

3. Проектування програмного забезпечення засобами UML.

3.3. Діаграма класів

Діаграма класів призначена для надання статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. Діаграма класів відображує різні взаємозв'язки між окремими сутностями предметної області, такими як об'єкти й підсистеми, а також описує їхню внутрішню структуру й типи відносин. На даній діаграмі не вказується інформація про часові аспекти функціонування системи.

Діаграма класів (class diagram) - діаграма на якій представлена сукупність декларативних або статичних елементів моделі, таких як класи з атрибутами й операціями, а також відношення, що їх з'єднують.

Клас

Клас (class) - абстрактний опис множини однорідних об'єктів, що мають одинакові атрибути, операції й відносини з об'єктами інших класів.

Графічно клас зображується у вигляді прямокутника, що додатково може бути розділений горизонтальними лініями на розділи або секції (рис. 3.3.1). У цих секціях можуть вказуватися ім'я класу, атрибути й операції класу.

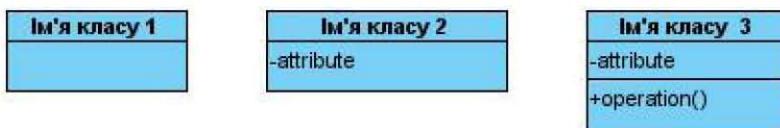


Рис. 3.3.1. Варіанти графічного зображення класу на діаграмі класів

На початкових етапах розробки діаграми окремі класи можуть позначатися простим прямокутником, у якому повинне бути зазначене ім'я відповідного класу (рис. 3.3.1). З деталізацією окремих компонентів діаграми опис класів доповнюється атрибутами і операціями. Передбачається, що остаточний варіант діаграми містить найбільш повний опис класів, який складається із трьох секцій.

Навіть якщо секції атрибутів й операцій порожні, у позначенні класу вони повинні бути виділені горизонтальною лінією, для того щоб відрізнисти клас від інших елементів мови UML. Приклади графічного зображення конкретних класів наведені на рис. 3.3.2. У першому випадку для класу Коло (Circle) зазначені тільки його атрибути – крапка на координатній площині, що визначає розташування його центра й радіус. Для класу Вікно (Window) зазначені тільки його операції.

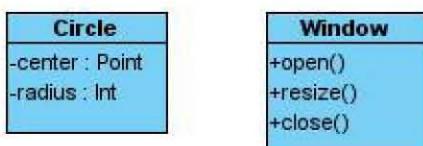


Рис. 3.3.2. Приклади графічного зображення конкретних класів

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

Конкретний клас (concrete class) - клас, на основі якого можуть бути безпосередньо створені екземпляри або об'єкти.

Абстрактний клас (abstract class) - клас, що не має екземплярів або об'єктів.

Розглянуті вище позначення відносяться до конкретних класів.

Ім'я класу

Ім'я класу має бути унікальним у межах пакета, що може містити одну або кілька діаграм класів. Ім'я вказується в самій верхній секції прямокутника, тому вона часто називається секцією імені класу. На додаток до загального правила іменування елементів мови UML, ім'я класу записується по центрі секції імені напівжирним шрифтом і повинне починатися із великої літери.

Для позначення імені абстрактного класу використається похилий шрифт (курсив). У мові UML прийнята загальна угода про те, що будь-який текст, що відноситься до абстрактного елемента, записується курсивом. Це має принципове значення, оскільки є семантичним аспектом опису абстрактних елементів мови UML.

У деяких випадках необхідно явно вказати, до якого пакета відноситься той або інший клас. Для цього використається спеціальний символ роздільник - подвійна двокрапка - (::). Синтаксис рядка імені класу в цьому випадку буде наступний: <Ім'я пакету>::<Ім'я класу>. Інакше кажучи, перед ім'ям класу повинне бути явно зазначене ім'я пакета, до якого його варто віднести. Наприклад, якщо визначено пакет з ім'ям Банк, то клас Рахунок у цьому банку може бути записаний у вигляді: Банк::Рахунок.

Атрибути класу

Атрибут (attribute) - змістовна характеристика класу, що описує множину значень, які можуть приймати окремі об'єкти цього класу.

Атрибут класу призначено для опису окремої властивості або ознаки, що є загальним для всіх об'єктів даного класу. Атрибути класу записуються в другій зверху секції прямокутника класу. Цю секцію часто називають секцією атрибутів.

Ім'я атрибута являє собою рядок тексту, що використається як ідентифікатор відповідного атрибута й тому повинна бути унікальної в межах даного класу. Ім'я атрибута - єдиний обов'язковий елемент синтаксичного позначення атрибута, повинне починатися з маленької літери й не повинне містити пробілів. Знак "/" перед ім'ям атрибута вказує на те, що даний атрибут є похідним від деякого іншого атрибута 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'єкта.

У мові UML прийнята певна стандартизація запису атрибутів класу. Кожному атрибуту класу відповідає окремий рядок тексту, що складається із квантора видимості атрибута, імені атрибута, його кратності, типу значень атрибута й, можливо, його вихідного значення. Загальний формат запису окремого атрибута класу наступний:

```
<квантор видимості> <ім'я атрибута> [кратність] :  
<тип атрибута> = <вихідне значення> {рядок-властивість}.
```

Видимість (visibility) - якісна характеристика опису елементів класу, що характеризує потенційну можливість інших об'єктів моделі впливати на окремі аспекти поведінки

Видимість у мові UML специфікується за допомогою квантора видимості (visibility), що може приймати одне з 4-х можливих значень і відображатися за допомогою спеціальних символів.

- Символ "+" - позначає атрибут з областю видимості типу загальнодоступний (public). Атрибут із цією областю видимості доступний або видний з будь-якого іншого класу пакета, у якому визначена діаграма.
- Символ "#" - позначає атрибут з областю видимості типу захищений (protected). Атрибут із цією областю видимості недоступний або невидний для всіх класів, за винятком підкласів даного класу.
- Символ "-" - позначає атрибут з областю видимості типу закритий (private). Атрибут із цією областю видимості недоступний або невидний для всіх класів без винятку.
- Символ "~" - позначає атрибут з областю видимості типу пакетний (package). Атрибут із цією областю видимості недоступний або невидний для всіх класів за межами пакета, у якому визначений клас-власник даного атрибута.

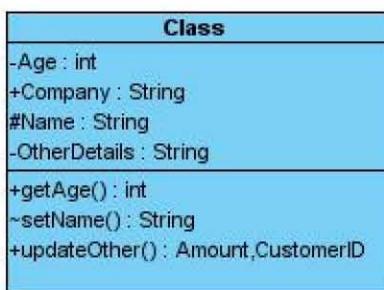


Рис. 3.3.3. Приклади графічного зображення видимості атрибутів та операцій класу

Вважається, що якісна об'єктно-орієнтована архітектура обмежує безпосередній доступ до атрибутів класу та пропонує методи, які дозволяють оперувати атрибутами за викликом. Такий підхід гарантує, що зміни даних відбуваються в одній точці та за певними правилами – для великих систем тягар підтримки цілісності коду з багатьма точками доступу до даних занадто важкий.

Кратність (multiplicity) атрибута характеризує загальну кількість конкретних атрибутів даного типу, що входять до складу окремого класу. У загальному випадку кратність записується у формі рядка тексту із цифр у квадратних дужках після імені відповідного атрибута, при цьому цифри розділяються двома крапками: [нижня границя .. верхня границя], де нижня й верхня границі позитивні цілі числа. Кожна така пара призначена для позначення окремого замкнутого інтервалу цілих чисел, у якого нижня

(верхня) границя дорівнює значенню нижня границя (верхня). У якості верхньої границі може використовуватися спеціальний символ "*" (зірочка), що означає довільне позитивне ціле число, тобто необмежене зверху значення кратності відповідного атрибута.

Інтервалів кратності для окремого атрибута може бути кілька. У цьому випадку їхнє спільне використання відповідає теоретико-множинному об'єднанню відповідних інтервалів. Значення кратності з інтервалу випливають у монотонно зростаючому порядку без пропуску окремих чисел, що лежать між нижньою та верхньою границями. При цьому дотримуються наступного правила: відповідні нижні й верхні граници інтервалів включаються в значення кратності.

Якщо як кратність указується одне значення, то кратність атрибута приймається рівною даному числу. Якщо ж указується єдиний знак "*", то це означає, що кратність атрибута може бути довільним позитивним цілим числом або нулем. У мові UML кратність широко використається також для завдання ролей асоціацій, складених об'єктів і значень атрибутів. Якщо кратність атрибута не зазначена, то за замовчуванням у мові UML приймається її значення рівне [1..1].

Операції класу

Операція (operation) - це сервіс, що надається кожним екземпляром або об'єктом класу на вимогу своїх клієнтів, якими можуть виступати інші об'єкти, у тому числі й екземпляри даного класу.

Операції класу записуються в третій зверху секції прямокутника класу, яку часто називають секцією операцій. Сукупність операцій характеризує функціональний аспект поведінки всіх об'єктів даного класу. Кожній операції класу відповідає окремий рядок, що складається із квантора видимості операції, імені операції, виразу типу, що повертає операція, значення й, можливо, рядок-властивість даної операції. Запис окремої операції класу має наступний вигляд:

<квантор видимості> <ім'я операції>(список параметрів):<виразу типу значення, що повертає,> {рядок-властивість}

Квантор видимості, як й у випадку атрибутів класу, може приймати одне із чотирьох можливих значень й, відповідно, відображається за допомогою спеціального символу, ключового слова або може бути опущений.

Ім'я операції - рядок тексту, що використовується як ідентифікатор відповідної операції. Воно повинно бути унікальним в межах даного класу. Ім'я операції - єдиний обов'язковий елемент синтаксичного позначення операції, повинне починатися з маленької літери і, як правило, записуватися без пробілів.

Список параметрів – перелік розділених комою формальних параметрів, кожний з яких може бути представлений у наступному виді:

<напрямок параметра> <ім'я параметра>: <вираз типу> = <значення параметра за замовчуванням>.

Параметр (parameter) - специфікація змінної операції, що може бути змінена, передана або повернута.

Параметр може включати ім'я, тип, напрямок і значення за замовчуванням.

Ім'я параметра - ідентифікатор відповідного формального параметра, при записі якого додержуються правил завдання імен атрибутів.

Вираз типу - специфікація типу даних для припустимих значень відповідного формального параметра.

Напрямок параметра – це одне із ключових слів `in`, `out` або `inout` зі значенням `in` за замовчуванням, у випадку якщо вид параметра не вказується.

Значення за замовчуванням у загальному випадку - деяке конкретне значення для цього формального параметра.

Вираз типу значення, що повертається, також вказує на тип даних значення, що повертається об'єктом після виконання відповідної операції. Дві крапки та вираз типу значення, що повертається, можуть бути опущені, якщо операція не повертає ніякого значення.

Розширення діаграми класів

Однією з характерних особливостей мови UML є наявність механізмів розширення, які дозволяють додати до розгляду додаткові графічні позначення, орієнтовані на рішення завдань із певної предметної області. Одним з таких розширень є профіль для процесу розробки програмного забезпечення (The UML Profile for Software Development Processes).

У рамках вищезазначеного профілю запропоновано три спеціальних графічних примітиви, які можуть бути використані для уточнення семантики окремих класів при побудові різних діаграм:

- Границний клас (`boundary class`) — клас, що розташовується на границі системи із зовнішнім середовищем і безпосередньо взаємодіє з акторами, але є складовою частиною системи. Границний клас може бути зображенний у формі прямокутника класу зі стереотипом `<<boundary>>` (рис. 3.3.4).

- Клас-сутність (`entity class`) — пасивний клас, що містить інформацію, яка має зберігатися постійно й не повинна знищуватися зі знищеннем об'єктів даного класу або припиненням роботи системи, що моделюється. Зазвичай, цей клас відповідає окремій таблиці бази даних. У цьому випадку його атрибути є полями таблиці, а операції – приєднаними або збереженими процедурами. Цей клас лише приймає повідомлення від інших класів моделі. Клас-сутність може бути зображенний формі прямокутника класу зі стереотипом `<<entity>>` (рис. 3.3.4).

- Керуючий клас (`control class`) — клас, відповідальний за координацію дій інших класів. На кожній діаграмі класів повинен бути хоча б один керуючий клас, причому кількість повідомлень, що посилають об'єктам керуючого класу – мала, у порівнянні із числом розсылаємих ними. Керуючий клас відповідає за координацію дій інших класів. Зазвичай, даний клас є активним й ініціює розсилання множини повідомлень іншим класам моделі. Крім спеціального позначення керуючий клас може бути зображенний у формі прямокутника класу зі стереотипом `<<control>>` (рис. 3.3.4).



Рис. 3.3.4. Графічне зображення класів для моделювання програмного забезпечення

Інтерфейс

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

Інтерфейс у контексті мови UML є спеціальним випадком класу, у якого є операції, але відсутні атрибути. Для позначення інтерфейсу використовується стандартний спосіб – прямокутник класу зі стереотипом <<interface>> (рис 3.3.5)

Як ім'я може бути використан іменник, що характеризує відповідну інформацію або сервіс, наприклад, "Датчик температури", "Форма вводу", "Сирена", "Відеокамера" (рис 3.3.5). З урахуванням мови реалізації моделі ім'я інтерфейсу, як й імена інших класів, рекомендується записувати англійською й починати із великої літери I, наприклад, ITemperatureSensor, IsecureInformation.

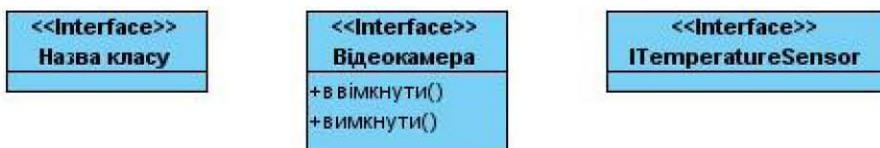


Рис. 3.3.5. Приклади графічного зображення інтерфейсів на діаграмах класів

Відношення на діаграмі класів

Крім внутрішніх принципів організації класів важливу роль при розробці проектированої системи мають різні відношення між класами, які також можуть бути зображені на діаграмі класів.

Базові відношення, що можуть бути зображені на діаграмі класів:

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

Кожне із цих відношень має власне графічне зображення, що відображує семантичний характер взаємозв'язку між об'єктами відповідних класів.

Відношення асоціації

Відношення **асоціації (association)** відповідає наявності довільного відношення або взаємозв'язку між класами. Це відношення позначається суцільною лінією зі стрілкою або без її й з додатковими символами, які характеризують спеціальні властивості асоціації.

Бінарна асоціація (binary association) - найбільш простий відношення асоціації, що служить для зображення довільного відношення між двома класами. Вона зв'язує в точності два різних класи й може бути ненаправленим (симетричним) або направленим відношенням. Окремий випадок бінарної асоціації - **рефлексивна асоціація**, що зв'язує клас із самим собою. Ненаправлена бінарна асоціація зображується лінією без стрілки. Для неї на діаграмі може бути зазначений порядок читання класів з використанням значка у формі трикутника поруч із ім'ям даної асоціації.

Як простий приклад ненаправленої бінарної асоціації можна розглянути відношення між двома класами - класом Компанія й класом Співробітник (рис. 3.3.6). Вони зв'язані між собою бінарною асоціацією «працює», ім'я якої зазначено на малюнку поруч із лінією асоціації. Для даного відношення визначений наступний порядок читання проходження класів - співробітник працює в компанії.



Рис. 3.3.6. Графічне зображення ненаправленої бінарної асоціації між класами

Ім'я асоціації - необов'язковий елемент її позначення. Але, якщо воно задано, то має бути записано поруч із лінією асоціації. При генерації коду з діаграми класів, іменований кінець асоціації стає змінною екземпляру цільового класу. Так, наприклад, "playsFor" змінною екземпляря (атрибутом) класу "Player" (рис. 3.3.7).

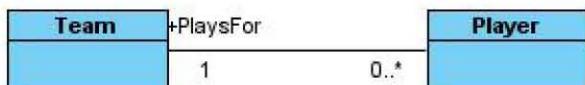


Рис. 3.3.7. Приклади графічного зображення відношення іменованої асоціації

Спрямована бінарна асоціація зображується суцільною лінією із простою стрілкою на одній з її кінцевих крапок. Напрямок цієї стрілки вказує на те, який клас є першим, а який - другим.

Як простий приклад спрямованої бінарної асоціації можна розглянути відношення між двома класами - класом Клієнт і класом Рахунок (рис. 3.3.8). Вони зв'язані між собою бінарною асоціацією з ім'ям Має, для якої визначений порядок проходження класів. Це означає, що конкретний об'єкт класу Клієнт завжди повинен указуватися першим при розгляді взаємозв'язку з об'єктом класу Рахунок. Інакше кажучи, ці об'єкти класів утворять кортеж елементів, наприклад, <клієнт, рахунок_1, рахунок_2,..., рахунок_n>.



Рис. 3.3.8. Графічне зображення спрямованої бінарної асоціації між класами

Окремий випадок відносини асоціації - так називається асоціація, що виключає (**Xor-association**). Семантика даної асоціації вказує на те, що з декількох потенційно можливих варіантів даної асоціації в кожен момент часу може використатися тільки один. На діаграмі класів асоціація, що виключає, зображується пунктирною лінією, що з'єднує дві й більше асоціацій (рис 3.3.9). Поруч з такою лінією записується обмеження у формі рядка тексту у фігурних дужках: {xor}.

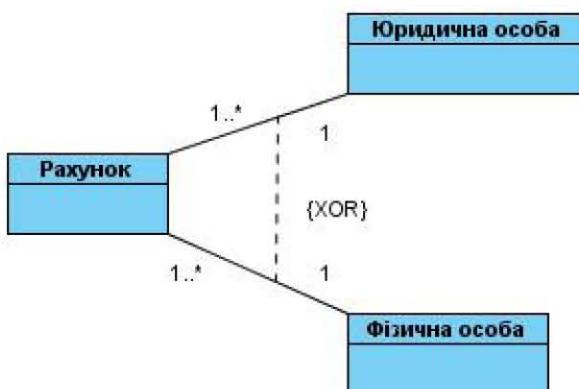


Рис. 3.3.9. Графічне зображення асоціації, що виключає, між трьома класами

Тернарна асоціація зв'язує відношенням три класи. Асоціація більше високої арності називається *n*-арною асоціацією.

Наступний елемент позначенень - кратність асоціації. Кратність ставиться до кінців асоціації й позначається у вигляді інтервалу цілих чисел, аналогічно кратності атрибутів й операцій класів, але без прямих дужок. Цей інтервал записується поруч із кінцем відповідної асоціації й означає потенційне число окремих екземплярів класу, які можуть мати місце, коли інші екземпляри або об'єкти класів фіксовані.

Наприклад, кратність "1" для класу Компанія означає, що кожен співробітник може працювати тільки в одній компанії. Кратність "1..*" для класу Співробітник означає, що в кожній компанії можуть працювати кілька співробітників, загальне число яких заздалегідь невідомо й нічим не обмежено. (рис. 3.3.6) Замість кратності "1..*" не можна записати тільки символ "*", оскільки останній означає кратність "0..*". Для даного прикладу це означало б, що окремі компанії можуть зовсім не мати співробітників у своєму штаті.

Для спрямованої асоціації стрілка може позначатися необов'язковим, але стандартним ключовим словом в лапках і необов'язковим індивідуальним ім'ям. Для відношення залежності зумовлені ключові слова, які позначають деякі спеціальні види залежностей. Ці ключові слова (стереотипи) записуються в лапках поряд із стрілкою, яка відповідає даній залежності. Приклади стереотипів для відношення залежності представлені нижче:

- "access" - служить для позначення доступності відкритих атрибутів і операцій класу-джерела для класів-клієнтів;
- "bind" - клас-клієнт може використовувати деякий шаблон для своєї подальшої параметризації;

- "derive" - атрибути класу-клієнта можуть бути обчислені за атрибутами класу-джерела;
- "import" - відкриті атрибути і операції класу-джерела стають частиною класу-клієнта, неначебто вони були оголошенні безпосередньо в нім;
- "refine" - указує, що клас-клієнт служить уточненням класу-джерела через причини історичного характеру, коли з'являється додаткова інформація в ході роботи над проектом.

Клас Асоціація

Клас асоціація – це логічна структура, яка дозволяє зв'язку асоціація мати атрибути ти методи. Наступний приклад ілюструє, що така структура несе в собі більше семантики ніж простий зв'язок між робітником та проектом: роль, яку грає працівник у проекті – це складна сутність, яка містить в собі характеристики, які не належать класу проект та класу робітник, тощо. Наприклад, працівник може приймати участь в різних проектах одночасно на різних посадах та з різними рівнями допуску.

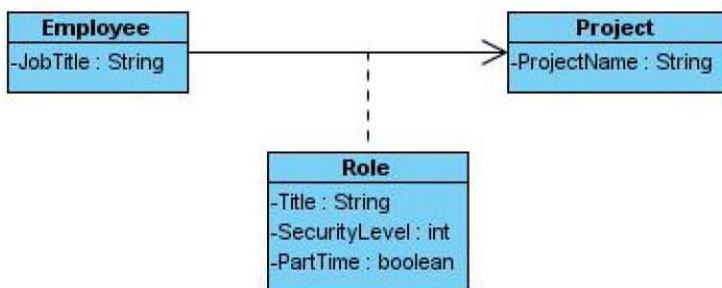


Рис. 3.3.10. Графічне зображення класу-асоціації

Шаблони або параметризований класи

Шаблон (template) або параметризований клас (parametrized class) призначений для позначення такого класу, який має один (або більше) нефіксований формальний параметр. Він визначає множину класів, кожний з яких може бути отриманий зв'язуванням цих параметрів з дійсними значеннями. Зазвичай параметрами шаблонів служать типи атрибутів класів, такі як цілі числа, перерахування, масив рядків та ін. У складнішому випадку формальні параметри можуть представляти і операції класу.

Графічно шаблон зображається прямоугольником, до верхнього правого кута якого приєднаний маленький прямоугольник з пунктирних ліній (Рис. 3.3.11), великий прямоугольник може бути роздільний на секції, аналогічно позначенню для класу. У верхньому прямоугольнику вказується список формальних параметрів для тих класів, які можуть бути отримані на основі даного шаблону. У верхній секції шаблону записується його ім'я за правилами запису імен для класів.

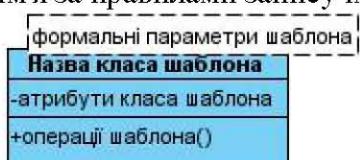


Рис. 3.3.11. Графічне зображення шаблону на діаграмі класів

Шаблон не може бути безпосередньо використаний як клас, оскільки містить невизначені параметри. Найчастіше як шаблон виступає деякий супер клас, параметри якого уточнюються в його класах-нащадках. Вочевидь, в цьому випадку між ними існує відношення залежності з ключовим словом "bind", коли клас-клієнт може використовувати деякий шаблон для своєї подальшої параметризації. На наступній діаграмі (рис 3.3.12) проілюстровано той факт, що клас "Адреса" може бути отримана з шаблону Зв'язний_список на основі актуалізації формальних параметрів "S, L, K" фактичними атрибутами "вулиця, будинок, квартира".

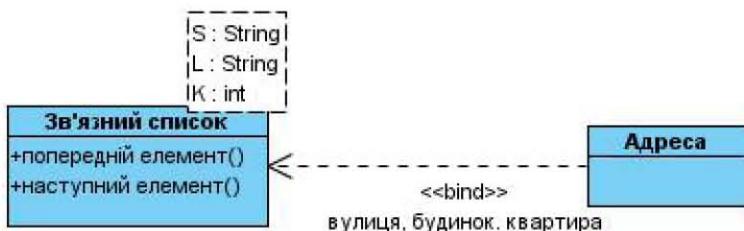


Рис. 3.3.12. Приклад використання шаблону на діаграмі класів

Відношення узагальнення

Узагальнення (generalization) – відношення між більше загальним поняттям і менш загальним поняттям.

Менш загальний елемент моделі повинен бути погоджений з більше загальним елементом і може містити додаткову інформацію. Дане відношення використовується для подання ієрархічних взаємозв'язків між різними елементами мови UML, такими як пакети, класи, варіанти використання.

Стосовно до діаграми класів дане відношення описує ієрархічна будова класів і наслідування їхніх властивостей і поведінки.

Наслідування (inheritance) - спеціальний концептуальний механізм, за допомогою якого спеціальні елементи містять у собі структуру й поведінку загальних елементів, тобто клас-нащадок має всі властивості й поведінку класу-предка, а також має власні властивості й поведінку, які можуть бути відсутні у класі-предку.

Батько, предок (parent) - відносно узагальнення більш загальний елемент.

Нащадок (child) - спеціалізація одного з елементів відношення узагальнення, називаного в цьому випадку батьком.

На діаграмах відношення узагальнення позначається суцільною лінією із трикутною стрілкою на одному з кінців (рис. 3.3.13). Стрілка вказує на загальний клас (клас-предок або супер клас), а її початок - на спеціальний клас (клас-нащадок або під клас).



Рис. 3.3.13. Графічне зображення відносини узагальнення в мові UML

Від одного класу-предка одночасно можуть успадковувати декілька класів-нащадків.

Наступна діаграма ілюструє відношення узагальнення між батьківським класом та класом-нащадком (3.3.14). Об'єкт Коло матиме атрибути координати: x_position, y_position та радіус (radius), а також метод display(). Зауважимо, що "Shape" - абстрактний клас.

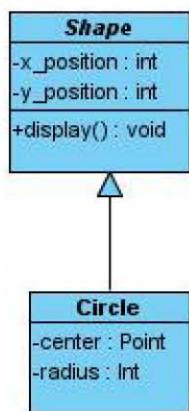


Рис. 3.3.14. Графічне зображення відносини узагальнення в мові UML між класами Circle та Shape

На додаток до простої стрілки узагальнення може бути приєднаний рядок тексту, що вказує на спеціальні властивості цього відношення у формі обмеження, що повинне бути записане у фігурних дужках. Цей текст буде ставитися до всіх ліній узагальнення, які йдуть до класів-нащадків. У якості обмежень 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. Тобто передбачається, що на діаграмі зазначені не всі класи-нащадки. Можливо, надалі розробник заповнить їхній 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. Передбачається, що окремі екземпляри класів-нащадків можуть належати одночасно декільком класам.

Відношення агрегації

Агрегація (aggregation) - спеціальна форма асоціації, що служить для відображення відношення типу "частина-ціле" між агрегатом (ціле) і його складовою частиною.

Відношення агрегації має місце між декількома класами в тому випадку, якщо один із класів являє собою сутність, що містить у собі у якості складових частин інші сутності.

Розподіл системи на складові частини може бути розглянуто як ієрархія. Проте ієрархія такого виду принципово відрізняється від тієї, що породжується відношенням узагальнення. Відмінність полягає в тім, що частини системи не мають успадковувати її властивості й поведінку, оскільки є самостійними сутностями. Більше того, частини цілого мають власні атрибути й операції, які можуть істотно відрізняються від атрибутів й операцій цілого.

Графічно відношення агрегації зображується суцільною лінією з не зафарбованим усередині ромбом на одному з своїх кінців. Цей ромб указує на той клас, що являє собою "ціле" або клас-контейнер. Інші класи є його "частинами" (рис. 3.3.15).



Рис. 3.3.15. Графічне зображення відношення агрегації в мові UML

Як приклад відношення агрегації можна розглянути взаємозв'язок типу "частина-ціле", що має місце між класом Системний блок персонального комп'ютера і його складових частин: Процесор, Материнська плата, Жорсткий диск і Відеокарта (рис. 3.3.16).



Рис. 3.3.16. Діаграма класів для ілюстрації відношення агрегації на прикладі системного блоку ПК

Відношення композиції

Композиція (composition) - різновид відношення агрегації, при якій складові частини цілого мають такий же час життя, що й саме ціле. Ці частини знищуються разом зі знищеннем цілого.

Композит (composite) - клас, що зв'язаний відношенням композиції з одним або більшим числом класів.

Графічно відношення композиції зображується суцільною лінією, з зафарбованим усередині ромбом на одному з своїх кінців. Цей ромб указує на клас-композит. Інші класи є його "частинами" (рис 3.3.17)



Рис. 3.3.17. Графічне зображення відношення композиції

Практичним прикладом відношення композиції може служити вікно графічного інтерфейсу програми, що може складатися з рядка заголовка, смуг прокручування, головного меню, робочої області й строки стану. Подібне вікно являє собою клас, проте його складові елементи також є окремими класами. Остання обставина характерна для відношення композиції, оскільки відображає різні способи зображення даного відношення.

Для відношення композиції й агрегації можуть використатися додаткові позначення, застосовані для відношення асоціації. А саме, можуть бути зазначені кратності окремих класів, що в загальному випадку не обов'язково. Стосовно до описаного вище прикладу клас Вікно програми є класом-композитом, а взаємозв'язки складових його частин можуть бути зображені діаграмою класів наступного вигляду (рис. 3.3.18).

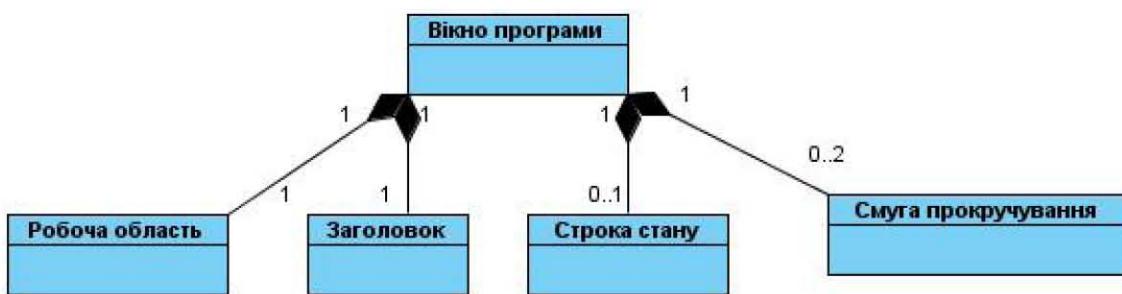


Рис. 3.3.18. Діаграма класів для ілюстрації відношення композиції на прикладі класу-композита Вікно програми

Нижченаведена діаграма ілюструє відносини композиції та агрегації. За допомогою агрегації показано, що клас Обліковий запис ("Account") використовує адресну книгу (AddressBook) проте не обов'язково містить її (рис. 3.3.19). Композиція підкреслює, що контакти (Contact) та групи контактів (ContactGroup) містяться в адресній книзі AddressBook.

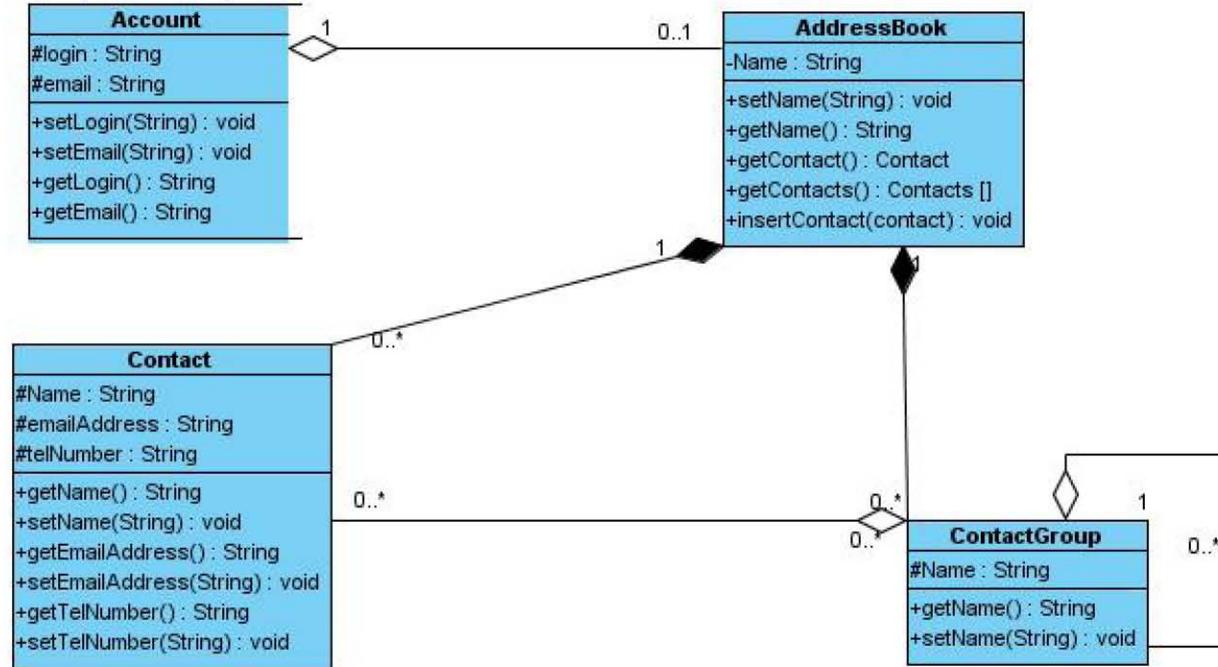


Рис. 3.3.19. Діаграма класів для ілюстрації відношення композиції та агрегації на прикладі роботи з адресною книгою

Ще одним прикладом відмінностей між застосуванням відношення агрегації та композиції може служити наступна діаграма взаємозв'язків між будівлею, приміщенням та обладнанням, яке розташоване у приміщенні (рис 3.3.20).

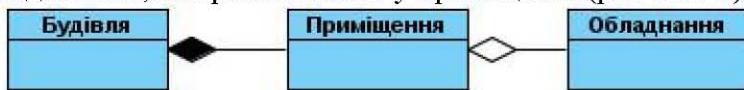


Рис. 3.3.20. Діаграма класів для ілюстрації відношення різниці між композицією та агрегацією

Зauważення: Для виявлення типу зв'язку – агрегація чи композиція варто проаналізувати, що буде з класом- частиною після знащення класу контейнеру. Чи матиме він окремий сенс в рамках поставленої задачі? Так, наприклад, після знащення Будівлі, Приміщення припиняє своє існування, а от Обладнання матиме сенс.

Відношення реалізації

Реалізація (Realization) - це семантичне відношення між класифікаторами, при якому один класифікатор визначає сутність, а інший гарантує її виконання.

Відносини реалізації в діаграмах класів зустрічаються між інтерфейсами і класами, що реалізовують їх. Зображається відношення реалізації у вигляді пунктирної лінії з незафарбованою стрілкою, як щось середнє між відносинами узагальнення та залежності (див. рис. 3.3.21).

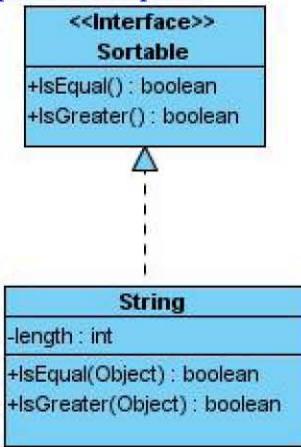


Рис. 3.3.21. Діаграма класів для ілюстрації відношення реалізації

Приклад:

Для наочності вищевикладеного матеріалу, розглянемо діаграму класів, яка ілюструє відому казку «Курка Ряба» (рис 3.3.22).

У казці Дід (клас Дід) та Баба (клас Баба) є Людьми (класи Дід та Баба наслідують від класу Людина), та вони є власниками (відношення асоціації) Курки (клас Курка) на ім'я Ряба. В свою чергу, Курка Ряба є твариною (тобто клас Курка наслідує від класа Тварина). Курка може відкладати яйця (клас Яйце, відношення асоціації «відкладати»). Зауважимо, що яйце в казці фігурувало як золоте, так і просте, що відображується на діаграмі за допомогою класу **<<enumeration>>** Вид. Ще однією дійовою особою казки є Миша (клас Миша), яка також є Твариною та володіє Хвостом (клас Хвіст) як інструментом (клас Інструмент) для розбиття яйця (відношення асоціації «розвиває»).

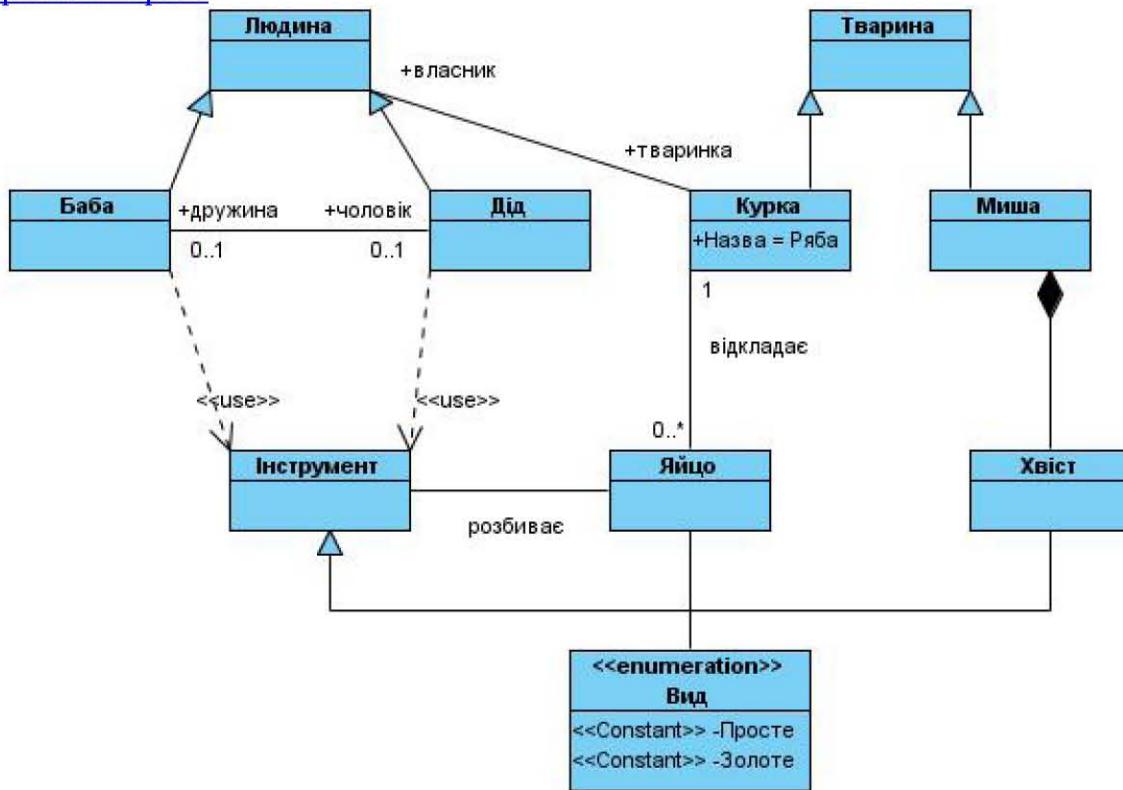


Рис. 3.3.22. Діаграма класів для ілюстрації казки «Курка Ряба»