Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Кафедра прикладної математики

Об’єктно-орієнтоване програмування

Робоча програма навчальної дисципліни (Силабус)

Реквізити навчальної дисципліни

Рівень вищої освіти	***Перший (бакалаврський)***
Галузь знань	*11 Математика і статистика*
Спеціальність	*113 Прикладна математика*
Освітня програма	*Наука про дані та математичне моделювання*
Статус дисципліни	*Вибіркова*
Форма навчання	*Очна (денна)*
Рік підготовки, семестр	*2 курс, осінній семестр*
Обсяг дисципліни	*4 кредити ЄКТС / 120 год.: (36 год. – лекції, 36 год. – лабораторні роботи, 48 год. – СРС)*
Семестровий контроль/ контрольні заходи	*Залік, МКР, лабораторні роботи*
Розклад занять	*https://schedule.kpi.ua/lecturers?lecturerId=a09cd61f-1cca-44a3-8cb5-9203e4f67234* *лекція – 1 раз на тиждень, лабораторні – 1 раз на тиждень*
Мова викладання	*Українська*
Інформація про керівника курсу / викладачів	Лектор: ас. Дмитренко Олександра Анатоліївна e-mail: o_dmytrenko@lll.kpi.ua Лабораторні: доц., к. ф.-м. н. Костюшко Ірина Анатоліївна, e-mail: kostushkoia5@gmail.com ас. Дмитренко Олександра Анатоліївна
Розміщення курсу	https://classroom.google.com/c/NzExMTMwMzQwNjYw?cjc=o7ghcm5

Програма навчальної дисципліни

1. Опис навчальної дисципліни, її мета, предмет вивчення та результати навчання

Метою *навчальної дисципліни полягає у формуванні у студентів знань та навичок з розробки програмного забезпечення на основі об'єктно-орієнтованої парадигми. Дисципліна спрямована на вивчення принципів та концепцій ООП, таких як інкапсуляція, наслідування та поліморфізм, а також їх практичне застосування при створенні програм. Крім того, ООП має на меті розвинути у студентів здатність проектувати та реалізовувати складні програмні системи, використовуючи об'єктно-орієнтований підхід для покращення структури, надійності та повторного використання коду.*

Предметом *навчальної дисципліни є*

- *основні концепції та принципи об'єктно-орієнтованого програмування;*
- *шаблони проектування та їх застосування в об'єктно-орієнтованій розробці;*

- *методи та інструменти проектування та розробки об'ємних програмних систем з використанням ООП;*
- *методи тестування ООП коду.*

Під час вивчення даної дисципліни студенти набудуть:
<u>І. загальних компетентностей:</u>
- *ЗК1. Здатність учитися і оволодівати сучасними знаннями;*
- *ЗК2. Здатність застосовувати знання у практичних ситуаціях;*
- *ЗК3. Здатність генерувати нові ідеї (креативність);*
- *ЗК4. Здатність бути критичним і самокритичним;*
- *ЗК5. Здатність проведення досліджень на відповідному рівні.*
- *ЗК6. Здатність до абстрактного мислення, аналізу та синтезу;*
- *ЗК7. Здатність до пошуку, оброблення та аналізу інформації з різних джерел;*
- *ЗК8. Знання та розуміння предметної області та розуміння професійної діяльності;*
- *ЗК10. Навички у використанні інформаційних і комунікаційних технологій.*
- *ЗК12. Визначеність і наполегливість щодо поставлених завдань і взятих обов'язків.*

<u>ІІ. фахових компетентностей:</u>
- *ФК3. Здатність обирати та застосовувати математичні методи для розв'язання прикладних задач, моделювання, аналізу, проектування, керування, прогнозування, прийняття рішень;*
- *ФК4. Здатність розробляти алгоритми та структури даних, програмні засоби та програмну документацію;*
- *ФК 5. Здатність проектувати бази даних, інформаційні системи та ресурси.*
- *ФК 6. Здатність розв'язувати професійні задачі за допомогою комп'ютерної техніки, комп'ютерних мереж та Інтернету, в середовищі сучасних операційних систем, з використанням стандартних офісних додатків.*
- *ФК7 Здатність експлуатувати та обслуговувати програмне забезпечення автоматизованих та інформаційних систем різного призначення.*
- *ФК8 Здатність використовувати сучасні технології програмування та тестування програмного забезпечення.*
- *ФК9. Здатність до проведення математичного і комп'ютерного моделювання, аналізу та обробки даних, обчислювального експерименту, розв'язання формалізованих задач за допомогою спеціалізованих програмних засобів;*
- *ФК10 Здатність створення документів встановленої звітності, використання нормативно-правових документів.*
- *ФК12. Здатність до пошуку, систематичного вивчення та аналізу науково-технічної інформації, вітчизняного й закордонного досвіду, пов'язаного із застосуванням математичних методів для дослідження різноманітних процесів, явищ та систем;*
- *ФК13 Здатність зрозуміти постановку завдання, сформульовану мовою певної предметної галузі, здійснювати пошук та збір необхідних вихідних даних.*
- *ФК14. Здатність сформулювати математичну постановку задачі, спираючись на постановку мовою предметної галузі, та обирати метод її розв'язання, що забезпечує потрібні точність і надійність результату;*
- *ФК15 Здатність брати участь у складанні наукових звітів із виконаних науководослідних робіт та у впровадженні результатів проведених досліджень і розробок.*
- *ФК16 Здатність до ефективної професійної письмової й усної комунікації українською мовою та однією з офіційних мов ЄС.*

Програмними результатами навчання є:

- *РН 10 Володіти методиками вибору раціональних методів та алгоритмів розв'язання математичних задач оптимізації, дослідження операцій, оптимального керування і прийняття рішень, аналізу даних.*
- *РН 11 Вміти застосовувати сучасні технології програмування та розроблення програмного забезпечення, програмної реалізації чисельних і символьних алгоритмів.*
- *РН 12 Розв'язувати окремі інженерні задачі та/або задачі, що виникають принаймні в одній предметній галузі: в соціології, економіці, екології та медицині.*
- *РН 14 Виявляти здатність до самонавчання та продовження професійного розвитку.*
- *РН 15 Уміти організувати власну діяльність та одержувати результат у рамках обмеженого часу.*
- *РН 17 Уміти здійснювати збір, опрацювання, аналіз, систематизацію науково-технічної інформації, уникаючи при цьому академічної недоброчесності.*
- *РН 18 Ефективно спілкуватися з питань інформації, ідей, проблем та рішень зі спеціалістами та суспільством загалом.*
- *РН 19 Збирати та інтерпретувати відповідні дані й аналізувати складності в межах своєї спеціалізації для донесення суджень, які відбивають відповідні соціальні та етичні проблеми.*
- *РН 20 Демонструвати навички професійного спілкування, включаючи усну та письмову комунікацію українською мовою та принаймні однією з офіційних мов ЄС.*
- *РН 22 Володіти основними принципами та методами побудови баз даних та інформаційних систем.*

2. Пререквізити та постреквізити дисципліни (місце в структурно-логічній схемі навчання за відповідною освітньою програмою)

Пререквізитами для вивчення дисципліни є дисципліни ЗО 19 «Програмування мовою Python», ЗО 20.1 «Програмування. Частина 1», ЗО 20.2 «Програмування. Частина 2», ЗО 22 «Алгоритми та структури даних». Особливо корисними для успішного опанування дисципліни є хороше розуміння програмування на Java, або ж C, C#, C++, Python.

Вивчення дисципліни є передумовою вивчення дисциплін ПО 08.1 Алгоритми і системи комп'ютерної математики, ПО 03 «Переддипломна практика».

3. Зміст навчальної дисципліни

РОЗДІЛ 1. ЗАГАЛЬНІ ПОНЯТТЯ ПРО ОБ'ЄКТНО ОРІЄНТОВАНИЙ СТИЛЬ ПРОГРАМУВАННЯ

Тема 1.1. Вступ та основні поняття ООП

Тема 1.2. Тестування ООП коду

Тема 1.3. Поняття чистого коду (clean code)

РОЗДІЛ 2. МОДЕЛЮВАННЯ ПРОЕКТУ ТА ШАБЛОНИ ПРОЕКТУВАННЯ

Тема 2.1. Схема класів проекту у форматі UML

Тема 2.2. Шаблони проектування та написання коду

МКР

РОЗДІЛ 3. СПОСОБИ РОЗРОБКИ ООП ПРОЕКТУ

Тема 3.1. Методи альтернативного погляду на написання коду

Тема 3.2. Способи розробки ПЗ

РОЗДІЛ 4. СТВОРЕННЯ ОБ'ЄМНОГО ООП ДИЗАЙНУ ТА ПРОЕКТУ

Тема 4.1. Архітектура Web проектів з використанням ООП

Тема 4.2. Інверсія контролю IoC. Spring. Тестування з використанням IoC.

Тема 4.3. Архітектура багатомодульних проектів

Залік

4. Навчальні матеріали та ресурси

Базова література

1. Head First Object-Oriented Analysis and Design / Brett McLaughlin, Gary Pollice, David West. – O'Reilly Media, Inc., 2006. – 603 с. – Режим доступу: https://www.oreilly.com/library/view/head-first-object-oriented/0596008678/ , https://theswissbay.ch/pdf/Gentoomen%20Library/Programming/O%27Reilly%20Desining%20Series/O%27Reilly%20Head%20First%20Object-Oriented%20Design%20and%20Analysis.pdf
2. Refactoring: Improving the Design of Existing Code / Martin Fowler – Addison-Wesley Professional; 2nd edition, 2018. – 448 с. Режим доступу: https://github.com/wuzhouhui/misc2/blob/master/Refactoring.Improving.the.Design.of.Existing.Code.2nd.Edition.2018.11.pdf
3. Clean Code: A Handbook of Agile Software Craftsmanship / Robert C. Martin. – Pearson, – 2008. – 464 с.– Режим доступу: https://www.oreilly.com/library/view/clean-code-a/9780136083238/ , https://github.com/martinmurciego/good-books/blob/master/Clean%20Code_%20A%20Handbook%20of%20Agile%20Software%20Craftsmanship%20-%20Robert%20C.%20Martin.pdf
4. Head First Design Patterns, 2nd Edition / Eric Freeman, Elisabeth Robson – O'Reilly Media, Inc. 2020. – 694 с. – Режим доступу: https://www.oreilly.com/library/view/head-first-design/9781492077992/ , https://github.com/ajitpal/BookBank/blob/master/%5BO%60Reilly.%20Head%20First%5D%20-%20Head%20First%20Design%20Patterns%202nd%20Edition%20-%20%5BFreeman%5D.pdf
5. Документація по Spring https://docs.spring.io/spring-framework/reference/index.html

Допоміжна література

6. Відомі вислови Мартіна Фаулера для натхнення на написання якісного коду https://www.goodreads.com/work/quotes/44258-refactoring-improving-the-design-of-existing-code
7. Шаблони проектування: https://refactoring.guru/design-patterns/catalog
8. Підходи до архітектура проектів: https://www.martinfowler.com/architecture/
9. Лекції з ООП, зокрема матеріал про UML діаграми класів: https://atomicobject.com/oo-programming/uml-notation

Навчальний контент

5. Методика опанування навчальної дисципліни (освітнього компонента)

5.1. Лекції

1	**Загальні поняття об'єктно-орієнтованого програмування з прикладами на Java** - Що таке об'єктно орієнтоване програмування. - Історія ООП. - Інші стилі програмування. - Основні принципи ООП: класи та об'єкти, поліморфізм, наслідування, інкапсуляція, абстракція. - Практична частина - класи, поля, методи, getters, setters, інтерфейси, абстрактні класи.
2	**Дизайн ООП застосунків. Вибір мови програмування з ООП можливостями** - Принципи дизайну ООП застосунків: SOLID, DRY, YAGNI, KISS. - Як вони реалізуються. - Основні характеристики Java. - Історія Java. - Інші мови програмування, в яких використовується ООП (C#, C++, Python, Kotlin). - Їх принади та відмінності від Java в плані розробки великих проектів.
3	**Реалізація поліморфізму та наслідування у Java** - Види класів та методів. - Наслідування класів та методів . - Специфікатори super та final. - Поліморфізм. - Реалізація поліморфізму. - Перевизначення методів. - Раннє та пізнє зв'язування.
4	**Тестування ООП коду** - Написання тестів до ООП коду. - Демо написання тестів на JUnit 5. - Інші інструменти для тестування, такі як TestNG та Mockito
5	**Слідкування за чистотою коду** - Основні принципи чистого коду (clean code). - Інструмент стеження за якістю коду - Sonarqube.
6	**Схема класів проекту у форматі UML** - Інтерфейси та класи у UML. - UML схема наслідування. - Композиція у UML.

	- Моделювання проекту за допомогою графічного представлення через UML (Unified Modeling Language).
7	**Шаблони проектування GRASP** - Загальна інформація про GRASP шаблони проектування. - Інформаційний експерт (Information Expert), - творець (Creator), - низька зв'язаність (Low Coupling) - високе зачеплення (High Cohesion), - стійкий до змін (Protected Variations), - контролер (Controller), - поліморфізм (Polymorphism), - чиста вигадка (Pure Fabrication)
8	**Шаблони-творці - Creational patterns** - Одинак - Singleton, - Фабрика - Factory - Абстрактна фабрика - Фабричний метод - Будівельник - Прототип
9	**Структурні шаблони - Structural patterns** - Адаптер - Декоратор - Міст - Фасад - Проксі - Композит
10	**Поведінкові шаблони - Behavioral patterns** - Оглядач - Observer - Стратегія - Ланцюг відповідальності - Команда - Ітератор - Шаблонний метод - Стратегія - Відвідувач - Медіатор - Стан
11	**Модульна контрольна робота**
12	**Методи альтернативного погляду на написання коду** - Парне програмування (PP), - Test Driven Development в теорії та на практиці
13	**Способи розробки ПЗ** - Екстремальне програмування (XP). - Адаптивна розробка програмного забезпечення. - Behavior Driven Development. - Вибір стилю програмування в залежності від структури проекту.
14	**Дизайн великого ООП проекту** - Об’єктно-орієнтований аналіз та дизайн, - MVC шаблон, - Domain-Driven Design, - Гексагональна архітектура
15	**Інверсія контролю та інструменти для її здійснення**

	- Інверсія контролю IoC - Spring - Інші інструменти для IoC
16	**Тестування з використанням IoC** - Тестування з допомогою IoC підходу - Використання Mockito.
17	**Архітектура великих проектів** - Мікросервісна архітектура, - використання баз даних - Події та орієнтоване на події управління поведінкою програми (Event Driven Development), - Aspect-oriented programming (AOP). - Підсумки про ООП, - за та проти ООП для різного роду проектів, - майбутній розвиток ООП
18	**Залік**

5.2. Основні та додаткові лабораторні роботи

№	Тема роботи	Бали
1	**Що таке ООП. Створення об'єктів** - написання проекту на Java з використанням базових структур мови	10
2	**Поліморфізм та чистий код. Вибір помічника (ШІ)** - Реалізувати поліморфізм через класи - через методи - застосувати створені класи та методи - шукаючи додаткову інформацію пропонується проаналізувати також підказки різних ШІ та обрати на майбутнє той, який найзручніший	8
3	**Шаблони проектування 1 + тести + чистий код** - реалізувати відповідні шаблони проектування до варіанту - до коду написати Unit-тести - код має бути оформлено за принципами clean code	10
4	**Шаблони проектування 2 + TDD + чистий код** - пишучи тести створювати код у формі TDD (test driven development) - реалізувати відповідні шаблони проектування до варіанту - код має бути оформлено за принципами clean code	12
5	**UML діаграма об'ємного ООП проекту** - відповідно до варіанту взяти предметну область та характеристики, які мають бути присутніми у проекті - розписати структуру коду у форматі UML діаграми - обрати технології для реалізації необхідного функціоналу	8

6	**Великий ООП проект з використанням обраних технологій, який відповідає затвердженій UML діаграмі. Unit-тести обов'язкові. Запропоновані технології: Maven, Gradle, MVC, Spring або інший IoC інструмент, DB, AOP, Events чи Messages, Postman та Swagger у випадку веб складової) + чистий код. На UI не рекомендується витрачати час.** - користуючись затвердженою UML діаграмою у минулій роботі, написати код	17
7	**Вести спостереження за якісним та кількісним виконанням лабораторних робіт (додаткова)** - заповнювати таблицю з зазначеними характеристиками, долучаючи її до кожного звіту - в фіналі, показати зведену таблицю та висновки стосовно власних характеристик та вподобань під час розробки коду	10

6. Самостійна робота студента

1	**Основні синтаксичні особливості мови Java** Класи, поля інтерфейси, методи, доступ до полів	8 год.
2	**Забезпечення інкапсуляції в межах класів, методів та пакетів** Створення полів, недоступних в інших класах. Створення класів в методах. Визначення їх області видимості.	8 год.
3	**Рефакторинг** Застосування рефакторингу до поганого коду. Вивчення методик рефакторингу.	8 год.
4	**Шаблони проектування** Вивчення шаблонів, визначених викладачем, які не встигли пройти на лекції.	8 год.
5	**Behavior Driven Development** Самостійно спробувати розробити шаблон в стилі Behavior Driven Development.	8 год.
6	**Розібратись, з рівнями Web застосунку** Що таке рівні Controller, Service, Repository DAO	8 год.

Політика та контроль

7. Політика навчальної дисципліни (освітнього компонента)

- *Відвідування лекцій є обов'язковим.*
- *На лекційних заняттях студенти повинні уважно слухати викладача, не перебиваючи його та не створюючи зайвого шуму. Студенти можуть ставити питання для з'ясування незрозумілих моментів та відповідати на запитання викладача. Усі гаджети має бути вимкнено або встановлено на беззвуковий режим, користуватися ними заборонено. Допустиме використання гаджетів для ведення конспектів для осіб зі спеціальними потребами.*
- *Лабораторні роботи студенти виконують мовою Java. Звіт оформлюють у Google Docs, за потреби надання відео воно має бути запропоноване у форматі посилання на YouTube. Викладати матеріали слід у Google Classroom. Захист відбувається в усній формі, в режимі конференції (за дистанційної форми навчання). На захисті студенти мають розповісти ідею та хід виконання роботи, сформулювати*

висновки, відповісти на запитання викладача. Роботи потрібно здавати та захищати вчасно, за запізнення оцінка знижується.

- *Модульну контрольну роботу та залік студенти виконують під час відповідного лекційного заняття протягом фіксованого часу через https://www.classmarker.com/. Оцінка за роботу виставляється цим інструментом автоматично. Роботу студент виконує самостійно. Спілкування та перехід на інші інтернет сторінки заборонений.*
- *Під час вивчення дисципліни студенти повинні дотримуватися правил академічної доброчесності, що передбачає неприпустимість плагіату, списування та інших способів видавання чужого доробку за свій. За недотримання цих правил передбачається покарання, що включає в себе виставлення оцінки «0» за звіти та код лабораторних робіт або «незадовільно», якщо порушення академічної доброчесності було зафіксовано під час проведення заліку чи МКР.*

8. Види контролю та рейтингова система оцінювання результатів навчання (РСО)

Поточний контроль: МКР, лабораторні роботи.

Календарний контроль: проводиться двічі на семестр на 8-ому та на 14-ому тижнях, як моніторинг поточного стану виконання вимог силабуса; студент отримує «зараховано», якщо він здав 2 та 5 лабораторних відповідно на оцінку, вищу за 0.

Семестровий контроль: залік.

Умови допуску до семестрового контролю: набрані 45 або більше балів під час семестру.

Рейтинг студента з кредитного модуля складається з:

1) балів за виконання та захист лабораторних робіт;
2) балів за виконання модульної контрольної роботи (МКР).

СИСТЕМА РЕЙТИНГОВИХ БАЛІВ

1. Бали за виконання та захист лабораторних робіт

Протягом семестру студенти виконують 6 лабораторних робіт. Максимальна кількість балів за всі роботи - 65. За кожну лабораторну роботу бали наведені в таблиці.

Бали нараховуються за:
- якість реалізації завдання та звіту: половина балів;
- відповідь під час захисту лабораторної роботи: половина балів.

Критерії оцінювання якості реалізації коду та звіту, якщо за роботу максимальна оцінка 10 балів. Якщо максимальна оцінка інакша, бали рахуються пропорційно:

5 балів — роботу виконано якісно, в повному обсязі;
4 бали — роботу виконано якісно, в повному обсязі, але вона має вади;
2-3 бали— роботу виконано не повністю або вона має суттєві помилки;
0 балів — роботу виконано неправильно, вона списана, (майже) повністю автозгенерована або відсутня.

Критерії оцінювання відповідей під час захисту лабораторної роботи:
5 балів — відповіді під час захисту повні, добре аргументовані;
4 бали — у цілому відповіді правильні, але мають вади чи незначні помилки;
2-3 бали — відповіді неповні або мають істотні вади;
1 бал - відповіді читаються, а не розповідаються
0 балів — немає відповідей, відповіді неправильні або чужі.

Важливо: Оцінка за код та звіт буде виставлена тільки за умови успішного усного захисту, тобто при наборі 2 або більше балів за вказаною шкалою. За умови неуспішного та/або задовгого захисту, максимальна оцінка буде знижена 1 бал на наступній спробі захисту.

Максимальна кількість балів за виконання та захист лабораторних робіт:
10 + 8 + 10 + 12 + 8 + 17 = **65 балів**.

2. Додаткова лабораторна робота

Студентам пропонується підвищити свої бали за рахунок виконання додаткової лабораторної роботи. Ця робота являє собою спостереження за своїм способом програмування, оцінкою часу, використанням додаткових ресурсів під час розробки кожної зі стандартних робіт. Звіт по цій роботі виглядає як додаткова заповнена таблиця з заданими характеристиками. Представляти цей звіт необхідно показуючи кожну роботу. Оцінка виставляється в кінці, при здачі лабораторної №6. Фінальний звіт має містити висновки за період виконання всіх лабораторних.

Робота оцінюється в 10 балів.

3. Модульний контроль

Студенти пишуть дві модульні контрольні роботи: в середині семестру та наприкінці. Обидві роботи містять запитання у тестовій формі з довільною кількістю відповідей. Запитань 15-20. Більшість запитань оцінюється в 1 бал; в залежності від складності запитання його оцінка може варіюватись від 0.5 до 1.5. В залежності від важливості запитання та формату відповідей, спосіб оцінювання може бути або пропорційним до кількості правильних відповідей, або двійковим (чи наведені всі правильні відповіді та жодної неправильної), або негативним (знімаються бали в разі відмічання неправильної відповіді). Мінімальна оцінка за одне питання - 0. Оцінка рахується автоматично в залежності від налаштувань системою *https://www.classmarker.com/ та стає відомою зразу після написання тесту. Приклади питань наведені в Додатку 1*

Максимальна кількість балів за перший модульний контроль: 20 балів.
Максимальна кількість балів за другий модульний контроль : 15 балів.

4. Залік

Неодмінною умовою допуску до заліку є набір 45 балів за семестр, адже це мінімальна кількість балів, щоб потенційно отримати 60 балів взагалом. Рекомендованою мінімальною оцінкою є 60 балів.

Якщо наприкінці семестру студент набрав не менше 60 рейтингових балів, а також виконав умови допуску до заліку, він отримує залік «автоматом» відповідно до таблиці.

У випадку, якщо сума рейтингових балів менша від 60, але виконано умови допуску до семестрового контролю, студент виконує залікову роботу, при цьому його рейтинг анулюють, після чого бали нараховують за результатами виконання додаткової залікової контрольної роботи.

Залікова робота складається з практичної задачі, а також містить 2 теоретичні питання вагою по 25 балів. Практичне питання вагою в 50 балів: 25 балів за код та 25 балів за усні коментарі та обґрунтування.

Робота пишеться з включеною камерою та відео екрану. Дозволено користуватись пошуковим сервісом, але заборонені засоби ШІ.

На відповідь на практичне питання виділяється 30 хвилин, на теоретичні, включно з захистом практичної частини - 15 хвилин.

Оцінювання залікової роботи 25 балів x 4 питання = **100 балів**.

5. Календарний контроль

Проводять двічі на семестр як моніторинг поточного стану виконання вимог силабуса. Студент отримує «атестовано», якщо зав необхідну кількість лабораторних робіт, відповідно до термінів здачі визначених викладачем.

Розрахунок шкали (R) рейтингу:

Рейтингова шкала з дисципліни складає

$$R = R_C = 65 \text{ балів} + 20 \text{ балів} + 15 \text{ балів} = 100 \text{ балів}.$$

Критерії оцінювання відповідей на кожну задачу:
25 балів — правильна та змістовна відповідь;
23–24 бали — відповідь правильна, змістовна, але має незначні вади;
19–22 балів — відповідь містить незначні помилки чи є неповною;
15–18 балів — відповідь містить кричущі помилки чи є неповною;
0 балів — немає відповіді.

Максимальна кількість балів за предмет: 100 балів

Кількість балів	*Оцінка*
100-95	Відмінно
94-85	Дуже добре
84-75	Добре
74-65	Задовільно
64-60	Достатньо
Менше 60	Незадовільно
Не виконано умов допуску	Не допущено

Робочу програму навчальної дисципліни (силабус):

Склала асистент кафедри ПМА Олександра ДМИТРЕНКО

Ухвалено кафедрою ПМА (протокол № 10 від 02.01.24).

Погоджено Методичною комісією факультету (протокол № 16 від 26.01.2024)

Додаток 1: Питання на МКР та залік

МКР 1

Тема 1: “ООП”

1) Що таке ООП. Яка різниця між класом, об’єктом, екземпляром об’єкту та типом в ООП. Для якої галузі найбільш підходить ООП? Для вирішення яких задач його створювали? Який код, для яких сфер найкраще писати в такому стилі?
2) На яких мовах програмування можна писати в ООП стилі.
3) Основні принципи ООП (поліморфізм, інкапсуляція, наслідування, абстракція), що вони означають, як пишуться в коді.
4) Як пов’язують об’єкти відношення Is-a, Has-a, як це написати на практиці? Порівняння наслідування, композиції, агрегації.
5) Абстракція - для чого інтерфейси та абстрактні класи, різниця між ними, який код там пишеться. Що робити, якщо не треба перебирати наслідніку всі методи батька?
6) Поліморфізм для класів та методів в чому виражається. Особливості застосування Override та Overload.

7) Принципи дизайну ООП застосунку - SOLID. Що означає кожен пункт, як виглядає на практиці.
8) Раннє та пізнє зв'язування.
9) Поясніть акроніми CRUD та DAO. Де та як застосовуються ці операції та об'єкти.
10) Як розшифровуються DRY, YAGNI, KISS. Що вони означають?
11) Питання практичного спрямування:
    a) При наслідуванні порядок виклику конструкторів, про super().
    b) Про що таке final та static, як вони працюють при наслідуванні в різних випадках (для класів, полів, методів)
    c) Поліморфізм та оверрайд ще і до статичних методів може відноситись.
    d) Клас Object та його методи.

Тема 2: "Тестування"

12) Для чого пишуться тести до коду? Коли треба їх писати? Який код необхідно тестувати? Скільки відсотків коду треба покривати тестами?
13) Що таке функціональне та нефункціональне тестування? Які їх підвиди?
14) Що в себе включають та якими засобами виконуються Unit, Integration, End-to-end тестування. Які їх пропорції в коді? Які тести виконують роль приймального (acceptance) тестування?
15) Які кроки мають бути в кожному тесті (Given, When, Then)? Які конструкції мови заборонені в тестах? Чи може бути програмна логіка в тестах? Як гарантувати, що тест тестує елемент коду?
16) Що таке принципи Fail Fast та Fail Safe? Навіщо CI/CD?

Тема 3. Великий проект. Рефакторинг

17) Які існують стилі/способи написання великого проекту з використанням? (Гіперболізація кожного з них призводить до спагетті, лазанья, равіолі та піца коду). До якого стилю належить завелике та непродумане використання об'єктів?
18) Підходи до дизайну та написання проекту та рефакторингу зверху-вниз та знизу-вверх.
19) Які бувають неприємні запахи в ООП коді? Як їх позбутись. Чи деякі конструкції коду завжди "смердять", чи є дозволені випадки їх використання (When to ignore)? Відповіді знаходяться за цим посиланням (всі розділи) https://refactoring.guru/refactoring/smells
20) Коли хороший час для рефакторингу? Які інструменти допомагають його виконувати та слідкувати за чистотою в проекті?

МКР 2

Тема 4: "Шаблони проектування GRASP"

21) Надайте загальну інформацію про GRASP шаблони проектування.
22) Інформаційний експерт (Information Expert),
23) творець (Creator),
24) низька зв'язаність (Low Coupling)
25) високе зачеплення (High Cohesion),
26) стійкий до змін (Protected Variations),

27) контролер (Controller),
28) поліморфізм (Polymorphism),
29) чиста вигадка (Pure Fabrication)

Тема 5: “Шаблони-творці - Creational patterns”

30) Одинак - Singleton,
31) Фабрика - Factory
32) Абстрактна фабрика
33) Фабричний метод
34) Будівельник
35) Прототип

Тема 6: Структурні шаблони - Structural patterns

36) Адаптер
37) Декоратор
38) Міст
39) Фасад
40) Проксі
41) Композит

Тема 7: Поведінкові шаблони - Behavioral patterns

42) Оглядач - Observer
43) Стратегія
44) Ланцюг відповідальності
45) Команда
46) Ітератор
47) Шаблонний метод
48) Стратегія
49) Відвідувач
50) Медіатор
51) Стан

Загальне питання до шаблонів
Описуючи шаблон, слід також вказати, чим він відрізняється від інших схожих в моментах реалізації та за цільовим призначенням. Малювання UML діаграми є частиною опису шаблону.

Залік

Теоретичні питання будуть по одному з МКР1 та МКР2.
Практичне питання буде реалізувати один з GOF (Gang of Four) шаблонів проектування, а також пояснити реалізацію користуючись принципами ООП та чистого коду.

Код має бути реалізовано за допомогою найкращих практик програмування в ООП стилі, з урахуванням JCP (Java Code Convention).

Якщо студент не може реалізувати код у відведений час, код буде оцінюватись в такому вигляді, як є, але студент може розказати, чого не вистачає, коментуючи роботу.