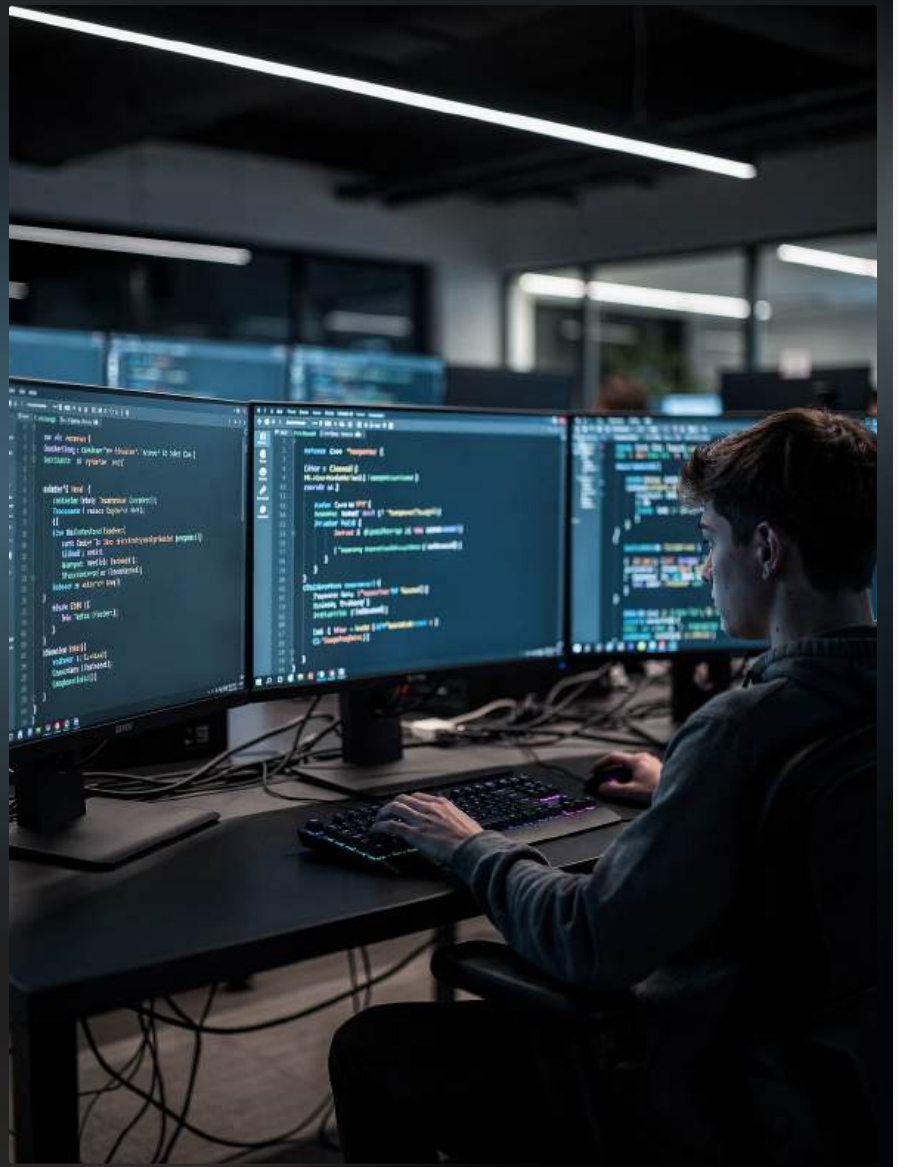


Силабус дисципліни «Об'єктно-орієнтоване програмування»

СПЕЦІАЛЬНІСТЬ 123 · КОМП'ЮТЕРНА ІНЖЕНЕРІЯ · БАКАЛАВР

120

Годин загальний обсяг дисципліни

4

Кредити ЄКТС відповідно до стандарту

4

Змістовних модулі 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 курсу

Мета дисципліни

Дисципліна «Об'єктно-орієнтоване програмування» формує у студентів спеціальності «Комп'ютерна інженерія» фундаментальні знання та практичні навички розробки програмного забезпечення з використанням об'єктно-орієнтованої парадигми. Курс спрямований на розвиток системного мислення, вміння проектувати ієрархії класів, застосовувати принципи інкапсуляції, успадкування та поліморфізму, а також шаблони проектування при розробці реальних програмних систем. Особлива увага приділяється формуванню практичних навичок написання якісного, ефективного та підтримуваного коду на мовах C++ та/або Java.

Форми навчання

- Лекції — 30 годин
- Лабораторні заняття — 30 годин
- Самостійна робота — 60 годин

Компетентності та результати навчання

Загальні компетентності

Здатність до аналізу, синтезу та критичного оцінювання технічної інформації: виявлення оптимальних архітектурних рішень, порівняння альтернативних підходів до проектування програмних систем, формування обґрунтованих висновків щодо вибору засобів реалізації. Уміння самостійно навчатися, планувати роботу та організовувати виконання завдань у визначені строки, зокрема під час виконання лабораторних робіт, індивідуальних проектів і підготовки до контрольних заходів.

Здатність до усної та письмової технічної комунікації: чітко описувати архітектурні рішення, обґрунтовувати вибір патернів проектування, оформлювати технічну документацію. Відповідальність, ініціативність та готовність до командної роботи над програмними проектами.

- Аналізувати технічні вимоги та обирати відповідні рішення
- Виділяти головне та узагальнювати результати проектування
- Планувати власну діяльність і дотримуватися дедлайнів
- Коректно взаємодіяти з викладачами та одногрупниками
- Брати участь у командній розробці програмного забезпечення

Фахові компетентності

Розуміння сутності та принципів об'єктно-орієнтованої парадигми програмування: інкапсуляція, успадкування, поліморфізм, абстракція. Знання ключових механізмів мов ООП — класів, об'єктів, конструкторів, деструкторів, перевантаження операторів, шаблонів, інтерфейсів та обробки виключень.

Здатність застосовувати принципи SOLID та шаблони проектування (GoF) при розробці програмних систем різної складності. Уміння будувати UML-діаграми, проектувати ієрархії класів та розробляти програмні системи з використанням сучасних середовищ розробки.

- Проектувати класи та ієрархії успадкування
- Застосовувати принципи SOLID у проектуванні
- Реалізовувати поліморфізм та шаблони проектування
- Будувати UML-діаграми програмних систем
- Розробляти та налагоджувати програмне забезпечення

Результати навчання

Після успішного завершення курсу студент повинен демонструвати такі результати навчання:

1

Знання та розуміння

Пояснювати сутність ключових концепцій ООП: клас, об'єкт, інкапсуляція, успадкування, поліморфізм, абстракція, інтерфейс, патерн проектування, виключення, шаблони. Характеризувати відмінності між структурним та об'єктно-орієнтованим підходами до програмування.

2

Аналіз та оцінювання

Аналізувати вимоги до програмної системи та обирати відповідні патерни проектування. Оцінювати якість архітектурних рішень, порівнювати різні підходи до реалізації ієрархій класів та обґрунтовувати вибір конкретного рішення.

3

Застосування знань

Розробляти програмне забезпечення із застосуванням принципів ООП та SOLID. Реалізовувати класичні патерни проектування, будувати UML-діаграми, налагоджувати програми та оформлювати технічну документацію.

4

Комунікація та рефлексія

Аргументовано захищати власні проектні рішення та описувати архітектуру розробленого програмного продукту. Формувати навички самоаналізу якості коду, рефакторингу та відповідального ставлення до програмної інженерії.

Модуль 1. Основи ООП та класи

ЗМІСТОВНИЙ МОДУЛЬ 1

Тема 1.1

Вступ до об'єктно-орієнтованого програмування. Парадигми програмування: структурна, процедурна, об'єктно-орієнтована. Основні концепції ООП. Огляд мов ООП (C++, Java, Python). Переваги ООП над процедурним підходом. Поняття класу та об'єкта.

Тема 1.2

Класи та об'єкти. Оголошення та визначення класів. Поля, методи, конструктори та деструктори. Специфікатори доступу: public, private, protected. Ключове слово this. Статичні члени класу. Дружні функції та класи.

Тема 1.3

Інкапсуляція та абстракція. Принцип інкапсуляції та його реалізація. Методи доступу (геттери та сеттери). Абстракція як інструмент спрощення складних систем. Абстрактні класи та інтерфейси. Проектування класів за принципом мінімальної відкритості.

Тема 1.4

Перевантаження операторів та методів. Концепція перевантаження. Перевантаження арифметичних, порівняльних та потокових операторів. Перевантаження методів (поліморфізм часу компіляції). Практика реалізації власних класів з перевантаженими операторами.

Модуль 2. Успадкування та поліморфізм

ЗМІСТОВНИЙ МОДУЛЬ 2

Тема 2.1

Успадкування. Принцип успадкування. Одиночне та множинне успадкування. Виклик конструкторів базового класу. Перевизначення методів. Захищені члени класу. Проблема ромбоподібного успадкування та віртуальне успадкування.

Тема 2.2

Поліморфізм. Концепція поліморфізму. Статичний та динамічний поліморфізм. Віртуальні функції та таблиця віртуальних методів. Чисто віртуальні функції та абстрактні класи. Пізнє зв'язування. Принцип підстановки Лісков (LSP).

Тема 2.3

Обробка виключень та управління пам'яттю. Механізм виключень (try, catch, throw). Ієрархія класів виключень. Стандартні виключення. Управління динамічною пам'яттю: new та delete. Розумні вказівники (smart pointers): unique_ptr, shared_ptr, weak_ptr. RAII-ідіома.

Модуль 3. Шаблони проектування та принципи SOLID

ЗМІСТОВНИЙ МОДУЛЬ 3

Тема 3.1

Принципи SOLID. Single Responsibility Principle — принцип єдиної відповідальності. Open/Closed Principle — відкритий/закритий принцип. Liskov Substitution Principle. Interface Segregation Principle. Dependency Inversion Principle. Застосування SOLID у реальних проектах.

Тема 3.2

Породжуючі та структурні патерни проектування. Огляд класифікації патернів GoF. Породжуючі патерни: Singleton, Factory Method, Abstract Factory, Builder, Prototype. Структурні патерни: Adapter, Bridge, Composite, Decorator, Facade, Proxy.

Тема 3.3

Поведінкові патерни та UML-моделювання. Поведінкові патерни: Observer, Strategy, Command, Iterator, Template Method, State. Введення у UML: діаграми класів, послідовності, прецедентів. Інструменти UML-моделювання. Зв'язок між патернами та UML-нотацією.

Модуль 4. Шаблони, STL та сучасне ООП

ЗМІСТОВНИЙ МОДУЛЬ 4

Тема 4.1

Узагальнене програмування та шаблони. Шаблони функцій та класів у C++. Параметри шаблонів. Спеціалізація шаблонів. Метапрограмування. Концепція узагальненого програмування. Generics у Java. Переваги та обмеження шаблонів.

Тема 4.2

Стандартна бібліотека шаблонів (STL). Контейнери: vector, list, deque, map, set, unordered_map. Ітератори та алгоритми STL. Функтори та лямбда-вирази. Застосування STL у проектах. Порівняння контейнерів за продуктивністю.

Тема 4.3

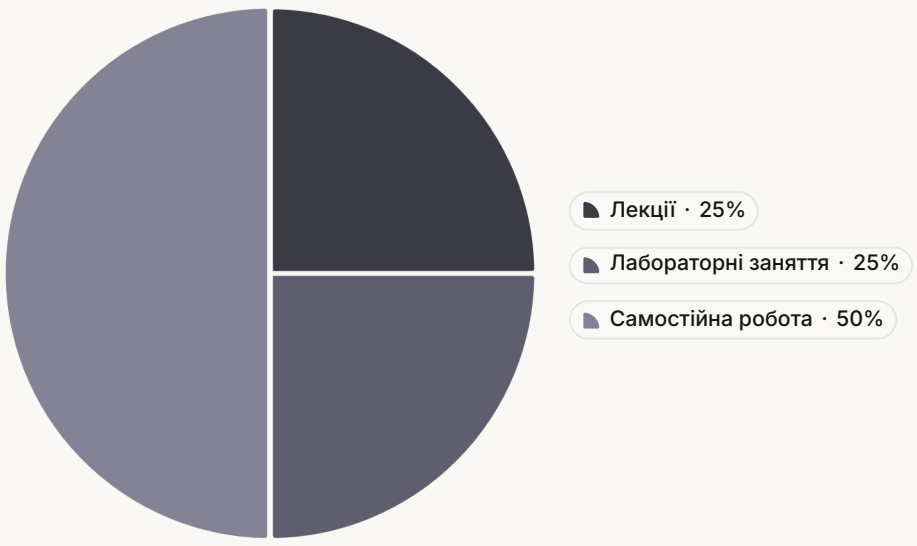
Сучасні стандарти C++ (C++17/20) та рефакторинг. Нові можливості C++17/20: structured bindings, if constexpr, concepts, ranges, coroutines. Поняття рефакторингу. Виявлення та усунення code smells. Інструменти статичного аналізу коду. Принципи написання чистого коду (Clean Code).

Тема 4.4

Тестування програмного забезпечення та командна розробка. Модульне тестування (Unit Testing). Тест-орієнтована розробка (TDD). Інструменти тестування: Google Test, Catch2. Системи контролю версій (Git). Основи командної розробки. Огляд CI/CD pipelines.

Розподіл навчального часу — 120 годин

Модуль	Тема	Лекції (год.)	Лабораторні (год.)	Самостійна робота (год.)
Модуль 1	Тема 1.1 — Вступ до ООП	2	2	4
	Тема 1.2 — Класи та об'єкти	2	2	4
	Тема 1.3 — Інкапсуляція та абстракція	2	2	4
	Тема 1.4 — Перевантаження операторів та методів	2	2	4
Модуль 2	Тема 2.1 — Успадкування	2	2	4
	Тема 2.2 — Поліморфізм	4	4	8
	Тема 2.3 — Обробка виключень та управління пам'яттю	2	2	4
Модуль 3	Тема 3.1 — Принципи SOLID	4	4	8
	Тема 3.2 — Породжуючі та структурні патерни	4	4	8
	Тема 3.3 — Поведінкові патерни та UML	2	2	4
Модуль 4	Тема 4.1 — Шаблони та узагальнене програмування	2	2	4
	Тема 4.2 — Стандартна бібліотека шаблонів (STL)	2	2	4
	Тема 4.3 — Сучасні стандарти C++17/20 та рефакторинг	2	2	4
	Тема 4.4 — Тестування та командна розробка	2	2	4
Разом		**30**	**30**	**60**



Структура навчального часу

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

- **Лекції (30 год. — 25%):** подача ключового теоретичного матеріалу
- **Лабораторні (30 год. — 25%):** практичне програмування, налагодження та захист робіт
- **Самостійна робота (60 год. — 50%):** поглиблене вивчення, виконання індивідуальних завдань

Самостійна робота становить **50% обсягу дисципліни** — студенти навчаються самостійно проектувати та реалізовувати програмні системи.

Самостійна робота студентів та методи викладання

Форми самостійної роботи

Опрацювання літератури

Поглиблене вивчення теоретичного матеріалу за підручниками та навчальними посібниками з ООП, специфікаціями мов програмування, технічною документацією бібліотек. Конспектування ключових положень, порівняння підходів різних авторів до реалізації патернів та принципів ООП.

Індивідуальні завдання

Виконання індивідуальних проектів з проектування та реалізації програмних систем із застосуванням патернів GoF, принципів SOLID, підготовка UML-діаграм, технічної документації та звіту з обґрунтуванням прийнятих архітектурних рішень.

Робота з інформаційними ресурсами

Опрацювання офіційної документації: cppreference.com, docs.oracle.com (Java). Аналіз відкритих репозиторіїв GitHub, вивчення коду реальних проектів. Використання онлайн-платформ: Compiler Explorer, LeetCode, Codeforces для практики програмування.

Підготовка до занять

Підготовка до лабораторних занять шляхом опрацювання теоретичного матеріалу та попереднього написання коду; підготовка до тестувань через вивчення ключових термінів і визначень ООП; систематизація знань та рефакторинг коду перед контрольними заходами.

Методи викладання

→ Лекції

Системне подання теоретичного матеріалу з використанням наочних схем, UML-діаграм, демонстрацій коду в IDE; студенти ведуть конспекти та задають уточнювальні запитання.

→ Кейс-стаді

Аналіз реальних програмних систем та архітектурних рішень; студенти виявляють застосовані патерни, пропонують альтернативи та обґрунтовують вибір підходу на основі вимог.

→ Code Review

Взаємний перегляд коду студентів: виявлення порушень принципів ООП та SOLID, пропозиції щодо рефакторингу, розвиток критичного технічного мислення та культури написання чистого коду.

→ Робота з репозиторіями

Аналіз коду відкритих проектів на GitHub; формування навичок читання чужого коду, використання систем контролю версій та написання коду згідно з прийнятими стандартами.

→ Самостійне програмування

Виконання індивідуальних та групових проектів із проектування та реалізації програмних систем; розвиток навичок самостійного архітектурного мислення та технічного документування.

Форми контролю знань

1

Поточний контроль

Завдання: практичні та тестові вправи за темами змістовного модуля. Оцінюються регулярність виконання та захисту лабораторних робіт, якість реалізованого коду та рівень засвоєння матеріалу. Поточний контроль формує накопичувальну оцінку протягом семестру. Включає усний захист лабораторних робіт, тестування та виконання аудиторних задач на програмування.

2

Проміжний контроль

Самостійна робота — комплексне проектне завдання або контрольна робота, яка охоплює теми змістовних модулів (теоретичні питання, UML-діаграми, реалізація класів і патернів). Оцінюються повнота рішення, коректність застосування принципів ООП та обґрунтованість архітектурних рішень. Проводиться після кожного змістовного модуля.

3

Підсумковий контроль

Іспит відповідно до навчального плану. Форма проведення — тестування та письмові відповіді на питання з можливим кодинг-завданням. Підсумкова оцінка враховує результати поточного, проміжного та підсумкового контролю відповідно до встановленої шкали оцінювання: 100-бальна система з переведенням у національну шкалу та ЄКТС (A, B, C, D, E, FX, F).

Усі методи контролю спрямовані на формування компетентностей, передбачених освітньо-професійною програмою спеціальності 123 «Комп'ютерна інженерія».

Рекомендована література та ресурси

Навчальна та наукова література (2022–2024)

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.
2. Гришанович Т. О., Глинчук Л. Я. Основи об'єктно-орієнтованого програмування : навч. посібник. — Луцьк : ВНУ імені Лесі Українки, 2022. — 120 с.
3. Порєв В. М. Об'єктно-орієнтоване програмування: комп'ютерний практикум : навч. посіб. — Київ : КПІ ім. Ігоря Сікорського, 2022. — 105 с.
4. Рачинська А. Л., Недєва О. А., Палій К. С. Об'єктно-орієнтоване програмування : електрон. метод. посіб. — Одеса : ОНУ ім. І. І. Мечникова, 2024. — 338 с.
5. Кузьма К. Т., Поздєєв В. О. Основи об'єктно-орієнтованого програмування мовою C# : навч. посібник. — Миколаїв : Миколаївський національний університет, 2022.
6. Баран С. В. Розробка програмного забезпечення з використанням патернів проектування : навч. посібник. — Кривий Ріг : Державний університет економіки і технологій, 2023. — 203 с.
7. Зеленський О. С., Лисенко В. С. Об'єктно-орієнтоване програмування на C++ : навч. посібник. — Запоріжжя : ЗНУ, 2022.
8. Васильєв О. М. Програмування в Python. Теорія і практика : навч. посібник. — Київ : Видавництво Ліра-К, 2023.
9. Жуковський С. С., Вакалюк Т. А. Об'єктно-орієнтоване програмування мовою C++ : навч. посіб. для студентів ЗВО. — Житомир, 2022.
10. Добровська Л. М., Аверьянова О. В. Проектування інформаційних систем : комп'ютерний практикум. — Київ : КПІ ім. Ігоря Сікорського, 2023. — 202 с.

Офіційні інтернет-ресурси

1. Довідник C++ (cppreference.com). — URL: **https://uk.cppreference.com/**
2. Документація Java SE (Oracle). — URL: **https://docs.oracle.com/en/java/**
3. Відкрита платформа GitHub (репозиторії з прикладами ООП). — URL: **https://github.com/**
4. Compiler Explorer (godbolt.org) — онлайн-компілятор C++/Java. — URL: **https://godbolt.org/**
5. Офіційний сайт стандарту ISO C++. — URL: **https://isocpp.org/**
6. Національна бібліотека України ім. В. І. Вернадського. — URL: **https://nbuv.gov.ua/**
7. Електронний архів КПІ ім. Ігоря Сікорського (ela.kpi.ua). — URL: **https://ela.kpi.ua/**
8. Платформа практики програмування LeetCode. — URL: **https://leetcode.com/**

Рекомендована література охоплює видання **2022–2024 років** українських авторів відповідно до вимог актуальності навчально-методичного забезпечення.