ЛАБОРАТОРНА РОБОТА 3

РОЗРОБКА UML-ДІАГРАМ ЛОГІЧНОГО РІВНЯ ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ: МОДЕЛЮВАННЯ СТАТИЧНИХ АСПЕКТІВ

Мета роботи – вивчити методику побудови UML-діаграм логічного рівня проектування ПЗ, які застосовуються для моделювання статичних аспектів побудови ПЗ.

3.1. Загальні відомості

Після виділення прецедентів, побудування діаграми послідовності та діаграми стійкості (див. л/р 2) можна приступати до більш детального проектування системи. На логічному рівні проектування ПЗ визначаються характеристики тих програмних компонентів, які входять до складу архітектури системи. При цьому мають бути розглянуті як статичні, так і динамічні аспекти цих компонентів. Для моделювання статичних (або структурних) аспектів ПЗ застосовуються такі UML-діаграми як [2-4]:

- діаграми класів (class diagram);
- діаграми об’єктів (object diagram).

Для моделювання динамічних аспектів ПЗ (тобто взаємодії компонентів) використовуються:

- діаграми станів (state chart diagram);
- діаграми діяльності (activity diagram);
- діаграми кооперації (communication diagram),

(про ці діаграми див. більш детально у л/р 4).

Діаграма класів (class diagram)

Клас (class) у мові UML потрібний для позначення сукупності об’єктів, які мають однакову структуру, поведінку та зв’язки з об’єктами з інших класів. Графічно клас зображується у вигляді прямокутника, який додатково має бути розділений горизонтальними лініями на розділи або секції. У цих розділах можуть вказуватися назва класу, атрибути (властивості) та операції (методи) цього класу.

Обов’язковим елементом позначення класу є його ім’я (name), яке має бути унікальним у межах відповідного пакета. На початкових етапах розробки діаграми

окремі класи можуть позначатися простим прямокутником з указівкою лише його імені. У міру опрацювання окремих компонентів діаграми класів доповнюються атрибутами та методами.

У другій зверху секції прямокутника класу записуються його атрибути (attributes). Кожному атрибуту класу відповідає окремий рядок тексту, який складається з квантора видимості атрибуту, імені атрибуту, його кратності, типу значень атрибуту і, можливо, його певного значення за умовчанням (by default).

Квантор видимості може набувати одне з чотирьох можливих значень: public, protected, private, package і відображується за допомогою спеціальних символів (їх набір залежить від версії відповідного CASE-засобу, наприклад, Visual Paradigm). Атрибут типу public є доступним або видимим з будь-якого іншого класу всієї програмної системи. Атрибут типу protected є доступним тільки для всіх класів, які є нащадками цього класу. Атрибут типу private є недоступним для всіх інших класів без виключення. І, нарешті, атрибут типу implementation є доступним тим класам, що входять до складу пакета (package), в якому знаходиться цей клас.

Операції класу є певними сервісами обробки даних, вони подібно атрибутам та-кож мають унікальні імена та квантори видимості (public, protected, private, implementation). На рис. 3.1 показано фрагмент діалогу в середовищі Visual Paradigm, в якому побудовано клас User, що має атрибути LastName, FirstName типу public, атрибут Age – типу protected, та атрибут Status – типу private. Відповідно, цей клас має також такі операції, як GetStatus() – типу protected, UpdateStatus() – типу private, та PrintStatus() – типу package.

Базовими типами відношень або зв’язками між класами в мові UML є: відношен-ня залежності (dependency relationship), відношення асоціації (association relationship), відношення узагальнення (generalization relationship) та відношення агрегації (aggregation relationship). Кожен з них має власне графічне позначення на діаграмі класів.

Відношення залежності у загальному випадку вказує на деякий семантичний зв’язок між певними класами, який не є, відповідно, жодним з інших типів зв’язків (див. вище).

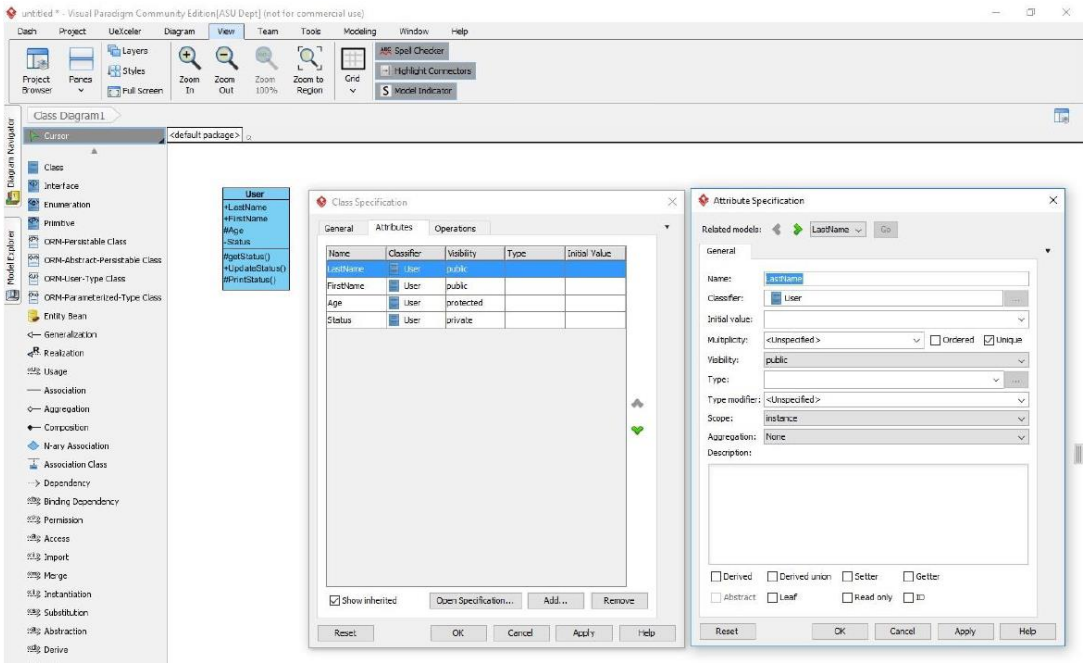

Рисунок 3.1 – Інтерфейс Visual Paradigm для визначення властивостей класу

Наприклад, відношення залежності використовується у такій ситуації, коли деяка зміна одного класу моделі може потребувати зміни певних параметрів (атрибутів і/або операцій) залежного від цього класу. Графічно це позначається пунктирною лінією між відповідними класами зі стрілкою на одному з її кінців, при цьому стріл-ка направлена від залежного класу, або від клієнта (client) до незалежного класу або класу-постачальника (supplier). На рис. 3.2 показано два класи: клас User Profile – це залежний клас-клієнт та клас Application – є клас-постачальник у цій залежності.

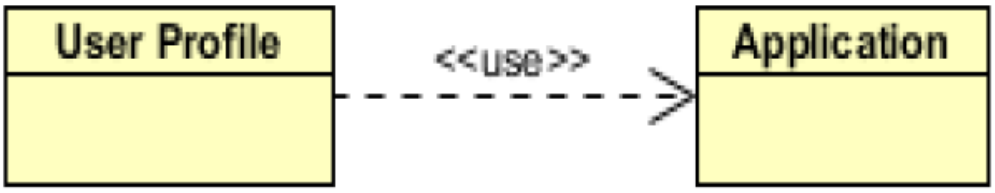

Рисунок 3.2 – Приклад відношення залежності

Відношення асоціації відповідає наявності деякої структурної залежності між класами. Воно позначається суцільною лінією з додатковими спеціальними символами, які характеризують окремі властивості конкретної асоціації, наприклад назва асоціації, її кратність та напрямок. Приклад асоціації наведено на рис. 3.3, де вона

визначена між двома класами: класом User і класом Account. Вони зв’язані між собою бінарною асоціацією subscribe, назва якої вказана поряд з лінією асоціації.

Рисунок 3.3 – Графічне зображення відношення бінарної асоціації

Додатково на діаграмі може бути кратність входження до відношення окремих класів. Вона позначається у вигляді інтервалу цілих чисел, який записується поряд з кінцем відповідної асоціації і може приймати наступні основні значення: ОДИН ДО ОДНОГО – 1:1, ОДИН ДО БАГАТЬОХ – 1:*, БАГАТО ДО БАГАТЬОХ – *:*. Крім того, є можливість вказувати мінімальне та максимальне значення кратності входження, наприклад, позначення кратності входження у вигляді: «0...1» – «1...*» , означає, що можливо один (або жоден) екземпляр класу на лівому боці відповідної асоціації залежить від щонайменше одного, або від декількох екземплярів класу на правому боці цієї ж асоціації.

Відношення агрегації має місце між декількома класами у тому випадку, якщо один з класів є таким, що включає в себе як складові частини деякі інші класи. Це відношення має фундаментальне значення для опису структури складних систем, оскільки застосовується для представлення системних взаємозв'язків типа «ціле – частина» («part – of»). Це відношення за своєю суттю описує декомпозицію або розбиття складної системи на простіші складові частини (компоненти), які також можуть бути піддані подальшій декомпозиції, якщо у цьому виникне необхідність. Ти-повим прикладом агрегації є взаємозв’язок, який має місце між об’єктом «Автомо-біль» і такими його компонентами, як «Двигун», «Шасі», «Кузов».

Графічно відношення агрегації зображується суцільною лінією, один з кінців якої є незабарвленим усередині ромбом. Цей ромб указує на той з класів, який є агрего-ваним класом, а інші класи є його частинами.

Додатково на діаграмі агрегації також може бути вказана кратність входження відповідних класів (рис. 3.4).

Окремим випадком відношення агрегації є композиція (composition), яка застосо-вується для моделювання класів, коли певні компоненти (частини) завжди знахо-дяться всередині цілого класу і специфіка взаємозв'язку між ними полягає у тому, що частини не можуть розглядатися у відриві від цілого, тобто зі знищенням цілого знищуються і всі його складові частини.

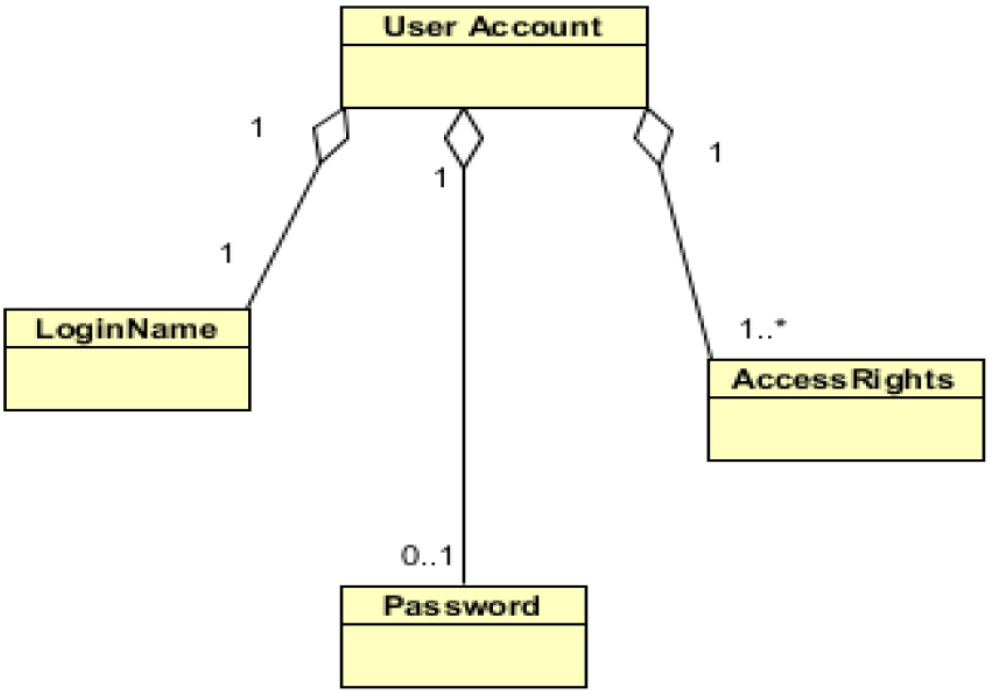

Рисунок 3.4 – Приклад діаграми класів для відношення агрегації

Прикладом цього відношення може бути вікно ділового інтерфейсу користувача певної програми, яке може складатися з заголовка, деяких кнопок управління, двох смуг прокручування та основної робочої області. З логічної точки зору подібне вікно є класом, а його компоненти є як підкласами, так і атрибутами або властивостями вікна. Графічно відношення композиції зображується суцільною лінією, один з кін-ців якої є зафарбованим усередині ромбом. Цей ромб вказує на той з класів, який є класом-композицією (рис. 3.5).

Відношення узагальнення є відношенням між деяким загальним класом – пред-ком (parent class), або суперкласом, і більш специфічними класами – нащадком (child class). Це відношення може використовуватися для представлення взаємозв'язків між пакетами, класами, варіантами використання та іншими елементами мови UML.

Щодо діаграми класів відношення узагальнення описує ієрархічну структуру класів і вказує на спадкоємність їх властивостей і поведінки. При цьому передбачається, що клас-нащадок володіє всіма властивостями і поведінкою класу-предка, а також має свої власні властивості і поведінку, які відсутні у класу-предка.

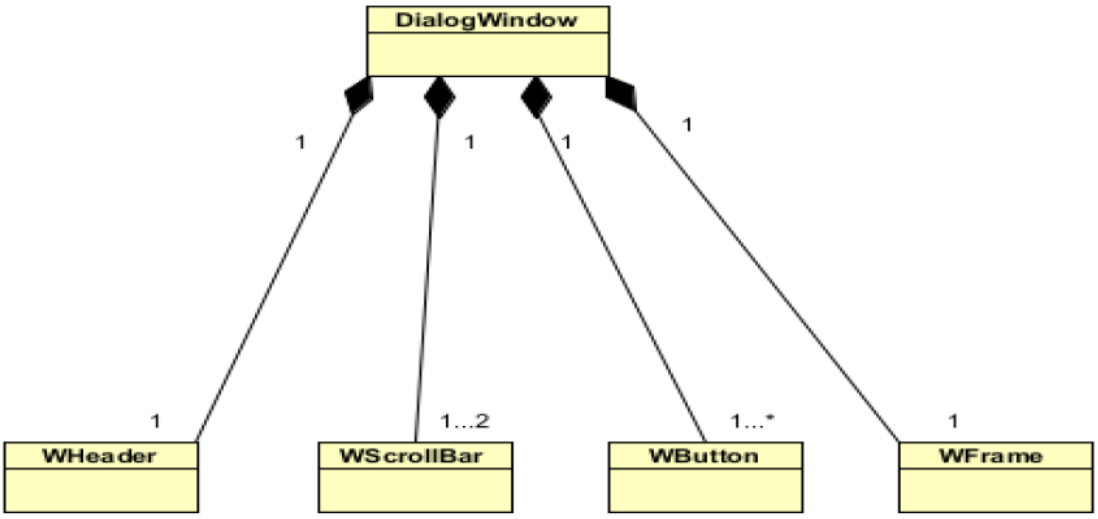

Рисунок 3.5 – Приклад діаграми класів для відношення композиції

На діаграмах відношення узагальнення позначається суцільною лінією з трикутною стрілкою на одному з кінців (рис. 3.6). Стрілка вказує на клас-предок або суперклас, а її відсутність – на більш спеціальний клас (клас-нащадок або підклас).

Діаграми об'єктів

Об'єкт є окремим представником певного класу, і кожен об'єкт має унікальну власну назву та конкретні значення своїх атрибутів. Для позначення об'єктів на діаграмі використовується той же символ, що й для діаграми класів, тільки з позначенням назви об'єкта у форматі назва об'єкта : назва класу, а кожен атрибут об'єкту має своє конкретне значення, наприклад, для об'єкту класу User, який має назву «Ivanov» та конкретні значення атрибутів LoginName = «Ivan», PW = «1023477892», (рис. 3.7).

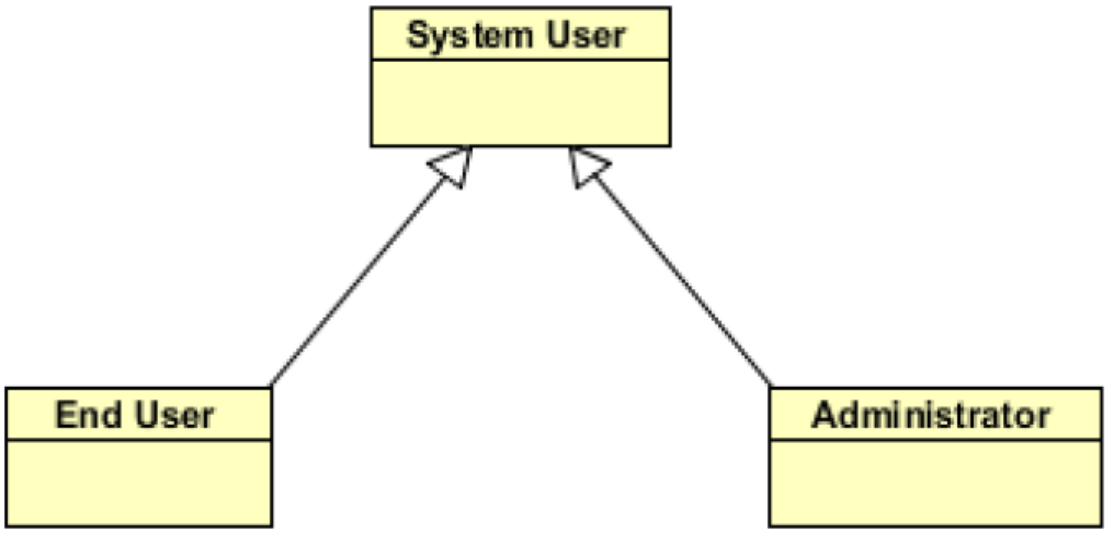

Рисунок 3.6 – Приклад діаграми класів для відношення узагальнення

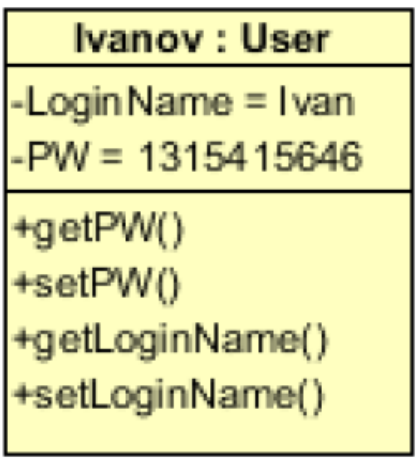

Рисунок 3.7 – Приклад зображення окремого об’єкта на діаграмі

Відношення між об’єктами на діаграмі мають ті ж самі типи, що й для відповідних класів, з тією різницею, що вони завжди позначаються суцільними лініями. Діаграми об’єктів доцільно використовувати для тих випадків у процесі розробки ПЗ, коли конкретні значення даних щодо їх атрибутів та зв’язків дають можливість більш ефективно врахувати їх особливості при подальшій програмній реалізації, тестуванні та супроводу системи. Об’єкти застосовуються також при побудові діаграм комунікації (communication diagram) – див. л/р 4.

3.2. Виконання лабораторної роботи

1. Розробити детальну діаграму класів для обраної предметної області.

2. Для окремих (найбільш важливих) її фрагментів побудувати відповідні діаграми об’єктів.

3.3. Результати виконання роботи та оформлення звіту

Зміст звіту:

1) титульний аркуш;

2) короткі теоретичні відомості;

3) результати практичного виконання, висновки та рекомендації.

Контрольні запитання

1. Які аспекти мають бути враховані при проектуванні програмної системи на логічному рівні? Які типи UML-діаграм застосовуються для цього?

2. Назвіть основі графічні елементи, що мають місце для побудови діаграми класів, поясніть їх призначення.

3. Які типи відношень існують між окремими класами?

4. У чому полягає семантична різниця між відношенням залежності та відношенням асоціації? Наведіть конкретні приклади цих відношень.

5. У чому полягає семантична різниця між відношенням агрегації та відношенням узагальнення? Наведіть конкретні приклади цих відношень.

6. Що означає відношення композиції? Чим воно відрізняється від відношення агрегації?

7. Що таке інтерфейс? Для чого він може бути застосований при розробці діаграми класів?

8. У чому полягає різниця між діаграмою класів та діаграмою об’єктів? Для чого доцільно використання діаграм об’єктів?