ЛЬВІВСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ БЕЗПЕКИ ЖИТТЄДІЯЛЬНОСТІ

Навчально-науковий інститут цивільного захисту

Кафедра інформаційних технологій та систем електронних комунікацій

ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

«БАЗОВІ ПРИНЦИПИ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ»

Методичні вказівки до виконання курсової роботи зі спеціальності 122 «Комп'ютерні науки» освітнього рівня «бакалавр»

Львів - 2025

Базові принципи об’єктно-орієнтованого програмування : методичні вказівки до виконання курсової роботи зі спеціальності 122 «Комп‘ютерні науки» освітнього рівня «бакалавр». Укл. Юлія Назар, Назарій Бурак – Львів : ЛДУ БЖД, 2025. – 23 с.

Укладачі: Юля НАЗАР, заступник начальника кафедри інформаційних технологій та систем електронних комунікацій, PhD

Бурак НАЗАРІЙ, начальник кафедри інформаційних технологій та систем електронних комунікацій, кандидат технічних наук, доцент

Рецензент: Тарас РАК, проректор ПЗВО «ІТ Степ Університет», доктор технічних наук, професор.

Затверджено на засіданні кафедри управління проектами, інформаційних технологій та телекомунікацій (протокол № *1* від «*27*» *серпня* 2025 року)

Ухвалено методичною радою навчально-наукового інституту цивільного захисту (протокол № *1* від «*15*» *вересня* 2025 року)

ЗМІСТ

ВСТУП

Для реалізації програмних продуктів застосовують різні технології програмування, такі як процедурно-орієнтовані, об'єктно-орієнтовані або проблемно-орієнтовані (логічні). Без виключень, найбільшої популярності серед прикладних програмістів набула технологія, що побудова на принципах об’єктно-орієнтованості. Слід нагадати, що ***об'єктно-орієнтоване програмування*** – це технологія створення складного програмного забезпечення, яке засноване на представленні програми у вигляді сукупності об'єктів, кожен з яких є екземпляром певного класу, а класи утворюють ієрархію із спадкоємством властивостей. Проте, для написання програмної логіки всередині класів в більшості випадків все ж таки використовується добре відомий процедурний підхід. Основна перевага об’єктно-орієнтованого програмування – це значне спрощення процесів модернізації та модифікації програмних систем на відміну від процедурного підходу. Значно легше маніпулювати десятками класів зі своїми методами, ніж тисячами функцій процедурної мови.

На сьогодні існують та набули широкої популярності багато мов програмування, які підтримують об’єктно-орієнтованість, або ж є виключно об’єктно-орієнтованими (C++, Objective-C, C#, Java, Python, PHP, Ruby тощо). До різновиду суто об’єктно-орієнтованих мов належить Java, яку обрано основною для виконання курсової роботи. Чому ми зупинились на виборі саме цієї мови? Відповідь на це питання лежить у рейтингу Java серед розробників (як Senior Developer так і початківців). В якості джерел рейтингу взято дані Всесвітньовідомого журналу IEEE Spectrum, що видається Інститутом інженерів електротехніки та електроніки (IEEE, USA), а також рейтинг TIOBE, який сформовано на основі суми запитів в пошуковій системі Google. Всі ці рейтинги об’єднує єдине – незмінна група лідерів, які лише чергуються між собою. Вагоме місце серед означеної групи лідерів займає саме об’єктно-орієнтована мова програмування Java. Зважаючи на високу популярність Java, відповідність її характеристик основним сучасним парадигмам програмування, а саме головне – кросплатформність, для написання курсової роботи рекомендовано використовувати саме цю мову програмування (версія не має значення, бажано SE 8 та вище).

З однойменного курсу добре відомо, що існує три парадигми об’єктно-орієнтованого програмування, а саме: поліморфізм, інкапсуляція та наслідування. Окремо виділяють, також, абстракцію. Власне ключовою задачею курсової роботи є закріплення навиків застосування основних принципів об’єктно-орієнтованого програмування під час реалізації реальних проектів. Проте перш ніж створювати об’єктно-орієнтовані моделі програних систем, рекомендовано спочатку

відображати їх логіку у вигляді алгоритму, а для кращої уяви про структуру та ієрархію класів будувати так звані діаграми класів (UML-діаграми). Зважаючи на це ***метою курсової роботи*** є засвоєння основних підходів та методів побудови алгоритмів, закріплення навиків програмної реалізації написаних алгоритмів із використанням принципів об’єктно-орієнтованого підходу, а також побудови об’єктно-орієнтованих структур за допомогою уніфікованої мови моделювання (Unified Modeling Language).

1. СТРУКТУРА ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Курсова робота повинна містити такі частини:

Титульна сторінка (додаток 1)

Анотація

Зміст до курсової роботи

Основна частина, яка складається з:

- вступу;
- розробки та опису алгоритму згідно з індивідуальним завданням;
- програмної реалізації алгоритму згідно з індивідуальним завданням;
- опису реалізованих методів поданих у вигляді коментарів/документації;
- схеми викликів методів або діаграми класів (UML-діаграми).

Висновки

Список використаної літератури

Додатки (за необхідності)

2. ОБОВ'ЯЗКОВИЙ ЗМІСТ ОСНОВНИХ РОЗДІЛІВ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Анотація (до 1 стор.). В анотації викладають короткі відомості про курсову роботу, а саме її тему, кількість сторінок, джерел використаної літератури, додатків тощо.

Зміст (1 стор.). У змісті вказуються назви розділів курсової роботи та їх сторінки (додатки не нумеруються).

Вступ (2-3 стор.). У вступі необхідно описати проблему у загальному вигляді, визначити її актуальність, сформулювати мету та завдання курсової роботи. Вступ має носити характер презентації власної ідеї та підходів щодо її реалізації. У вступі має бути описана загальна логіка та функціонал, реалізацію якого задумано під час виконання курсової роботи.

1. Розробка та опис алгоритму. Метою розділу є відпрацювання базових навиків побудови архітектури програмної системи, відтворення її логіки шляхом подання та опису алгоритму роботи.

У цьому розділі потрібно навести опис алгоритму та оцінити його складність. Також рекомендується проаналізувати особливості алгоритму з погляду об'єктно-орієнтованого програмування.

Розділ повинен містити представлення алгоритму у вигляді псевдокоду та/або у вигляді блок-схеми алгоритму з необхідними поясненнями (спосіб представлення алгоритму обирається на власний розсуд).

Слід зауважити, що формальних правил написання псевдокоду не існує. Псевдокод – не неформальний запис алгоритму програми, який використовує загальну структуру найбільш розповсюджених мов програмування нехтуючи деталями та тонкощами синтаксису, що не впливають на розуміння алгоритму (наприклад тип даних, виклик підпрограм, обов'язкові розділові знаки тощо). Псевдокод є полегшеним представленням програмного коду. На відміну від мов програмування, на синтаксис псевдокодів не накладається жодних правил чи обмежень. В псевдокод можна включати математичні вирази, так, як їх прийнято подавати в математиці. Деякі елементи псевдокоду можуть містили фрагменти природньої мови (української, англійської).

Призначення блок-схеми алгоритму полягає у представлені алгоритму розв'язування поставленої задачі за допомогою геометричних елементів (блоків), які стандартизовані відповідно до ISO 5807-85 "Обробка інформації. Символи і умовні позначення блок-схем даних, програм та систем, схем програмних мереж і системних ресурсів". Для наочності, основні стандартизовані елементи блок-схеми алгоритму наведено в додатку Б.

2. Програмна реалізація алгоритму. Основною метою розділу є закріплення практичних навиків написання програмного коду із застосуванням загальних принципів об'єктно-орієнтованого підходу.

В даному розділі за допомогою об'єктно-орієнтованої мови програмування (рекомендовано Java) необхідно реалізувати представлений алгоритм та продемонструвати процес реалізації в пояснювальній записці у вигляді лістингів окремих частин програми. Лістинги необхідно подавати окремими блоками (методами) та структурувати за класами.

До виконання індивідуального завдання виданого викладачем слід застосовувати творчий та креативний підхід (на що буде звернено увагу під час захисту власного проекту). Виконавець роботи самостійно будує логіку програми, кількість та різновиди методів, способи їх виклику, кількість та різновиди змінних, необхідність застосування конструкторів, визначення модифікаторів доступу тощо. Основне завдання – це реалізація задекларованої у вступі та відображеної в алгоритмі ідеї функціонування програми.

Для підтвердження коректності та працездатності написаних методів в пояснювальній записці необхідно подавати скрін консолі із результатом роботи методу.

До програмного коду висуваються лише такі вимоги:

- мінімальна кількість класів програми має сягати не менше ***шести***;
- кількість класів, що міститимуть лише один метод має сягати не більше ***двох***;

- між класами, їх методами та змінними має забезпечуватись структурований взаємозв'язок (не припустимо будувати логіку програми виключно на функціонуванні незалежних класів та їх методів);
- мінімальна кількість класів, де необхідно застосувати принципи інкапсуляції із можливістю управління доступом (через методи get() та set()) має складати не менше ***двох***;
- мінімальна кількість класів, де необхідно застосувати принципи поліморфізму шляхом перевизначення конструкторів та/або методів, має сягати не менше ***чотирьох***;
- мінімальна кількість класів, де необхідно застосувати принципи унаслідування із можливістю використання змінних, методів та конструкторів батьківського класу, має сягати не менше ***одного (може бути абстрактний)***;
- мінімальна кількість інтерфейсів, де необхідно задавати поведіку програми, з подальшою імплементацією у класах, має сягати не менше ***двох***;
- логіка написаної програми має передбачати застосування ***циклів, умовних розгалужень if/else (можливо swich), масивів (стеки, черги або інші структури даних), рекурсивних методів, бібліотек готових класів (наприклад Math, String, Scanner) у довільній кількості***;
- логіка написаної програми обов'язково має включати використання ***колекцій (List, Set, Que або Map)*** для групування та збереження даних;
- при роботі із колекціями та анонімними класами обов'язково використовувати ***лямбда вирази*** та ***Stream API***;
- окрім застосування базових принципів ООП під час курсової роботи необхідно обов'язково використовувати ***опрацювання виключень***, а також, за можливості, багатопотокове програмування, роботу із датою;
- за бажанням можна створити UI-частину застосунку із використанням бібліотеки Spring та підключити базу даних до проєкту, які будуть вивчатись протягом другої половини навчального курсу.

3. Опис реалізованих методів. Основною метою розділу є відпрацювання навиків написання коментарів до програмного коду.

У цьому розділі наводиться опис програмної реалізації поставленого завдання, зокрема опис методів, змінних та структур даних, які вони використовують тощо. Основна ідея розділу полягає у стислому описі внутрішнього функціоналу написаних методів з посиланнями на лістинги програм другого розділу. До приладу, якщо метод містить цикл, то необхідно стисло, у вигляді коментаря, навести інформацію про умови циклу, його ітераційну складову, призначення, кінцевий результат роботи тощо. Якщо метод використовує дані, які є результатом виконання іншого методу, або ж навпаки метод генерує дані, які є вихідними для подальшого

виконання програми – то таку інформацію необхідно відображати у вигляді стислої характеристики методу.

Основна мета означеного розділу – навчитись формувати документацію (коментарі) до власноруч написаних методів з унікальною логікою. Мова для створення документації може бути ***українська*** або ***англійська***.

4. Представлення діаграми класів. Метою розділу є закріплення базових понять побудови архітектури класів програмної системи за допомогою Unified Modeling Language.

З метою висвітлення структури (архітектури) програми у цьому розділі рекомендовано подати так звану діаграму класів за допомогою уніфікованої мови моделювання (UML). Нагадаємо, що уніфікована мова моделювання (Unified Modeling Language, UML) – це графічна мова для специфікації, візуалізації, конструювання і документування програмних систем. За допомогою UML можна розробити детальний план програмної системи, який відображатиме і концептуальні елементи системи (системні функції та бізнес-процеси), і особливості її реалізації (класи, схеми баз даних, програмні компоненти багаторазового використання тощо). На сьогодні існує низка інструментальних засобів для побудови UML-діаграм, зокрема Rational Rose Enterprise, Objec-teering, Magic Draw UML, MS Visio, Visual UML та ін. Наведені програмні засоби дозволяють легко будувати структуру програмної системи на стадії її проектування. Проте існує й інший спосіб побудови UML-діаграми за умови вже наявного програмного коду. Такою властивістю володіють більшість середовищ розробки за умови встановлення додаткових плагінів. Зокрема для середовища Eclipse розроблено низку плагінів, які володіють властивістю генерування UML-діаграм. До їх переліку можна віднести ObjectAid, EclipseUML, Papyrus та ін. Для середовища IntelliJ IDEA теж створені плагіни для зручної роботи з UMl діаграмами класів, зокрема найпопулярнішим є плагін PlantUML, який дозволяє створювати UML-діаграми за допомогою простого текстового синтаксису. Також в даному середовищі розробки є можливість генерувати UML діаграму класів на основі вже існуючого коду. Для цього достатньо відкрити Project Structure (Ctrl + Alt + Shift + S), вибрати вкладку Diagrams → Show Diagram, після чого IntelliJ IDEA побудує UML-діаграму на основі коду Java

Виконавці курсової роботи можуть самостійно обирати інструментальні засоби для побудови UML-діаграм.

Висновки (1 стор.) У висновках наводять основні результати та досягнення курсової роботи, описують основні результати роботи програми та її ефективність, надають рекомендації з практичного застосування розробленої програми.

Список використаної літератури. З метою запобігання академічного плагіату в кінці курсової роботи слід надати перелік усієї літератури, якою

виконавець користувався в процесі виконання курсової роботи. До переліку також слід включати усі електронні ресурси, які використані для реалізації задуму власного проекту.

Додатки (за необхідності). Якщо в ході реалізації проекту у виконавця виникають ідеї, що потребують виконання додаткових операцій не передбачених завданням на курсову роботу, то результати їх виконання можна виносити у додатки із обов'язковим відображенням у змісті до курсової роботи. Така практика вітається та буде врахована при оцінюванні результатів виконання курсової роботи.

3. ФОРМАТИ ВИКОНАННЯ КУРСОВОЇ РОБОТИ

Курсова робота може виконуватись як індивідуальна праця згідно виданого завдання (методичні вказівки: розділ 4) або як комплексна робота, виконання якої потребує командної організації у кількості 2-4 студентів.

Командне виконання курсової роботи можливе шляхом спільного вирішення комплексної проблеми в рамках написання студентської (курсантської) наукової роботи та/або реалізації студентських R&D проектів.

У випадку командного виконання курсової роботи група студентів, за погодженням із керівником, спільно працює над вирішенням комплексного завдання із самостійним розподілом обсягів необхідної роботи серед учасників. Оформлення пояснювальної записки, згідно визначеного завдання, проводиться індивідуально усіма учасниками команди за результатами виконання попередньо розподіленого обсягу індивідуальної роботи. Виключенням можуть бути лише розділи 1 та 4 де дозволяється представлення спільно напрацьованих алгоритму роботи програмної системи та діаграми її класів. За умови командної реалізації роботи можливе незначне відхилення від обов'язкового змісту основних розділів пояснювальної записки, що має бути погоджено з керівником.

Орієнтовні напрямки командного виконання курсової роботи подано у розділі 5 методичних вказівок. Процедура захисту курсової роботи виконаної у складі команди описана у розділі 7 методичних вказівках.

4. ІНДИВІДУАЛЬНІ ЗАВДАННЯ ДЛЯ ВИКОНАННЯ КУРСОВОЇ РОБОТИ

В якості індивідуального завдання студентові надається тематичний напрямок в якому необхідно реалізувати програмне рішення з унікальною логікою та дотриманням вимог, що представлені в розділі 3 методичних вказівок. Тематичний напрямок для індивідуального завдання обирається у відповідності до порядкового номеру в академічному журналі. Дозволяється обирати тематичний напрямок

(індивідуальне завдання) для виконання курсової роботи за окремим погодженням із керівником курсової роботи (у випадку самостійного виконання студентської наукової роботи, індивідуального проекту в рамках навчальної практики тощо). Ідея для реалізації програмної системи за одержаним тематичним напрямком обирається здобувачем освіти самостійно, за необхідності консультуючись із керівником роботи. Унікальність та креативність ідеї, а також підходів до її реалізації беруться до уваги комісією під час захисту курсової роботи.

Варіанти індивідуальних тематичних напрямів:

№ варіанту	***Тематичні напрямки для вирішення прикладного завдання***
1	Програма для ведення медичних карт пацієнтів
2	Симуляція роботи банку
3	Система бронювання готелю
4	Програма для управління автопарком
5	Онлайн-магазин з обліком товарів і замовлень
6	Менеджер завдань із пріоритетами
7	Система управління медичними записами пацієнтів
8	Програма для управління фінансами та бюджетом
9	Програма для створення та управління розкладом занять
10	Система бронювання авіаквитків
11	Система управління футбольною лігою
12	Симулятор міста з транспортною системою
13	Система управління рестораном із замовленнями
14	Система обліку та управління донорами крові
15	Система моніторингу рівня забруднення повітря та води
16	Система обліку та контролю медикаментів у лікарнях та аптеках
17	Менеджер особистих фінансів з категоризацією витрат
18	Система управління курсами та викладачами
19	Система управління кінотеатром
20	Програма для організації подорожей
21	Програма для оренди велосипедів або самокатів
22	Програма для організації спортивних змагань
23	Система для управління бібліотекою
24	Програма для обліку майна в компанії
25	Програма для складання та управління списком справ

Для кращого розуміння шляхів вирішення прикладного завдання за одержаним тематичним напрямком згідно варіанту розглянемо приклад: перелік основних класів та методів для декількох програмних систем.

Тематичний напрямок «Служба прокату авто»:

Основні класи:

- авто (марка, тип, рік випуску, категорія), список авто;
- замовник (дані замовника, категорія), список замовників;
- замовлення (замовник, авто, дата, час, тривалість замовлення), список замовлень.

Основні методи:

- ведення списку авто, пошук авто за різними ознаками, ведення довідника типів авто (вантажні, легкові тощо) та категорій (VIP, бізнес-клас тощо);
- ведення списку замовників, пошук замовників за різними ознаками, ведення довідника категорій замовників (постійні клієнти, VIP тощо);
- ведення списку замовлень, визначення вартості замовлення, пошук замовлення за різними ознаками тощо.

Тематичний напрямок «Деканат»:

Основні класи:

- студент (ПІБ, № заліковки), список студентів; список результатів успішності «студент-дисципліна»;
- дисципліна (назва, список форм контролю по семестрах (№ семестру, форми контролю (залік, іспит, курсова)), список дисциплін.

Основні методи:

- ведення списку студентів та списку результатів успішності, визначення відмінників, боржників, пошук студентів за різними ознаками;
- ведення списку дисциплін, визначення форм контролю навчання за номером семестру, пошук дисциплін за різними ознаками.

5. ГРУПОВІ ЗАВДАННЯ ДЛЯ ВИКОНАННЯ КУРСОВОЇ РОБОТИ

Завдання для командної реалізації курсової роботи обирається самоорганізованою групою студентів та погоджується з керівником курсової роботи. Завдання для командного виконання курсової роботи подається у вигляді проблеми та шляхів її вирішення. Склад команди, обсяги командної роботи, їх розподіл серед учасників, а також кінцевий результат та технології його досягнення погоджуються з керівником курсової роботи.

Тематичні напрямки для командної реалізації курсової роботи обираються відповідно до актуальних наукових, науково-прикладних або інженерних завдань,

якими займається випускова кафедра. До прикладу, командна робота може бути орієнтована на вирішення окремих частин загальної проблематики таких проектів як: «Розумний університет» (розробка системи замовлення довідок, електронний журнал, електронна залікова книжка, чат-бот для комунікації «студент-деканат», чат-бот «розклад» тощо); «Безпечне місто» (система обліку об'єктів укриття, перевірка відповідності протипожежного стану об'єкта за його параметрами тощо); «Безпечний будинок» та інші. Група студентів має право ініціювати власну тему (напрямок розробки) обґрунтувавши її актуальність та унікальність.

6. ВИМОГИ ДО ОФОРМЛЕННЯ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Пояснювальна записка до курсової роботи оформляється на аркушах паперу форматом А4 з одного боку. Заголовки обов'язкових розділів мають бути виділеними від іншого тексту роботи. Робочі мови для виконання курсової роботи: українська, англійська. Текстова частина має бути подана грамотно, без помилок у відповідності до наданих вказівок.

Текст у пояснювальній записі (окрім програмного коду) розміщується рівномірно за шириною сторінки з дотриманням відступів з усіх сторін по 20 мм. Шрифт для відображення текстової частини – TimesNewRomanCyr, розмір шрифту – 14, міжрядковий інтервал – 1,15. Шрифт для відображення програмного коду – Consolas, розмір шрифту – 12, міжрядковий інтервал 1. Сторінки пояснювальної записки нумеруються. Титульна сторінка вважається першою і не нумерується.

Обов'язкові розділи пояснювальної записки нумеруються. Після номера розділу подається його назва (наприклад: 1. Розробка та опис алгоритму). Вступ, висновки та список використаних джерел не нумеруються. Кожен розділ необхідно розпочинати з нової сторінки. Розділ за необхідності може складатися з підрозділів (номер підрозділу записується через крапку після номера розділу, наприклад: 1.1. Псевдокод алгоритму; 1.2. Блок-схема алгоритму).

Формули, на які здійснюється посилання, а також усі таблиці або рисунки повинні мати наскрізну нумерацію. Формули нумеруються у круглих дужках по лівому краю сторінки. Таблиці нумеруються на початку її подання та мають містити назву (Таблиця 1 – Відповідність методів та їх модифікаторів доступу). Рисунки нумеруються після їх представлення у тексті та мають містити назву (Рисунок 1 – Блок схема алгоритму програми). Окремі частини (скріни) програмного коду подаються як лістинги. Лістинг програми також має мати нумерацію та назву (Лістинг 1 – Рекурсивний метод визначення факторіалу числа).

Список літератури виконується згідно з вимогами стандартів. Література може бути розміщена за алфавітом або в порядку посилання на неї. Посилання в тексті

пояснювальної записки здійснюється вказанням номера джерела літератури у квадратних дужках. У списку необхідно наводити лише використану літературу та електронні ресурси.

Додатки також нумеруються або позначаються великими літерами у відповідності до їх порядкового представлення в алфавіті (наприклад: Додаток А; Додаток Б; … ; Додаток Д).

Пояснювальна записка подається на рецензування керівникові курсової роботи у папці-скоросшивач.

7. ЗАХИСТ КУРСОВОЇ РОБОТИ

Пояснювальну записку до курсової роботи подають керівнику на рецензування не пізніше ніж за два тижні до підсумкового контролю (диференційованого заліку або екзамену). Керівник перевіряє відповідність виконаної роботи поставленому завданню, наявність обов'язкових складових пояснювальної записки (постановка завдання та шляхи її вирішення, алгоритм, програмний код, коментарі до коду, UML-діаграма, висновки тощо) та працездатність окремих частин або в цілому програмної системи. За умови виявлення неточностей, помилок або невідповідність роботи завданню та методичним вказівкам – пояснювальна записка повертається на доопрацювання (*особливу увагу слід звернути на оформлення пояснювальної записки – розділ 6 методичних вказівок*). За умови відповідності роботи завданню та вимогам методичних вказівок, вона допускається до її презентації та захисту на оцінку. Дата захисту курсової роботи визначається керівником та узгоджується із навчальною групою та деканатом. Захисти робіт відбуваються у складі комісії до якої входить керівник курсової роботи та 2-3 викладачі випускової кафедри за напрямом розробки програмного забезпечення.

Процедура захисту індивідуальної курсової роботи проводиться в форматі презентації власного проекту. Для захисту роботи студент може використовувати підготовлену презентацію або демонструвати працездатність програми використовуючи інші зручні способи (інтегроване середовище розробки, на прикладі готового програмного продукту тощо).

Процедура захисту командної роботи проводиться у форматі спільного (колективного) представлення загальної проблеми та шляхів її вирішення. Лідер команди презентує загальну структуру та принцип роботи програмної системи використовуючи алгоритм та діаграму класів, а решта учасників почергово презентують свій індивідуальний вклад у вирішенні загальної проблеми, обґрунтовують прийняті рішення, демонструють працездатність коду тощо. Рівень

індивідуального виконання кожного учасника команди та його долучення до реалізації спільного проекту визначається під час процедури захисту та оцінюється окремо.

Під час презентації курсової роботи необхідно охарактеризувати ідею, яка закладена в проект, представити розроблений алгоритм та його характеристику. Для відображення структури та загальної архітектури програмної системи автор демонструє UML-діаграму. Основна увага під час захисту роботи має бути орієнтована на демонстрацію програмної системи. Автор (член команди) зупиняється на основних моментах, пояснюючи свої рішення (рішення команди), обґрунтовуючи засоби, методи і технології, які були використані під час створення програмного продукту.

В ході презентації комісія визначає відповідність виконаного завдання поставленим у методичних вказівках вимогам та підготовленість здобувача освіти до реалізації реальних проектів на практиці. За необхідності члени комісії можуть задавати уточнюючі або узагальнюючі питання, які допомагають визначити самостійність виконання проекту.

Презентація проекту (захист роботи) здійснюється студентом лише один раз. За незадовільної оцінки студенту видається нове завдання на курсову роботу та можливість повторно виконати проект.

Найбільш успішні роботи рекомендуються для подальшого розвитку в рамках реалізації студентських R&D проектів або студентської наукової роботи з їх представленням на Всеукраїнських конкурсах, ідеатонах, хакатонах тощо.

8. КРИТЕРІЇ ОЦІНЮВАННЯ КУРСОВОЇ РОБОТИ

Під час оцінювання курсової роботи до уваги приймаються три критерії, що в сумі формують загальну оцінку: якість реалізації програмного продукту ($К_{код}$); якість оформлення пояснювальної записки ($К_{записка}$); рівень презентації та захисту результатів власної роботи ($К_{захист}$). Оцінювання проводиться за 100-бальною шкалою за формулою $К = К_{код} + К_{записка} + К_{захист}$.

Оцінювання програмного продукту $К_{код}$ (діапазон балів 0-50):

- 40 – 50 балів – програмний продукт повністю відповідає усім вимогам методичних рекомендацій, відтворює функціональність за отриманим тематичним напрямком, характеризується стійкістю (захистом від некоректного введення даних, стійкістю роботи тощо);

- 30 – 40 балів – програмний продукт містить незначні неточності у реалізації принципів об’єктно-орієнтованого підходу, алгоритм або діаграма класів не в повній мірі відповідають поданому коду, або структура програмного продукту частково не

відповідає вимогам методичних вказівок. Загалом продукт є працездатним та відповідає тематичному напрямку;

- 20 – 30 балів – програмний продукт працездатний проте з суттєвими зауваженнями щодо стійкості його роботи. Архітектура та реалізація програмної системи не відповідає поставленим у методичних рекомендаціях вимогам (не більше ніж на 40%) або програмна реалізація частково не відповідає тематичному напрямку;

- 0 – 20 балів – програмний код не відповідає поставленим у методичних вказівках вимогам більше ніж на 40%, проте орієнтований на реалізацію ідеї згідно одержаного завдання. Програмна система не відтворює задекларованої функціональності та/або є частково не працездатно. Програмний продукт видає некоректні результати роботи та/або є не стійким при введенні різних вхідних даних;

- 0 балів: програмний продукт не представлено в повній мірі на захисті, він є не працездатним, або запозичений з відкритих ресурсів.

Оформлення пояснювальної записки $K_{записка}$ (діапазон балів 0-30):

- 25 – 30 балів: пояснювальна записка в повній мірі відповідає вимогам розділів 2 «Обов'язковий зміст основних розділів пояснювальної записки» та 6 «Вимоги до оформлення пояснювальної записки» методичних вказівок, містить увесь необхідний ілюстративний та графічний матеріал, подано усі лістинги програмних модулів, коментарі змістовно та лаконічно відтворюють функціонал окремих методів;

- 20 – 25 балів – пояснювальна записка відповідає вимогам розділів 2 «Обов'язковий зміст основних розділів пояснювальної записки» та 6 «Вимоги до оформлення пояснювальної записки» методичних вказівок не менше ніж на 80%, має незначні зауваження щодо повноти забезпечення ілюстративним та графічним матеріалом, або до якості його виконання, коментарі не в повній мірі відтворюють функціонал окремих модулів програм або мають неточності чи невідповідність поданим лістингам;

- 10 – 20 балів – робота має значні зауваження щодо якості оформлення та відповідності вимогам розділів 2 «Обов'язковий зміст основних розділів пояснювальної записки» і 6 «Вимоги до оформлення пояснювальної записки» методичних вказівок, проте всі обов'язкові складові (розділи) подано у пояснювальній записці. В пояснювальній записці наявні не всі лістинги програмного коду, коментарі сформульовано не зрозуміло та/або у невідповідності поданим лістингам;

- 0 – 10 балів – пояснювальна записка не містить усіх необхідних компонентів передбачених вимогам розділу 2 «Обов'язковий зміст основних розділів пояснювальної записки» методичних рекомендацій. Програмний код наведено не в повній мірі у результаті чого не можливо перевірити його працездатність.

Графічний та ілюстративний матеріал відсутній або не відповідає програмному коду чи завданню на курсову роботу;

- 0 балів – пояснювальну записку не подано на рецензування, або вона не відповідає поставленому завданню та ключовим вимогам методичних вказівок.

Захист курсової роботи $К_{захист}$ (діапазон балів 0-20):

- 15 – 20 балів – здобувач освіти демонструє вільне володіння матеріалом курсової роботи, оперує термінами та демонструє розуміння основних понять об'єктно-орієнтованого програмування, презентована ідея та шляхи її реалізації є унікальними та креативними, вірно відповідає на усі додаткові запитання;

- 10 – 15 балів – здобувач відповідає основним критеріям на оцінку (15-20) проте під час відповідей на додаткові або уточнюючі питання допускає певні неточності;

- 0 – 10 балів – під час презентації допускаються певні неточності або невідповідності поданого в пояснювальній записці матеріалу, при наданні додаткових або уточнювальних питань відповіді є не вірними і не точними, що може свідчити про виконання роботи з використанням готових алгоритмів без їх творчого опрацювання;

- 0 балів – здобувач не може пояснити поданий на захист матеріал та коректно його прокоментувати, що може свідчити про несамостійність виконаної роботи.

Здобувач, що отримав 0 балів з першого або третього критерію, вважається таким що не виконав завдання на курсову роботу та обирає нову тему з її подальшим перезахистом.

Загальна оцінка за курсову роботу з 100-бальної шкали переводиться у національну шкалу у відповідності до поданої таблиці:

Сума балів	100-91	90-81	80-71	70-61	60-51	50-35	34-0
Оцінка за нац.шкалою	«відмінно»	«добре»	«добре»	«задо-вільно»	«задо-вільно»	« не задо-вільно»	«не задо-вільно»

9. ДОТРИМАННЯ АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ

Пояснювальна записка та окремі фрагменти програмного коду можуть проходити перевірку на наявність ознак академічного плагіату відповідно до відкритих систем для он-лайн перевірки текстових запозичень.

Встановлення факту академічної недоброчесності у курсових роботах здобувачів освіти є підставою для повторного написання зазначеної письмової роботи за іншим варіантом.

Перевірка роботи на наявність академічного плагіату проводиться на випусковій кафедрі після її попереднього рецензування керівником або за умови

виникнення підозри щодо можливого факту академічної недоброчесності. Перевірка здійснюється керівником курсової роботи або секретарем кафедри.

Рекомендовані показники оригінальності тексту у пояснювальній записці до курсової роботи:

– понад 70% – текст пояснювальної записки курсової роботи є оригінальним (несуттєвий обсяг запозичень);

– від 60 до 70% – оригінальність тексту пояснювальної записки курсової роботи задовільна (незначний обсяг запозичень), проте слід переконатися у наявності і правильному оформленні цитувань та посилань на використані джерела;

– від 40 до 60% – курсова робота приймається до захисту після доопрацювання автором та наявності і вірного оформлення цитувань та посилань на використані джерела, оскільки має значний обсяг запозичень;

– менше 40% – письмова робота до захисту не приймається, оскільки має суттєвий обсяг запозичень, що трактується як плагіат.

Якщо за результатами перевірки на наявність академічного плагіату встановлено коректність посилань на використані джерела, то робота допускається до захисту та вважається такою, що пройшла внутрішнє рецензування.

За для забезпечення можливості перевірки пояснювальної записки та/або окремих частин програмного коду на наявність академічного плагіату, записка та програмний код (архів усіх класів) здаються на випускову кафедру в електронному вигляді шляхом завантаження у відповідну категорію електронного освітнього середовища «Віртуальний університет».

СПИСОК ЛІТЕРАТУРИ

1. Програмування та алгоритмічні мови програмування : методичні вказівки для виконання курсового проекту для студентів галузі знань 12 «Інформаційні технології» спеціальності 122 «Комп'ютерні науки», 124 «Системний аналіз» / уклад.: І. В. Назарчук, Г. Г. Шварко – К.: НТУУ «КПІ», 2017. – 32 с.

2. Об'єктно-орієнтоване програмування : методичні вказівки до виконання курсової роботи / уклад.: Г. В. Красовська – К.: КНУБА, 2011. – 28 с.

3. Дизайн-патерни – просто, як двері : підручник / Андрій Будай : «Developer's SUCCESS», 2012. – 90 с.

4. Об'єктно-орієнтоване моделювання програмних систем : навчальний посібник. – Львів: Видавничий центр ЛНУ імені Івана Франка, 2007. – 108 с.

5. Основи програмування на Java : методичні вказівки до лабораторного практикуму та самостійної роботи, частина перша / Бивойно П. Г., Бивойно Т. П. – Чернігів: ЧНТУ, 2014. – 79 с.

6. Head First Java (изучаем Java) : пер. с англ. / Kathy Sierra, Bert Bates. – М. : «Эксмо», 2012. – 718 с.

Додаток А

Державна служба України з надзвичайних ситуацій

Львівський державний університет безпеки життєдіяльності

Кафедра ІТтаСЕК

Курсова робота

З дисципліни: “Об’єктно-орієнтоване програмування”

на тему:

“Базові принципи об’єктно- орієнтованого програмування”

Індивідуальне завдання: “ ”

Виконала:
ст./курсант групи КН21к

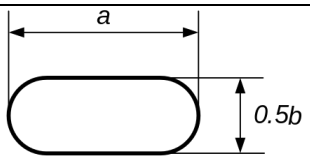
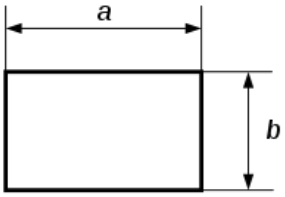
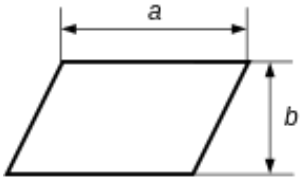
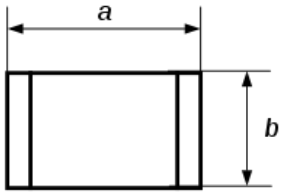
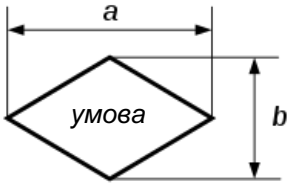
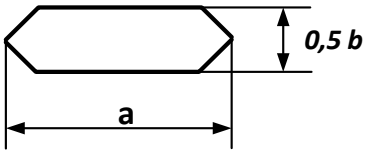
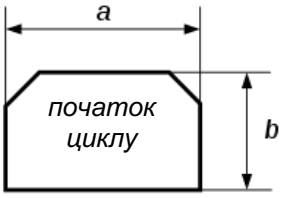
_____

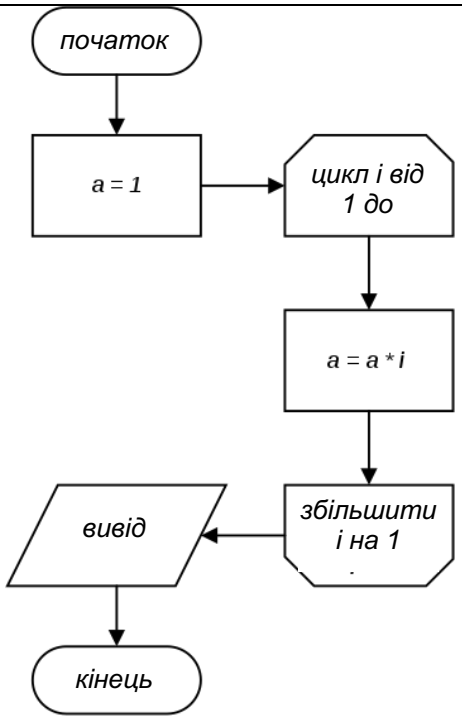
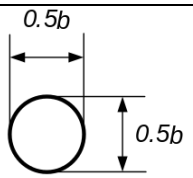
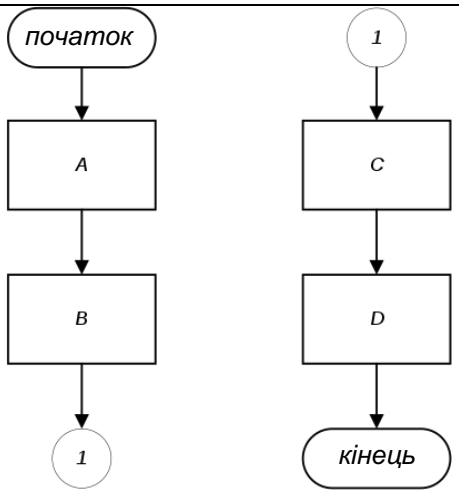
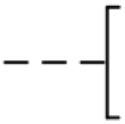
Прийняла:
заст.нач. каф. ІТтаСЕК
доктор філософії
Юлія НАЗАР

Львів 2025

Додаток Б

Основні елементи блок-схеми алгоритму

Початок/кінець алгоритму (вхід з зовнішнього середовища та вихід в зовнішнє середовище)	
Дія, операція (арифметична операція, оголошення та ініціалізація змінної, обробка даних тощо)	
Ввід/вивід даних (символ не визначає носія даних). Елемент відображає перетворення у форму придатну для обробки даних або відображення результатів обробки	
Функція / метод, який складається з декількох операцій. Елемент може відображати виконання певної підрограми	
Умовне розгалуження (if / else). Символ відтворює функцію яка має один вхід і n виходів, лише один з яких може бути виконаний у випадку, якщо умова справджується	
Цикл з параметром. Через крапку з комою вказуються ім'я змінної з початковим значенням, граничне значення параметра (або умова виконання циклу), крок зміни параметра (ітерація)	
Розширене представлення циклу. Операції, що виконуються всередині циклу, розміщуються між елементами. Умови циклу і збільшення записуються всередині символу початку або кінця циклу - в залежності від типу організації циклу. *Часто для зображення на блок-схемі циклу замість цього символу використовують символ рішення, вказуючи в ньому умову, а одну з ліній виходу замикають вище в блок-схемі*	 

(перед операціями циклу).	
Приклад використання циклу	
Сполучний символ. Символ відображає вихід з однієї частини алгоритму та вхід до іншої частини алгоритму (у випадку надмірно великої структури алгоритму, який не може бути розміщеним а межах одного листа)	
Приклад застосування з’єднувача	
Коментар	

Паралельні дії. Синхронізація двох або більше паралельних задач. Паралельні дії можуть застосовуватись для асинхронних операцій або операцій, які не залежать одна від одної