Національний університет “Львівська політехніка”
Кафедра автоматизованих систем управління

Методичні вказівки до курсової роботи з дисципліни “Прикладне програмування” версія від 20.11.2021 р.

(автор – Ю.В. Цимбал)

1. Основні цілі курсової роботи

- навчитись формулювати задачі для програмних проектів, розкладати їх на підзадачі, вибирати методи та алгоритми для їх розв’язання;
- навчитись визначати, порівнювати, вибирати і використовувати сучасні засоби та технології програмування для розв’язування конкретних прикладних задач;
- навчитись реалізовувати, відлагоджувати та розгортати програмні проекти з використанням сучасних технологій розробки;
- навчитись описувати програмні проекти, як кінцеві результати, так і процес виконання.

2. Вимоги до змісту та структури курсової роботи

Зміст **пояснювальної записки** до роботи може бути таким:

Титульна сторінка.
Завдання на курсову роботу.
Зміст.
Вступ.
1. Постановка задачі.
2. Огляд літератури.
3. Опис етапу проектування.
4. Програмне рішення.
5. Опис проведених експериментів.

Висновки.
Список використаних джерел.
Додатки.

Рекомендований обсяг пояснювальної записки – 20-25 сторінок (без додатків). Викладення матеріалу має бути збалансованим, розмір кожного розділу не має перевищувати 20% загального обсягу роботи.

Вимоги до оформлення роботи наступні:

1. Шрифт Times New Roman, розмір шрифту 14. Заголовки розділів – 16, bold. Кожен розділ починати з нової сторінки.
2. Міжрядковий інтервал – 1.5 лінії.
3. Абзац – 1.25 см.
4. Вирівнювання тексту – по ширині (Justify)
5. Зображення (рисунки) мають бути хорошої якості, з контрастним читабельним текстом на них і бажано світлим фоном, і підписані (наприклад, “ Рис. 1. Діаграма випадків використання”). Великі зображення, наприклад, діаграму класів можна розбити на декілька менших, або розмістити у додатку на альбомній сторінці (Landscape).

3. Вимоги до виконання окремих розділів пояснювальної записки курсової роботи

У **вступі** слід описати мету, яка ставиться в роботі, її актуальність, можливості вирішення сучасними програмними засобами, майбутнє застосування результатів роботи на практиці. Вступ можна написати після завершення роботи.

Постановка задачі – це чітке визначення функціональних можливостей програмної розробки, вимоги до неї з боку користувача, зокрема, тип програмного рішення – настільний (desktop), веб-орієнтований або мобільний застосунок.

Огляд літератури розкриває особливості предметної області, для якої розробляється програмний проект, визначає місце проекту серед схожих за призначенням інших проектів, знаходить близькі прототипи та підходи до розв’язання аналогічних задач. Включає порівняльний аналіз можливих варіантів розв’язання поставленої задачі та вибір оптимального варіанту.

Мета **етапу проектування** – максимально спростити задачу, розбиваючи її на під задачі, розробити загальну архітектуру та окремі частини програмної системи. Для опису результатів можна використати UML-діаграми, зокрема, діаграма прецедентів (Use Case), діаграма компонентів, діаграми класів та схема бази даних, які описують будову програмної системи та її функціонування і формують об’єктну модель системи (Докладніше про діаграми - в окремому розділі цих методичних вказівок).

Програмне рішення – це докладний опис Вашої роботи над програмною реалізацією поставлених задач. Зокрема, звернути увагу на розробку графічного інтерфейсу користувача і взаємодію з базою даних. Також зазначити, які бібліотеки (крім стандартних) та приклади програм, написані іншими програмістами, були використані в роботі. При цьому можливість такого використання має бути дозволена з погляду авторського права і погоджена з керівником роботи.

Експериментальна частина включає опис послідовності дій, потрібних для використання програмної розробки – з погляду користувача (як завантажити і використовувати), адміністратора (як розгорнути на новому комп'ютері або сервері), програміста (як продовжити роботу над проектом за іншим комп'ютером). Також навести невеликі екранограми (знімки екрану роботи) з прикладами виконання різноманітних практичних задач.

Список використаних джерел має містити не менше 10 позицій (у т.ч. іноземних та з мережі Інтернет) з обов'язковими посиланнями у тексті пояснювальної записки.

Висновки це підсумок проведеної роботи із узагальненим оглядом задачі, яка була вирішена, перспективами розвитку програмної системи, коротким описом набутих Вами знань та вмінь.

Додатки містять тексти програмних модулів, таблиці вхідних та вихідних даних та великі екранограми, які ілюструють інтерфейс користувача розробленої програми.

4. Захист роботи

Перед захистом роботи керівник розглядає програмну та описову частину роботи, зокрема, перевіряє їх оригінальність. Для цього попередньо розмістіть **програмний код** на відкритому для керівника і закритому для інших репозиторії, наприклад, GitHub, а **пояснювальну записку** - на сторінці курсової роботи у Віртуальному навчальному середовищі.

Критерії оцінювання роботи:

1. 50% – програмна частина;
2. 50% – описова частина.

Захист може бути разом із презентацією Вами роботи. Тоді критерії оцінювання є такими:

1. 40% – програмна частина;
2. 40% – описова частина.
3. 20% – доповідь із презентацією

Оцінка виставляється при особистому спілкуванні з автором. Тобто важливими є як виконання та письмове оформлення роботи (50 %), так і усна компонента (50 %). Можливо, що захист роботи буде відбуватися перед комісією з трьох викладачів кафедри.

ДОДАТОК. Діаграми реалізації програмного забезпечення

Це діаграми, за допомогою яких описується архітектура певної програмної системи.

Діаграма класів

Діаграма класів (Class diagram) визначає склад класів об’єктів і їх зв’язків. Крім того, діаграма класів може включати коментарі та обмеження. Обмеження можуть неформально задаватися на природній мові. Діаграма задається зображенням, на якому класи позначаються поділеними на три частини прямокутниками, а зв’язки – лініями, що з’єднують прямокутники. Це відповідає візуальному зображенню понять і зв’язків між ними. Верхня частина прямокутника – обов’язкова, в ній записується назва класу. Друга і третя частини прямокутника визначають відповідно список атрибутів класу (полів) і операцій (методів), що можуть мати такі специфікатори доступу (рис. 1):

- *public* (загальний) позначає операцію або атрибут, доступ до яких здійснюється з будь-якої частини програми будь-яким об’єктом системи;
- *protected* (захищений) позначає операцію або атрибут, доступ до яких здійснюється об’єктами того класу, в якому вони оголошені, або об’єктами класів-нащадків,
- *private* (приватний) позначає операцію або атрибут, доступ до яких здійснюється тільки об’єктами того класу, в якому вони визначені.

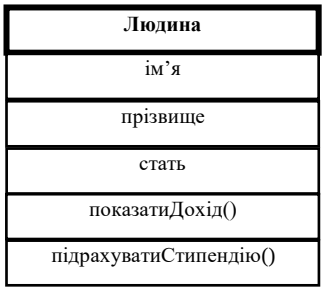

Рис. 1. Приклад наочного подання класу у UML

Користувач може визначати специфічні для нього атрибути. Під операцією розуміють сервіс, який екземпляр класу може виконувати, якщо до нього буде спрямовано відповідний

виклик. Операція має назву і список аргументів. Для неї може також задаватися тип значення, яке вона повертає.

Класи можна побудувати з наступними відношеннями або зв'язками, які позначаються різними символами на діаграмі.

Асоціація – взаємна залежність між об'єктами різних класів, кожен з яких є її рівноправним учасником. Асоціації є механізмом, який надає об'єктам змогу обмінюватися даними між собою.

Асоціації можуть виконувати роль, яка визначає призначення асоціації і може бути одно- чи двосторонньою. Вона може позначати кількість екземплярів об'єктів кожного класу, які беруть участь у зв'язку (0 – якщо жодного, 1 – якщо один, N – якщо багато).

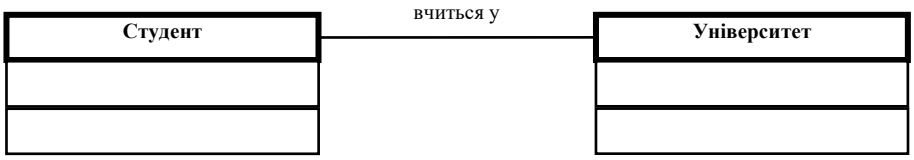

Рис. 2. Приклад асоціації у UML

У UML *узагальнення* між двома класами розташовує їх у вузлах ієрархії, яка відповідає концепції наслідування класу-нащадка від базового класу.

Наслідування є однією з фундаментальних рис об'єктно-орієнтованого програмування, коли клас "отримує" всі атрибути і операції класу, нащадком якого він є, і може перевизначати (змінювати) деякі з них, а також додавати власні атрибути і операції.

У UML узагальнення (наслідування) показують у вигляді лінії зі стрілочкою, що поєднує два класи і яку спрямовано до базового класу.

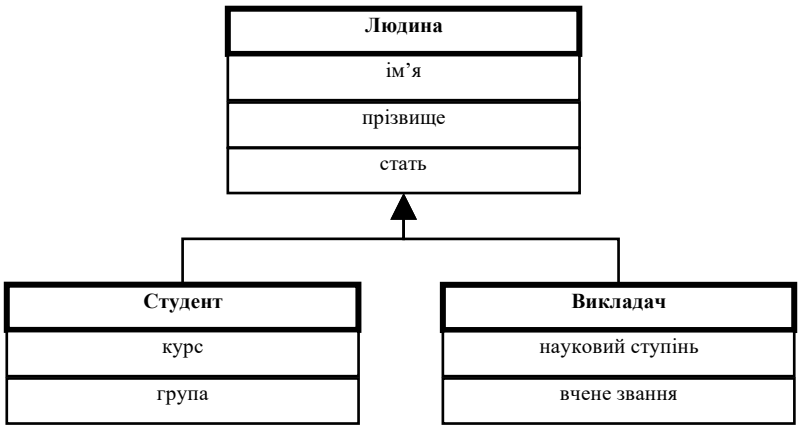

Рис. 3. Наслідування класів у UML

Залежність – відношення між класами, при якому клас-клієнт може використовувати певну операцію або сервіс іншого класу; класи можуть бути зв'язані відношеннями

задежності, якщо один клас трансформується в іншій внаслідок виконання певного процесу життєвого циклу програми.

Агрегація є особливим типом асоціації, за якого два класи, які беруть участь у зв'язку не є рівнозначними, вони мають зв'язок типу "ціле-частина" (приклад: "автомобіль-фара").

Композиції – це асоціації, які відповідають дуже сильній агрегації. Це означає, що у композиціях також маємо справу з співвідношенням "ціле-частина", але тут зв'язок є настільки сильним, що ціле не може існувати без частини (приклад: "автомобіль-двигун").

Діаграма компонентів

Цей тип діаграм призначений для розподілення класів по компонентах під час фізичного проектування системи.

В UML – це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними. Діаграма компонентів відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватися. На діаграмах компонентів зображуються входження класів і об'єктів у програмні компоненти системи (модулі, бібліотеки, бази даних і т.д.). Приклади діаграми наведено на рис. 4.

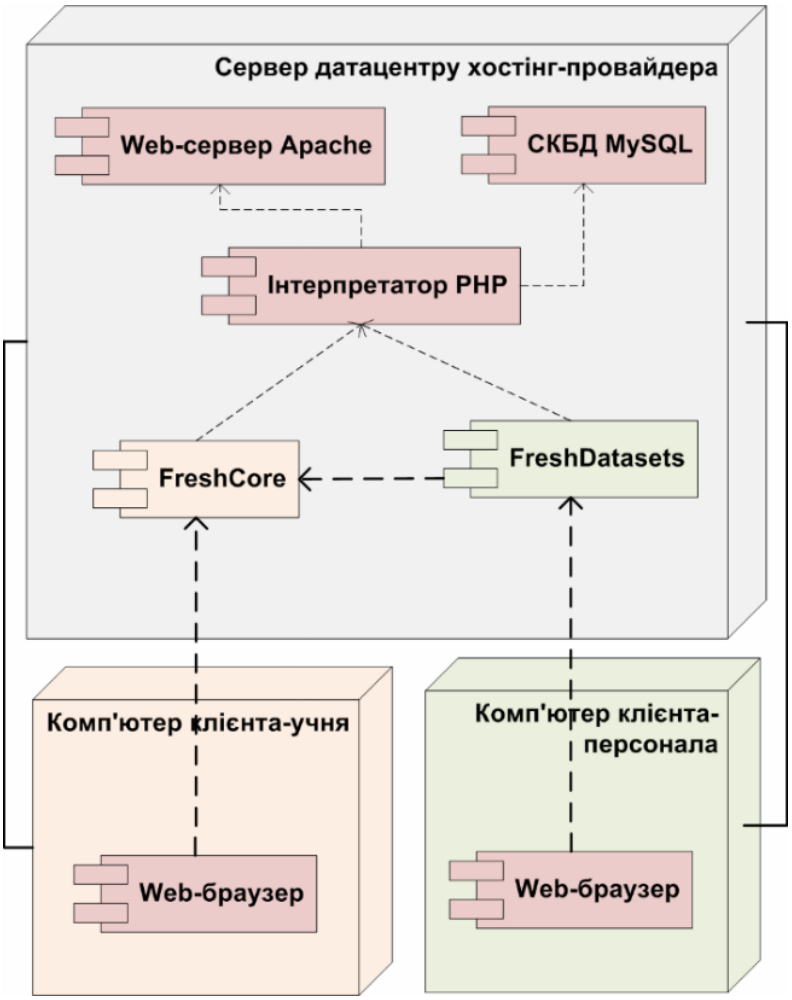

Рис. 4. Приклад діаграми компонентів