ЗАКЛАД ВИЩОЇ ОСВІТИ
«УНІВЕРСИТЕТ ТРАНСФОРМАЦІЇ МАЙБУТНЬОГО»

СИЛАБУС НАВЧАЛЬНОЇ ДИСЦИПЛІНИ
«Об'єктно-орієнтоване програмування»

КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ

https://uft.in.ua/
Телефон: +38 (073) 047-26-26
E-mail: info@uft.in.ua

Викладач – Литвин Олександр Олександрович, кандидат технічних наук

Галузь знань	G «Інженерія, виробництво та будівництво»
Шифр та назва спеціальності	G7 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»
Назва освітньо-професійної програми	«Автоматизація, комп'ютерно-інтегровані технології та робототехніка»
Рівень вищої освіти	Перший (бакалаврський)
Статус навчальної дисципліни (обов'язкова/вибіркова)	Вибіркова
Мова викладання	Українська

Форма навчання	Семестр викладання	Обсяг навчальної дисципліни	Лекції (годин)	Практичні заняття (годин)	Самостійна робота (годин)	Підсумковий контроль
Денна	5-8 семестр	120 год, 4 кредитів ЄКТС	32	32	56	Диференційований залік
Заочна	5-8 семестр		10	10	100	

ЗАГАЛЬНА ІНФОРМАЦІЯ ПРО НАВЧАЛЬНУ ДИСЦИПЛІНУ	
Анотація дисципліни	Навчальна дисципліна спрямована на вивчення принципів, методів і засобів об'єктно-орієнтованого програмування. У межах курсу розглядаються базові поняття ООП: класи, об'єкти, інкапсуляція, наслідування, поліморфізм, абстракція, а також основи проєктування програмних систем.
Мета і завдання дисципліни	Метою дисципліни є формування у здобувачів знань і практичних навичок розроблення програм із використанням об'єктно-орієнтованого підходу. Основними завданнями є: засвоєння основних принципів ООП; набуття вмінь проєктувати класи та об'єкти; опанування методів створення, тестування та налагодження програм; розвиток навичок побудови структурованого, зрозумілого та повторно використовуваного коду.
Компетентності, формуванню яких сприяє дисципліна	Здатність застосовувати сучасні методи програмування; Здатність розробляти алгоритми та програмні модулі; Здатність аналізувати, проєктувати та реалізовувати програмні системи; Здатність працювати з технічною документацією та програмними засобами; Здатність до самонавчання та професійного розвитку.
Результати навчання	Після вивчення дисципліни здобувач повинен: знати основні поняття та принципи об'єктно-орієнтованого програмування, уміти створювати класи, об'єкти, методи та властивості; застосовувати інкапсуляцію, наслідування, поліморфізм і абстракцію; розробляти, тестувати та налагоджувати програми; аналізувати структуру програмного коду та вдосконалювати його.
Навички Soft skills	Логічне та критичне мислення; уміння працювати в команді; відповідальність за результат; навички комунікації; самоорганізація та управління часом; здатність до самостійного розв'язання практичних задач.
Методи навчання	Навчання проводиться у вигляді класичних, мультимедійних та інтерактивних лекцій і консультацій з викладачами. Лабораторні та практичні заняття Вивчення дисципліни здійснюється самостійно за допомогою навчальної літератури, конспектів та електронних ресурсів. Розв'язування програмних задач. Виконання індивідуальних завдань. Консультації та обговорення.
Матеріально-технічні ресурси	Комп'ютерний клас або персональні комп'ютери. Середовище розробки програмного забезпечення. Мультимедійне обладнання. Навчально-методичні матеріали та електронні ресурси. Доступ до глобальної мережі Internet. Хмарні технології підтримки освітнього процесу. Google Classroom / Moodle (підтримка практичних занять). Google Meet / Zoom (онлайн-лекції,

	практичні заняття).

ІНФОРМАЦІЯ ПРО КОНСУЛЬТАЦІЇ

Відповідно до окремого графіка

ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Назви змістових модулів	Зміст модулів
Змістовий модуль 1. Теоретичні основи об'єктно-орієнтованого програмування	**Тема 1. Вступ до об'єктно-орієнтованого програмування.** Поняття парадигм програмування. Передумови виникнення об'єктно-орієнтованого підходу. Основні переваги ООП порівняно з процедурним програмуванням. Сфери застосування об'єктно-орієнтованого програмування.
	Тема 2. Основні поняття ООП: клас, об'єкт, властивості, методи. Поняття класу та об'єкта. Атрибути і методи класу. Створення об'єктів, доступ до їхніх властивостей і методів. Життєвий цикл об'єкта.
	Тема 3. Принципи об'єктно-орієнтованого програмування. Інкапсуляція, наслідування, поліморфізм, абстракція як базові принципи ООП. Їх роль у побудові гнучких, масштабованих і повторно використовуваних програмних систем.
	Тема 4. Конструктори, деструктори та керування станом об'єктів. Призначення конструкторів і деструкторів. Ініціалізація об'єктів. Перевантаження конструкторів. Забезпечення коректного функціонування об'єктів у програмі.
	Тема 5. Модифікатори доступу та організація інтерфейсу класу. Закриті, захищені та відкриті елементи класу. Формування публічного інтерфейсу. Приховування реалізації та захист даних об'єкта.
Змістовий модуль 2. Практичне застосування об'єктно-орієнтованого програмування	**Тема 6. Основи проєктування класів.** Виділення сутностей предметної області. Побудова класів відповідно до вимог задачі. Визначення зв'язків між класами. Принципи якісного проєктування програмних компонентів.
	Тема 7. Наслідування та ієрархії класів. Побудова базових і похідних класів. Розширення і спеціалізація функціональності. Повторне використання коду на основі успадкування. Особливості створення ієрархій класів.

	Тема 8. Поліморфізм і перевизначення методів. Статичний і динамічний поліморфізм. Перевантаження та перевизначення методів. Використання поліморфізму для забезпечення універсальності програмних рішень.
	Тема 9. Абстрактні класи та інтерфейси. Поняття абстрактного класу. Інтерфейси як засіб формалізації поведінки об’єктів. Використання абстракцій для побудови гнучких програмних систем.
	Тема 10. Обробка винятків і надійність програм.Поняття виняткових ситуацій. Засоби обробки помилок у програмах. Забезпечення стійкості та надійності програмних продуктів.
	Тема 11. Колекції об’єктів і робота зі складними структурами даних. Організація масивів, списків, стеків, черг та інших контейнерів для зберігання об’єктів. Основні підходи до керування наборами даних в об’єктно-орієнтованих програмах.
	Тема 12. Основи створення прикладних програм засобами ООП. Розроблення невеликих програмних проєктів із застосуванням класів, об’єктів, успадкування, поліморфізму та механізмів обробки винятків. Тестування і налагодження програм.

СПИСОК РЕКОМЕНДОВАНИХ ДЖЕРЕЛ

Основна література:

1. Буч Г., Максимчук Р. А., Енгл М. В., Янг Б. Дж., Коналлен Дж., Х’юстон К. *Object-Oriented Analysis and Design with Applications*. 3rd ed. Boston: Addison-Wesley / Pearson Education, 2007. Для вивчення тем 1–6 рекомендовано використовувати с. 29–144; для UML і проєктування — с. 147–246.
2. Седжвік Р., Уейн К. *Computer Science: An Interdisciplinary Approach*. 1st ed. Boston: Addison-Wesley Professional, 2016. Для тем базового програмування — с. 1–328; для об’єктно-орієнтованого програмування — с. 329–492.
3. Шилдт Г. *Самовчитель з об’єктно-орієнтованого програмування* [видання уточнити]. Для тем ООП доцільно використовувати розділи про класи, наслідування, інтерфейси та винятки; сторінки слід уточнити за наявним у бібліотеці виданням. Орієнтиром може бути видання Шилдта *Java: The Complete Reference*, 7th ed., с. 143, 157, 189, 217, 249.
4. Лафоре Р. *Object-Oriented Programming in C++*. 4th ed. Indianapolis: Sams Publishing, 2001/2002. Для тем 2–12 рекомендовано використовувати с. 215–848 вибірково за змістовими модулями.
5. Еккель Б. *Thinking in Java*. 4th ed. Upper Saddle River: Prentice Hall / Pearson, 2006. Для тем інкапсуляції, наслідування, поліморфізму й винятків рекомендовано використовувати с. 15–60, 145+, 171+, 197+, 333–370.
6. Дейтел Х., Дейтел П. *Java How to Program* / *C++ How to Program*. Для Java-орієнтованого курсу — с. 367–530; для C++-орієнтованого — с. 431–746

Допоміжна література:

1. Мартін Р. Чистий код. Створення і аналіз програмного забезпечення.

Оригінальне видання: Martin, Robert C. *Clean Code: A Handbook of Agile Software Craftsmanship*. Для дисципліни з ООП цю книгу доцільно використовувати насамперед для тем, пов'язаних зі стилем коду, іменуванням, функціями, класами, інкапсуляцією, обробкою помилок і побудовою систем. Рекомендовані сторінки: розд. 1 "Clean Code" — с. 1–15; розд. 2 "Meaningful Names" — с. 17–30; розд. 3 "Functions" — с. 31–65; розд. 6 "Objects and Data Structures" — с. 93–101; розд. 7 "Error Handling" — с. 103–112; розд. 10 "Classes" — с. 135–151; розд. 11 "Systems" — с. 153–169.

2. Мейєр Б. Об'єктно-орієнтоване конструювання програмних систем.

Базове академічне видання: Meyer, Bertrand. *Object-Oriented Software Construction*. 2nd ed. Це одне з найсильніших джерел для теоретичних основ ООП, модульності, абстрактних типів даних, класів, об'єктів, контрактного програмування, винятків і спадкування. Для курсу доцільно використовувати такі розділи: Chapter 2 — с. 21–34; Chapter 3 — с. 39–64; Chapter 4 — с. 67–99; Chapter 5 — с. 101–119; Chapter 6 — с. 121–160; Chapter 7 — с. 165–215; Chapter 8 — с. 217–277; Chapter 10 — с. 317–330; Chapter 11 — с. 331–408; Chapter 12 — с. 411–438; Chapter 14 — с. 459–517; Chapter 16 — с. 569–610.

3. Фаулер М. Архітектура корпоративних програмних застосунків.

Оригінальне видання: Fowler, Martin. *Patterns of Enterprise Application Architecture*. Це джерело варто використовувати як поглиблене для тем про проєктування класів, доменну логіку, розподіл відповідальностей між об'єктами та архітектурні патерни. Рекомендовані сторінки: Introduction — с. 1–13; Part 1 "The Narratives" — с. 15–106; Chapter 9 "Domain Logic Patterns" — с. 109–142; Chapter 10 "Data Source Architectural Patterns" — с. 143–182; Chapter 11 "Object-Relational Behavioral Patterns" — с. 183–214; Chapter 12 "Object-Relational Structural Patterns" — с. 215–304; Chapter 13 "Object-Relational Metadata Mapping Patterns" — с. 305–327.

4. Гамма Е., Хелм Р., Джонсон Р., Вліссідес Дж. Прийоми об'єктно-орієнтованого проєктування. Патерни проєктування.

Класичне видання: Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John. *Design Patterns: Elements of Reusable Object-Oriented Software*. Для дисципліни ООП це ключове джерело для тем про повторне використання коду, композицію, делегування, інтерфейси та шаблони проєктування. У видавничому TOC чітко наведені такі стартові сторінки патернів: Abstract Factory — с. 87, Builder — с. 97, Factory Method — с. 107, Prototype — с. 117, Singleton — с. 127, Adapter — с. 139, Bridge — с. 151, Composite — с. 163, Decorator — с. 175, Facade — с. 185, Flyweight — с. 195, Proxy — с. 207. Для загального ознайомлення також варто опрацювати розд. 1 "Introduction", розд. 2 "A Case Study: Designing a Document Editor", а також розд. 5 "Behavioral Patterns".

5. Мартін Р. Чиста архітектура.

Оригінальне видання: Martin, Robert C. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Цю книгу доцільно використовувати як сучасне допоміжне джерело для тем про принципи SOLID, межі модулів, залежності та архітектурне мислення в об'єктно-орієнтованих системах. Найкорисніші для курсу сторінки: Chapter 5 "Object-Oriented Programming" — с. 33–47; Chapter 7 "SRP" — с. 61–67; Chapter 8 "OCP" — с. 69–75; Chapter 9 "LSP" — с. 77–82; Chapter 10 "ISP"

— с. 83–86; Chapter 11 “DIP” — с. 87–91; Chapter 22 “The Clean Architecture” — с. 201–209; Chapter 23 — с. 211–215; Chapter 24 — с. 217–220; Chapter 25 — с. 221–228.

6. Фаулер М. Рефакторинг. Поліпшення наявного коду.

7. Актуальне англомовне видання: Fowler, Martin. *Refactoring: Improving the Design of Existing Code*. 2nd ed.

Для курсу ООП книга особливо корисна в частині покращення структури класів і методів, усунення “запахів коду”, інкапсуляції та роботи зі спадкуванням. Рекомендовані сторінки: Chapter 1 — с. 1–43; Chapter 2 — с. 45–70; Chapter 3 “Bad Smells in Code” — с. 71–84; Chapter 4 “Building Tests” — с. 85–99; Chapter 5 — с. 101–104; Chapter 6 “A First Set of Refactorings” — с. 105–160; Chapter 7 “Encapsulation” — від с. 161; Chapter 12 “Dealing with Inheritance” — с. 349–405. Якщо потрібен саме каталог окремих рефакторингів, у TOC preview наведені стартові сторінки, зокрема: Extract Function — 106, Encapsulate Variable — 132, Extract Class — 182, Move Function — 198, Replace Conditional with Polymorphism — 272, Pull Up Method — 350, Extract Superclass — 375, Replace Superclass with Delegate — 399.

Інформаційні джерела та Internet-ресурси:

1. Oracle Docs / dev.java — офіційна документація Java. Для вивчення тем з ООП використовувалися розділи “Objects, Classes, Interfaces, Packages, and Inheritance” на dev.java, а також матеріали Java Tutorials “Classes and Objects”, “Inheritance”, “Object as a Superclass” і специфікація Java Language Specification, Chapter 9 “Interfaces”. Ці джерела доцільні для тем про класи, об’єкти, інкапсуляцію, наслідування, інтерфейси та перевизначення методів.

2. Microsoft Learn — офіційна документація C# та .NET. Використовувалися офіційні матеріали “Classes, structs, and records”, “Inheritance”, “Polymorphism”, а також навчальний маршрут “Implement inheritance and polymorphism”. Саме ці розділи доцільно застосовувати для тем про класи, базові й похідні типи, інтерфейси, поліморфізм і реалізацію принципів ООП у C#.

3. Python Docs — офіційна документація Python. Для курсу використовувалися The Python Tutorial і насамперед розділ “Classes”. Саме ці матеріали охоплюють створення класів, екземплярів, атрибутів, методів, спадкування та основні об’єктні механізми Python.

4. cppreference — довідковий ресурс з C++. Для тем з ООП у C++ використовувалися статті “Derived classes”, “virtual function specifier” та “final specifier”. Ці матеріали доцільні для опрацювання спадкування, віртуальних методів, динамічного поліморфізму та заборони подальшого перевизначення або наслідування.

5. GitHub — приклади програмних проєктів і навчальних репозиторіїв. Використовувалися тематичні добірки GitHub Topics: “object-oriented-programming”, “oop” та “oop-examples” для підбору демонстраційних прикладів, навчальних проєктів, шаблонів класів і невеликих студентських застосунків. Це доречно використовувати як джерело прикладів до лабораторних і самостійних завдань.

6. Stack Overflow — професійна спільнота для аналізу типових задач програмування. Фактично використовувалися не довільні обговорення, а насамперед tag wiki і типові питання в розділах java, python та c++-faq. Це доцільно для аналізу поширених помилок у темах спадкування, перевизначення методів, роботи з класами, поліморфізму та синтаксичних особливостей мов.

7. Відкриті освітні курси з програмування на платформах Coursera, edX, Prometheus. Із Coursera доцільно зазначити

курси “Object Oriented Programming”, “Object-Oriented Programming Concepts” та “Object Oriented Programming Specialization”; з edX — “Introduction to Object-Oriented Programming with Java” і “Object-Oriented Programming”; з Prometheus — “Програмування для всіх: основи Python”, “Основи Python” та “Основи програмування”. Ці курси використовуються для поглиблення тем про базові принципи ООП, класи, об’єкти, наслідування, поліморфізм і практичне програмування.

ШКАЛА ОЦІНЮВАННЯ

Сума балів за всі види навчальної діяльності	Оцінка ECTS	Оцінка за національною шкалою
90 – 100	**A**	відмінно
80 – 89	**B**	добре
65 – 79	**C**	
55 – 64	**D**	задовільно
50 – 54	**E**	
35 – 49	**FX**	незадовільно з можливістю повторного складання
1 – 34	**F**	незадовільно з можливістю повторного вивчення дисципліни

ЗАСОБИ ОЦІНЮВАННЯ

	кількість	бал (за одиницю)	всього балів	кількість	бал (за одиницю)	всього балів
Робота на практичних заняттях	16	1	**16**	5	1	**5**
Презентація та захист результатів виконаних індивідуальних завдань	2	5	**10**	2	5	**10**
Модульні контрольні роботи	2	12	**24**	2	12	**24**
Розрахунково-аналітична робота	-	-	**-**	1	11	**11**
Всього			**50**			**50**
	100					