Чернівецький національний університет імені Юрія Федьковича
(повне найменування закладу вищої освіти)

Факультет математики та інформатики
(назва факультету/навчально-наукового інституту)

Кафедра математичного моделювання
(назва кафедри, що забезпечує викладання)

“ЗАТВЕРДЖУЮ”
Декан факультету
математики та інформатики
_____ Ольга МАРТИНЮК
“28” серпня 2026 року

РОБОЧА ПРОГРАМА
навчальної дисципліни
Проєктування програмних систем
(назва навчальної дисципліни)

обов’язкова
(вказати: обов’язкова)

Освітньо-професійна програма Системний аналіз
(назва програми)

Спеціальність 124 Системний аналіз
(вказати: код, назва)

Галузь знань 12 Інформаційні технології
(вказати: шифр, назва)

Рівень вищої освіти перший (бакалаврський)
(вказати: перший (бакалаврський) / другий (магістерський) / третій (освітньо-науковий)

Факультет математики та інформатики
(назва факультету/ навчально-наукового інституту, на якому здійснюється підготовка фахівців за вказаною освітньою програмою)

Мова навчання українська
(вказати: на якій мові читається дисципліна)

Чернівці 2026 рік

Робоча програма навчальної дисципліни «*Проєктування програмних систем*» складена відповідно до освітньо-професійної програми Системний аналіз

Розробники:
Піддубна Лариса Андріївна, доцент кафедри математичного моделювання, кандидат фіз.-мат. наук, доцент;
Перцов Андрій Сергійович, доцент кафедри математичного моделювання, кандидат фіз.-мат. наук, доцент

Викладачі, що забезпечують читання даної навчальної дисципліни:
Піддубна Лариса Андріївна, доцент кафедри математичного моделювання, кандидат фіз.-мат. наук, доцент;
Перцов Андрій Сергійович, доцент кафедри математичного моделювання, кандидат фіз.-мат. наук, доцент

Погоджено з гарантом ОП _____ **Андрій ПЕРЦОВ**

Затверджено на засіданні кафедри математичного моделювання
Протокол № 1 від «27» серпня 2026 року

Завідувач кафедри _____ **Ігор ЧЕРЕВКО**

Схвалено методичною радою факультету математики та інформатики
Протокол № 1 від «28» серпня 2026 року

Голова методичної ради _____ **Тоня ФРАТАВЧАН**

Мета навчальної дисципліни. Дисципліна "Проєктування програмних систем" спрямована на узагальнене ознайомлення із інструментами, методиками, технологіями, системами розробки програмного забезпечення, вивчення систем управління проектними розробками цільового програмного забезпечення.

Знання і досвід, набуті в цій дисципліні, будуть корисними в майбутній практичній діяльності студентів при проектуванні чи розробці програм чи систем із залученням учнів/студентів навчальних закладів, де вивчається інформатика.

Метою дисципліни є опанування студентами теоретичних знань архітектури системного програмного забезпечення, побудови, функціонування, використання засобів операційних систем, технології контейнерів для реалізації упаковки, розгортання та функціонування програмного забезпечення.

Пререквізити. Навчальні дисципліни: "Програмування", "Бази даних та інформаційні системи".

Результати навчання. У результаті вивчення навчальної дисципліни студенти повинні

знати основні теоретичні та практичні питання теорії архітектури програмних засобів;

вміти формувати структуру програмного засобу відповідно до обраної архітектури, оформляти документацію, коди відповідно до визнаних стандартів, згідно зі стандартом вищої освіти з урахуванням таких загальних і фахових компетентностей, а також програмних результатів навчання:

ЗК02. Здатність застосовувати знання у практичних ситуаціях

ЗК03. Здатність планувати і управляти часом

ЗК04. Знання та розуміння предметної області та розуміння професійної діяльності

ЗК07. Здатність до пошуку, оброблення та аналізу інформації з різних джерел

ЗК08. Здатність бути критичним і самокритичним

ЗК11. Здатність генерувати нові ідеї (креативність)

ЗК12. Здатність працювати в команді

ФК6. Здатність до комп'ютерної реалізації математичних моделей реальних систем і процесів; проєктувати, застосовувати і супроводжувати програмні засоби моделювання, прийняття рішень, оптимізації, обробки інформації, інтелектуального аналізу даних.

ФК7. Здатність використовувати сучасні інформаційні технології для комп'ютерної реалізації математичних моделей та прогнозування поведінки конкретних систем а саме: об'єктно-орієнтований підхід при проєктуванні складних систем різної природи, прикладні математичні пакети, застосування баз даних і знань.

ФК8. Здатність організовувати роботу з аналізу та проєктування складних систем, створення відповідних інформаційних технологій та програмного забезпечення.

ПР8. Володіти сучасними методами розробки програм і програмних комплексів та прийняття оптимальних рішень щодо складу програмного забезпечення, алгоритмів процедур і операцій.

ПР10. Знати архітектуру сучасних обчислювальних систем і комп'ютерних мереж.

ПР11. Знати і вміти застосовувати на практиці системи управління базами даних і знань та інформаційні системи.

ПР13. Проєктувати, реалізовувати, тестувати, впроваджувати, супроводжувати, експлуатувати програмні засоби роботи з даними і знаннями в комп'ютерних системах і мережах.

Опис навчальної дисципліни
Загальна інформація

Форма навчання	Рік підготовки	Семестр	Кількість		Кількість годин						Вид підсумкового контролю
			кредитів	годин	лекції	практичні	семінарські	лабораторні	самостійна робота	індивідуальні завдання	
Денна	3	6	5	150	30	–	–	30	90	–	екзамен
Заочна											

Структурний зміст навчальної дисципліни

Назви змістових модулів і тем навчальних занять	Кількість годин											
	денна форма						Заочна форма					
	усього	у тому числі					усього	у тому числі				
		л	п	лаб	інд	с.р.		л	п	лаб	інд	с.р.
1	2	3	4	5	6	7	8	9	10	11	12	13
Змістовий модуль 1. Види та парадигми проєктування												
Тема 1.1 . Базові питання. Види програмного забезпечення. Моделювання проєкту з UML: діаграми статичні та динамічні, логічні та фізичні. Види проєктування: архітектурне (верхній рівень) та деталізоване проєктування (класів, атрибутів, операцій), проєктування інтерфейсу користувача	12	4		4		4	–	–	–	–	–	–
Тема 1.2. Парадигми проєктування: функціональна декомпозиція згори вниз, архітектура, орієнтована на дані, об'єктно-орієнтований аналіз та проєктування, подієво-керована архітектура.	18	4		4		10	–	–	–	–	–	–

Життєвий цикл і етапи розробки програмного забезпечення.												
Разом за змістовим модулем 1.	**30**	**8**		**8**		**14**	–	–	–	–	–	–
Змістовий модуль 2. Реалізація архітектурних моделей												
Тема 2.1. Ідентифікація класів предметної області. UML-діаграми ієрархії класів: моделювання підсистем, класів та зв'язків між ними	22	4		4		14	–	–	–	–	–	–
Тема 2.2 . Проєктування сценаріїв реалізації варіантів використання на основі UML-діаграм послідовностей та комунікації Роль архітектури. Стандартні архітектури: клієнт-серверна та n-рівнева архітектура, Model View Controller	14	2		2		10						
Тема 2.3 Архітектурні моделі та патерни проєктування (Abstract Factory, Facade, Decorator, Flyweight, Visitor, Observer, Proxy,Strategy, Chain of Responsibility)	22	4		4		14						
Тема 2.4. Вимоги до оформлення коду: стиль, розбиття на структуровані одиниці, найменування	20	4		4		12						

змінних, класів, об’єктів тощо Засоби автоматичної ґенерації програмного коду на основі трансформацій UML-модель – код ООП-мовою, повторне використання коду ПЗ.												
Тема 2.5. Налагодження: Точки зупинки (Breakpoints), Спостереження за змінними (Variable Watch), Виведення на консоль (Console Output), Налагоджувач (Debugger), Аналізатори коду (Code Analyzers).	10	2				8						
Тема 2.6 . Керування конфігурацією програмного забезпечення та контроль версій.	11	2		4		5						
Тема 2.7. Забезпечення якості: тестування, верифікація, валідація Призначення, спільне та відмінності процесів тестування, верифікації, валідації Види тестів: модульні, інтеграційні, регресійні, системні, валідаційні Тестування методами білої та чорної скрині	7	2				5	–	–	–	–	–	–

Розробка через тестування (Test-driven development)												
Тема 2.8 . Додаткові техніки верифікації та валідації: інспекція коду, перевірка на відповідність стандартам і вимогам, оцінювання зручності використання та користувацького досвіду, перевірка продуктивності та масштабованості	14	2		4		8	–	–	–	–	–	–
Разом за змістовим модулем 2	**120**	**22**		**22**		**76**	–	–	–	–	–	–
Всього годин	**150**	**30**		**30**		**90**	–	–	–	–	–	–
Підсумковий модуль	**екзамен**											

Тематика лекційних занять з переліком питань

№	Назва теми з основними питаннями
1	Тема 1.1 Базові питання. Види програмного забезпечення. Моделювання проєкту з UML: діаграми статичні та динамічні, логічні та фізичні. Види проєктування: архітектурне (верхній рівень) та деталізоване проєктування (класів, атрибутів, операцій), проєктування інтерфейсу користувача
2	Тема 1.2 Парадигми проєктування: функціональна декомпозиція згори вниз, архітектура, орієнтована на дані, об'єктно-орієнтований аналіз та проєктування, подієво-керована архітектура. Життєвий цикл і етапи розробки програмного забезпечення.
3	Тема 2.1. Ідентифікація класів предметної області. UML-діаграми ієрархії класів: моделювання підсистем, класів та зв’язків між ними
4	Тема 2.2 Проєктування сценаріїв реалізації варіантів використання на основі UML-діаграм послідовностей та комунікації Роль архітектури. Стандартні архітектури: клієнт-серверна та n-рівнева архітектура, Model View Controller
5	Тема 2.3 Архітектурні моделі та патерни проєктування (Abstract Factory, Facade, Decorator, Flyweight, Visitor, Observer, Proxy,Strategy, Chain of Responsibility)
6	Тема 2.4 Вимоги до оформлення коду: стиль, розбиття на структуровані одиниці, найменування змінних, класів, об’єктів тощо

	Засоби автоматичної ґенерації програмного коду на основі трансформацій UML-модель – код ООП-мовою, повторне використання коду ПЗ.
7	Тема 2.5 Налагодження: Точки зупинки (Breakpoints), Спостереження за змінними (Variable Watch), Виведення на консоль (Console Output), Налагоджувач (Debugger), Аналізатори коду (Code Analyzers).
8	Тема 2.6 Керування конфігурацією програмного забезпечення та контроль версій.
9	Тема 2.7 Забезпечення якості: тестування, верифікація, валідація Призначення, спільне та відмінності процесів тестування, верифікації, валідації Види тестів: модульні, інтеграційні, регресійні, системні, валідаційні Тестування методами білої та чорної скрині Розробка через тестування (Test-driven development)
10	Тема 2.8 Додаткові техніки верифікації та валідації: інспекція коду, перевірка на відповідність стандартам і вимогам, оцінювання зручності використання та користувацького досвіду, перевірка продуктивності та масштабованості

Детальні презентації до кожної лекції наведено на сайті електронного навчання на сторінці курсу https://moodle.chnu.edu.ua/course/view.php?id=4954

Тематика лабораторних занять з переліком питань

№	Назва теми (завдання)
1	*Лабораторна робота №1*. Розробка технічного завдання **Завдання.** 1. Формально описати вимоги «замовника». 2. Описати функціональність продукту. 3. Описати інтерфейс потенційних користувачів. 4. Скласти рекомендації щодо потреб збереження інформації.
2	*Лабораторна робота* №2. Написання UseCases **Завдання.** 1. Відповідно до обраної предметної області на основі проведеного аналізу вимог замовника (лабораторна робота №1) розробити три різні usecases (по одному в короткій, поверхневій та повній формах відповідно) для свого проекту. Повна форма опису має містити всі пункти наведені в таблиці 1. Головний успішний сценарій повинен мати не менше 10 кроків. Передбачити не менше 5 альтернативних сценаріїв. 2. У середовищі app.diagrams.net створити діаграму варіантів використання для обраного варіанта комп'ютерної системи. Діаграма повинна містити усіх акторів (користувачів системи) та по три варіанти використання для кожного актора. Пов'язати варіанти використання та акторів, при цьому використати усі види зв'язків (unidirectional association, generalization, extend relationship, include relationship)
3	*Лабораторна робота №3* Побудова діаграм класів (Class Diagrams) Завдання 1. 1. Для всіх об'єктів на діаграмах взаємодії призначити (створити) певний клас; для кожного повідомлення призначити (створити) відповідний метод (операцію) для класу об'єкта-приймача.

	2. Розташувати створені класи з переліком операцій на діаграмі класів. 3. Для кожної операції визначити атрибути, які вона використовує та при необхідності додати їх до списку атрибутів класу. 4. Для кожного атрибуту задати логічний тип даних, для кожної операції логічний тип даних для return value та для переліку аргументів, якщо вони присутні. 5. Пов'язати класи на діаграмі класів, використовуючи різні типи відношень (асоціацію, агрегацію, композицію, наслідування, інстанціювання). Вимоги 1. Діаграма класів повинна містити не менше 10 класів. 2. Для кожного класу визначити не менше 5 атрибутів та 5 операцій. 3. По можливості використати всі типи відношень між класами. Завдання 2. Створити одну діаграму станів для опису процесу функціонування обраної системи в цілому і дві діаграми для конкретних елементів системи. Використовувати діаграму станів для авторизації користувачів забороняється. Вимоги 1. Кожна діаграма повинна містити не менше 6 станів. 2. По можливості використати обидва типи переходів (звичайний і рефлексивний). 3. Для кожного переходу визначити хоча б одну з характеристик (тригер, гранична умова, дія).
4	*Лабораторна робота №4* Застосування патернів програмування Завдання 1. Аналізуючи діаграми класів (Лабораторна робота №3), обґрунтувати застосування підібраних для реалізації патернів. 2. Використовуючи лабораторні роботи із дисциплін професійної підготовки, створити репозиторій, розмістити файли проектів. Надати доступ викладачам на github / gitlab ресурсі.

Детальні завдання до кожної лабораторної роботи наведено на сайті електронного навчання на сторінці курсу https://moodle.chnu.edu.ua/course/view.php?id=4954

Контрольні питання до лабораторних робіт

Лабораторна робота №1

1. Що таке програмний засіб? **(1 бал)**
2. Перерахуйте етапи розвитку технології програмування. **(1 бал)**
3. Які особливості розробки програмних продуктів? **(1 бал)**
4. Що називається життєвим циклом ПЗ? **(1 бал)**
5. Які стадії входять до ЖЦ ПЗ? **(1 бал)**
6. Опишіть етапи стадії розробки ПЗ. **(1 бал)**
7. Опишіть етапи стадії організації ПЗ. **(1 бал)**
8. Опишіть стадію вивід з експлуатації. **(1 бал)**
9. Що називається розробкою програмного забезпечення? **(1 бал)**

Лабораторна робота №2

1. Для чого призначена діаграма класів (class diagram)? **(1 бал)**
2. Назвіть її основні елементи. **(1 бал)**
3. Дайте визначення класу. Які типи класів ви знаєте? **(1 бал)**
4. Чим відрізняється абстрактний клас від конкретного? **(1 бал)**
5. Чим відрізняється абстрактний клас від інтерфейсу? **(1 бал)**
6. Що таке параметризований клас? Дайте визначення процесу інстанціювання. Наведіть приклад. **(2 бал)**
7. .Назвіть основні характеристики класу. **(1 бал)**
8. Що таке атрибути класу? Які типи атрибутів ви знаєте? **(1 бал)**

Лабораторна робота №3

1. Що таке діяльність? **(1 бал)**
2. Чим діяльність відрізняється від стану? **(1 бал)**
3. Що таке стан прийняття рішення? **(1 бал)**
4. Що таке стан синхронізації? **(1 бал)**
5. Для чого призначена діаграма послідовності (sequence diagram)? Наведіть її основні елементи. **(1 бал)**
6. Дайте характеристику компонента. Що може виступати в ролі компонента? Наведіть приклад. **(1 бал)**
7. Назвіть головну відмінність між пакетами та компонентами. Шаблони проектування програмного забезпечення. Призначення. Коротка історія створення. **(1 бал)**
8. Класифікація шаблонів проектування ПЗ. **(1 бал)**

Лабораторна робота № 4

1. Що таке патерн (шаблон) проектування? Яку основну проблему він вирішує? **(1 бал)**
2. Назвіть основні типи патернів (за каталогом GoF) і поясніть, чим вони відрізняються. **(1 бал)**
3. Чому важливо обґрунтовувати вибір конкретного патерну, а не використовувати його скрізь? **(1 бал)**
4. Які проблеми/недоліки ви знайшли в оригінальній діаграмі класів (з абораторної роботи №3), які вирішуються за допомогою патернів? **(1 бал)**
5. Як саме обраний патерн покращив архітектуру програми? (Наприклад: зменшив зв'язаність, збільшив сприйняття, спростив тестування). **(1 бал)**
6. Чи має обраний вами патерн будь-які недоліки? Як вони можуть вплинути на вашу програму в майбутньому? **(1 бал)**
7. Як створювався ваш репозиторій? Чи був він ініціалізований з .gitignore файлом? **(1 бал)**
8. Опишіть вашу робочу гілку (branch). Чи використовували ви окремі гілки для різних функцій (feature branches)? **(1 бал)**
9. Як виглядає історія ваших комітів? Чи є коміти дрібними та логічно завершеними, з зрозумілими повідомленнями? **(1 бал)**
10. Наведіть приклад повідомлення коміту, яке відображає внесені зміни. Чи є у вашому репозиторії файл README.md? Що в ньому міститься? **(1 бал)**

Завдання для самостійної роботи студентів

Самостійна робота складається з повторення матеріалу, засвоєного на лекціях, самостійного опанування частини теоретичного матеріалу, роботи з контрольними запитаннями та завданнями.

Студенти можуть отримувати до 1 бала в рахунок виконання завдань СРС під час кожного лекційного заняття за правильні відповіді на запитання лектора, активне обговорення багатоваріантних підходів до рішення представленої лектором проблеми (для активізації пошукової та дослідної діяльності студентів).

№	Назва теми	Завдання для самостійної роботи	К-сть год.
1	Теми 1.1-2.8	Підготовка до лекційних занять	30
2	Теми 1.1-2.6	Підготовка до лабораторних занять	40
3	Теми 1.1-2.8	Підготовка до підсумкового модуль-контролю	20

Зміст завдань для самостійної роботи

№	Тема	Зміст	Год.
1	Тема 1.1 Базові питання. Види програмного забезпечення. Моделювання проєкту з UML: діаграми статичні та динамічні, логічні та фізичні. Види проєктування: архітектурне (верхній рівень) та деталізоване проєктування (класів, атрибутів, операцій), проєктування інтерфейсу користувача	Класифікація ПЗ: Скласти таблицю з видами програмного забезпечення (системне, прикладне, інструментальне тощо). Надайте по 2-3 конкретних приклади для кожного типу та охарактеризуйте їх призначення. Визначте, до якого типу ПЗ належать програми: операційна система, текстовий редактор, компілятор, гра, CRM-система. UML-діаграми: Створіть порівняльну таблицю для статичних (наприклад, діаграма класів, компонентів) та динамічних (наприклад, послідовності, станів) діаграм UML. Вкажіть: Їхню мету. Основні елементи. У яких фазах проектування вони використовуються. Поясніть, чому діаграма розгортання є фізичною, а діаграма класів — логічною.	5
2	Тема 1.2 Парадигми проєктування: функціональна декомпозиція згори вниз, архітектура, орієнтована на дані, об'єктно-орієнтований аналіз та проєктування, подієво-керована архітектура. Життєвий цикл і етапи розробки програмного забезпечення.	Складіть таблицю, в якій охарактеризуйте кожну з парадигм: Функціональна декомпозиція (зверху вниз) Архітектура, орієнтована на дані Об'єктно-орієнтований аналіз та проектування (ООП) Подієво-керована архітектура Для кожної парадигми вкажіть: Основні принципи та підходи. Переваги та недоліки. Типові сфери застосування (наприклад, ООП — GUI, ігри; подієво-керована архітектура — реальний час, IoT).	10
3	Тема 2.1. Ідентифікація класів предметної області. UML-діаграми ієрархії класів: моделювання підсистем, класів та зв’язків між ними	Складіть список методів ідентифікації класів у предметній області (аналіз іменників, вимог, сценаріїв використання тощо). Поясніть, які критерії використовуються для відбору класів-кандидатів у фінальну модель (наприклад, відповідальність, атомарність, незалежність). Типи зв'язків між класами: Створіть таблицю з типами зв'язків у UML (асоціація, агрегація, композиція, успадкування, залежність). Для кожного типу наведіть визначення, позначення на діаграмі та конкретний приклад з предметної області (наприклад, композиція: «автомобіль – двигун»).	5

4	Тема 2.2 . Проєктування сценаріїв реалізації варіантів використання на основі UML-діаграм послідовностей та комунікації Роль архітектури. Стандартні архітектури: клієнт-серверна та n-рівнева архітектура, Model View Controller	UML-діаграми поведінки: Складіть порівняльну таблицю діаграм послідовностей та діаграм комунікації (колишніх діаграм співробітництва). Вкажіть: Основні елементи (об'єкти, повідомлення, актори, лінія життя тощо). Назначення кожної діаграми. Переваги та недоліки. Поясніть, які діаграми краще використовувати для моделювання часових аспектів, а які — для акценту на зв'язках між об'єктами. Архітектурні шаблони: Дайте визначення таким архітектурним підходам: Клієнт-серверна архітектура. N-рівнева архітектура. Model-View-Controller (MVC). Для кожного підходу наведіть: Основні компоненти та їхню роль. Переваги та недоліки. Типові сфери застосування.	8
5	Тема 2.3 Архітектурні моделі та патерни проєктування (Abstract Factory, Facade, Decorator, Flyweight, Visitor, Observer, Proxy,Strategy, Chain of Responsibility)	Класифікація та призначення: Складіть таблицю з патернів, вказавши для кожного: Тип (породжувальний, структурний, поведінковий). Основне призначення (яку проблему вирішує). Ключові учасники (головні класи/інтерфейси). Простий приклад аналогії з реального життя (наприклад, Facade - кнопка "все включити" на пульті). Патерни для аналізу: Abstract Factory, Facade, Decorator, Flyweight, Visitor, Observer, Proxy, Strategy, Chain of Responsibility.	8
6	Тема 2.4 Вимоги до оформлення коду: стиль, розбиття на структуровані одиниці, найменування змінних, класів, об’єктів тощо Засоби автоматичної ґенерації програмного коду на основі трансформацій UML-модель – код ООП-мовою, повторне використання коду ПЗ.	Вимоги до якості коду: Складіть конспект-пам'ятку основних правил оформлення коду (на основі рекомендацій для обраної мови програмування, наприклад, PEP 8 для Python, Google Java Style Guide тощо). Включіть: Правила іменування (змінні, класи, методи, константи). Вимоги до відступів, довжини рядків, коментарів. Принципи розбиття коду на функції/класи/модулі (DRY, KISS, SOLID). Автоматична генерація коду: Дослідіть принцип роботи засобів автоматичної генерації коду з UML-діаграм (наприклад, зв'язок між UML-класами та кодом на мові програмування). Складіть таблицю популярних інструментів для генерації коду (наприклад, Enterprise Architect, Visual Paradigm, онлайн-генератори) та їхніх можливостей.	6

7	Тема 2.5 Налагодження: Точки зупинки (Breakpoints), Спостереження за змінними (Variable Watch), Виведення на консоль (Console Output), Налагоджувач (Debugger), Аналізатори коду (Code Analyzers).	Інструменти налагодження: Складіть таблицю основних інструментів налагодження. Для кожного інструменту вкажіть: Призначення (наприклад, точка зупинки - для паузи виконання програми). Як використовується (практичний приклад). Переваги та обмеження. Інструменти: точки зупинки (breakpoints), спостереження за змінними (watch), консольний вивід, налагоджувач (debugger). Аналізатори коду: Дослідіть, що таке аналізатори коду (наприклад, ESLint для JavaScript, Pylint для Python, SonarQube).	4
8	Тема 2.6 Керування конфігурацією програмного забезпечення та контроль версій.	1. **Основні поняття:** o Дайте визначення таким поняттям: – Контроль версій (Version Control). – Система контролю версій (VCS). – Централізована та розподілена VCS. – Управління конфігурацією програмного забезпечення (SCM). o Поясніть, чому керування конфігурацією є важливим етапом у життєвому циклі розробки ПЗ. 2. **Огляд інструментів:** o Складіть порівняльну таблицю популярних систем контролю версій (Git, Subversion (SVN), Mercurial). Вкажіть: – Тип системи (централізована/розподілена). – Основні переваги та недоліки. – Сфери застосування.	4
9	Тема 2.7 Забезпечення якості: тестування, верифікація, валідація Призначення, спільне та відмінності процесів тестування, верифікації, валідації Види тестів: модульні, інтеграційні, регресійні, системні, валідаційні Тестування методами білої та чорної скрині Розробка через тестування (Test-driven development)	Верифікація, валідація та тестування: Складіть таблицю, що відображає призначення, спільне та відмінності між: Верифікацією (чи правильно ми робимо продукт?) Валідацією (чи робимо ми правильний продукт?) Тестуванням (процес виявлення дефектів). Наведіть конкретні приклади кожного процесу для проекту "Інтернет-магазин". Види тестування: Дайте визначення та опишіть призначення наступних видів тестування: Модульне (Unit testing). Інтеграційне (Integration testing). Системне (System testing). Регресійне (Regression testing). Валідаційне (Acceptance testing). Для кожного виду наведіть приклад тестового сценарію для функції "Додати товар у кошик".	5

10	Тема 2.8 Додаткові техніки верифікації та валідації: інспекція коду, перевірка на відповідність стандартам і вимогам, оцінювання зручності використання та користувацького досвіду, перевірка продуктивності та масштабованості	Інспекція коду (Code Review): Дайте визначення інспекції коду. Складіть список критеріїв, на які звертають увагу під час інспекції (читабельність, відповідність стандартам, наявність помилок, безпека тощо). Опишіть процес проведення інспекції (підготовка, проведення, виправлення недоліків). Перевірка на відповідність: Які стандарти та вимоги можуть бути важливими для ПЗ? (наприклад, код-стайл, ISO, GDPR, accessibility). Які інструменти можна використовувати для автоматизації перевірки на відповідність? Користувацький досвід (UX): Поясніть, чому оцінка зручності використання є частиною валідації. Які методи використовуються для оцінки UX? (наприклад, A/B тестування, юзабіліті-тестування, збір зворотного зв'язку). Продуктивність і масштабованість: Дайте визначення продуктивності та масштабованості. Які інструменти використовуються для тестування продуктивності? (наприклад, JMeter, LoadRunner).	5

Самостійна робота студентів використовується при вивченні наступних тем і передбачає опрацювання теоретичного матеріалу, результати якого застосовуються під час виконання лабораторних робіт.

Методи навчання

Методи навчання та викладання: лекції, лабораторні заняття, електронне навчання з використанням системи Moodle, тестування.

Методи формування професійної компетентності: розповідь, пояснення, бесіда, демонстрація, візуалізація, дискусія тощо.

Методи формування практичних умінь та навичок: розв’язування задач лабораторних робіт, виконання завдань, розробка та аналіз алгоритмів і програмного коду, захист звітів з лабораторних робіт.

Система контролю та оцінювання

Засобами оцінювання та демонстрування результатів навчання є: стандартизовані тести; аналітичні звіти з лабораторних робіт; презентації результатів виконаних завдань та досліджень, усний контроль у вигляді індивідуального та фронтального опитування на лекціях та лабораторних заняттях.

Формами поточного контролю є усна чи письмова (тестування, лабораторна робота) відповідь студента.

Формою підсумкового контролю є екзамен.

Критерії оцінювання поточного та підсумкового контролю

Критерієм підсумкового оцінювання є досягнення студентом мінімальних порогових рівнів оцінок (балів) за кожним передбаченим результатом навчання. Мінімальний пороговий рівень оцінки варто визначати за допомогою якісних критеріїв і трансформувати його в мінімальну позитивну оцінку використовуваної числової (рейтингової) шкали.

Система оцінювання рівня навчальних досягнень ґрунтується на принципах ECTS та є накопичувальною. Протягом семестру студенти виконують 4 лабораторних робіт. Кожна лабораторна робота оцінюється по 15 балів.

Виконуючи завдання лабораторної роботи, студент повинен оформити і завантажити для подальшої перевірки на сайт електронного навчання звіт разом із працездатними файлами програмної реалізації завдань ЛР (правила оформлення наведені на сторінці навчальної дисципліни на сайті).

50% балів, відведених на оцінювання ЛР, студент отримує за працюючий програмний продукт, в якому реалізовано всі завдання ЛР та оформлений звіт. Решта 50% балів виставляється після захисту студентом виконаного звіту. На захисті звіту з ЛР студент має відповісти на питання щодо власної реалізації. При відповіді на теоретичні питання та питання щодо своєї розробки БД у випадку неістотної помилки знімається 10-20% балів, а у випадку істотної 20-40% балів, якщо ж студент не опанував теоретичний матеріал, плутається в означеннях, наводить логічно невірні твердження, то знімається до 50% балів від усієї суми балів за ЛР.

Максимальна кількість, яку можна набрати на підсумковому модулі (тестування) – 40 балів.

Підсумкова оцінка виставляється за результатами суми балів, набраних на змістових модулях під час семестру та підсумковому модулі згідно з нижче наведеною таблицею.

Розподіл балів, які отримують студенти

Поточне оцінювання: *аудиторна та самостійна робота*						Модуль контроль	Сума
Змістовий модуль №1		Змістовий модуль № 2					
Тема 1.1	Тема 1.2	Тема 2.3	Тема 2.4	Тема 2.5	Тема 2.6		
15	15	15		15		40	100

Шкала оцінювання: національна та ЄКТС

Оцінка за національною шкалою	**Оцінка за шкалою ECTS**	
	Оцінка (бали)	**Пояснення за розширеною шкалою**
Відмінно	A (90-100)	відмінно
Добре	B (80-89)	дуже добре
	C (70-79)	добре
Задовільно	D (60-69)	задовільно
	E (50-59)	достатньо

Незадовільно	FX (35-49)	(незадовільно) з можливістю повторного складання
	F (1-34)	(незадовільно) з обов'язковим самостійним опрацюванням освітнього компоненту до перескладання

Перелік питань для самоконтролю та підсумкового контролю навчальних досягнень студентів

ЗРАЗКИ ТЕОРЕТИЧНИХ ПИТАНЬ

1. Що таке програмний засіб?
2. Перерахуйте етапи розвитку технології програмування.
3. Які особливості розробки програмних продуктів?
4. Що називається життєвим циклом ПЗ?
5. Які стадії входять до ЖЦ ПЗ?
6. Опишіть етапи стадії розробки ПЗ.
7. Опишіть етапи стадії організації ПЗ.
8. Опишіть стадію вивід з експлуатації.
9. Що називається розробкою програмного забезпечення?
10. Для чого призначена діаграма класів (class diagram)? Назвіть її основні елементи.
11. Дайте визначення класу. Які типи класів ви знаєте?
12. Чим відрізняється абстрактний клас від конкретного?
13. Чим відрізняється абстрактний клас від інтерфейсу?
14. Що таке параметризований клас? Дайте визначення процесу інстанціювання. Наведіть приклад.
15. Назвіть основні характеристики класу.
16. Що таке атрибути класу? Які типи атрибутів ви знаєте?
17. Що таке операції класу? Які типи операцій ви знаєте?
18. Дайте визначення статичних атрибутів та операцій класу.
19. Дайте визначення наслідування. Назвіть основні типи наслідування.
20. Яким чином реалізується інкапсуляція для атрибутів та операцій класу? Назвіть типи модифікаторів доступу до елементів класу.
21. Яким чином реалізується поліморфізм для класів, що знаходяться в єдиній ієрархії? Наведіть приклад.
22. Що таке віртуальні функції, для чого вони призначені?
23. Яким чином можна створити класи та їх операції з діаграм взаємодії?
24. В якому форматі відображають атрибути та операції на діаграмі класів?
25. Перелічіть можливі відношення між класами на діаграмі класів.
26. Які характеристики можна визначити для відношення асоціації?
27. Як відношення асоціації можна перетворити в агрегацію/композицію?
28. У чому відмінність між відношеннями агрегації та композиції?
29. Який клас є базовим, а який похідним при визначенні відношення наслідування?
30. Які типи класів поєднує відношення інстанціювання?

31.Для чого призначена діаграма станів та переходів (statechart diagram)?
32.Назвіть основні елементи statechart diagram.
33.Що таке стан? Перелічіть список тригерів для внутрішніх дій стану.
34.Перелічіть специфічні стани та наведіть їх призначення.
35.Для чого призначені переходи на statechart diagram?
36.Які характеристики властиві для кожного переходу? Чи всі вони обов'язкові?
37.Які типи переходів існують на діаграмі? Чим вони відрізняються?
38.Для чого призначена діаграма діяльності (activity diagram)? Перелічіть основні елементи.
39.У чому відмінність між діаграмою діяльності і діаграмою станів та переходів?
40.Що таке діяльність? Чим діяльність відрізняється від стану?
41.Що таке стан прийняття рішення?
42.Що таке стан синхронізації?
43.Для чого призначена діаграма послідовності (sequence diagram)? Наведіть її основні елементи.
44.Дайте характеристику компонента. Що може виступати в ролі компонента? Наведіть приклад.
45.Назвіть головну відмінність між пакетами та компонентами. Шаблони проектування програмного забезпечення. Призначення. Коротка історія створення.
46.Класифікація шаблонів проектування ПЗ.
47.Призначення структурних шаблонів проектування ПЗ.
48.Коротка характеристика кожного структурного шаблону.
49.Назви, призначення та мотивація шаблону Composite.
50.Структура шаблону Composite та його учасники.
51.Особливості реалізації шаблону Composite. Результат використання шаблону.
52.Назви, призначення та мотивація шаблону Decorator.
53.Структура шаблону Decorator та його учасники.
54.Особливості реалізації шаблону Decorator. Результат використання шаблону.
55.Назви, призначення та мотивація шаблону Proxy.
56.Структура шаблону Proxy та його учасники.
57.Особливості реалізації шаблону Proxy. Результат використання шаблону.
58.Види шаблону Proxy.
59.Шаблони, які використовуються сумісно з Composite, Decorator та Proxy.
60.Відмінність Decorator та Proxy в специфікації конструктора.
61.Структура шаблону Flyweight та його учасники.
62.Особливості реалізації шаблону Flyweight. Результат використання шаблону.
63.Назви, призначення та мотивація шаблону Adapter.
64.Структура шаблону Adapter та його учасники.
65.Особливості реалізації шаблону Adapter. Результат використання шаблону.
66.Назви, призначення та мотивація шаблону Bridge.
67.Структура шаблону Bridge та його учасники.
68.Особливості реалізації шаблону Bridge. Результат використання шаблону.
69.Назви, призначення та мотивація шаблону Facade.
70.Структура шаблону Facade та його учасники.
71.Особливості реалізації шаблону Facade. Результат використання шаблону.
72.Відмінність Adapter, Decorator та Proxy в специфікації конструктора.

73.Види адаптерів. Двосторонній та динамічний (pluggable) адаптери.
74.Шаблони, які використовуються сумісно з Flyweight, Adapter, Bridge, Facade.
75.Що таке рефакторинг коду? Основні принципи та підходи.

Зарахування результатів неформальної/інформальної освіти

Здобувачі вищої освіти має право на участь у неформальній/інформальній освіті.

У межах поточного контролю можуть визнаватися результати неформальної/інформальної освіти за умови наявності сертифікату або освітньої декларації про результати неформальної/інформальної освіти з питань, що відповідає тематиці курсу («Порядок визнання у Чернівецькому національному університеті імені Юрія Федьковича результатів навчання, здобутих шляхом неформальної та/або інформальної освіти», https://www.chnu.edu.ua/media/4g5fzssb/poriadok-vyznannia-rezultativ-navchannia-zdobutykh-shliakhom-neformalnoi-ta-abo-informalnoi-osvity.pdf).

Студентам можуть бути зараховані додаткові бали, отримані через неформальну освіту, до загальної суми балів, набраної з освітньої компоненти, за умови, що результати з проблеми, за якою відбувалося навчання, відповідають тематиці курсу.

Рекомендована література

1. Фаулер Мартін. Шаблони корпоративних додатків// Електронна книга
2. Scott Chacon and Ben Straub Pro Git book, – Apress // https://git-scm.com/book/en/v2
3. Pro Git. Режим доступу: https://git-scm.com/book/uk/v2
4. Керівництва GitHub. Режим доступу: https://guides.github.com/
5. Кніберг Х. Scrum та XP // Електронна книга
6. Рубін К. Основи Scrum. Практичне керівництво для гнучкої розробки ПЗ / Рубін Кеннет С. // Електронна книга
7. Мартін Р. Чистий код: створення і рефакторинг за допомогою Agile / пер. з англ. І. Бондар-Терещенко. — Харків : Вид-во «Ранок» : Фабула, 2023. — 448 с.
8. Мартін Роберт. Чиста архітектура: Мистецтво розроблення програмного забезпечення / пер. з англ. І. Бондар-Терещенко. — Харків : Вид-во «Ранок» : Фабула, 2023. — 368 с

Інформаційні ресурси

1. https://stud.com.ua/97384/informatika/oglyad_suchasnih_tehnologiy_rozrobki_programnogo_zabezpechennya_ponyattya
2. https://techexpert.ua/it-services/rozrobka-prykladnogo-pz/
3. https://armedsoft.com/ua/services/rozrobka-programnogo-zabezpechennya-ta-it-rishen
4. https://moodle.chnu.edu.ua/course/view.php?id=4954

Політика академічної доброчесності

Дотримання політики щодо академічної доброчесності учасниками освітнього процесу при вивченні освітнього компонента регламентовано чинними документами ЧНУ:

- "Кодекс академічної доброчесності та етики Чернівецького національного університету імені Юрія Федьковича" https://www.chnu.edu.ua/universytet/normatyvni-dokumenty/kodeks-akademichnoi-dobrochesnosti-ta-etyky-chernivetskoho-natsionalnoho-universytetu-imeni-yuriia-fedkovycha/

- "Положення про виявлення та запобігання академічного плагіату у Чернівецькому національному університеті імені Юрія Федьковича" https://www.chnu.edu.ua/universytet/normatyvni-dokumenty/polozhennia-pro-vyiavlennia-ta-zapobihannia-akademichnomu-plahiatu-u-chernivetskomu-natsionalnomu-universyteti-imeni-yuriia-fedkovycha/
- "Політика використання штучного інтелекту в Чернівецькому національному університету імені Юрія Федьковича" https://www.chnu.edu.ua/universytet/normatyvni-dokumenty/polityka-vykorystannia-shtuchnoho-intelektu-v-chernivetskomu-natsionalnomu-universyteti-imeni-yuriia-fedkovycha-zi-zminamy-zatverdzhenymy-vchenoiu-radoiu-chnu-imeni-yuriia-fedkovycha/

Використання здобувачами освіти інструментів штучного інтелекту (ШІ) під час виконання навчальних завдань:

- *допускається* за умови дотримання принципів академічної доброчесності (здобувач освіти несе відповідальність за зміст, достовірність і результати поданої роботи, а у разі використання ШІ зобов'язаний зазначити факт і спосіб його застосування);
- *не допускається* використання ШІ як заміни самостійного виконання завдань або з метою порушення вимог академічної доброчесності;
- *додаткові обмеження чи вимоги* щодо використання ШІ можуть визначатися викладачем в освітньому процесі для конкретних видів навчальної діяльності відповідно до особливостей дисципліни.