Робоча програма дисципліни «Об'єктно-орієнтоване програмування»

СПЕЦІАЛЬНІСТЬ 123 · КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

БАКАЛАВР

120

Годин

Загальний обсяг дисципліни

4

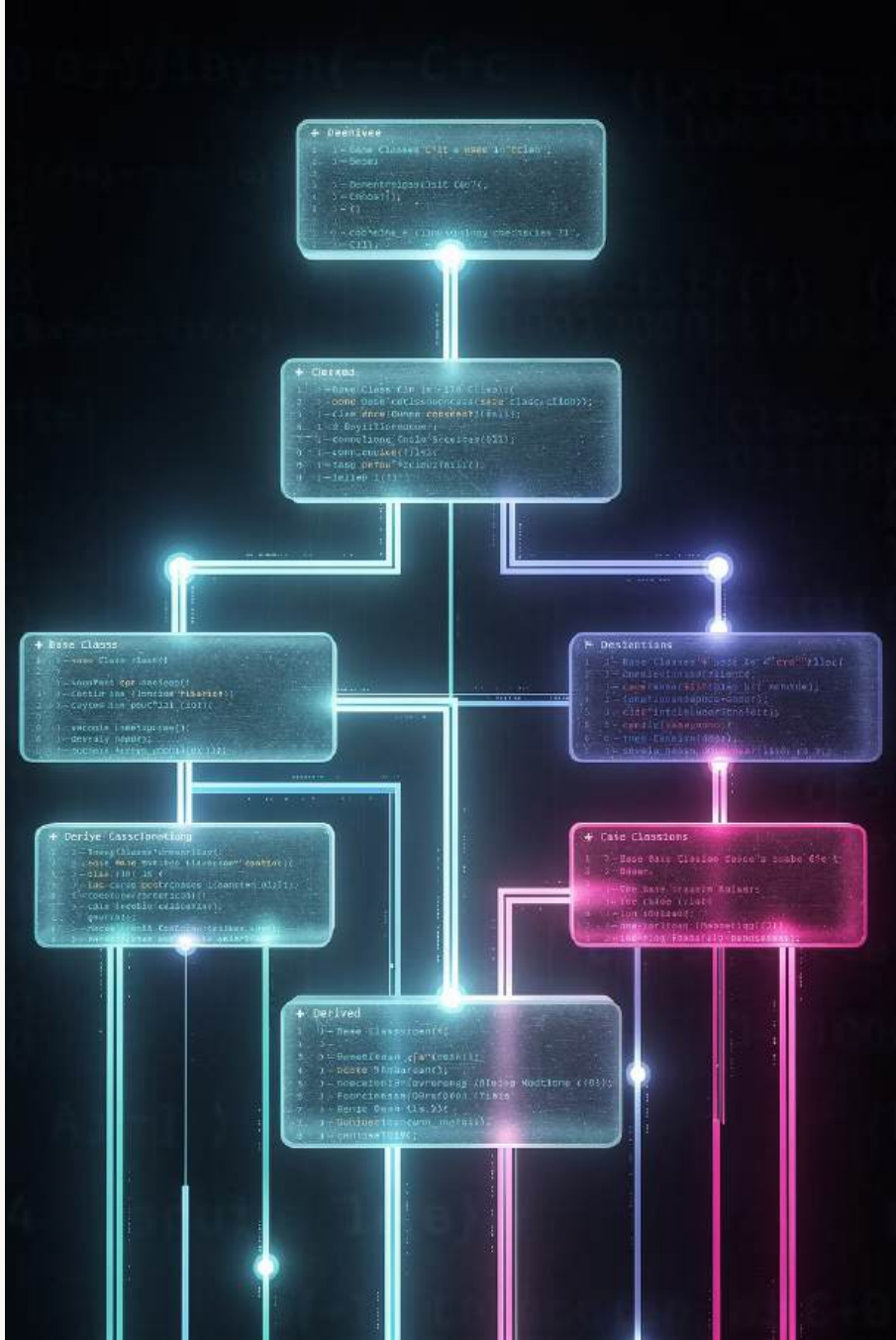
Кредити ЄКТС

Відповідно до стандарту

4

Модулі

Змістовних блоки курсу



Загальна інформація про дисципліну

Ідентифікація

Назва: Об'єктно-орієнтоване програмування

Спеціальність: 123 Комп'ютерна інженерія

Ступінь: Бакалавр

Обсяг: 120 годин / 4 кредити ЄКТС

Модулів: 4 змістовних блоки

Призначення дисципліни

Дисципліна «Об'єктно-орієнтоване програмування» формує у студентів системне розуміння об'єктно-орієнтованої парадигми програмування, принципів проєктування та реалізації програмних систем з використанням мов програмування Java, C++ або Python. Курс охоплює ключові концепції ООП: інкапсуляцію, наслідування, поліморфізм, абстракцію, патерни проєктування та принципи SOLID.

Дисципліна є обов'язковою для підготовки бакалаврів зі спеціальності «Комп'ютерна інженерія» та забезпечує формування фахових компетентностей, передбачених освітньо-професійною програмою.

Загальні компетентності

Аналітичне мислення

Здатність до аналізу, синтезу та критичного оцінювання програмних рішень: виявлення залежностей між компонентами системи, порівняння альтернативних підходів до реалізації, формування обґрунтованих технічних рішень.

Комунікація

Здатність до технічної комунікації: чітко пояснювати архітектурні рішення та алгоритми, документувати програмний код, оформлювати технічні специфікації та звіти з лабораторних робіт.

Самоорганізація

Уміння самостійно вивчати нові технології, планувати розробку програмного забезпечення та організовувати виконання завдань у визначені строки, зокрема під час виконання лабораторних та індивідуальних проєктів.

Командна робота

Відповідальність, ініціативність та готовність працювати в команді під час групового розроблення програмних проєктів, обговорення підходів до проєктування та спільного рецензування коду (code review).

Студент повинен вміти: аналізувати вимоги до програмного забезпечення; проєктувати ієрархії класів; реалізовувати об'єктно-орієнтовані рішення мовами програмування; застосовувати патерни проєктування; документувати та тестувати програмний код; працювати в команді над спільними проєктами.

Фахові компетентності

Розуміння парадигми ООП

Розуміння сутності та принципів об'єктно-орієнтованого підходу до розробки програмного забезпечення. Знання ключових концепцій: класи, об'єкти, інкапсуляція, наслідування, поліморфізм, абстракція. Вміння застосовувати ООП-підхід для вирішення практичних задач.

Проєктування програмних систем

Здатність проєктувати об'єктно-орієнтовані програмні системи: будувати ієрархії класів, визначати інтерфейси та абстрактні класи, застосовувати принципи SOLID та патерни проєктування GoF для створення гнучких і масштабованих систем.

Реалізація та тестування ПЗ

Знання методів реалізації ООП-концепцій засобами мов програмування (C++, Java або Python). Здатність розробляти, налагоджувати та тестувати об'єктно-орієнтовані програми, застосовувати методи модульного тестування та рефакторингу.

Використання інструментів розробки

Орієнтування в сучасних інструментах розробки програмного забезпечення: IDE (Visual Studio, IntelliJ IDEA, Eclipse), системи контролю версій (Git), засоби документування коду (Doxygen, Javadoc) та інструменти тестування (JUnit, Google Test).

Результати навчання

01

Пояснення концепцій ООП

Пояснювати сутність ключових понять і механізмів: клас, об'єкт, конструктор, деструктор, інкапсуляція, наслідування, поліморфізм, абстракція, інтерфейс, перевантаження операторів, шаблони.

02

Проєктування ієрархій класів

Проєктувати об'єктно-орієнтовані програмні системи із застосуванням UML-діаграм класів, визначати відношення між класами (асоціація, агрегація, композиція, наслідування), реалізовувати принципи SOLID.

03

Застосування патернів проєктування

Аналізувати програмні задачі та обирати відповідні патерни проєктування (Singleton, Factory, Observer, Strategy, Decorator та ін.), обґрунтовувати вибір та реалізовувати патерни у власних проєктах.

04

Розробка та тестування програм

Розробляти, налагоджувати та тестувати об'єктно-орієнтовані програми мовою C++/Java/Python; застосовувати засоби модульного тестування; оформлювати технічну документацію та звіти.

МОДУЛЬ 1

Основи об'єктно-орієнтованого програмування

1 — Тема 1.1

Парадигми програмування. Введення в ООП. Класи та об'єкти

2 — Тема 1.2

Інкапсуляція. Конструктори та деструктори. Специфікатори доступу

3 — Тема 1.3

Наслідування. Ієрархії класів. Перевизначення методів

4 — Тема 1.4

Поліморфізм. Віртуальні функції. Абстрактні класи та інтерфейси

МОДУЛЬ 1 · ТЕМА 1.1

Парадигми програмування. Класи та об'єкти

Лекційний матеріал (2 год.)

Лекція 1: Основні парадигми програмування: процедурне, функціональне, об'єктно-орієнтоване. Переваги ООП. Поняття класу та об'єкта. Оголошення класу, визначення методів і полів. Специфікатори доступу: public, private, protected. Стан і поведінка об'єкта.

Лекція 2: Конструктори: за замовчуванням, параметризований, копіювання. Деструктори. Список ініціалізації. Ключове слово this. Статичні члени класу. Константні методи та об'єкти.

Ключові питання лекції

- Відмінності між парадигмами програмування
- Поняття класу як типу даних і об'єкта як екземпляра класу
- Специфікатори доступу та принцип інкапсуляції
- Типи конструкторів та їх призначення
- Статичні члени класу та їх використання

Лабораторне заняття (2 год.)

Завдання 1. Розробка класу «Студент» з полями (ім'я, прізвище, залікова книжка, оцінки) та методами (отримати середній бал, вивести інформацію). Реалізація конструкторів та деструктора.

Завдання 2. Реалізація класу з демонстрацією роботи специфікаторів доступу, статичних членів та константних методів.

Завдання 3. Тестування: базові поняття ООП (10 питань).

Самостійна робота (7 год.)

- Опрацювання: Бобков В. Б. та ін. «Програмування-2. Об'єктно-орієнтоване програмування» — КПІ, 2023 — розділ «Класи та об'єкти»
- Підготовка звіту з лабораторної роботи з описом розробленого класу
- Складання таблиці «Порівняння парадигм програмування»

Очікуваний результат: студент вміє оголошувати класи, реалізовувати конструктори та деструктори, пояснювати призначення специфікаторів доступу.

МОДУЛЬ 1 · ТЕМА 1.2

Інкапсуляція. Конструктори та деструктори

1

Приховування даних

Закриті поля класу (private). Геттери та сеттери. Захист від некоректного доступу.

2

Конструктори

Конструктор за замовчуванням, параметризований, копіювання. Список ініціалізації.

3

Деструктори

Призначення деструктора. Звільнення динамічної пам'яті. Правило трьох/п'яти.

4

Перевантаження операторів

Перевантаження арифметичних, порівняльних операторів та оператора виведення.

Лекційний матеріал (2 год.)

Інкапсуляція як принцип ООП: приховування реалізації, геттери та сеттери. Дружні функції та класи (friend). Перевантаження операторів: унарних, бінарних, оператора виведення/введення. Динамічне виділення пам'яті в класах: правило трьох (деструктор, конструктор копіювання, оператор присвоєння). Правило п'яти у C++11.

Лабораторне заняття (2 год.) та СРС (7 год.)

Лаборатна: Реалізація класу «Матриця» з перевантаженням операторів (+, *, =, <<, >>); застосування правила трьох для коректного керування пам'яттю.

СРС: Опрацювання: Гришанович Т. О., Глинчук Л. Я. «Основи об'єктно-орієнтованого програмування» — ВНУ ім. Лесі Українки, 2022; підготовка аналітичної записки «Перевантаження операторів: переваги та ризики».

МОДУЛЬ 1 · ТЕМИ 1.3–1.4

Наслідування та поліморфізм

Тема 1.3 — Наслідування (Лекція 5, 2 год.)

Поняття наслідування. Відкрите, закрите та захищене наслідування. Ієрархії класів. Конструктори та деструктори при наслідуванні. Множинне наслідування: можливості та проблема «ромбу». Вирішення проблеми через віртуальне наслідування. Абстрактні класи та чисто віртуальні методи.

Лабораторна (2 год.): Розробка ієрархії класів «Фігури» (базовий клас Shape, похідні: Circle, Rectangle, Triangle). Реалізація методів обчислення площі та периметру. Демонстрація різних типів наслідування.

СРС (7 год.): Опрацювання: Рачинська А. Л., Недєва О. А., Палій К. С. «Об'єктно-орієнтоване програмування» — ОНУ, 2024; підготовка звіту «Множинне наслідування: переваги та альтернативи».

Тема 1.4 — Поліморфізм (Лекція 6, 2 год.)

Поліморфізм як ключовий принцип ООП. Статичний поліморфізм (перевантаження функцій та операторів). Динамічний поліморфізм: віртуальні функції, таблиця віртуальних методів (vtable). Чисто віртуальні функції та абстрактні класи. Інтерфейси. Вказівник і посилання на базовий клас. RTTI: dynamic_cast, typeid.

Лабораторна (2 год.): Розширення ієрархії «Фігури»: реалізація віртуального методу draw(); демонстрація динамічного поліморфізму через масив вказівників на базовий клас. Тестування: наслідування та поліморфізм.

СРС (7 год.): Опрацювання: Бобков В. Б. та ін. «Програмування-2. ООП» — КПІ, 2023 — розділ «Поліморфізм»; підготовка порівняльної таблиці «Статичний та динамічний поліморфізм».

МОДУЛЬ 2

Шаблони, STL та виключення у C++

Тема 2.1

Шаблони функцій та класів. Узагальнене програмування. Специфікація та інстанціювання шаблонів.

Тема 2.2

Стандартна бібліотека шаблонів (STL): контейнери, ітератори, алгоритми, функтори.

Тема 2.3

Обробка виключень. Механізм try-catch-throw. Ієрархія класів виключень. Гарантії безпеки.

Тема 2.4

Сучасний C++ (C++11/14/17/20): переміщення, лямбда-вирази, smart pointers, семантика переміщення.

МОДУЛЬ 2 · ТЕМА 2.1

Шаблони функцій та класів

Лекційний матеріал (2 год.)

Поняття узагальненого програмування. Шаблони функцій: синтаксис, інстанціювання, спеціалізація. Шаблони класів: параметри типів та нетипові параметри. Шаблонні методи. Часткова спеціалізація шаблонів. Шаблони зі змінною кількістю аргументів (variadic templates). Концепти у C++20.

Ключові питання

- Навіщо потрібні шаблони та яка їх роль в узагальненому програмуванні
- Відмінності між шаблонами функцій і класів
- Механізм інстанціювання шаблонів компілятором
- Повна та часткова спеціалізація шаблонів
- Variadic templates та концепти C++20

Лабораторне заняття (2 год.)

Завдання 1. Розробка шаблонного класу «Стек» (Stack<T>) з операціями push, pop, top, isEmpty. Тестування з різними типами даних (int, double, string).

Завдання 2. Реалізація шаблонних алгоритмів пошуку та сортування; демонстрація спеціалізації для типу char*.

Завдання 3. Тестування: шаблони та узагальнене програмування (10 питань).

Самостійна робота (7 год.)

- Опрацювання: Порєв В. М. «ООП: комп'ютерний практикум» — КПІ, 2022 — розділ «Шаблони»
- Підготовка звіту «Шаблонний клас черги (Queue<T>) — реалізація та тестування»
- Складання порівняльної таблиці «Шаблони vs. поліморфізм: коли що застосовувати»

Очікуваний результат: студент вміє розробляти шаблонні функції та класи, виконувати спеціалізацію шаблонів, пояснювати механізм їх інстанціювання.

МОДУЛЬ 2 · ТЕМА 2.2

Стандартна бібліотека шаблонів (STL)

Контейнери

Послідовні: vector, list, deque. Асоціативні: map, set, unordered_map.

Алгоритми

sort, find, count, transform, for_each, copy, remove_if та ін.

Рядки та потоки

std::string, stringstream. Файловий ввід/вивід. Форматування виведення.

Ітератори

Категорії ітераторів. Прямий, зворотний, const-ітератор. Range-based for.

Функтори

Функціональні об'єкти. Лямбда-вирази. std::function, std::bind.

Лекційний матеріал (2 год.)

Архітектура STL: контейнери, ітератори, алгоритми. Послідовні контейнери: vector, list, deque, array. Асоціативні контейнери: map, multimap, set, multiset. Невпорядковані контейнери: unordered_map, unordered_set. Ітератори та їх категорії. Алгоритми STL. Лямбда-вирази та функтори.

Лабораторна (2 год.) та СРС (7 год.)

Лабораторна: Розробка програми управління базою студентів з використанням STL-контейнерів (map, vector); сортування та пошук з використанням STL-алгоритмів та лямбда-виразів.

СРС: Опрацювання: Рачинська А. Л. та ін. «ООП» — ОНУ, 2024; підготовка аналітичної записки «STL як інструмент ефективної розробки програмного забезпечення».

МОДУЛЬ 2 · ТЕМИ 2.3–2.4

Обробка виключень та сучасний C++

Тема 2.3 — Обробка виключень (Лекція 7, 2 год.)

Механізм обробки виключень у C++: try, catch, throw. Стандартні класи виключень (std::exception та ієрархія). Власні класи виключень. Гарантії безпеки: базова, сувора, no-throw. Специфікатор noexcept. Порядок обробки виключень. Виключення у конструкторах та деструкторах.

Лабораторна (2 год.): Доопрацювання класу «Матриця»: додавання обробки виключень для некоректних операцій (вихід за межі, несумісні розміри). Реалізація власної ієрархії виключень.

СРС (7 год.): Опрацювання: Гришанович Т. О., Глинчук Л. Я. «Основи ООП» — ВНУ ім. Лесі Українки, 2022; підготовка есе «Обробка виключень: кращі практики та поширені помилки».

Тема 2.4 — Сучасний C++ (Лекція 8, 2 год.)

Нові можливості C++11/14/17/20: семантика переміщення (move semantics), rvalue-посилання, конструктор переміщення. Розумні вказівники: unique_ptr, shared_ptr, weak_ptr. Лямбда-вирази: синтаксис, захоплення змінних, використання. Авто-виведення типів (auto, decltype). Ініціалізація списком. Структуровані прив'язки (C++17). Корутини та концепти (C++20).

Лабораторна (2 год.): Рефакторинг попередніх програм з використанням сучасних можливостей C++: заміна сирих вказівників на smart pointers, застосування move semantics, лямбда-виразів та range-based for.

СРС (7 год.): Підготовка аналітичної записки «Еволюція C++: від C++03 до C++20 — ключові зміни та їх вплив на стиль програмування».

МОДУЛЬ 3

Патерни проєктування та принципи SOLID

Тема 3.1

Принципи SOLID. Чистий код. Рефакторинг. UML-моделювання: діаграми класів, послідовності, прецедентів.

Тема 3.2

Породжувальні патерни: Singleton, Factory Method, Abstract Factory, Builder, Prototype.

Тема 3.3

Структурні патерни: Adapter, Decorator, Facade, Composite, Proxy, Bridge, Flyweight.

Тема 3.4

Поведінкові патерни: Observer, Strategy, Command, Iterator, Template Method, State.

МОДУЛЬ 3 · ТЕМА 3.1

Принципи SOLID та UML–моделювання

Лекційний матеріал (2 год.)

Принципи SOLID: Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion. Антипатерни: God Object, Spaghetti Code, Magic Numbers. Поняття чистого коду (Clean Code). Рефакторинг: визначення, техніки, коли застосовувати. UML як мова моделювання: діаграми класів (зв'язки: асоціація, агрегація, композиція, залежність, реалізація), діаграми прецедентів та послідовності.

Ключові питання

- Зміст і практичне застосування кожного принципу SOLID
- Типові антипатерни та способи їх усунення
- UML діаграма класів: нотація та відношення між класами
- Рефакторинг без зміни функціональності
- Зв'язок між SOLID та патернами проєктування

Лабораторне заняття (2 год.)

Завдання 1. Аналіз прикладів коду, що порушують принципи SOLID. Рефакторинг для дотримання SRP та OCP.

Завдання 2. Побудова UML-діаграми класів для системи управління замовленнями в інтернет-магазині.

Завдання 3. Тестування: принципи SOLID та UML-нотація (10 питань).

Самостійна робота (7 год.)

- Опрацювання: Кузьма К. Т., Поздєєв В. О. «Основи ООП мовою C#» — МНУ, 2022 — розділ «Принципи SOLID»
- Підготовка реферату «Принципи SOLID: теорія та приклади застосування в реальних проєктах»
- Побудова UML-діаграм для власного навчального проєкту

Очікуваний результат: студент вміє аналізувати код на відповідність принципам SOLID, виконувати рефакторинг та будувати UML-діаграми класів.

МОДУЛЬ 3 · ТЕМА 3.2

Породжувальні патерни проєктування

Лекційний матеріал (2 год.)

Поняття патерну проєктування. Класифікація GoF: породжувальні, структурні, поведінкові. Породжувальні патерни: Singleton (одиночка) — забезпечення єдиного екземпляра класу; Factory Method — делегування створення підкласам; Abstract Factory — сімейства взаємопов'язаних об'єктів; Builder — покрокова побудова складних об'єктів; Prototype — клонування об'єктів.

Лабораторна (2 год.) та СРС (7 год.)

Лабораторна: Реалізація патерну Factory Method для створення різних типів UI-елементів; реалізація Singleton для класу конфігурації програми. Порівняння підходів зі звичайним створенням об'єктів.

СРС: Опрацювання: Баран С. В. «Розробка ПЗ з використанням патернів проєктування» — ДУЕТ, 2023; підготовка аналітичної записки «Породжувальні патерни: коли застосовувати і яких помилок уникати».

Очікуваний результат: студент вміє описувати та реалізовувати породжувальні патерни проєктування, обирати відповідний патерн для заданої задачі.

МОДУЛЬ 3 · ТЕМИ 3.3–3.4

Структурні та поведінкові патерни

Тема 3.3 — Структурні патерни (Лекція 11, 2 год.)

Структурні патерни GoF: Adapter — адаптація несумісних інтерфейсів; Decorator — динамічне додавання функціональності; Facade — спрощений інтерфейс до підсистеми; Composite — деревоподібні структури; Proxy — замісник для контролю доступу; Bridge — відокремлення абстракції від реалізації; Flyweight — спільне використання дрібних об'єктів.

Лабораторна (2 год.): Реалізація патерну Decorator для додавання логування та кешування до класу репозиторія даних; реалізація Facade для спрощення роботи з підсистемою.

СРС (7 год.): Опрацювання: Баран С. В. «Розробка ПЗ з використанням патернів проєктування» — ДУЕТ, 2023; підготовка порівняльної таблиці структурних патернів.

Тема 3.4 — Поведінкові патерни (Лекція 12, 2 год.)

Поведінкові патерни GoF: Observer — сповіщення про зміни стану; Strategy — взаємозамінні алгоритми; Command — інкапсуляція запитів; Iterator — послідовний обхід; Template Method — схема алгоритму в базовому класі; State — зміна поведінки залежно від стану; Chain of Responsibility — ланцюжок обробників.

Лабораторна (2 год.): Реалізація системи сповіщень з патерном Observer; реалізація різних стратегій сортування з патерном Strategy з можливістю перемикання під час виконання.

СРС (7 год.): Підготовка аналітичної записки «Поведінкові патерни: порівняльний аналіз Observer, Strategy та Command»; розробка діаграм класів для обраних патернів.

МОДУЛЬ 4

Розробка програмних систем: тестування, документування та проєкт

Тема 4.1

Модульне тестування ООП-систем. Фреймворки тестування (Google Test, JUnit). Test-Driven Development (TDD).

Тема 4.2

Документування коду. Doxygen/Javadoc. Системи контролю версій Git. Робота в команді (GitHub/GitLab).

Тема 4.3

Основи Java та Python у контексті ООП: порівняння з C++. Garbage collection. Рефлексія. Анотації.

Тема 4.4

Комплексний курсовий проєкт: розробка об'єктно-орієнтованої системи з використанням патернів, тестування та документування.

МОДУЛЬ 4 · ТЕМА 4.1

Модульне тестування та TDD

Лекційний матеріал (2 год.)

Поняття тестування ПЗ. Рівні тестування: модульне, інтеграційне, системне. Фреймворки модульного тестування: Google Test (C++), JUnit (Java), pytest (Python). Структура тесту: Arrange-Act-Assert. Test-Driven Development (TDD): цикл Red-Green-Refactor. Mock-об'єкти та заглушки. Покриття коду тестами. Безперервна інтеграція (CI).

Ключові питання

- Рівні та типи тестування програмного забезпечення
- Структура та написання юніт-тестів з Google Test
- Принцип TDD та його переваги для ООП-розробки
- Mock-об'єкти: призначення та реалізація
- Метрика покриття коду тестами

Лабораторне заняття (2 год.)

Завдання 1. Написання модульних тестів для класу «Стек» та «Матриця» з використанням Google Test. Перевірка граничних випадків та виключень.

Завдання 2. Практика TDD: реалізація класу «Банківський рахунок» за підходом Red-Green-Refactor.

Завдання 3. Аналіз покриття коду тестами та підвищення покриття до 80%+.

Самостійна робота (7 год.)

- Опрацювання: Порєв В. М. «ООП: комп'ютерний практикум» — КПІ, 2022 — розділ «Тестування»
- Підготовка звіту «TDD у розробці ООП-систем: переваги та виклики»
- Написання тестів для власних класів курсового проєкту

Очікуваний результат: студент вміє писати модульні тести з використанням Google Test, застосовувати підхід TDD та аналізувати покриття коду.

МОДУЛЬ 4 · ТЕМА 4.2

Документування коду та Git

Лекційний матеріал (2 год.)

Документування програмного коду: коментарі, самодокументований код. Інструменти генерації документації: Doxygen (C++), Javadoc (Java). Анотації та теги документації. Система контролю версій Git: init, add, commit, push, pull, merge, rebase. Гілки та їх стратегії (Git Flow, GitHub Flow). Платформи для командної розробки: GitHub, GitLab. Code review: принципи та інструменти.

Лабораторна (2 год.) та СРС (7 год.)

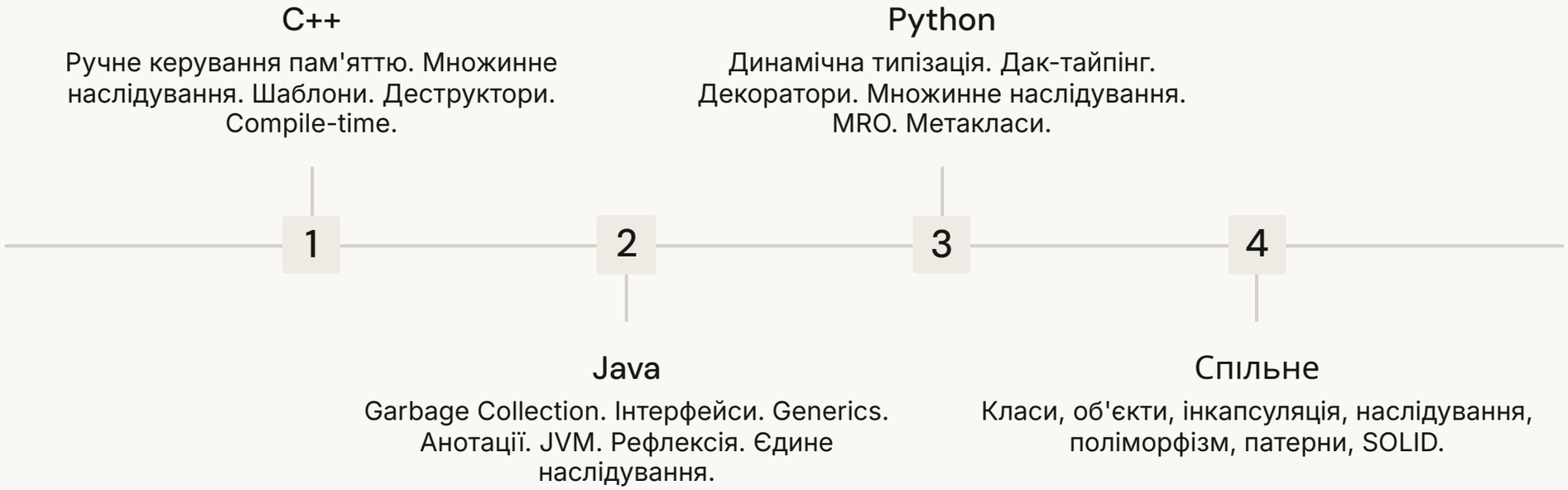
Лабораторна: Оформлення документації до власних класів з використанням Doxygen; генерація HTML-документації. Робота з Git: створення репозиторію, гілок, pull request та code review в команді.

СРС: Опрацювання: Рачинська А. Л. та ін. «ООП» — ОНУ, 2024; підготовка аналітичної записки «Git Flow vs. GitHub Flow: порівняльний аналіз стратегій розгалуження»; документування власного курсового проєкту.

Очікуваний результат: студент вміє документувати програмний код за допомогою Doxygen, працювати з Git та платформами GitHub/GitLab у командній розробці.

МОДУЛЬ 4 · ТЕМА 4.3

ООП у Java та Python: порівняльний аналіз



Лекційний матеріал (2 год.)

ООП у Java: відсутність множинного наслідування класів, інтерфейси та default-методи, Generics, Garbage Collection, рефлексія. ООП у Python: динамічна типізація та дак-тайпінг, декоратори, MRO (Method Resolution Order), метакласи, dataclasses. Порівняльний аналіз реалізацій ООП у C++, Java та Python. Вибір мови для конкретних задач.

Лабораторна (2 год.) та СРС (7 год.)

Лабораторна: Реалізація однакової задачі (система управління бібліотекою) засобами двох мов (C++ та Java або Python). Порівняння синтаксису, особливостей та продуктивності.

СРС: Опрацювання: Кузьма К. Т., Поздєєв В. О. «Основи ООП мовою C#» — МНУ, 2022; підготовка порівняльного звіту «ООП у C++, Java та Python: синтаксис, концепції, інструменти».

МОДУЛЬ 4 · ТЕМА 4.4

Комплексний курсовий проєкт

Лекційний матеріал (2 год.)

Методологія розробки програмних систем: від вимог до реалізації. Аналіз вимог та проєктування системи. Вибір та обґрунтування патернів проєктування. Ітеративна розробка. Рецензування коду (code review). Презентація та захист програмного проєкту. Оформлення технічної документації.

Ключові питання

- Методологія розробки ООП-проєкту від ідеї до реалізації
- Вибір та обґрунтування архітектурних рішень
- Інтеграція патернів проєктування у реальну систему
- Методи тестування готового програмного продукту
- Презентація та захист технічних рішень

Лабораторне заняття (2 год.)

Завдання 1. Презентація та захист комплексного курсового проєкту: демонстрація функціональності, опис архітектурних рішень, обґрунтування вибору патернів.

Завдання 2. Code review: взаємне рецензування проєктів у групах з 2–3 осіб. Аргументована оцінка відповідності принципам SOLID та ООП.

Завдання 3. Фінальне тестування: комплексний тест за всіма модулями курсу.

Самостійна робота (7 год.)

- Завершення реалізації курсового проєкту: усунення зауважень після code review
- Підготовка технічної документації з Doxygen та README.md
- Підготовка презентації та демонстраційного відео проєкту

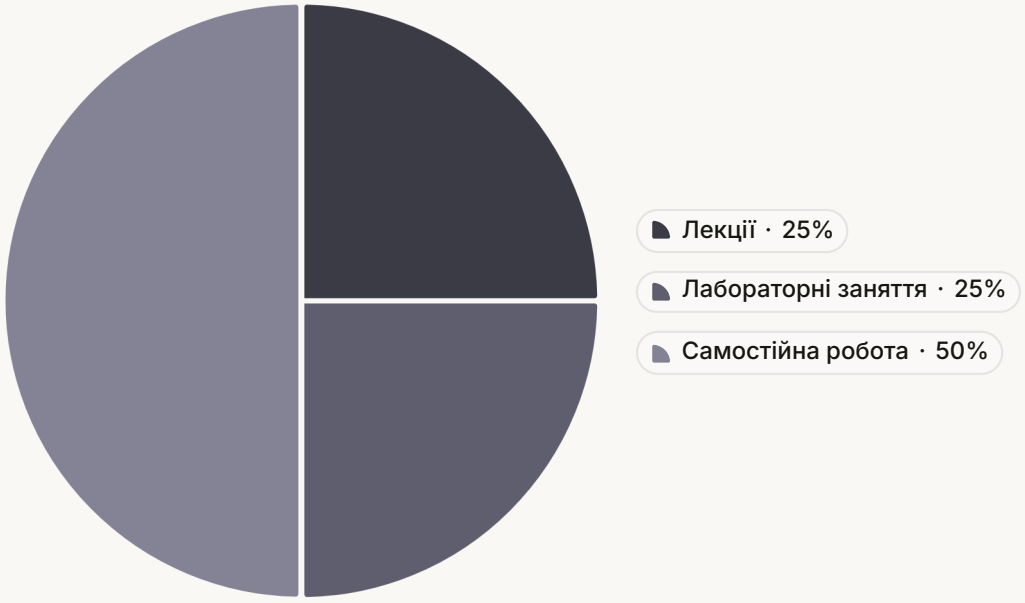
Очікуваний результат: студент вміє розробляти повноцінну об'єктно-орієнтовану програмну систему із застосуванням патернів проєктування, тестування та документування.

Розподіл навчального часу — 120 годин

Модуль	Тема	Лекції (год.)	Лабораторні (год.)	СРС (год.)
Модуль 1	Тема 1.1 — Класи та об'єкти	2	2	7
	Тема 1.2 — Інкапсуляція та конструктори	2	2	7
	Тема 1.3 — Наслідування	2	2	7
	Тема 1.4 — Поліморфізм та абстрактні класи	2	2	7
Модуль 2	Тема 2.1 — Шаблони класів і функцій	2	2	7
	Тема 2.2 — STL: контейнери, ітератори, алгоритми	2	2	7
	Тема 2.3 — Обробка виключень	2	2	7
	Тема 2.4 — Сучасний C++ (C++11/14/17/20)	2	2	7
Модуль 3	Тема 3.1 — Принципи SOLID та UML	2	2	7
	Тема 3.2 — Породжувальні патерни	2	2	7
	Тема 3.3 — Структурні патерни	2	2	7
	Тема 3.4 — Поведінкові патерни	2	2	7
Модуль 4	Тема 4.1 — Модульне тестування та TDD	2	2	7
	Тема 4.2 — Документування та Git	2	2	7
	Тема 4.3 — ООП у Java та Python	2	2	7
	Тема 4.4 — Комплексний курсовий проєкт	2	2	4
Разом	**16 тем**	**30**	**30**	**60**

Загальний обсяг: 120 годин = 4 кредити ЄКТС. Розподіл є орієнтовним і може коригуватися відповідно до специфіки навчального процесу.

Структура навчального часу



Коментар до розподілу

Загальний обсяг — **120 годин (4 кредити ЄКТС)** — розподілено між трьома формами роботи для забезпечення балансу між теорією та практикою.

Лекції — 30 год. (25%)

Системне подання теоретичного матеріалу з ООП: концепції, патерни, принципи SOLID, UML-моделювання з використанням наочних схем, діаграм класів та прикладів коду.

Лабораторні — 30 год. (25%)

Відпрацювання практичних навичок програмування: розробка класів, реалізація патернів, написання тестів, робота з Git. Рівна частка з лекціями підкреслює практичну спрямованість курсу.

Самостійна робота — 60 год. (50%)

Поглиблене вивчення, виконання завдань, розробка курсового проєкту, робота з науковою та технічною літературою. Самостійна робота становить **50% обсягу дисципліни**.

Самостійна робота студентів — Детальний опис

Опрацювання літератури

Поглиблене вивчення теоретичного матеріалу за підручниками та посібниками з ООП (Бобков, Гришанович, Рачинська, Порєв, Кузьма та ін.), технічною документацією мов програмування, науковими статтями у фахових виданнях. Студент конспектує ключові положення та аналізує приклади коду.

Виконання індивідуальних завдань

Виконання індивідуальних завдань з програмування: реалізація класів, ієрархій, патернів проєктування; написання звітів з лабораторних робіт з описом архітектурних рішень; підготовка аналітичних записок та рефератів.

Робота з інформаційними ресурсами

Опрацювання офіційних ресурсів: cppreference.com, docs.oracle.com (Java), docs.python.org, репозиторіїв з прикладами патернів проєктування на GitHub, електронного репозиторію НБУВ (nbuv.gov.ua) та баз наукових статей.

Підготовка до занять та контролю

Підготовка до захисту лабораторних робіт; підготовка до тестувань (ключові терміни, концепції, патерни); розробка та вдосконалення курсового проєкту; підготовка до модульних контрольних робіт.

Форми контролю знань

1	2	3
Поточний контроль	**Проміжний контроль**	**Підсумковий контроль**
Захист лабораторних робіт: усне опитування з демонстрацією та поясненням реалізованого коду. Тестування за ключовими термінами теми (10–15 питань) після кожної теми. Оцінюються правильність реалізації, розуміння концепцій ООП та якість оформлення коду. Формує накопичувальну оцінку протягом семестру.	Модульна контрольна робота після завершення кожного змістовного модуля: теоретичні питання, аналіз та написання коду, розробка UML-діаграми. Оцінюються глибина розуміння матеріалу, правильність реалізації ООП-концепцій, якість архітектурних рішень та обґрунтованість висновків.	Іспит відповідно до навчального плану + захист курсового проєкту. Форма: тестування та розв'язання практичних задач з програмування. Підсумкова оцінка враховує результати поточного, проміжного та підсумкового контролю: **100-бальна система** з переведенням у національну шкалу та ЄКТС (A, B, C, D, E, FX, F).

Методи викладання

Викладання дисципліни «Об'єктно-орієнтоване програмування» поєднує класичні академічні та інтерактивні методи, що забезпечує формування не лише міцної теоретичної бази, а й практичних навичок розробки програмного забезпечення, необхідних для майбутньої професійної діяльності у сфері комп'ютерної інженерії.

Лекції

Системне подання теоретичного матеріалу з демонстрацією прикладів коду, UML-діаграм та порівняльних таблиць; студенти ведуть конспекти та задають запитання.

Лабораторні роботи

Практична реалізація ООП-концепцій: написання класів, тестів, застосування патернів; кожне заняття завершується захистом із демонстрацією та поясненням коду.

Кейс-стаді

Аналіз реальних програмних архітектур відомих систем; виявлення патернів, оцінка архітектурних рішень та їх обґрунтування на основі принципів SOLID.

Code Review

Взаємне рецензування коду: розвиток критичного мислення, навичок аргументації, культури програмування та поваги до технічних рішень колег.

Самостійне дослідження

Підготовка аналітичних записок, звітів та курсового проєкту; розвиток навичок самостійного пошуку технічних рішень та їх документування.

Командний проєкт

Розробка фрагментів курсового проєкту в малих групах: розподіл ролей, планування, інтеграція компонентів, використання Git для командної роботи.

Рекомендована література — Підручники та посібники (2022–2024 рр.)

1. Бобков В. Б., Грудзинський Ю. Є., Крилов К. В.
Програмування-2. Об'єктно-орієнтоване програмування : навч. посіб. — Київ : КПІ ім. Ігоря Сікорського, 2023. — 653 с. (електрон. вид.).

2. Рачинська А. Л., Недєва О. А., Палій К. С.
Об'єктно-орієнтоване програмування : електрон. метод. посіб. до лекц. та лаб. занять. — Одеса : ОНУ ім. І. І. Мечникова, 2024. — 338 с.

3. Гришанович Т. О., Глинчук Л. Я.
Основи об'єктно-орієнтованого програмування : навч. посіб. — Луцьк : ВНУ ім. Лесі Українки, 2022. — 120 с.

4. Кузьма К. Т., Поздєєв В. О.
Основи об'єктно-орієнтованого програмування мовою C# : навч. посіб. — Миколаїв : Миколаївський національний університет, 2022. — 180 с.

5. Порєв В. М.
Об'єктно-орієнтоване програмування: комп'ютерний практикум : навч. посіб. — Київ : КПІ ім. Ігоря Сікорського, 2022. — 105 с. (електрон. вид.).

6. Баран С. В.
Розробка програмного забезпечення з використанням патернів проєктування : навч. посіб. — Кривий Ріг : Державний університет економіки і технологій, 2023. — 203 с.

7. Васильєв О. М.
Програмування в Python. Теорія і практика : навч. посіб. — Київ : Видавництво Ліра-К, 2023. — 320 с.

8. Зеленський О. С., Лисенко В. С.
Об'єктно-орієнтоване програмування на C++ : навч. посіб. — Запоріжжя : ЗНУ, 2022. — 215 с.

Додаткова література та наукові статті

9. Алхімова С. М.

Об'єктно-орієнтоване програмування : підручник. У 2-х ч. Ч. 2. Об'єктно-орієнтований підхід до розробки програмного забезпечення. — Київ : КПІ ім. Ігоря Сікорського, вид-во «Політехніка», 2022. *Рекомендується для модулів 1–2.*

10. Жуковський С. С., Вакалюк Т. А.

Об'єктно-орієнтоване програмування мовою C++ : курс лекцій. — Житомир : ЖДУ ім. Івана Франка, 2022. — 180 с. *Рекомендується для тем 1.1–2.4.*

11. Доценко С. І.

Організація та системи керування базами даних : навч. посіб. — Харків : УкрДУЗТ, 2023. — 117 с. *Рекомендується для тем 4.2–4.4 (інтеграція ООП з базами даних).*

12. Добровська Л. М., Аверьянова О. В.

Проєктування інформаційних систем : комп'ютерний практикум. — Київ : КПІ ім. Ігоря Сікорського, 2022. — 202 с. *Рекомендується для модулів 3–4 (UML та архітектура).*

Нормативно-правова база та офіційні ресурси

- Конституція України

  Прийнята 28 червня 1996 р. Основний Закон держави. **zakon.rada.gov.ua**

- Закон «Про освіту» від 05.09.2017 № 2145-VIII

  Визначає засади освітньої діяльності в Україні. **zakon.rada.gov.ua**

- Стандарт вищої освіти зі спеціальності 123 «Комп'ютерна інженерія»

  Затверджений МОН України. Визначає компетентності та результати навчання бакалавра. **mon.gov.ua**

- ISO/IEC 14882 — Стандарт мови C++

  Міжнародний стандарт мови програмування C++. Актуальна версія: C++20 (ISO/IEC 14882:2020). **isocpp.org**

- Закон «Про вищу освіту» від 01.07.2014 № 1556-VII

  Регулює відносини у сфері вищої освіти, визначає рівні та ступені вищої освіти. **zakon.rada.gov.ua**

Офіційні ресурси та фахові видання

cppreference.com

cppreference.com — повна технічна документація мови C++ та стандартної бібліотеки STL.

Документація Java SE

docs.oracle.com — офіційна документація Java Standard Edition з прикладами та специфікаціями API.

Документація Python

docs.python.org — офіційна документація Python, туторіали, бібліотека стандартних модулів.

Національна бібліотека України ім. В. І. Вернадського

nbuv.gov.ua — електронні ресурси, наукові статті та монографії з програмування та комп'ютерних наук.

Repozitorii КПІ (ela.kpi.ua)

ela.kpi.ua — електронний архів КПІ ім. Ігоря Сікорського: навчальні посібники, методичні матеріали.

GitHub Education

education.github.com — безкоштовні інструменти для студентів та викладачів, репозиторії з навчальними матеріалами.

Міністерство освіти і науки України

mon.gov.ua — освітні стандарти, навчальні програми, методичні матеріали для спеціальності 123.

Журнал «Комп'ютерно-інтегровані технології»

cit.mntu.edu.ua — фахове видання з питань програмування, інженерії ПЗ та комп'ютерних наук.

Зв'язок дисципліни зі спеціальністю 123 «Комп'ютерна інженерія»

Чому ООП для комп'ютерних інженерів?

Дисципліна «Об'єктно-орієнтоване програмування» є базовою для підготовки бакалаврів зі спеціальності 123 «Комп'ютерна інженерія». Сучасні вбудовані системи, драйвери, системне програмне забезпечення та прикладні застосунки розробляються із застосуванням ООП-підходу.

Знання ООП є фундаментом для вивчення дисциплін: «Алгоритми та структури даних», «Операційні системи», «Архітектура програмного забезпечення», «Комп'ютерна графіка», «Вбудовані системи» та виробничої практики.

Практичне значення для майбутньої кар'єри

- Розробка програмного забезпечення для вбудованих систем і мікроконтролерів мовою C++
- Архітектура та проєктування складних програмних систем із застосуванням патернів GoF
- Командна розробка ПЗ з використанням Git, code review та CI/CD
- Тестування та забезпечення якості програмного забезпечення (QA/QE)
- Розробка кросплатформних застосунків на Java або Python
- Підготовка до технічних співбесід: знання ООП та патернів є обов'язковою вимогою IT-компаній

Ключові компетентності курсу — Підсумок

Курс охоплює всі ключові аспекти об'єктно-орієнтованого програмування — від базових концепцій до патернів проєктування та командної розробки — і формує у студентів цілісне розуміння сучасних підходів до розробки якісного програмного забезпечення.

Бажаємо успіхів у вивченні ООП!

120

Годин

Загальний обсяг дисципліни

4

Кредити ЄКТС

Відповідно до стандарту

16

Тем

У 4 змістовних модулях

8+

Джерел

Рекомендованої літератури 2022–2024 рр.

> «Будь-яка досить розвинена технологія нерозрізнима від магії. Але на відміну від магії, добрий код можна прочитати, зрозуміти і вдосконалити.»

Спеціальність 123 · Комп'ютерна інженерія · Бакалавр · 4 кредити ЄКТС