $"загальна зарплата: {payment.CalculateSalaries(emp)}");
        }
    }
}

class Program
{
    static void Main(string[] args) // Вхід в головну функцію програми
    {
        Console.OutputEncoding=UTF8Encoding.UTF8;
        // створіть список (List<T>) рядків і виведіть його вміст у консоль
        List<string> listEmployees = new List<string> { "Олексій", "Павло",
            "Ганна", "Марія", "Олена" };
        Console.WriteLine("Елементи списка:");
        foreach (var item in listEmployees)
        {
            Console.WriteLine(item);
        }

        /*Створіть словник (Dictionary<TKey, TValue>) з ключами та значеннями типу
int;
         * додайте кілька елементів і виведіть їх у консоль*/

        Dictionary<string, int> dictEmplExper = new Dictionary<string, int>();
        for (int i = 0; i < listEmployees.Count; i++)
        {
            dictEmplExper.Add(listEmployees[i], i * 10);
        }
        ICollection<string> keys = dictEmplExper.Keys;
        Console.WriteLine("\nЕлементи словника:");
        foreach (var key in keys)
        {
            Console.WriteLine($"{key} має {dictEmplExper[key]} годин досвіду");
        }

        /*Створіть чергу (Queue<T>) з об'єктами вашого вибору; додайте та видаліть
         * елементи та виведіть змінений стан черги*/

        Queue<int> queueBirthDate = new Queue<int>();
        int date = 2005;
        for (int i = 0; i < listEmployees.Count; i++)
        {
            queueBirthDate.Enqueue(date);
            date -= 5;
        }
        Console.WriteLine("\nЧерга дат народження:");
        foreach (var birth in queueBirthDate)
        {
            Console.WriteLine($"Дата народження - {birth}");
        }
        Console.WriteLine("\nНова черга - кому більше 25?");
        for (int i = 0; i < listEmployees.Count; i++)
        {
            int birth = queueBirthDate.Dequeue();
            if (DateTime.Now.Year - birth >= 25)
            {
```

```
                queueBirthDate.Enqueue(birth);
                Console.WriteLine(birth);
            }
        }

        /*Створіть стек (Stack<T>) з об'єктами вашого вибору; додайте та видаліть
         * елементи та виведіть змінений стан стеку*/
        Stack<int> stackSalary = new Stack<int>();
        int counter = 1000;
        for (int i = 0; i < listEmployees.Count; i++)
        {
            stackSalary.Push(counter += 500);
        }
        Console.WriteLine("\nСтек зарплати:");
        for (int i = 0; i < listEmployees.Count; i++)
        {
            Console.WriteLine(stackSalary.Pop());
        }
        try
        {
            stackSalary.Peek();
        }
        catch (InvalidOperationException)
        {
            Console.WriteLine("\nСпроба переглянути елементи в стеку, але стек
порожній");
        }
        /*У клієнтському коді створіть екземпляр колекції, додайте декілька об’єктів
та виведіть їхні дані у консоль*/
        // Створення об'єктів Employee
        Employee empl1 = new Employee("Петров", "Петро", "Петрович", DateTime.Now,
"чоловік", 20581, 2300, 3, 560, 23);
        Employee empl2 = new Employee("Петрова", "Олена", "Петрівна", DateTime.Now,
"жінка", 18456, 2000, 2, 420, 20);
        Employee empl3 = new Employee("Зеленський", "Володимир", "Олександрович",
DateTime.Now, "чоловік", 89230, 2100, 4, 730, 18);

        Payment payment = new Payment(); // Створення об'єкта Payment
        EmployeeList employeeList = new EmployeeList(); // Створення об'єкта
EmployeeList

        employeeList.AddEmployee(empl1);
        employeeList.AddEmployee(empl2);
        employeeList.AddEmployee(empl3);
        // Передача payment як параметра
        employeeList.DisplayEmployee(payment);
        employeeList.RemoveEmployee(empl3);
        employeeList.DisplayEmployee(payment); // Передача payment як параметра
    }
}
```

2.4 Постановка лабораторної роботи «Параметризовані класи та індексатори»

Мета роботи: ознайомитися з концепцією індексаторів та параметризованих класів, навчитися створювати параметризовані класи та обробляти з використанням індексаторів.

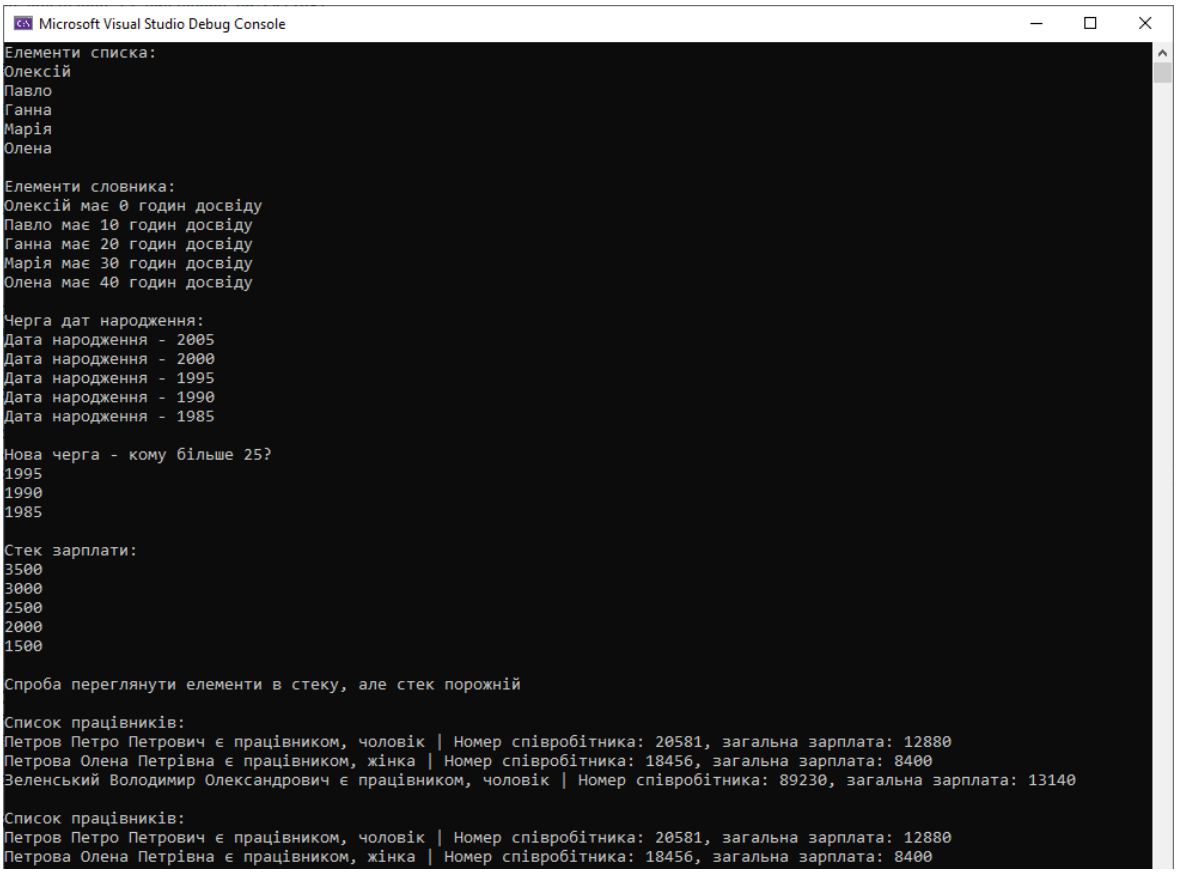

Рисунок 2.11 – Приклад виконання програми

Постановка завдання. На основі опису ієрархії класів індивідуального варіанту виконайте наступні дії.

1. До базового класу додати нього поле-масив. Розмір масиву повинен зберігатися в окремій властивості. У класі визначити індексатор для вихідного масиву. Додати ще один масив і визначити індексатор і для нього. Розробити конструктор для ініціалізації масиву.

2. У коді головної точки входу static void Main() продемонструвати роботу з об’єктами параметризованого класу: створити 2 об'єкти з різними параметрами, наприклад рядковий тип і числовий тип, вивести у консоль елементи масивів через індексатор за допомогою циклу foreach.

3. Побудувати UML-діаграму ієрархії класів за допомогою сервісу Draw.io за синтаксисом PlantUML.

4. Проведіть тестування консольного додатку з фіксацією результатів скріншотами.

5. У звіті навести опис предметної області, UML-діаграму ієрархії класів, опис класів та їх атрибутів, програмний код на мові C# з коментарями, приклад виконання програми у IDE Visual Studio.

Приклад №1 виконання лабораторної роботи. Опис предметної області: на прикладі класу «Тварина» продемонструвати роботу з полями-масивами та індексаторами.

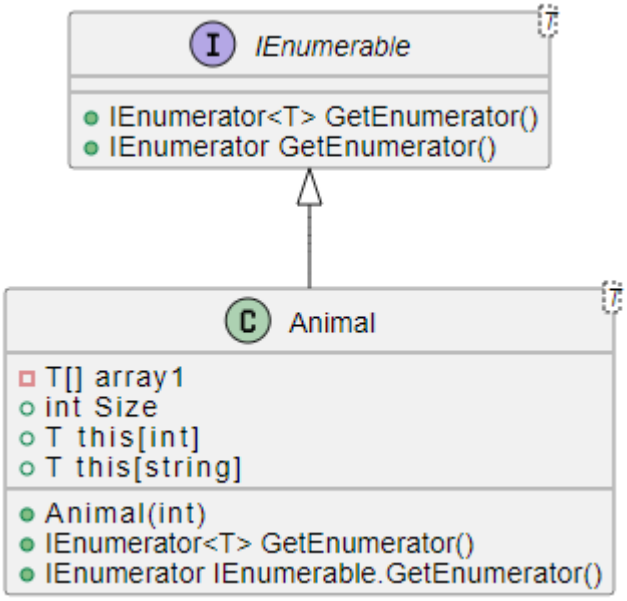

Рисунок 2.12 – UML-діаграма ієрархії класів

Далі наведено описи класів та їх атрибутів.

Клас Animal<T> – це типизований клас, який представляє колекцію тварин або значень, пов'язаних з тваринами. Він використовує параметр типу T, щоб зберігати дані будь-якого типу (зокрема назви тварин або кількість ніг). Клас реалізує інтерфейс IEnumerable<T>, що дозволяє перебирати його елементи за допомогою циклів foreach.

Атрибути класу: array1 – приватне поле T[], яке є масивом типу T. Воно використовується для зберігання даних про тварин або значень, пов'язаних з тваринами. Властивість Size повертає розмір масиву array1. Вона є тільки для читання і визначається під час створення об'єкта класу Animal<T>. Індексатор T[int index] надає доступ до елементів масиву array1 за індексом типу int. Він використовується для отримання або встановлення значення елемента масиву

за допомогою індексу. Індексатор T [string index] надає доступ до елементів масиву array1 за індексом типу string. Метод GetEnumerator()реалізує інтерфейс IEnumerable<T>. Він повертає ітератор, який дозволяє перебирати елементи масиву array1 за допомогою циклу foreach. Метод IEnumerable.GetEnumerator() реалізує інтерфейс IEnumerable. Він викликає типізовану версію методу GetEnumerator і дозволяє використовувати об'єкти класу Animal<T> в циклах foreach, не вказуючи конкретний тип.

Програмний код на мові C# для виконання завдань лабораторної роботи:

```
using System.Collections;
using System.Collections.Generic;
// Генеричний клас Animal<T> для зберігання масиву значень типу T
class Animal<T> : IEnumerable<T>
{
    // Приватне поле для елементів масиву (назви тварин або кількість ніг)
    private T[] array1;
    // Властивість для зберігання розміру масиву
    public int Size { get; }
    // Конструктор, який ініціалізує масив заданого розміру
    public Animal(int size)
    {
        Size = size;
        array1 = new T[Size];
    }

    // Індексатор для доступу до елементів масиву за індексом типу int
    public T this[int index]
    {
        get { return array1[index]; }
        set { array1[index] = value; }
    }
    // Індексатор для доступу до елементів масиву за індексом типу string
    public T this[string index]
    {
        get { return array1[int.Parse(index)]; }
        set { array1[int.Parse(index)] = value; }
    }

    // Метод для ітератора перебору елементів масиву
    public IEnumerator<T> GetEnumerator()
    {
        foreach (var item in array1)
        {
            yield return item; // Повертає кожний елемент масиву по черзі
        }
    }
    // Реалізація інтерфейсу IEnumerable
    IEnumerator IEnumerable.GetEnumerator()
    {
        return GetEnumerator(); // Викликає реалізацію типізованого ітератора
    }
}

class Program
{
    static void Main(string[] args)
```

```
    {
        // Створення об'єкта Animal для зберігання назв тварин
        Animal<string> animal1 = new Animal<string>(3);
        animal1[0] = "Павук";
        animal1[1] = "Собака";
        animal1[2] = "Пташка";
        Console.WriteLine("Елементи першого масиву:");
        foreach (var item in animal1)
        {
            Console.WriteLine(item); // Виведення елементів масиву назв тварин
        }
        // Створення об'єкта Animal для зберігання кількості ніг у тварин
        Animal<int> animal2 = new Animal<int>(3);
        animal2[0] = 6; // Павук має 6 ніг
        animal2[1] = 4; // Собака має 4 ноги
        animal2[2] = 2; // Пташка має 2 ноги

        Console.WriteLine("Елементи другого масиву:");
        foreach (var item in animal2)
        {
            Console.WriteLine(item); // Виведення елементів масиву кількості ніг
        }
    }
}
```

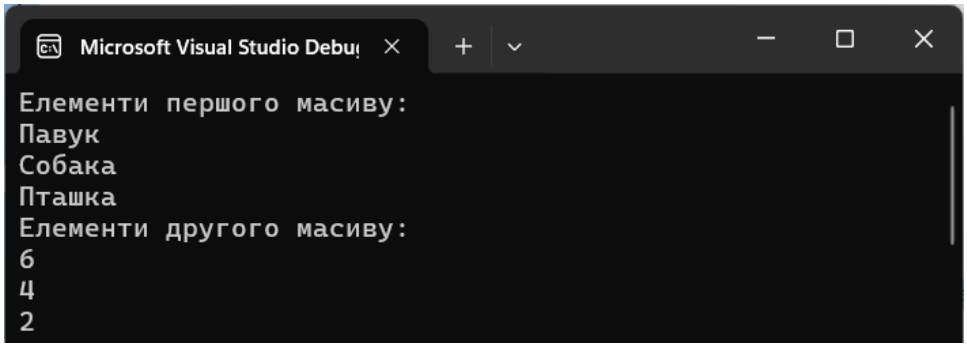

Рисунок 2.13 – Приклад виконання програми

Приклад №2 виконання лабораторної роботи. Опис предметної області: на ієрархії класів для збереження інформації про працівників продемонструвати роботу з полями-масивами та індексаторами.

Далі наведено описи класів та їх атрибутів.

Класи Person<T> і Employee<T> є узагальненими, тобто вони використовують параметр типу T для визначення типу елементів, що зберігаються в масиві array. Це робить ці класи універсальними і дозволяє працювати з будь-яким типом даних.

Клас Person містить автоматичні властивості, що зберігають: Surname – прізвище людини, Name – ім'я людини, Patronymic – по батькові людини, BirthDate – дату народження, Gender – стать людини, Size – розмір масиву.

Приватні поля класу: array – масив типу T, який використовується для зберігання даних. Індексатори this[] дозволяють отримувати або встановлювати значення в масиві array за індексом типу int або string.

Ітератори GetEnumerator() – типізований ітератор для обходу елементів масиву array та IEnumerable.GetEnumerator() – нетипізований ітератор для підтримки інтерфейсу IEnumerable, що також повертає типізований ітератор.

Клас Employee містить автоматичні властивості, що зберігають: EmployeeNumber – номер співробітника, Salary – розмір заробітної плати, WorkExperience – досвід роботи співробітника в роках, WorkedHours – кількість відпрацьованих годин, HourlyRate – погодинну ставку співробітника.

Ця структура класів забезпечує гнучкість та розширюваність для створення об'єктів, що представляють як загальних осіб, так і конкретних співробітників, з додатковими властивостями та методами для обробки їхніх даних (рис. 2.14).

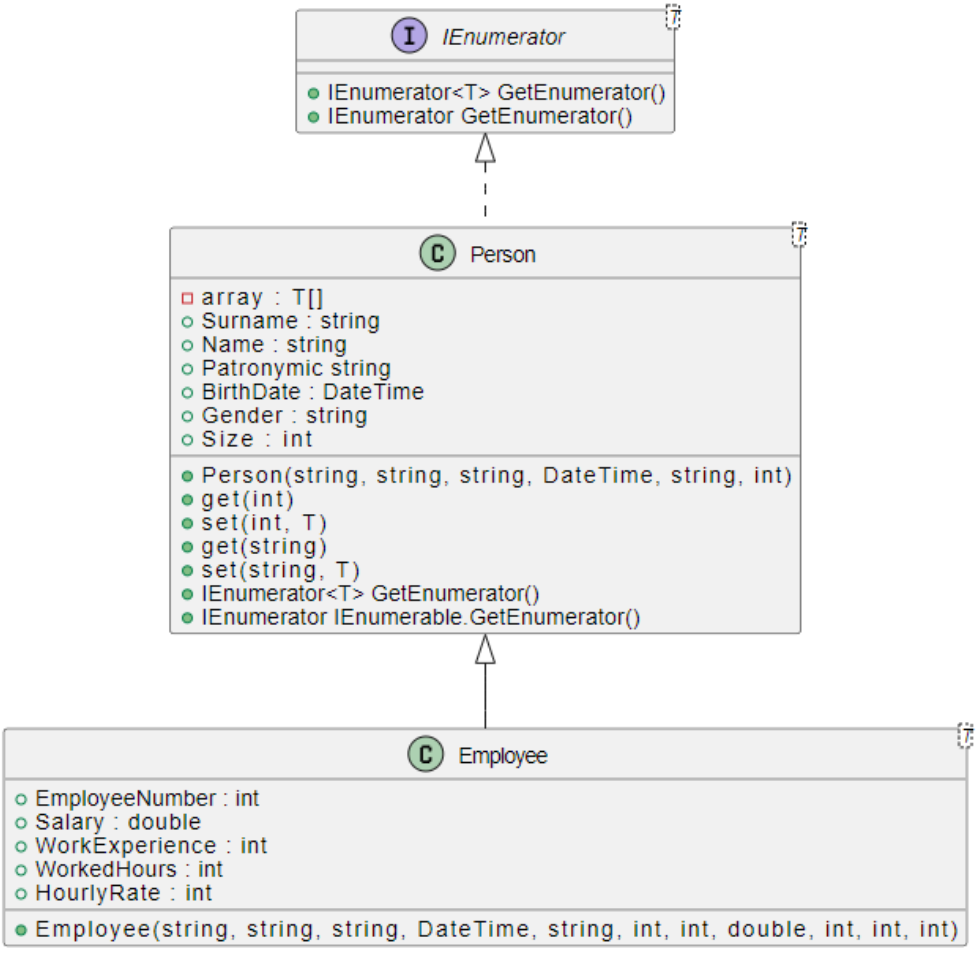

Рисунок 2.14 – UML-діаграма ієрархії класів

Програмний код на мові C# для виконання завдань лабораторної роботи:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Text;
using static System.Runtime.InteropServices.JavaScript.JSType;

public class Person<T> : IEnumerable<T>
{
    // Створюємо автоматичні властивості для класу Person
    public string Surname { get; set; }
    public string Name { get; set; }
    public string Patronymic { get; set; }
    public DateTime BirthDate { get; set; }
    public string Gender { get; set; }
    // Оголошуємо масив типу T
    private T[] array;
    // Розмір масиву повинен зберігатися в окремій властивості
    public int Size { get; }

    // Створюємо конструктор ініціалізації класу Person
    public Person(string surname, string name, string patronymic,
        DateTime birthDate, string gender, int size)
    {
        // Присвоюємо приватним полям їх значення, які передаються як параметри
        Surname = surname;
        Name = name;
        Patronymic = patronymic;
        BirthDate = birthDate;
        Gender = gender;
        // Ініціалізація масиву
        Size = size;
        array = new T[size];
    }

    // Індексатор для масиву типу int
    public T this[int index]
    {
        get { return array[index]; }
        set { array[index] = value; }
    }

    // Індексатор для масиву типу string
    public T this[string index]
    {
        get { return array[int.Parse(index)]; }
        set { array[int.Parse(index)] = value; }
    }

    // Типизований ітератор для перегляду елементів поля array
    public IEnumerator<T> GetEnumerator()
    {
        foreach (var item in array)
        {
            yield return item;
        }
    }

    // Нетипизований ітератор для перегляду елементів поля array
    IEnumerator IEnumerable.GetEnumerator()
    {
        return GetEnumerator();
    }
}
```

```
/*Похідний клас Employee від базового класу Person*/
public class Employee<T> : Person<T>
{
    // Створюємо автоматичні властивості для класу Employee
    public int EmployeeNumber { get; set; }
    public double Salary { get; set; }
    public int WorkExperience { get; set; }
    public int WorkedHours { get; set; }
    public int HourlyRate { get; set; }

    /*Створюємо конструктор ініціалізації класу Employee.
     * Ключове слово base передає параметри у базовий клас Person*/
    public Employee(string surname, string name, string patronymic,
        DateTime birthDate, string gender, int size, int employeeNumber,
        double salary, int workExperience, int workedHours, int hourlyRate) :
        base(surname, name, patronymic, birthDate, gender, size)
    {
        // Присвоюємо приватним полям їх значення, які передаються як параметри
        EmployeeNumber = employeeNumber;
        Salary = salary;
        WorkExperience = workExperience;
        WorkedHours = workedHours;
        HourlyRate = hourlyRate;
    }
}
class Program
{
    static void Main(string[] args) // Вхід в головну функцію програми
    {
        /* Створення екземпляру колекції, додавання декількох об’єктів
         * та виведення їхніх даних у консоль */
        Console.OutputEncoding=UTF8Encoding.UTF8;
        Employee<int> empl1 = new Employee<int>("Петренко", "Петр", "Петрович",
            DateTime.Now, "чоловік", 3, 20581, 2300, 3, 560, 23);
        Employee<string> empl2 = new Employee<string>("Петренко", "Олена",
           "Петрівна", DateTime.Now, "жінка", 4, 18456, 2600, 4, 420, 20);
        empl1[0] = 7;
        empl1[1] = 9;
        empl1[2] = 10;
        int counter = 1;

        foreach (var item in empl1)
        {
            Console.WriteLine($"Рейтинг {empl1.Name} {empl1.Surname} " +
                $"за {counter} роки(ів) {item}");
            counter++;
        }
        /*Console.WriteLine('\n');*/
        empl2[0] = "трейні";
        empl2[1] = "джуніор";
        empl2[2] = "мідл";
        empl2[3] = "сеньор";
        counter = 1;

        foreach (var item in empl2)
        {
            Console.WriteLine($"Стаж роботи {empl2.Name} {empl2.Surname} на посаді"+
                $"{item} складає {counter} роки(ів)");
            counter++;
        }
    }
}
```

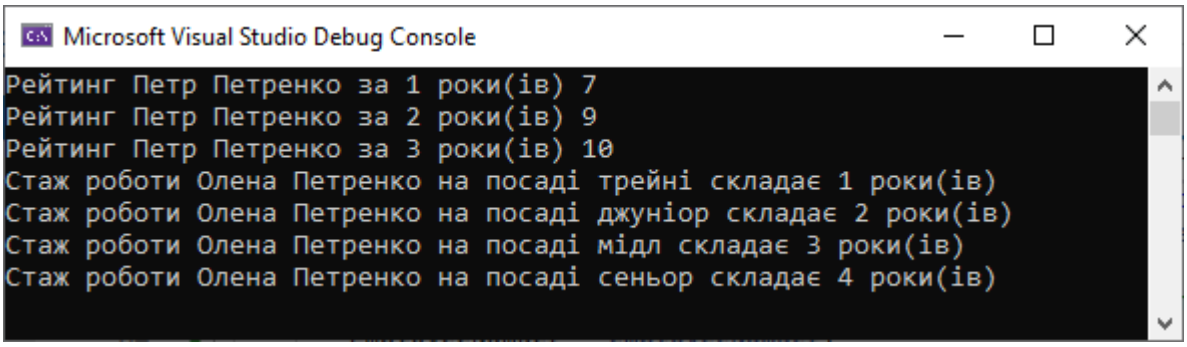

Рисунок 2.15 – Приклад виконання програми

Висновки до розділу

У розділі виконано аналіз характеристик об'єктів галузі проєктування об’єктно-орієнтованої ієрархії класів прикладних програмних систем з використанням інтерфейсів, параметризованих класів та узагальнених колекцій. Встановлено, що ці концепції є основними будівельними блоками для створення складних та ефективних додатків у C#. Вони дозволяють писати код, який легше підтримувати, розширювати та масштабувати.

Виконана постановка нових лабораторних робіт:

1. Інтерфейси. Наслідування інтерфейсів.

2. Узагальнені колекції та ітератори.

3. Параметризовані класи та індексатори.

Розроблено завдання лабораторних робіт, приклади виконання робіт та оформлення звітів, зокрема опис предметної області, опис класів та їх атрибутів, UML-діаграма ієрархії класів, 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.

РОЗДІЛ 3 ВИМОГИ ДО КАДРОВОГО ЗАБЕЗПЕЧЕННЯ ОБ'ЄКТУ ГАЛУЗІ

Для підготовки фахівців цифрових технологій, які викладатимуть освітній модуль «Робота з інтерфейсами, колекціями та індексаторами на мові C#» у закладах вищої освіти, важливо мати комплексний підхід.

Для аналізу цього процесу необхідно у першу чергу оцінити актуальність підготовки фахівців цифрових технологій. Сучасний ринок праці вимагає від спеціалістів не лише глибоких технічних знань, але й навичок, необхідних для передачі знань студентам. Особливо це стосується мов програмування, таких як C#, які є широко розповсюдженими у сфері розробки програмного забезпечення. Підготовка викладачів з метою викладання освітніх модулів сприяє підвищенню якості освіти та розвитку інноваційних методик навчання.

Основними концепціями освітнього модуля є інтерфейси, колекції та індексатори. У мові програмування C# інтерфейси використовуються для визначення контрактів для класів. Вони дозволяють розробникам створювати гнучкі і масштабовані програми, спрощуючи підтримку і оновлення коду. Підготовка викладачів включає розгляд таких понять, як створення та використання інтерфейсів, реалізація множинних інтерфейсів, а також використання стандартних інтерфейсів.

Освітній модуль охоплює вивчення колекцій в C#, таких як списки, словники, множини та черги. Вони використовуються для організації та управління даними в програмі. Фахівці мають навчитися вибирати відповідну колекцію залежно від завдань, оптимізувати її використання та розуміти концепції асимптотичної складності алгоритмів.

Майбутним фахівцям треба розумітися на індексаторах у C#, тобто механізму, який дозволяє об'єктам класів працювати як масиви. Індексатори забезпечують зручний синтаксис для доступу до елементів класу та використовуються для створення більш інтуїтивного і читабельного коду.

Підготовка фахівців включає не лише теоретичні знання, але й практичні навички. Викладачі повинні бути здатні створювати навчальні плани, розробляти інтерактивні вправи, використовувати сучасні педагогічні підходи, такі як проєктне навчання, фліп-класи та колаборативне навчання. Це дозволить студентам глибше засвоїти матеріал і ефективніше застосовувати знання на практиці.

Метою підготовки є навчання викладачів таким чином, щоб вони могли допомогти студентам створювати реальні проєкти на основі вивчених концепцій. Це можуть бути програми для управління даними, інтерфейси користувача або інші додатки, що базуються на використанні колекцій та індексаторів у C#. Практичний аспект забезпечує розуміння того, як використовувати концепції в реальних задачах, які можуть виникати у професійному середовищі.

Впровадження сучасних технологій у викладання має велике значення. Використання інтерактивних інструментів, віртуальних лабораторій та платформ для онлайн-навчання дозволяє забезпечити гнучкість процесу навчання. Викладачі, які володіють цими технологіями, можуть ефективно адаптувати методики навчання до потреб студентів, надаючи їм доступ до матеріалів у зручному форматі.

Одним із головних викликів у підготовці фахівців з цифрових технологій є швидкий розвиток технологій, що вимагає постійного оновлення знань викладачів. Також важливим є вміння адаптувати складні технічні концепції до рівня розуміння студентів, які можуть мати різний рівень підготовки. Крім того, існує необхідність у розробці нових методичних матеріалів та посібників, які будуть актуальними для навчального процесу.

Аналіз останніх досліджень [23-26] дозволив скласти загальні рекомендації щодо вдосконалення процесу підготовки:

1. Постійне навчання та розвиток. Запровадження систематичних тренінгів та курсів для викладачів.

2. Інтеграція з професійним середовищем. Викладачі мають бути

залучені до реальних проєктів у сфері ІТ, що сприятиме їх професійному росту та кращому розумінню сучасних вимог ринку.

3. Підтримка міждисциплінарного підходу. Залучення знань з різних галузей (наприклад, управління проєктами, бізнес-аналізу) для створення більш комплексного підходу до навчання.

У Європі існує багато вищих навчальних закладів, які спеціалізуються на підготовці фахівців з цифрових технологій. До найбільш відомих можна віднести такі.

1. Швейцарська вища технічна школа Цюриха є одним з найкращих технічних університетів у світі, з особливо сильними програмами у галузі інформаційних та комунікаційних технологій, зокрема комп'ютерні науки, інженерія програмного забезпечення, робототехніка та цифрові технології.

2. Кембриджський університет має потужну дослідницьку базу в галузі штучного інтелекту, машинного навчання та цифрових інновацій, зокрема він пропонує підготовку бакалаврів на програмах «Інформатика та комп'ютерні науки», «Математика для цифрових технологій».

3. Імперський лондонський коледж має одну з найкращих програм з інформатики та технологій, що поєднує фундаментальні дослідження з практичними застосуваннями у сфері ІТ. До основних програм підготовки можна віднести цифрові технології, наука про дані, кібербезпека.

4. Технічний університет Мюнхена відомий своїми програмами, спрямованими на розвиток цифрових технологій та інновацій у сфері ІТ, зокрема інженерія програмного забезпечення, штучний інтелект, інформатика.

5. Політехнічна школа є одним з найкращих технічних інститутів у Франції, який пропонує комплексну підготовку фахівців з цифрових технологій за програмами комп'ютерні науки, цифрові технології та системи.

Ці навчальні заклади пропонують програми, які включають не тільки теоретичну базу, але й практичні курси та проєкти, спрямовані на розвиток навичок у сфері цифрових технологій.

Підготовка викладачів з цифрових технологій у Європі має кілька

ключових особливостей, які відображають загальноєвропейські тенденції у сфері освіти та розвитку цифрових компетенцій. Далі наведено основні аспекти цієї підготовки.

Європейські країни активно впроваджують цифрові технології в навчальні програми для підготовки викладачів. Це включає використання онлайн-ресурсів, цифрових інструментів для викладання, платформ дистанційного навчання та мобільних додатків.

Європейська рамка цифрових компетенцій для освітян визначає необхідні цифрові компетенції для викладачів, розподілені за шістьма рівнями компетентності. Вона допомагає освітнім установам та урядам розробляти навчальні програми та оцінювати рівень володіння цифровими технологіями серед викладачів. У багатьох європейських країнах існують національні програми підвищення кваліфікації викладачів, які фокусуються на розвитку цифрових навичок. Це може бути як формальне навчання (курси, тренінги), так і неформальні можливості (вебінари, онлайн-курси).

Використання цифрових технологій в освіті передбачає не тільки технічне володіння, але й розуміння того, як ці інструменти можна застосовувати для поліпшення навчального процесу. Тому викладачів навчають методикам інтерактивного навчання, гейміфікації, використанню мультимедійних матеріалів та адаптивних освітніх технологій.

Європейський Союз активно фінансує різноманітні програми та ініціативи, спрямовані на підвищення рівня цифрових навичок серед викладачів. Це включає такі програми, як Erasmus+ [27], Horizon Europe [28], а також інші можливості для розвитку та обміну досвідом між країнами.

Підготовка викладачів у Європі також зосереджена на тому, щоб зробити цифрові технології доступними для всіх студентів, незалежно від їхніх можливостей або особливих потреб. Це передбачає використання інклюзивних інструментів та платформ, а також адаптацію навчальних матеріалів для студентів з обмеженими можливостями.

У країнах Європи існують стандартизовані процеси оцінки цифрових

компетенцій викладачів, що дозволяє оцінювати рівень їхніх навичок і надавати сертифікацію. Це сприяє створенню єдиних стандартів у цифровій освіті та підвищує якість викладання.

Ці особливості формують сучасну систему підготовки викладачів у Європі, орієнтовану на забезпечення високого рівня цифрових компетенцій і готовність до використання новітніх технологій у навчальному процесі.

Висновки до розділу

Підготовка фахівців цифрових технологій для викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#» є важливим етапом у забезпеченні високоякісної освіти у закладах вищої освіти. Такий підхід дозволяє не лише підвищити рівень знань студентів, але й забезпечити їм конкурентоспроможні навички для подальшого професійного розвитку.

Виконано огляд ключових аспектів кадрового забезпечення фахівців для викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#», зокрема встановлено наступне:

1. сучасний ринок праці потребує від викладачів не тільки технічних знань у сфері програмування, таких як C#, але й здатності ефективно передавати ці знання студентам;

2. сучасна система підготовки викладачів у Європі орієнтована на забезпечення високого рівня цифрових компетенцій і готовність до використання новітніх технологій у навчальному процесі;

3. використання сучасних педагогічних підходів і цифрових технологій у навчанні підвищує якість освіти та сприяє розвитку інноваційних методик навчання;

4. постійний розвиток технологій вимагає від викладачів регулярного оновлення своїх знань і вмінь, щоб залишатися конкурентоспроможними та відповідати вимогам сучасної освіти.

РОЗДІЛ 4 МЕТОДИКА ПРОФЕСІЙНОЇ ПІДГОТОВКИ ФАХІВЦІВ З ЦИФРОВИХ ТЕХНОЛОГІЙ ДЛЯ ВИКЛАДАННЯ ОСВІТНЬОГО МОДУЛЯ «РОБОТА З ІНТЕРФЕЙСАМИ, КОЛЕКЦІЯМИ ТА ІНДЕКСАТОРАМИ НА МОВІ C#» У ЗАКЛАДАХ ВИЩОЇ ОСВІТИ. ДИДАКТИЧНИЙ ПРОЕКТ КОНСУЛЬТАТИВНОГО ЗАНЯТТЯ З ТЕМИ «ІНТЕРФЕЙСИ. ВИКОРИСТАННЯ ІНТЕРФЕЙСУ ICOMPARABLE. ІНТЕРФЕЙС ЯК ПОСИЛАЛЬНИЙ ТИП ДАНИХ. СПАДКУВАННЯ ІНТЕРФЕЙСІВ» ДИСЦИПЛІНИ «ПРОГРАМНА ІНЖЕНЕРІЯ» ДЛЯ ЗДОБУВАЧІВ ВИЩОЇ ОСВІТИ СПЕЦІАЛЬНОСТІ 015 ПРОФЕСІЙНА ОСВІТА (ЦИФРОВІ ТЕХНОЛОГІЇ)

4.1 Вихідні дані

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: «Програмна інженерія»;

тема: «Інтерфейси. Використання інтерфейсу IComparable. Інтерфейс як посилальний тип даних. Спадкування інтерфейсів».

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 – виконання курсової роботи та 186 годин підготовки до аудиторних занять, контрольних заходів тощо) 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 «Програмна інженерія» 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 «Програмна інженерія».

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 [45].

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 «Інтерфейси. Використання інтерфейсу IComparable. Інтерфейс як посилальний тип даних. Спадкування інтерфейсів» такі, як Лістинг, IInfo, GetDescription, ItemMark, PrintUpperDescr, вміти називати набір інтерфейсів, які містяться в бібліотеці C# які можна використовувати у своїх програмах.	Знати визначення понять «інтерфейс» «стандартна бібліотека мови c#», c# колекція та «середовище .net framework», знання сутності поняття «лістинг».	Правильно названі з переліку основні поняття теми «Інтерфейси. Використання інтерфейсу IComparable. Інтерфейс як посилальний тип даних. Спадкування інтерфейсів» як Лістинг, IInfo, GetDescription, ItemMark, PrintUpperDescr, вміти називати набір інтерфейсів, які містяться в бібліотеці C# які можна використовувати у своїх програмах.
2	Уміти характеризувати основні елементи, які включає стандартна бібліотека C#, формувати описи об'єктів які ускладнюють і заплутують структуру програми, класифікувати товари, уміти складати	Виконання дій першого рівня: правильно названі з переліку основні поняття теми «Інтерфейси. Використання інтерфейсу IComparable. Інтерфейс як посилальний тип даних. Спадкування інтерфейсів» як Лістинг, IInfo, GetDescription,	Правильно охарактеризований логічний та зрозумілий підхід використання інтерфейсу, застосування інтерфейсу в зовнішній та внутрішній цикл, правильне оголошення інтерфейсів.

Продовження табл. 4.1

1	2	3	4
	загальний список товарів.	ItemMark, PrintUpperDescr, вміти називати набір інтерфейсів, які містяться в бібліотеці C# які можна використовувати у своїх програмах.	
3	Уміти аналізувати нескінченний цикл та робити висновки про доцільність використання інтерфейсу як посильний тип даних.	Виконання дій першого і другого рівнів.	Правильно проаналізований нескінченний цикл та зроблені висновки, щодо доцільність використання інтерфейсу як посильний тип даних.
4	Проектувати, розробляти та аналізувати алгоритми розв'язання обчислювальних та логічних задач, оцінювати ефективність та складність алгоритмів на основі застосування формальних моделей алгоритмів та функцій.	Виконання дій першого, другого і третього рівнів.	Правильно розроблена програма з використанням інтерфейсу.

Таким чином, були розроблені цілі консультативного заняття з теми «Інтерфейси. Використання інтерфейсу IComparable. Інтерфейс як посилальний тип даних. Спадкування інтерфейсів» дисципліни «Програмна інженерія» 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 «Програмна інженерія».

Нижче представлено перелік основної та допоміжної літератури, а також інформаційні ресурси для вивчення дисципліни та підготовки до консультативного заняття.

Рекомендована література:

1. 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 IT-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: ННППІ 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 «Програмна інженерія».

4.4 Визначення найбільш складних для розуміння та засвоєння питань

Визначимо найбільш складні для розуміння та засвоєння здобувачами вищої освіти питання теми «Інтерфейси. Використання інтерфейсу IComparable. Інтерфейс як посилальний тип даних. Спадкування інтерфейсів» дисципліни «Програмна інженерія» та сформулюємо можливі відповіді на них (табл. 4.2) [44].

Таблиця 4.2

Обрання питань для консультування та формулювання відповідей на можливі питання

Теми (або тема) дисципліни	Зміст програми за кожною темою	Найбільш складні питання за темами (темою)	Відповіді на питання
1	2	3	4
«Інтерфейси. Використання інтерфейсу IComparable. Інтерфейс як посилальний тип даних. Спадкування інтерфейсів».	1. Інтерфейси. 2. Використання інтерфейсу IComparable. 3. Інтерфейс як посилальний тип даних. 4. Спадкування інтерфейсів. 5. Колекції. 6. Інтерфейси узагальнених колекцій. 7. Класи узагальнених колекцій.	1.Охарактеризуйте класи у лістингу?	1. У лістингу оголошено два класи. Клас MobilePhone описує товари виду "мобільний телефон", а клас FoodProcessor – товари виду "кухонний комбайн". Кожен клас має метод, який повертає опис об'єкта, що формується на основі його полів. При цьому для опису екземплярів різних класів використовуються різні поля різних типів.

Продовження табл. 4.2

1	2	3	4
		2. Які оголошення інтерфейсу може містити модифікатори доступу?	2. Оголошення інтерфейсу може містити модифікатори доступу public, protected, internal чи private. Проте елементи інтерфейсу неявно є тільки публічними і не можуть містити модифікаторів доступу (включаючи public). Клас, який впроваджує інтерфейс, обов'язково повинен реалізувати всі елементи інтерфейсу. При цьому код для реалізації елементів інтерфейсу може як реалізуватися самим класом, так і спадкуватися від базових класів. Хоча оголошення інтерфейсу не містить реалізації його методів, документація для інтерфейсу повинна містити опис того, що вони виконують.
		3. Що таке інтерфейс?	3. *Інтерфейс* – це посилальний тип, який задає множину функціональних елементів, але не реалізовує їх. Інтерфейс можуть реалізовувати (впроваджувати) класи та структури. Семантично інтерфейси схожі на абстрактні класи, але жоден метод інтерфейсу не може мати реалізації. Інтерфейс показує, що клас повинен робити, але не визначає, як саме. При впровадженні інтерфейсу у клас мають бути реалізовані всі методи інтерфейсу.

Отже, на даному етапі визначили найбільш складні для розуміння та засвоєння питання з теми «Інтерфейси. Використання інтерфейсу IComparable. Інтерфейс як посилальний тип даних. Спадкування інтерфейсів» дисципліни «Програмна інженерія» для здобувачів вищої освіти спеціальності 015 Професійна освіта (Цифрові технології).

4.5 Вибір дидактичних методів активізації навчальної діяльності студентів на консультації

Виберемо методи активізації навчальної діяльності здобувачів вищої освіти на консультації (табл. 4.3) [46].

Таблиця 4.3

Методи активізації навчальної діяльності студентів 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 «Інтерфейси. Використання інтерфейсу IComparable. Інтерфейс як посилальний тип даних. Спадкування інтерфейсів», та використання плакатів «Колекції. Інтерфейси узагальнених колекцій. Класи узагальнених колекцій».
Мотиваційні методи	Для реалізації мотивації використаємо: тип: внутрішня мотивація; вид: вступна мотивація; метод: мотивуючий вступ; прийом: віднесення до особистості. Повідомлення важливості вивчення даної теми: «Вивчення сьогоднішньої теми «Інтерфейси. Використання інтерфейсу IComparable. Інтерфейс як посилальний тип даних. Спадкування інтерфейсів» для вас, як майбутніх програмістів, є достатньо актуальним. Відповідно до вашої майбутньої професійної діяльності знання цього навчального матеріалу знадобляться вам для програмування на підприємстві, а саме це дозволить правильно використовувати певні набори операторів і призначення керуючих конструкцій розгалуження. Все це є одним з основних завдань вашої майбутньої діяльності. Від того, наскільки успішно ви справитеся з даним завданням, вже працюючи на підприємстві, залежить його благополуччя, а вам дозволить рости у якості високоякісного програміста в області програмування та стверджуватимуть вас як висококваліфікованого фахівця».
Проблемні методи	Використання проблемного питання. Проблемне питання: «Напишіть програму використання класів, які описують різні товари»
Комунікативні методи	Імітація ситуацій з реального життя. Усі наші знання необхідні для подальшої вашої професійної діяльності, тому що використання інтерфейсу дає можливість об'єднати деяку послідовність окремих операцій. Отже, наприклад: Використання класів, які описують різні товари

Продовження табл. 4.3

1	2

	class MobilePhone // Мобільний телефон
	
	{
	public string Manufacturer; // Виробник
	
	public string Model; // Модель
	
	public double ScreenSize; // Розмір екрану
	
	public double MemorySize; // Обсяг пам'яті
	
	public string PhoneDescription() // Опис
	{
	return String.Format("{0} {1} {2}\" {3}Гб", Manufacturer, Model,
	ScreenSize, MemorySize);
	}
	class FoodProcessor // Кухонний комбайн
	{
	public string Manufacturer; // Виробник
	public string ModelInfo; // Модель
	public double Power; // Потужність
	public string FoodProcessorDescription() // Опис
	{
	return String.Format("{0} {1} {2}Вт", Manufacturer, ModelInfo, Power);
	}
	static void PrintUpperDescr(MobilePhone m)
	{
	Console.WriteLine(m.PhoneDescription().ToUpper());
	}
	static void Main(string[] args)
	{
	MobilePhone mp = new MobilePhone()
	{
	Manufacturer = "PhoneMan",
	Model = "KL-289i+",
	MemorySize = 2,
	ScreenSize = 7
	};
	FoodProcessor fp = new FoodProcessor()
	{
	Manufacturer = "KitchenTec",
	ModelInfo = "Astra FV-202",
	Power = 12
	};
	Console.WriteLine(mp.PhoneDescription());
	Console.WriteLine(fp.FoodProcessorDescription());
	PrintUpperDescr(mp);
	}

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 [44].

Таблиця 4.4

Варіанти організації консультативного заняття

№ варіанта	Етапи організації заняття	Характеристика варіанта
1	2	3
1	- вступне слово лектора; - відповіді на питання здобувачів 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 освіти.
3	- видача завдань на самостійне вивчення матеріалу теми; - підготовка питань лектору; - відповіді і їх обговорення.	В цьому випадку консультування грає функцію додаткового інформування зі складних питань і пояснення незрозумілого навчального матеріалу.
4	- повідомлення теми; - консультування декількома фахівцями в певній області науки і техніка з актуальних питань науки і нової техніки.	Цей варіант 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) [46].

Таблиця 4.5

Сценарій консультативного заняття

Етапи проведення консультативного заняття	Дії викладача	Дії здобувачів освіти
1	2	3
Організа-ційний момент	Вітання, фіксація відсутніх, перевірка зовнішньої обстановки в аудиторії	Вітання викладача. Підтвердження присутності у момент переклички, настрій на здійснення навчальної діяльності
Повідомлення теми і мети уроку	Повідомлення теми заняття «Вивчення сьогоднішньої теми «Інтерфейси. Використання інтерфейсу IComparable. Інтерфейс як посилальний тип даних. Спадкування інтерфейсів», формулювання його цілей: сформувати уміння давати визначення поняттю «Інтерфейс», правильно використовувати інтерфейс як посилальний тип даних.	Фіксація теми, сприйняття цілей, представлення результатів засвоєння матеріалу теми даного заняття
Мотивація мети	Повідомлення важливості вивчення даної теми: «Вивчення сьогоднішньої теми «Інтерфейси. Використання інтерфейсу IComparable. Інтерфейс як посилальний тип даних. Спадкування інтерфейсів» для вас, як майбутніх програмістів, є достатньо актуальним. Відповідно до вашої майбутньої професійної діяльності знання цього навчального матеріалу знадобляться вам для програмування на підприємстві, а саме це дозволить правильно використовувати певні набори операторів і призначення керуючих конструкцій розгалуження. Все це є одним з основних завдань вашої	Сприйняття важливості і актуальності вивчення теми, прояв інтересу до неї.

Продовження табл. 4.5

1	2	3
	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».	
Актуалізація знань	Викладач проводить фронтальне усне опитування з метою перевірки базових знань: 1. Назвіть два класи у лістингу? 2. Чим відрізняються класи від інтерфейсу?	Здобувачі освіти беруть участь у опитуванні та відповідають на поставлені питання. Передбачувані відповіді: 1. У лістингу оголошено два класи. Клас MobilePhone описує товари виду "мобільний телефон", а клас FoodProcessor – товари виду "кухонний комбайн". Кожен клас має метод, який повертає опис об'єкта, що формується на основі його полів. При цьому для опису екземплярів різних класів використовуються різні поля різних типів. 2. На відміну від класів, які можуть успадкувати тільки один базовий клас, інтерфейси можуть спадкувати довільну кількість базових інтерфейсів. У свою чергу, кожен базовий інтерфейс може спадкувати інші інтерфейси.
Формування ООД	Викладач проводить консультацію згідно плану, за допомогою методу - пояснення: 1. Інтерфейси. 2. Використання інтерфейсу IComparable. 3. Інтерфейс як посилальний тип даних. 4. Спадкування інтерфейсів. 5. Колекції. 6. Інтерфейси узагальнених колекцій. 7. Класи узагальнених колекцій.	Слухають пояснення, конспектують.

Продовження табл. 4.5

1	2	3
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. Колекції. 2. Інтерфейси узагальнених колекцій. 3. Класи узагальнених колекцій.	Здобувачі освіти запитують: 1. Поясніть, будь-ласка, головні преваги колекцій? 2.Які недоліки та переваги інтерфейсу?
Підведення підсумків	Викладач підводить підсумки проведення консультації: «Сьогодні ми розглянули незрозумілі вам питання теми для самостійного вивчення. Зараз перевіримо як ви засвоїли	Здобувачі освіти слухають, відповідають: «Стандартна бібліотека шаблонів (STL) - підмножина стандартної бібліотеки C++ і містить контейнери, алгоритми, ітератори, об'єкти-функції і т. д.». Здобувачі освіти прощаються.

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

На заключному етапі представлено контурний конспект з теми «Інтерфейси. Використання інтерфейсу IComparable. Інтерфейс як посилальний тип даних. Спадкування інтерфейсів» дисципліни «Програмна інженерія» 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 «Інтерфейси. Використання інтерфейсу IComparable. Інтерфейс як посилальний тип даних. Спадкування інтерфейсів» дисципліни «Програмна інженерія» для здобувачів вищої освіти спеціальності 015 Професійна освіта. (Цифрові технології) представлено у Додатку.

Висновки до розділу

У розділі розроблено дидактичний проект консультативного заняття з теми «Інтерфейси. Використання інтерфейсу IComparable. Інтерфейс як посилальний тип даних. Спадкування інтерфейсів» дисципліни «Програмна інженерія» 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 «Програмна інженерія». 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 «Робота з інтерфейсами, колекціями та індексаторами на мові 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 «Робота з інтерфейсами, колекціями та індексаторами на мові C#» у закладах вищої освіти є актуальною у професійній педагогіці.

Проаналізовано ступінь актуальності проблеми професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Робота з інтерфейсами, колекціями та індексаторами на мові C#».

Виконано аналіз характеристик об'єктів галузі проєктування об'єктно-орієнтованої ієрархії класів прикладних програмних систем з використанням інтерфейсів, параметризованих класів та узагальнених колекцій.

Виконана постановка нових лабораторних робіт:

1. Інтерфейси. Наслідування інтерфейсів.
2. Узагальнені колекції та ітератори.
3. Параметризовані класи та індексатори.

Розроблено завдання лабораторних робіт, приклади виконання робіт та оформлення звітів, зокрема опис предметної області, опис класів та їх атрибутів, UML-діаграма ієрархії класів, 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.

Виконано огляд ключових аспектів кадрового забезпечення фахівців для

викладання освітнього модуля «Робота з інтерфейсами, колекціями та індексаторами на мові C#», зокрема встановлено, що сучасна система підготовки викладачів у Європі орієнтована на забезпечення високого рівня цифрових компетенцій і готовність до використання новітніх технологій у навчальному процесі.

Розроблено дидактичний проект консультативного заняття з теми «Інтерфейси. Використання інтерфейсу IComparable. Інтерфейс як посилальний тип даних. Спадкування інтерфейсів» дисципліни «Програмна інженерія» 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 «Програмна інженерія». Подано список використаних джерел та відповідні посилання.

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

За основними результатами дослідження виконана публікація тези доповіді на конференції:

– Никитських О.С. Лабораторний практикум «Робота з інтерфейсами, колекціями та індексаторами» // Матеріали VIII міжн. наук.-практ. конф. здобувачів вищої освіти та молодих учених «Студенти та молодь – для майбутнього країни» (м. Харків, 14-15 листопада 2024 р.) [упоряд. Г. Г. Михальченко]. Харків, 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. Поліщук А.М., Ніколюк П.К. Технологія програмування та основні етапи її розвитку. Комп’ютерні технології обробки даних, 2022, 89-91.

16. 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.

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

18. 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).

19. 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.

20. 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.

21. 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.

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

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

24. Collins A. M. Rethinking education in the age of technology. In: Intelligent Tutoring Systems: 9th International Conference, ITS 2008, Montreal, Canada, June 23-27, 2008 Proceedings 9. Springer Berlin Heidelberg, 2008. p. 1-2.

25. Elayyan S. The future of education according to the fourth industrial revolution. Journal of Educational Technology and Online Learning, 2021, 4.1: 23-30.

26. Digital Competence Framework for Educators (DigCompEdu). URL: https://joint-research-centre.ec.europa.eu/digcompedu_en

27. Еразмус+ - Програма Європейського Союзу у сфері освіти, професійної підготовки, молоді та спорту. URL: https://erasmusplus.org.ua/

28. Horizon Europe. URL: https://research-and-innovation.ec.europa.eu/funding/funding-opportunities/funding-programmes-and-open-calls/horizon-europe_en

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

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

31. 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.

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

33. 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.).

34. 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.).

35. 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

36. 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

37. 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

38. 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.

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

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

41. 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.

42. 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

43. 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.

44. 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 : [б. в.], 2022. – 92 с.

45. 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.

46. 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.

47. 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.

48. 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.

49. 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: [б. в.], 2017. - 100 с.

50. 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: [б. в.], 2017. - 101 с.

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

52. 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

Інтерфейси. Використання інтерфейсу IComparable. Інтерфейс як посилальний тип даних. Спадкування інтерфейсів.

Інтерфейс – це посилальний тип, який задає множину функціональних елементів, але не реалізовує їх. Інтерфейс можуть реалізовувати (впроваджувати) класи та структури. Семантично інтерфейси схожі на абстрактні класи, але жоден метод інтерфейсу не може мати реалізації. Інтерфейс показує, що клас повинен робити, але не визначає, як саме. При впровадженні інтерфейсу у клас мають бути реалізовані всі методи інтерфейсу, кожен клас реалізувати їх по-іншому.

Розглянемо приклад, який висвітлить недоліки, що усуваються при використанні інтерфейсу. Припустимо, що ми маємо достатньо великий набір товарів різних видів. Кожен вид має притаманні йому характеристики. Так, товар виду "мобільний телефон" має одні характеристики, а товар виду "кухонний комбайн" – інші. На основі існуючого набору товарів потрібно скласти їх загальний список. При цьому кожен елемент списку міститиме коротку характеристику товару і формуватиметься на основі характеристик свого виду.

У лістингу 6.1 оголошено два класи. Клас MobilePhone описує товари виду "мобільний телефон", а клас FoodProcessor – товари виду "кухонний комбайн". Кожен клас має метод, який повертає опис об'єкта, що формується на основі його полів. При цьому для опису екземплярів різних класів використовуються різні поля різних типів.

Зауважимо, що для розглянутого підходу формування описів об'єктів характерні наступні риси, які ускладнюють і заплутують структуру програми:

1. Методи, які повертають інформацію про об'єкт, належать різним класам і можуть мати різну назву. Щоб вивести опис, потрібно пам'ятати назви прийнятних для даного випадку методів (клас може містити багато методів).

2. Потрібний для опису об'єкта метод у класі взагалі може бути

відсутнім.

3. Екземпляри різних класів неможливо передати у інший метод для обробки через один параметр, оскільки вони мають різні типи. Так, у метод PrintUpperDescr можна передати тільки об'єкт класу MobilePhone.

Лістинг 6.1. Використання класів, які описують різні товари

```
class MobilePhone          // Мобільний телефон
{
   public string Manufacturer;   // Виробник
   public string Model;                  // Модель
   public double ScreenSize;     // Розмір екрану
   public double MemorySize;           // Обсяг пам'яті
   public string PhoneDescription() // Опис
   {
      return String.Format("{0} {1} {2}\" {3}Гб", Manufacturer, Model,
                  ScreenSize, MemorySize);
   }
}
class FoodProcessor        // Кухонний комбайн
{
   public string Manufacturer;   // Виробник
   public string ModelInfo;       // Модель
   public double Power;                                 // Потужність
   public string FoodProcessorDescription()          // Опис
   {
      return String.Format("{0} {1} {2}Вт", Manufacturer, ModelInfo,
Power);
   }
}
static void PrintUpperDescr(MobilePhone m)
{
```

```
    Console.WriteLine(m.PhoneDescription().ToUpper());
}
static void Main(string[] args)
{
    MobilePhone mp = new MobilePhone()
    {
        Manufacturer = "PhoneMan",
        Model = "KL-289i+",
        MemorySize = 2,
        ScreenSize = 7
    };
    FoodProcessor fp = new FoodProcessor()
    {
        Manufacturer = "KitchenTec",
        ModelInfo = "Astra FV-202",
        Power = 12
    };
    Console.WriteLine(mp.PhoneDescription());
    Console.WriteLine(fp.FoodProcessorDescription());
    PrintUpperDescr(mp);
}
```

Використання інтерфейсу дозволяє реалізувати логічніший та зрозуміліший підхід. Для цього слід оголосити інтерфейс, який матиме метод, призначений для формування опису об'єкта. Після цього кожен з класів, що представляє об'єкти різних видів, повинен реалізувати цей інтерфейс. Застосування інтерфейсу дозволить уникнути трьох приведених недоліків.

Загальний синтаксис оголошення інтерфейсу такий:

```
interface IInterfaceName
{
    // Заголовки функціональних елементів інтерфейсу
```

}

Назви інтерфейсам **прийнято** починати з великої літери "I". Елементами інтерфейсу можуть бути методи, властивості, індексатори та події. Інтерфейс не може містити елементів даних та статичних елементів. В інтерфейсі розміщують виключно **оголошення** функціональних елементів, а їх **тіла** повинні бути реалізовані у класах, які спадкують інтерфейс. За змістом інтерфейс нагадує абстрактний клас, але не використовує ключового слова abstract.

Оголошення інтерфейсу може містити модифікатори доступу public, protected, internal чи private. Проте елементи інтерфейсу неявно є тільки публічними і не можуть містити модифікаторів доступу (включаючи public). Клас, який впроваджує інтерфейс, обов'язково повинен реалізувати **всі** елементи інтерфейсу. При цьому код для реалізації елементів інтерфейсу може як реалізуватися самим класом, так і спадкуватися від базових класів.

Хоча оголошення інтерфейсу не містить реалізації його методів, документація для інтерфейсу повинна містити опис того, що вони виконують.

Щоб впровадити інтерфейс у клас, слід при оголошенні класу вказати назву інтерфейсу в переліку базових класів:

class TheClass : IInterfaceName

{ ... }

Якщо клас, у який впроваджують інтерфейс, є похідним, то його базовий клас у списку успадкованих класів повинен бути першим. Після назви базового класу через кому можна вказати довільну кількість інтерфейсів (у той же час базовий клас може бути тільки один):

class TheClass : TheBaseClass, IInterface1, IInterface2

{ ... }

У лістингу 6.2 оголошено інтерфейс IInfo (рядки 1…5) з одним методом GetDescription рядкового типу та однією властивістю ItemMark рядкового типу тільки для читання. Метод GetDescription має повертати опис об'єкта, а властивість ItemMark – "марку" об'єкта з базовою інформацією про виробника

та модель. Інтерфейс IInfo містить оголошення методу та властивості без їх реалізації.

Лістинг 6.2. Оголошення та реалізація інтерфейсу

```
interface IInfo       // Оголошення інтерфейсу
{
   string GetDescription(); // Метод
   string ItemMark { get; } // Властивість
}
// Назва інтерфейсу
class MobilePhone : IInfo         // Клас реалізує інтерфейс IInfo
{
   public string Manufacturer;
   public string Model;
   public double ScreenSize;
   public double MemorySize;
   public string ItemMark
   {
      get
      {
         return String.Format("{0} {1}", Manufacturer, Model);
      }
   }
   public string GetDescription()
   {
      return String.Format("{0} {1} {2}\" {3}Гб", Manufacturer, Model,
                           ScreenSize, MemorySize);
   }
}
// Назва інтерфейсу
class FoodProcessor : IInfo // Клас реалізує інтерфейс IInfo
```

```
        {
            public string Manufacturer;
            public string ModelInfo;
            public double Power;
            public string ItemMark
            {
                get
                {
                    return String.Format("{0} {1}", Manufacturer, ModelInfo);
                }
            }
            public string GetDescription()
            {
                return String.Format("{0} {1} {2}Вт", Manufacturer, ModelInfo,
Power);
            }
        }
        // У параметр передаємо інтерфейс
        static void PrintUpperDescr(IInfo obj)
        {
            Console.WriteLine(obj.GetDescription().ToUpper());
        }
        static void Main(string[] args)
        {
            MobilePhone mp = new MobilePhone()
            { Manufacturer = "PhoneT", Model = "KL289", MemorySize = 2,
ScreenSize = 7 };
            FoodProcessor fp = new FoodProcessor()
            {
                Manufacturer = "KitchenTec", ModelInfo = "Astra FV-202", Power = 12
```

```
        };
        Console.WriteLine("Описи:");
        Console.WriteLine(mp.GetDescription());
        Console.WriteLine(fp.GetDescription());
        Console.WriteLine("Марки:");
        Console.WriteLine(mp.ItemMark);
        Console.WriteLine(fp.ItemMark);
        Console.WriteLine("Описи великими літерами:");
        PrintUpperDescr(mp);
        PrintUpperDescr(fp);
    }
```

Всі класи, які описують товари різних видів, повинні реалізувати інтерфейс IInfo. При оголошенні класу в переліку базових класів вказуємо назву інтерфейсу (рядки 7, 26). При цьому в класі обов'язково потрібно реалізувати всі елементи інтерфейсу. Якщо цього не зробити, виникне помилка компіляції. У лістингу 6.2 обидва класи, які описують різні види товарів (класи MobilePhone та FoodProcessor), містять метод GetDescription (рядки 19…23 та 37…41). Сігнатура цих методів (тип результату, назва, параметри) жорстко задана інтерфейсом, але реалізація може бути різна. У цьому прикладі методи GetDescription різних класів на основі полів свого класу формують опис об'єкта. Властивість ItemMark також реалізована в обох класах (рядки 14…18 та 32…36). Зауважимо, що структура методу GetDescription та властивості ItemMark (назва, тип, кількість і тип параметрів тощо) відповідають їх опису в інтерфейсі.

Крім цього, екземпляри всіх класів, які реалізують один інтерфейс, можна легко передавати в один метод для додаткової обробки даних. У лістингу 6.2 оголошено метод PrintUpperDescr (рядки 44…47) з параметром obj типу IInfo. Через цей параметр в метод можна передати будь-який клас, що реалізовує вказаний інтерфейс. При цьому посилання на об'єкт автоматично перетворюється у посилання на інтерфейс, який у ньому реалізовано. В даному

випадку метод PrintUpperDescr просто виводить на екран опис переданого через параметр об'єкта великими літерами.

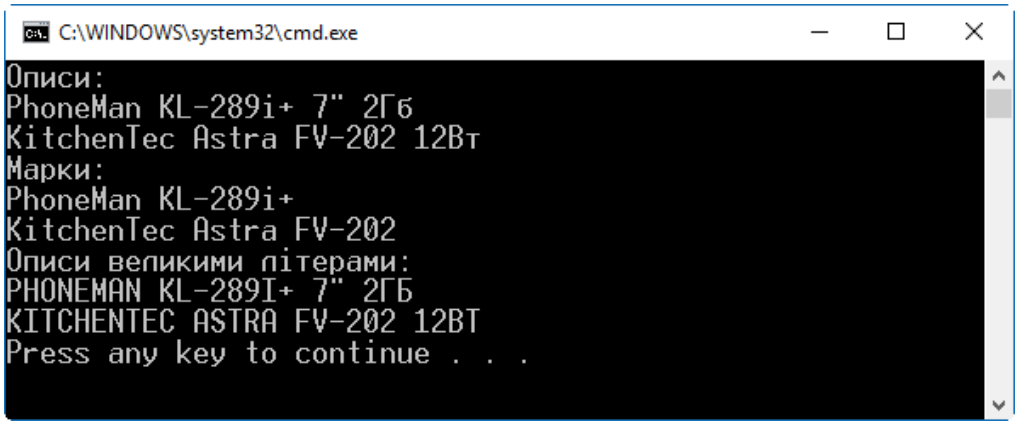

Рис. 6.1. Результат виконання виводу даних про об'єкти через інтерфейс IInfo

У методі Main створено екземпляри класів MobilePhone і FoodProcessor (рядки 51…57) та продемонстровано використання реалізованих у класах елементів інтерфейсу IInfo. В рядках 60…61 викликаємо методи інтерфейсу GetDescription, в рядках 64…65 – виводимо у вікно значення властивостей інтерфейсу ItemMark, а в рядках 68…69 – передаємо методу PrintUpperDescr об'єкти різних типів, які реалізують інтерфейс IInfo.

Використання інтерфейсу IComparable. Бібліотека C# містить набір інтерфейсів, які можна використовувати у своїх програмах. Один із них – інтерфейс IComparable, – реалізує метод, що дозволяє порівнювати екземпляри будь-якого класу між собою. Цей інтерфейс використовується методом Sort класу Array, який представляє масив.

Розглянемо такий приклад коду:

```
var intArr = new[] { 20, 30, 10, 80, 50, 40, 90 };
Array.Sort(intArr);
foreach (int i in intArr)
    {
        Console.Write("{0} ", i);
    }
```

У цьому фрагменті програми спочатку оголошуємо масив цілих чисел

intArr і відразу ініціалізуємо його несортованою множиною значень. Далі викликаємо статичний метод Sort класу Array, який сортує масив за зростанням. І в кінці виводимо відсортований масив на екран, щоб пересвідчитися, що значення масиву справді сортовані.

Таким чином, відсортувати значення масиву дуже легко. Але якщо спробувати сортувати масив не з числовими значеннями, а з екземплярами свого класу, то при виконанні програми виникне виняткова ситуація "System.ArgumentException: At least one object must implement IComparable" (Щонайменше один з об'єктів повинен реалізувати IComparable). Справа у тому, що метод Sort "не знає", як порівняти між собою об'єкти довільного визначеного програмістом класу і як їх впорядкувати. Метод Sort класу Array залежить від інтерфейсу, який називається IComparable і визначений у BCL.

Інтерфейс IComparable оголошено так:

```
public interface IComparable
{
   int CompareTo(object obj);
}
```

Цей інтерфейс має всього один метод CompareTo. Оголошення методу задає, що він отримує один параметр типу object і повертає результат типу int.

Хоча сам інтерфейс не містить реалізації цього методу, але документація .NET описує, що цей метод має робити, якщо інтерфейс IComparable буде впроваджений у клас. Згідно документації, метод CompareTo має повертати такі значення:

– від'ємне значення, якщо у впорядкованому (відсортованому) списку поточний об'єкт повинен бути розташований перед об'єктом, заданим параметром object.

– Додатне значення, якщо поточний об'єкт повинен бути розташований після об'єкта object.

– Нуль, якщо обидва об'єкти при впорядкуванні займають одну і ту ж позицію.

Алгоритм, використаний методом Sort класу Array, основує свою роботу на методі CompareTo інтерфейсу IComparable. Він порівнює два елементи масиву між собою. Тип int реалізує цей інтерфейс, тому сортування масиву цілих чисел пройшло успішно. Але довільний визначений у програмі клас не містить реалізації інтерфейсу IComparable, тому спроба виклику методу Sort спричинить виняткову ситуацію.

Щоб метод Sort міг сортувати екземпляри певного класу, цей клас повинен реалізувати інтерфейс IComparable. Для цього при оголошенні класу назву інтерфейсу слід вказати в переліку базових класів, а також забезпечити реалізацію методу CompareTo. Якщо клас впроваджує інтерфейс, він має реалізувати всі елементи інтерфейсу. Оскільки інтерфейс IComparable містить тільки один метод CompareTo, то потрібно реалізувати лише його.

Лістинг 6.3. Реалізація інтерфейсу IComparable у класі Computer

```
class Computer : IComparable
{
   public string MotherBoard;
   public string CPUModel;
   public double CPUGHz;
   public int CompareTo(object obj)
   {
      return CPUGHz > (obj as Computer).CPUGHz ? 1 :
            CPUGHz < (obj as Computer).CPUGHz ? -1 : 0;
   }
}
static void Main(string[] args)
{
   Computer[] compArr = new[]
   { new Computer() { MotherBoard = "JHs-298", CPUModel = "Intel", CPUGHz = 2.2},
     new Computer() { MotherBoard = "93-HxO92", CPUModel = "AMD",
```

```
CPUGHz = 2.1},
            new Computer() { MotherBoard = "H989", CPUModel = "Intel",
CPUGHz = 2.7},
            new Computer() { MotherBoard = "KL-092z", CPUModel = "CPU",
CPUGHz = 2.0}
        };
        Console.WriteLine("Сортування масиву об'єктів:");
        Array.Sort(compArr);
        foreach (Computer comp in compArr)
        {
          Console.WriteLine("{0}, {1}, {2:0.00} GHz", comp.MotherBoard,
                        comp.CPUModel, comp.CPUGHz);
        }
    }
```

У лістингу 6.3 описано клас Computer, у який впроваджено інтерфейс IComparable (рядки 1…12). Метод CompareTo інтерфейсу, реалізований у класі Computer, порівнює значення поля CPUGHz поточного об'єкта із значенням цього ж поля іншого об'єкта, переданого у параметрі obj. При цьому об'єкт obj приводимо до типу Computer за допомогою оператора as. Таким чином, реалізований у лістингу 6.3 метод CompareTo забезпечить сортування об'єктів Computer за значенням поля CPUGHz (тактовою частотою процесора).

У методі Main оголошено та ініціалізовано масив compArr (рядки 16…25), елементи якого є екземплярами класу Computer. Для сортування, як і в попередньо розглянутому випадку з масивом цілих чисел, викликаємо статичний метод Sort класу Array (рядок 28), передавши йому посилання на масив compArr. Щоб пересвідчитися, що масив відсортовано правильно, у циклі виводимо описи кожного об'єкта Computer у масиві (рядки 29…31). Результат роботи програми приведено на рис. 6.2. Бачимо, що об'єкти справді відсортовані за зростанням значення поля CPUGHz (останнє значення кожного рядка з даними).

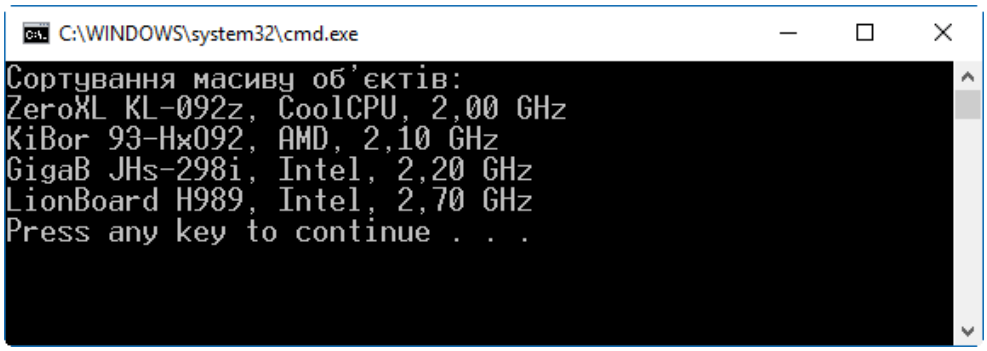

Рис. 6.2. Результат виконання сортування об'єктів класу Computer

Спадкування інтерфейсів. Інтерфейс, як і клас, може спадкувати структуру іншого інтерфейсу. Таким чином, на основі одних інтерфейсів можна створювати інші інтерфейси. Щоб вказати, що інтерфейс успадковує інші інтерфейси, імена базових інтерфейсів слід вказати так само, як при спадкуванні класів – в окремому переліку:

```
interface IInterface : IInterface1, IInterface2
{ ...}
```

На відміну від класів, які можуть успадкувати тільки один базовий клас, інтерфейси можуть спадкувати довільну кількість базових інтерфейсів. У свою чергу, кожен базовий інтерфейс може спадкувати інші інтерфейси. Результуючий інтерфейс крім своїх елементів містить елементи всіх успадкованих інтерфейсів. Відповідно клас, який впроваджує такий інтерфейс, також повинен реалізувати всі елементи всіх успадкованих інтерфейсів.

ДОДАТОК Б ПУБЛІКАЦІЯ ЗА РЕЗУЛЬТАТАМИ ДОСЛІДЖЕННЯ

ЛАБОРАТОРНИЙ ПРАКТИКУМ «РОБОТА З ІНТЕРФЕЙСАМИ, КОЛЕКЦІЯМИ ТА ІНДЕКСАТОРАМИ»

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

Метою дослідження є обґрунтування доцільності модернізації лабораторного практикуму з дисципліни «Програмна інженерія» на тему «Робота з інтерфейсами, колекціями та індексаторами».

Автором виконано аналіз характеристик об'єктів галузі проєктування об’єктно-орієнтованої ієрархії класів прикладних програмних систем з використанням інтерфейсів, параметризованих класів та узагальнених колекцій. Встановлено, що ці концепції є основними будівельними блоками для створення складних та ефективних додатків у C# [1]. Вони дозволяють писати код, який легше підтримувати, розширювати та масштабувати.

Виконана постановка нових лабораторних робіт: 1) інтерфейси в C#, наслідування інтерфейсів; 2) узагальнені колекції та ітератори в C#; 3. параметризовані класи та індексатори в C#.

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

Апробація результатів дослідження виконана під час виконання проведення лабораторних занять з дисципліни «Програмна інженерія» бакалаврами заочної форми навчання спеціальності 015 Професійна освіта (Цифрові технології) БННПІ ХНУ.

Встановлено, що опанування здобувачами вищої освіти запропонованого лабораторного практикуму забезпечить ґрунтовне формування програмних результатів навчання [2]. Це, у свою чергу, сприятиме розширенню їхнього професійного кругозору, підвищенню кваліфікації та розвитку навичок, необхідних для виконання складних завдань у різних сферах діяльності. Здобуті знання та вміння забезпечать їм можливість ефективно адаптуватися до вимог сучасного ринку праці, підвищать їхню конкурентоспроможність, а також сприятимуть активній участі у професійних і наукових спільнотах, що вимагають високого рівня підготовки й експертності.

Список використаних джерел

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

2. 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.