ТЕХНОЛОГІЇ СТВОРЕННЯ ПРОГРАМНИХ ПРОДУКТІВ

Основи UML:
діаграма класів

Теоретичний блок
до лабораторного заняття №3

3. Діаграми класів

Центральне місце в методології ООАП займає розробка логічної моделі системи у вигляді діаграми класів. Діаграма класів відображає різні взаємозв'язки між окремими елементами, такими як об'єкти і підсистеми з атрибутами і операціями, а також описує їх внутрішню структуру і відношень.

Клас (class) — абстрактний опис множини однорідних об'єктів, що мають однакові атрибути, операції і відношення з об'єктами інших класів.

3.1. Елементи графічної нотації діаграми класів

Графічно клас в мові UML зображується у вигляді прямокутника, в якому має бути вказане ім'я відповідного класу (рис. 3.1, а). В процесі опрацювання окремих компонентів діаграми опис класів доповнюється атрибутами (рис. 3.1, б) і операціями (рис. 3.1, в). Четверта секція (рис. 3.1, г) не обов'язкова і служить для розміщення додаткової інформації довідкового характеру, наприклад, про виключення або обмеження класу. Передбачається, що остаточний варіант діаграми містить якнайповніший опис класів, які складаються з трьох або чотирьох секцій.

Навіть якщо секції атрибутів і операцій порожні, в позначенні класу вони мають бути виділені горизонтальною лінією, для того, аби відрізнити клас від інших елементів мови UML. Приклади

Рис. 3.1. Варіанти графічного зображення класу на діаграмі класів

графічного зображення конкретних класів наведені на рис. 3.2. У першому випадку для класу «Коло» (рис. 3.2, а) вказані лише його атрибути. Для класу «Вікно» (рис. 3.2, б) вказані лише його операції, при цьому секція його атрибутів залишена порожньою. Для класу «Рахунок» (рис. 3.2, в) додатково показана четверта секція, в якій вказана вимога – реалізувати резервне копіювання об'єктів цього класу.

Рис. 3.2. Приклади графічного зображення конкретних класів

Ім'я класу має бути унікальним в межах пакету, який може містити одну або декілька діаграм класів. Ім'я класу записується по центру секції імені напівжирним шрифтом і повинно починатися із прописної літери. Рекомендується як імена класів використовувати іменники, записані без пропусків. У секції імені класу можуть також знаходитися стереотипи або посилання на стандартні шаблони, від яких утворений даний клас і, відповідно, від яких він успадковує атрибути і операції.

Клас може мати або не мати екземплярів (або об'єктів). Залежно від цього в мові UML розрізняють конкретні і абстрактні класи.

Конкретний клас (concrete class) — клас, на основі якого можуть бути безпосередньо створені екземпляри або об'єкти. Розглянуті вище позначення відносяться до конкретних класів.

Абстрактний клас (abstract class) — клас, який не має екземплярів або об'єктів. Прикладами абстрактних класів є «перетворювач» (доки не визначена його функція перетворення), «лінія» (доки не визначена її форма) тощо.

Для позначення імені абстрактного класу використовується косий шрифт (курсив). У мові UML прийнята загальна домовленість про те, що будь-який текст, що відноситься до абстрактного елементу, записується курсивом.

В деяких випадках необхідно явно вказати, до якого пакету відноситься той або інший клас. Для цієї мети використовується спеціальний символ роздільник – подвійна двокрапка – «**Ім'я пакету::Ім'я класу**». Наприклад, якщо визначений пакет з ім'ям «**Банк**», то клас «**Рахунок**» в цьому банку може бути записаний у вигляді: «**Банк::Рахунок**».

На рис.3.3 показані класи пакету «**Проект інформаційної технології оптимізації СК**»

Рис. 3.3. Класи пакету IT

Атрибути класу

Атрибут (attribute) — необхідний для опису окремої властивості або ознаки, яка є загальною для всіх об'єктів даного класу. Атрибути класу записуються в другій зверху секції прямокутника класу.

Кожному атрибуту класу відповідає окремий рядок тексту. Загальний формат запису окремого атрибуту класу такий:

«квантор видимості» «ім'я атрибуту» [кратність] : «тип атрибуту» = «вихідне значення» {рядок-властивість}.

Видимість (visibility) — якісна характеристика опису елементів класу, що характеризує потенційну можливість інших об'єктів моделі впливати на окремі аспекти поведінки даного класу. Видимість описується за допомогою квантора видимості (visibility), який може приймати одне з 4-х можливих значень і відображуватися за допомогою спеціальних символів.

- "+" – позначає атрибут загальнодоступний (public). Атрибут з цією зоною видимості доступний або видний з будь-якого іншого класу пакету, в якому визначена діаграма.
- "#" – позначає атрибут захищений (protected). Атрибут з цією зоною видимості недоступний або не видний для всіх класів, за винятком підкласів даного класу.
- "-" – позначає атрибут закритий (private). Атрибут з цією зоною видимості недоступний або невидний для всіх класів без виключення.
- "~" - позначає атрибут пакетний (package). Атрибут з цією зоною видимості недоступний або не видний для всіх класів за межами пакету, в якому визначений клас-власник даного атрибуту.

Квантор видимості може бути опущений. Його відсутність означає, що видимість атрибуту не вказується. Замість умовних графічних позначень можна записувати відповідне ключове слово: public, protected, private, package.

Ім'я атрибуту - єдиний обов'язковий елемент синтаксичного позначення атрибуту, повинно починатися з малої літери і не повинно містити пропусків.

Кратність (multiplicity) — характеризує загальну кількість конкретних атрибутів даного типу, що входять до складу окремого класу. У загальному випадку кратність записується у формі: [нижня границя .. верхня границя]. Як верхня границя може використовуватися спеціальний символ "*" (зірочка), який означає довільне позитивне ціле число, тобто необмежене зверху значення

кратності відповідного атрибуту.

Інтервалів кратності для окремого атрибуту може бути декілька. В цьому випадку їх спільне використання відповідає теоретико-множинному об'єднанню відповідних інтервалів.

Якщо як кратність вказується одним значенням, то кратність атрибуту приймається рівною даному числу. Якщо ж вказується єдиний знак "*", то це означає, що кратність атрибуту може бути довільним позитивним цілим числом або нулем. Якщо кратність атрибуту не вказана, то за умовчанням приймається 1.

Тип атрибуту інколи визначається залежно від мови програмування, яку передбачається використовувати для реалізації даної моделі. У простому випадку тип атрибуту вказується рядком тексту, що має осмислене значення в межах пакету або моделі, до яких відноситься даний клас.

Вихідне значення служить для задавання початкового значення відповідного атрибуту у момент створення окремого екземпляра класу.

При описі атрибутів можуть бути використані додаткові синтаксичні конструкції— це підкреслення рядка атрибуту, текст пояснення у фігурних дужках і коса риска перед ім'ям атрибуту.

Знак "/" перед ім'ям атрибуту вказує на те, що даний атрибут є похідним від деякого іншого атрибуту цього ж класу.

Похідний атрибут (derived element) — атрибут класу, значення якого для окремих об'єктів може бути обчислене за допомогою значень інших атрибутів цього ж об'єкту.

Операції класу

Сукупність ***операцій (operation)*** характеризує функціональний аспект поведінки всіх об'єктів даного класу. Запис операцій класу в мові UML також стандартизований і підкоряється певним синтаксичним правилам. Загальний формат запису окремої операції класу:

«квантор видимості» «ім'я операції» список параметрів):

«вираз типу повертаного значення» {властивість}

Квантор видимості, як і в разі атрибутів класу, може приймати одне з чотирьох можливих значень і, відповідно, відображується за допомогою спеціального символу або ключового слова. Квантор видимості для операції може бути опущений. Замість умовних графічних позначень також можна записувати відповідне ключове слово: public, protected, private, package.

Ім'я операції - єдиний обов'язковий елемент синтаксичного позначення операції, починається з маленької літери.

Список параметрів є переліком розділених комою формальних параметрів, кожен з яких, у свою чергу, може бути представлений в наступному вигляді:

«напрям параметру» «ім'я параметра» : «вираз типу» =

«значення параметра за умовчанням».

Напрям параметра — одне з ключових слів *in, out* або *inout* із значенням *in* за умовчанням, у

випадку, якщо вигляд параметра не вказується.

Вираз типу повертаного значення також вказує на тип даних значення, яке повертається об'єктом після виконання відповідної операції. Дві крапки і вираз типу повертаного значення можуть бути опущені, якщо операція не повертає жодного значення. Для вказівки декількох повертаних значень даний елемент специфікації операції може бути записаний у вигляді списку окремих виразів.

Операція із областю дії на весь клас показується підкресленням імені і рядка виразу типу.

Рядок-властивість може бути відсутнім, якщо властивості не специфіковані.

Список формальних параметрів і тип *повертаного* значення не обов'язковий. Квантор видимості атрибутів і операцій може бути вказаний у вигляді спеціального значка або символу, які використовуються для графічного представлення моделей в інструментальному засобі. Імена операцій, так само, як атрибутів і параметрів, записуються з ***рядкової літери***, а типи параметрів — із ***прописної літери***. При цьому обов'язковою частиною рядка запису операції є наявність імені операції і круглих дужок.

Інтерфейс

Інтерфейс (interface) — іменована множина операцій, які характеризують поведінку окремого елемента моделі.

Інтерфейс у мові UML є спеціальним випадком класу, в якому є операції, але відсутні атрибути. Для позначення інтерфейсу використовується спеціальний графічний символ – коло, або стандартний – прямокутник класу із стереотипом <<interface>> (рис. 3.4).

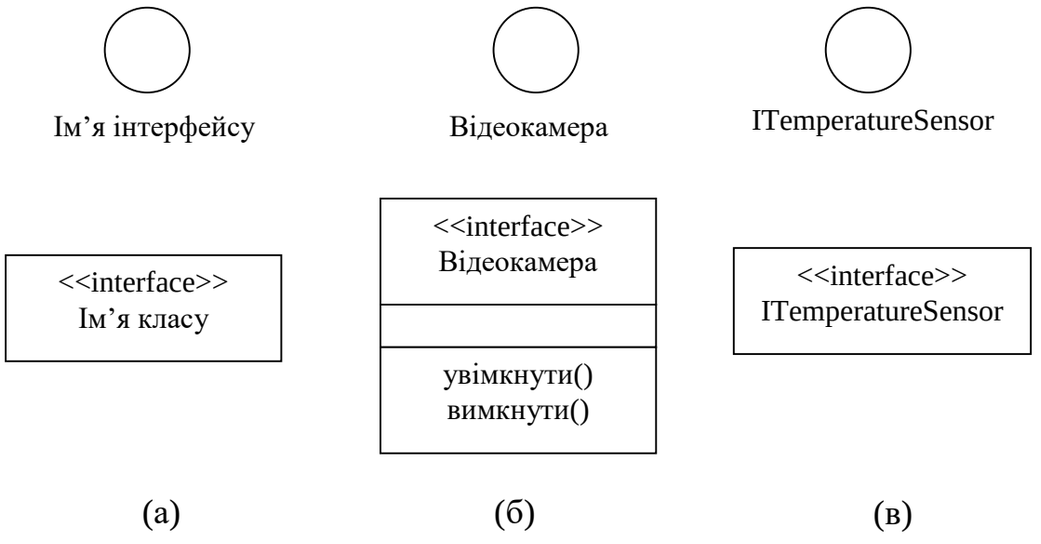

Рис. 3.4. Приклади графічного зображення інтерфейсів на діаграмах класів

На діаграмі варіантів використання інтерфейс зображується у вигляді маленького кола, поряд з яким записується його ім'я (рис. 3.4, а). Як ім'я може використовуватися іменник, який

характеризує відповідну інформацію або сервіс, наприклад, "Сенсор температури", "Форма введення", "Сирена", "Відеокамера" (рис. 3.4, б). З врахуванням мови реалізації моделі ім'я інтерфейсу, як і імена інших класів, рекомендується записувати англійською і починати із заголовної букви I, наприклад, ITemperatureSensor, IsecureInformation (рис. 3.4, в).

Інтерфейси на діаграмі служать для специфікації таких елементів моделі, які видимі ззовні, але їх внутрішня структура залишається прихованою від клієнтів. Інтерфейси не можуть містити ні атрибутів, ні станів, ні направлених асоціацій. Вони містять лише операції без вказівки особливостей їх реалізації. Формально інтерфейс не лише відділяє специфікацію операцій системи від їх реалізації, але і визначає загальні кордони проектованої системи. У подальшому інтерфейс може бути уточнений явною вказівкою тих операцій, які специфікують окремий аспект поведінки системи. Графічні зображення інтерфейсів у формі кола можуть використовуватися і на інших типах канонічних діаграм, наприклад, діаграмах компонентів і розгортання.

3.2. Відношення і їх графічне зображення на діаграмі класів

Окрім внутрішнього устрою класів важливу роль при розробці системи мають різні відношення між класами, які також можуть бути показані на діаграмі класів. Сукупність допустимих типів таких відношень строго фіксована в мові UML. Базові відношення, що показуються на діаграмах класів:

- Відношення асоціації (association relationship)
- Відношення узагальнення (generalization relationship)
- Відношення агрегації (aggregation relationship)
- Відношення композиції (composition relationship)

Відношення асоціації

Відношення асоціації відповідає наявності довільного відношення або взаємозв'язку між класами. Дане відношення позначається суцільною лінією із стрілкою або без неї і з додатковими символами, які характеризують спеціальні властивості асоціації. Як додаткові спеціальні символи можуть використовуватися ім'я асоціації, символ навігації, а також імена і кратність класів-ролей асоціації.

Ім'я асоціації - необов'язковий елемент її позначення. Проте, якщо воно задане, то записується із заголовної букви поряд з лінією асоціації.

Найбільш простий випадок даного відношення - *бінарна асоціація* (binary association), яка служить для подання довільного відношення між двома класами. Вона може бути ненаправленим (симетричним) або направленим відношенням. Окремий випадок бінарної асоціації - *асоціація рефлексії*, яка пов'язує клас з самим собою. Ненаправлена бінарна асоціація зображується лінією без стрілки. Для неї на діаграмі може бути вказаний порядок читання класів з використанням

значка у формі трикутника поряд з ім'ям даної асоціації.

Як простий приклад ненаправленої бінарної асоціації можна розглянути відношення між двома класами - класом «Компанія» і класом «Співробітник» (рис. 3.5). Вони пов'язані між собою бінарною асоціацією *Працює*, ім'я якої вказане на рисунку поряд з лінією асоціації. Для даного відношення визначений наступний порядок читання діаграми - співробітник працює в компанії.

Направлена бінарна асоціація зображується суцільною лінією з простою стрілкою на одному з її кінців. Напрям цієї стрілки вказує на те, який клас є першим, а який - другим.

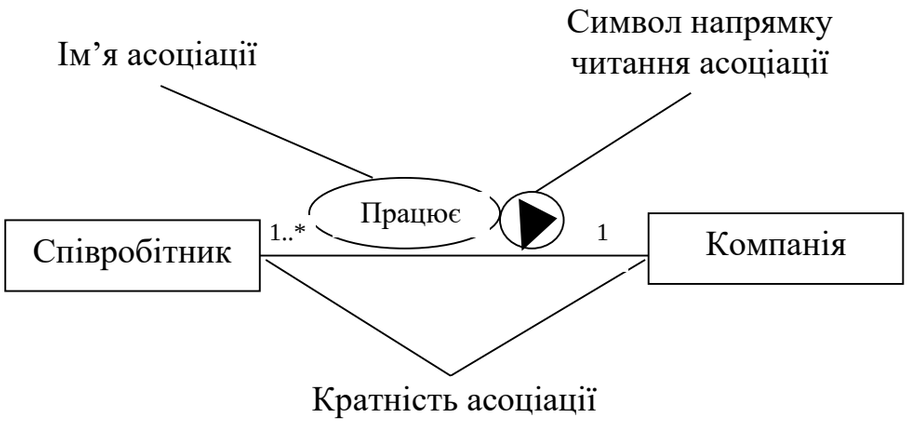

Рис. 3.5. Графічне зображення ненаправленої бінарної асоціації між класами

Як простий приклад направленої бінарної асоціації можна розглянути відношення між двома класами - класом «Клієнт» і класом «Рахунок» (рис. 3.6). Вони зв'язані між собою бінарною асоціацією з ім'ям *Має*, для якої визначений порядок згадування класів. Це означає, що конкретний об'єкт класу «Клієнт» завжди повинен вказуватися першим при розгляді взаємозв'язку з об'єктом класу «Рахунок». Іншими словами, ці об'єкти класів утворюють кортеж елементів, наприклад, «клієнт, рахунок_1, рахунок_2, ..., рахунок_n».

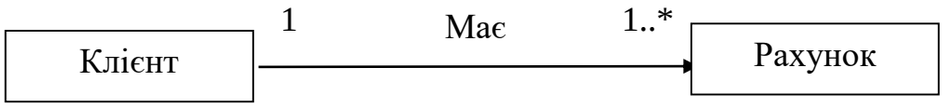

Рис. 3.6. Графічне зображення направленої бінарної асоціації між класами

Окремий випадок відношення асоціації - так звана *виключаюча асоціація* (Xor-association). Семантика даної асоціації вказує на те, що з декількох потенційно можливих варіантів даної асоціації в кожен момент часу може використовуватися лише один. На діаграмі класів виключаюча асоціація зображується пунктирною лінією, що сполучає дві і більше асоціацій (рис. 3.7), поряд з якою записується обмеження у формі рядка тексту у фігурних дужках: {xor}.

Тернарна асоціація зв'язує відношенням три класи. Асоціація вищої арності називається *n-арной асоціацією* **(*n-ary association*)**.

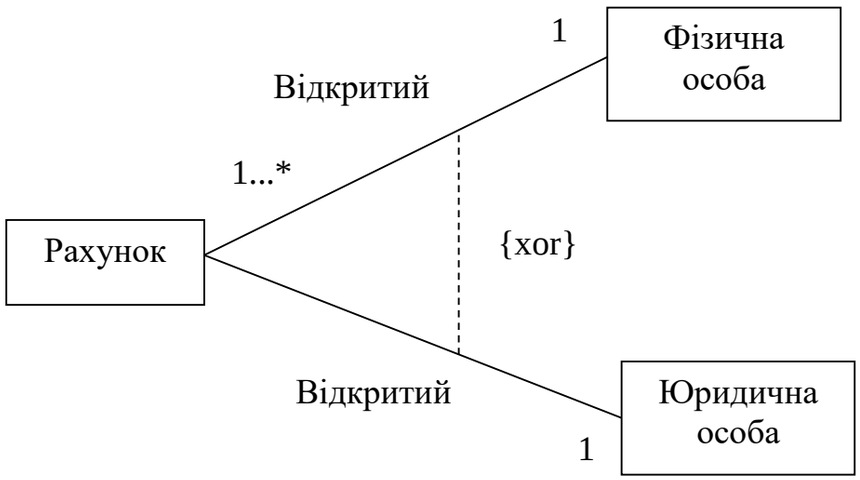

Рис. 3.7. Графічне зображення виключаючої асоціації між трьома класами

Кожен екземпляр такої асоціації є впорядкованим набором (кортежем), що містить *n* екземплярів з відповідних класів. Така асоціація пов'язує відношенням більш ніж три класи, при цьому клас може брати участь в асоціації більш ніж один раз. Кожен екземпляр *n*-арної асоціації є *n*-арним кортежем, що складається з об'єктів відповідних класів. Бінарна асоціація є окремим випадком *n*-арної асоціації, коли значення *n=2*, але вона має власне позначення.

Графічно *n*-арна асоціація позначається ромбом, від якого ведуть лінії до символів класів даної асоціації. Сам же ромб з'єднується з символами класів суцільними лініями. Зазвичай лінії проводяться від вершин ромба або від середини його сторін. Ім'я *n*-арної асоціації записується поряд з ромбом. Проте порядок класів в *n*-арній асоціації на діаграмі не фіксується.

Як приклад тернарної асоціації можна розглянути відношення між трьома класами: «Співробітник», «Компанія» і «Проект». Дана асоціація вказує на наявність відношення між цими трьома класами, яке може показувати інформацію про проекти, що реалізовуються в компанії, і про співробітників, які беруть участь у виконанні окремих проектів (рис. 3.8).

Клас може бути приєднаний до лінії асоціації пунктирною лінією. Це означає, що даний клас

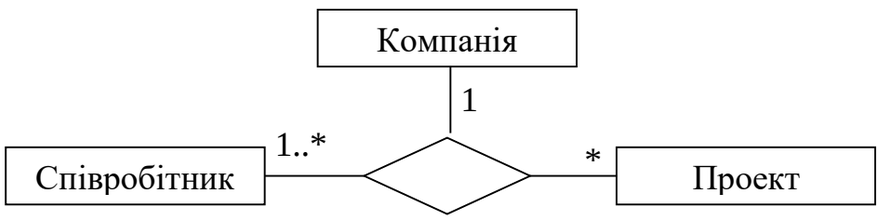

Рис. 3.8. Графічне зображення тернарної асоціації між трьома класами

забезпечує підтримку властивостей відповідної *n*-арної асоціації, а сама *n*-арна асоціація має атрибути, операції і асоціації. Іншими словами, така асоціація є класом з відповідним позначенням у вигляді прямокутника і самостійним елементом мови UML - асоціативним класом (Association Class).

Роль (role) - іменована специфічна поведінка деякого елементу. Роль може бути архітектурною або функціональною**.**

Ім'я ролі є рядком тексту поряд з кінцем асоціації для відповідного класу. Вона вказує на специфічну роль, яку грає клас, що є кінцем даної асоціації. Ім'я ролі – не обов'язковий елемент позначень і може бути відсутнім на діаграмі.

Кратність асоціації відноситься до кінців асоціації і позначається у вигляді інтервалу цілих чисел, аналогічно кратності атрибутів і операцій класів, але без дужок. Цей інтервал записується поряд з кінцем відповідної асоціації і означає потенційне число окремих екземплярів класу, які можуть мати місце.

Так, для прикладу (рис. 3.8) кратність "1" для класу «Компанія» означає, що кожен співробітник може працювати лише в одній компанії. Кратність "1..*" для класу «Співробітник» означає, що в кожній компанії можуть працювати декілька співробітників, загальне число яких заздалегідь невідоме і нічим не обмежене. Замість кратності "1..*" не можна записати лише символ "*", оскільки останній означає кратність "0..*". Для даного прикладу це означало б, що окремі компанії можуть зовсім не мати співробітників в своєму штаті.

Асоціація є найбільш загальною формою відношення в мові UML. Всіх інші типи відношень можна вважати окремими випадками даного відношення.

Відношення узагальнення

Відношення узагальнення є відношенням між загальнішим елементом (батьком або предком) і окремим або спеціальним елементом (дочірнім або нащадком).

Згідно одному з головних принципів методології ООАП - спадкоємству, клас-нащадок (***child***) володіє всіма властивостями і поведінкою класу-предка **(*parent*)**, а також має власні властивості і поведінку, які можуть бути відсутніми у класа-предка.

На діаграмах відношення узагальнення позначається суцільною лінією з трикутною стрілкою на одному з кінців (рис. 3.9). Стрілка вказує на загальніший клас (клас-предок або суперклас), а її початок - на більш спеціальний клас (клас-нащадок або підклас).

Від одного класу-предка одночасно можуть успадковувати декілька класів-нащадків. Наприклад, клас «*Транспортний засіб*» (курсив позначає абстрактний клас) може виступати як суперклас для підкласів, які відповідають конкретним транспортним засобам, таким як:

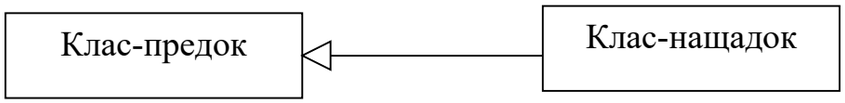

Рис. 3.9. Графічне зображення відношення узагальнення в мові UML

«Автомобіль», «Автобус», «Трактор» тощо. Це може бути зображено графічно у формі діаграми класів рис. 3.10.

З метою спрощення позначень на діаграмі класів і зменшення числа стрілок-трикутників і

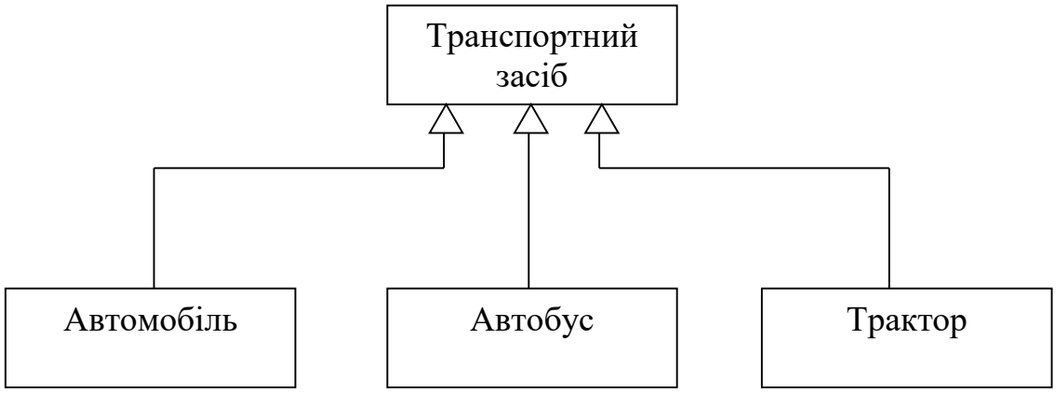

Рис. 3.10. Приклад графічного зображення відношення узагальнення для декількох класів-нащадків

сукупності ліній, що позначають одне і те ж відношення узагальнення, може бути просто змальована стрілка.

Графічне зображення відношення узагальнення формою відповідає графу типу «ієрархічне дерево».

На додаток до стрілки узагальнення може бути приєднаний рядок тексту, який відзначає спеціальні властивості цього відношення у формі обмеження у фігурних дужках.

Як **обмеження** можуть бути використані наступні ключові слова мови UML:

- {complete} - означає, що в даному відношенні узагальнення специфіковані всі класи-нащадки, і інших класів-нащадків у даного класу-предка бути не може.
- {incomplete} - означає випадок, протилежний першому, а саме, передбачається, що на діаграмі вказані не всі класи-нащадки. У подальшому можливо розробник заповнить їх перелік, не змінюючи вже побудовану діаграму.

• {disjoint} - означає, що класи-нащадки не можуть містити об'єктів, що одночасно є екземплярами двох або більше класів.

• {overlapping} - випадок, протилежний попередньому, а саме, передбачається, що окремі екземпляри класів-нащадків можуть належати одночасно декільком класам.

З врахуванням додаткового використання стандартного обмеження діаграма класів (рис. 3.10) може бути уточнена (рис. 3.12).

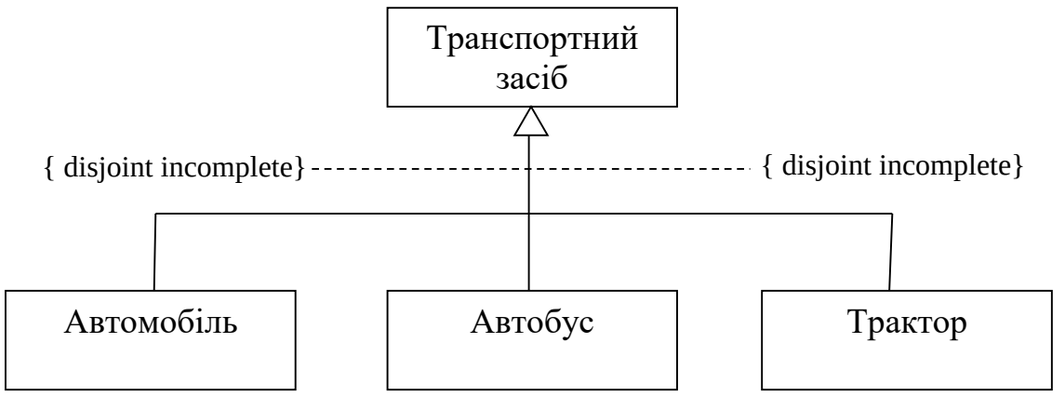

Рис. 3.12. Варіант уточненого графічного зображення відношення узагальнення класів з використанням рядка-обмеження

Відношення агрегації

Агрегація (aggregation) - спеціальна форма асоціації, яка служить для зображення відношення типа «ціле – частина».

Відношення агрегації має місце між декількома класами в тому випадку, якщо один з класів включає як складові частини інші елементи. Дане відношення має фундаментальне значення для опису структури складних систем. Розкриваючи внутрішню структуру системи, відношення агрегації показує, з яких елементів складається система, і як вони пов'язані між собою.

В очевидь, розділення системи на складові частини є ієрархією, але принципово відмінну від тієї, яка породжується відношенням узагальнення. Відмінність полягає в тому, що частини системи ніяк не зобов'язані успадковувати її властивості і поведінку, оскільки є самостійними елементами. Більш того, частини цілого володіють власними атрибутами і операціями, які істотно відрізняються від атрибутів і операцій цілого.

Графічно відношення агрегації зображається суцільною лінією, один з кінців якої є незафарбованим ромбом. Цей ромб вказує на той клас, який є "цілим" або клас-контейнер. Решта

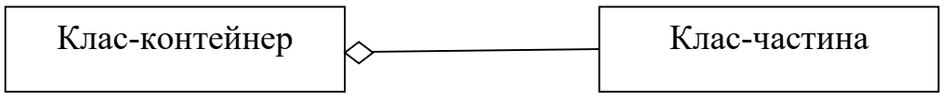

Рис. 3.13. Графічне зображення відношення агрегації в мові UML

класів є його "частинами" (рис. 3.13).

Відношення агрегації покажемо на прикладі взаємозв'язку між класом «Системний блок персонального комп'ютера» і його складовими частинами: «Процесор», «Материнська плата», «Оперативна пам'ять», «Жорсткий диск» і «Дисковод». Використовуючи позначення мови UML, компонентний склад системного блоку можна представити у вигляді відповідної діаграми класів (рис. 3.14).

Відношення композиції

Композиція (composition) - різновид відношення агрегації, при якому складові частини цілого мають такий же час життя, що і саме ціле. Ці частини знищуються разом із знищенням цілого. Особливість цього взаємозв'язку полягає в тому, що частини не можуть виступати у відриві від цілого.

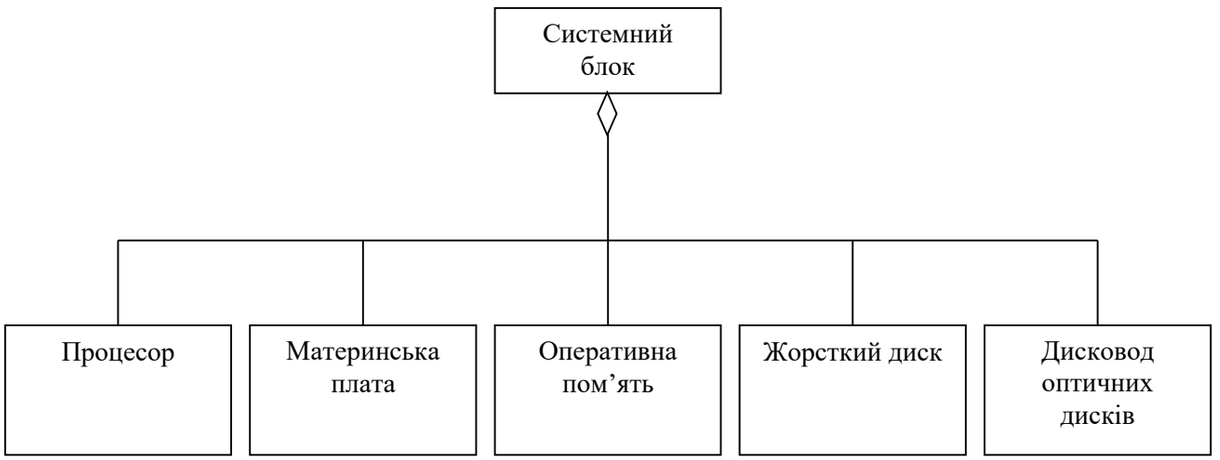

Рис. 3.14. Діаграма класів для ілюстрації відношення агрегації на прикладі системного блоку ПК

Графічно відношення композиції зображається суцільною лінією, один з кінців якої є закрашеним усередині ромбом. Цей ромб вказує на той клас, який є класом-композитом (рис. 3.15).

Можливо, найнаочнішим прикладом цього відношення є жива клітина в біології, у відриві від

Рис. 3.15. Графічне зображення відношення композиції в мові UML

якої не можуть існувати її складові частини. Інший практичний приклад - вікно графічного інтерфейсу програми, яке може складатися з рядка заголовка, смуг прокрутки, головного меню, робочої області і рядка стану. Подібне вікно є класом, а його складові елементи також є окремими

класами.

Для відношень композиції і агрегації можуть використовуватися додаткові позначення, які використовуються у відношеннях асоціації. А саме, можуть вказуватися кратності окремих класів, які в загальному випадку не обов'язкові. Стосовно описаного вище прикладу клас «Вікно програми» є класом-композитом, а взаємозв'язки складових його частин можуть бути зображені діаграмою класів рис. 3.16.

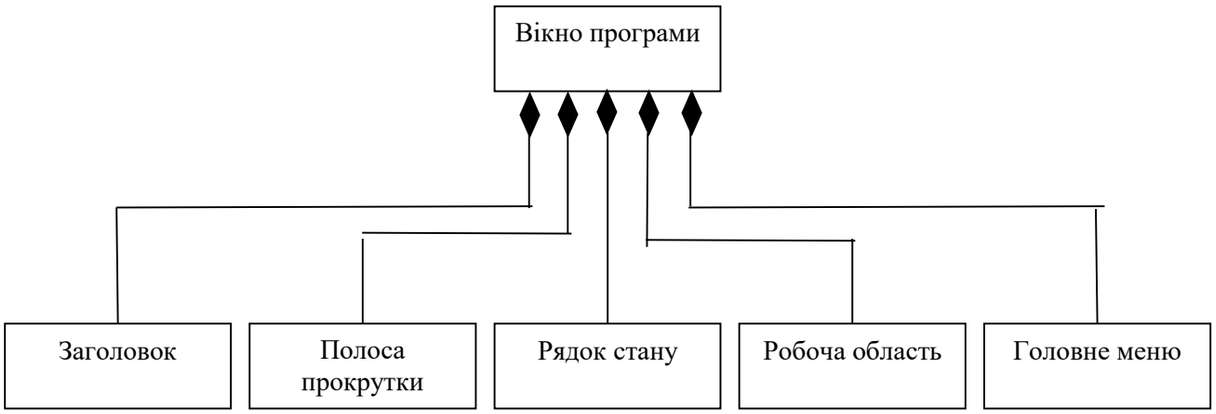

Рис. 3.16. Діаграма класів для ілюстрації відношення композиції на прикладі класу-композиту «Вікно програми»