**Лекція 6. Нотація UML. Структурне моделювання**

1. Модель класів

2. Компоненти синтаксису моделі класів

**1.                 Модель класів**

Моделювати систему краще з трьох різних точок зору, пов'язаних між собою. Кожна модель описує важливі аспекти системи, але для більш повного опису потрібно все три моделі:

– модель класів

– модель стану

– модель взаємодії

*Модель класів* становить статичні, структурні аспекти системи, пов'язані з даними.

*Модель стану* представляє тимчасові, поведінкові, управлінські аспекти системи.

*Модель взаємодії* представляє кооперацію окремих об'єктів, тобто всі аспекти системи, пов'язані з взаємодіями.

Розглянемо детальніше модель класів.

Класи – найбільш важливі будівельні блоки будь-якої об'єктно-орієнтованої системи.

*Клас –* це опис безлічі об'єктів з однаковими атрибутами, операціями, зв'язками і семантикою.

Моделювання системи включає в себе ідентифікацію сутностей, які важливі для конкретного уявлення. Ці сутності формують словник системи, яка моделюється. Кожна з цих сутностей відрізняється від іншої і має набір певних властивостей.

Наприклад, при побудові будинку вам як майбутньому власникові не байдуже, якими будуть стіни, двері, вікна, кабінети, системи освітлення та ін. Стіни мають якусь висоту, ширину і є суцільними. Двері характеризуються тими ж ознаками, але їх ще й відрізняється специфічною поведінкою: вони відкриваються в одну сторону. Вікна частково схожі з дверима, оскільки теж утворюють отвір в стіні, але їх функціональність неоднакова. Окремі стіни, двері і вікна рідко існують самі по собі, тому потрібно також розглянути, як конкретні екземпляри цієї суті поєднуються один з одним. В

 UML всі ці сутності моделюються класами. Клас – це абстракція сутності, що є частиною словника системи. Клас – не індивідуальний об'єкт, а уявлення чималої кількості об'єктів: скажімо, можна концептуально представляти «стіну» як клас об'єктів з низкою загальних властивостей (висота, довжина, товщина, чи є стіна несучої тощо).

Наприклад:



Рисунок 11.1 – Приклади класів

**2.     Компоненти синтаксису моделі класів**

*Операція* (operation) – це реалізація послуги, яка може бути запрошена у будь-якого об'єкта даного класу, щоб викликати певну його поведінку. Іншими словами, операція – це абстракція чогось, що ви можете зробити з конкретним об'єктом і взагалі з усіма об'єктами даного класу.

Клас може мати будь-яке число операцій або не мати жодної. Графічно операції представлені в розділі списку, наведеного під атрибутами класу. Допускається зазначення тільки імен операцій. Наприклад:



Щоб краще організувати довгі списки атрибутів і операцій, бажано забезпечити префіксом (ім'ям стереотипу) кожну категорію в них.



Обов'язок (responsibility) – це угода або зобов'язання класу. Коли ви створюєте клас, висувається припущення, що всі його об'єкти характеризуються однаковим станом і однаковою поведінкою.

На більш абстрактному рівні відповідні атрибути і операції є просто засобами, завдяки яким клас виконує свої обов'язки.

Наприклад, клас Wall (Стіна) відповідає за інформацію щодо висоти, ширини, товщини; клас FraudAgent (Агент За запобігання шахрайству), який можна зустріти в додатку, що обробляє кредитні карти, - за обробку запитів і визначення їх обґрунтованості, підозрілості або незаконність.

У класу може бути скільки завгодно обов'язків, хоча на практиці кожен добре структурований клас має як мінімум один обов'язок і як максимум – невеликий їх набір. Графічно обов'язки можуть бути представлені в спеціально відведеному для них розділі, в нижній частині піктограми класу.



Класи рідко існують самі по собі. Коли будуються моделі, то, як правило, фокусується увагу на групах класів, що взаємодіють один з одним. В UML такі спільноти класів формують кооперації і зазвичай візуалізують в діаграмах класів.

При моделюванні системи необхідно не тільки ідентифікувати сутності, що формують її словник, а й змоделювати відносини один до одного. В об'єктноорієнтованому моделюванні існують три види зв'язків між класами, які найбільш важливі: залежність, узагальнення, асоціації.

*Залежність* представляє зв'язки використання між класами (включаючи уточнення, трасування і зв'язування).

*Узагальнення*, яке пов'язує узагальнені класи з їх спеціалізаціями.

*Асоціації*, що описують структурні зв'язки об'єктів.

Кожна з цих різновидів представляє окремий спосіб комбінування абстракцій.

Розглянемо зв'язки щодо метафори будівництва будинку. Стіни, двері, вікна, вбудовані шафи і системи освітлення формують частину словника системи. Всі ці об'єкти (стіни, двері, вікна, шафи та освітлювальні прилади) в сукупності формують більш високорівневі сутності – кімнати.

*Залежність* являє собою зв'язок використання. Наприклад, труби залежать від водонагрівача для підігріву води, яка по ним передається.

*Асоціація* – це структурний зв'язок між екземплярами. Наприклад, кімнати складаються зі стін та інших об'єктів; в стіни вмонтовані двері і, можливо, вікна; через стіни можуть тягнутися труби.

Узагальнення пов'язує узагальнені класи з більш спеціалізованими і тому відомі як зв'язки успадкування («клас-підклас», або «батько-нащадок»). Наприклад, вітраж – це вікно з дуже великими, жорстко фіксованими панелями; патіо – різновид вікна, що відкривається убік.

Залежність (dependency) – це зв'язок, який встановлює, що одна сутність, наприклад клас Window (Вікно), використовує інформацію і сервіс (операцію або послугу), що подаються іншою сутністю, наприклад класом Event (Подія), але не обов'язково – навпаки.

Залежність зображується у вигляді пунктирної лінії зі стрілкою, спрямованої на залежну сутність.



Вибирається залежність, коли потрібно показати, що одна сутність використовує іншу. Найчастіше цей тип зв'язку застосовується для того, щоб показати, що один клас використовує операції іншого класу (або використовує змінні чи аргументи типу іншого класу).

*Узагальнення* (generalization) – це зв'язок між сутністю загального характеру (званої суперкласом, або батьком) і більш специфічною сутністю (званої подклассом, дочірнім класом або нащадком). Іноді узагальнення називають зв'язком типу «є». Графічно узагальнення представлено суцільною лінією зі стрілкою в формі великого порожнього трикутника, що вказує на батька. Використовуйте узагальнення, коли необхідно зобразити зв'язок «батько-нащадок».

Клас, що не має батьків, але має одну або кількох нащадків, називається *кореневим* (root) або *базовим*. Клас, що не має нащадків, називається *листовим* (leaf).

Про клас, у якого є тільки один батько, кажуть, що він використовує *одиночне спадкоємство*, на відміну від класу, у якого більш ніж один батько – *множинне спадкування*.

Наприклад:



*Асоціація* – це структурний зв'язок, який вказує, що об'єкти однієї сутності з'єднуються з об'єктами іншої. Так, маючи асоціацію між двома класами, можна з'єднати об'єкти одного класу з об'єктами іншого.

Цілком припустимо, щоб обидва кінці асоціації з'єднували один і той же клас – іншими словами, один об'єкт класу може зв'язуватися з іншим об'єктом того ж класу. Асоціація, що зв'язує два класи, називається *бінарною*. Зустрічаються так звані *n-арні асоціації*, які з'єднують більше двох класів. Графічно асоціація представлена суцільною лінією, два кінці якої з'єднують один або різні класи.



Асоціація може мати ім'я, яке використовується для опису природи зв'язку. Тому значення імені не повинно бути двозначним. Використовуючи стрілочку в формі трикутника, ви можете вказати напрямок, в якому слід читати це ім'я.

Коли клас бере участь в асоціації, він виконує в зв'язку з цим конкретну роль. Роль – це «обличчя» класу, яка знаходиться на дальньому кінці асоціації, представлене класу, що знаходиться на її ближньому кінці. Можна явно іменувати роль, яку виконує клас в асоціації. Роль, яку відіграє клас, що знаходиться на кінці асоціації, називається кінцевим ім'ям.

Наприклад, клас Person (Людина), який грає роль employee (працівник), асоційований з класом Company (Компанія), що грає роль employer (роботодавець).



У багатьох ситуаціях моделювання важливо знати, скільки об'єктів може бути з'єднане одним екземпляром асоціації. Цей параметр називається множинністю ролі асоціації. *Множинність* (multiplicity) представляє діапазон цілих чисел, який вказує можливу кількість пов'язаних об'єктів.

Множинність може бути визначена як одиниця (1), нуль або один (0..1), багато (0 .. \*), один або кілька (1 .. \*). Можна задавати діапазон цілих значень, наприклад 2..5, або встановлювати точне число, наприклад 3 (еквівалент записи 3..3).

Наприклад, кожна компанія може наймати одного або декількох осіб (множинність 1 .. \*); кожній людині зіставлено 0 або більше компаній-роботодавців (множинність \* – еквівалент записи 0 .. \*).



Проста асоціація між двома класами представляє структурну зв'язок між рівноправними елементами: обидва класу концептуально знаходяться на одному рівні – жоден з них не може вважатися важливіше іншого.

Можна також змоделювати зв'язок «ціле-частина», в якій один клас представляє велику сутність (ціле), що містить в собі більш дрібні (частини). Цей тип зв'язків, заснованих на відносинах володіння, називається *агрегацією* і має на увазі, що об'єкт-ціле володіє об'єктами-частинами.

По суті, *агрегація* – це особливий вид асоціації, тому зображується вона лінією простої асоціації, до якої доданий порожній ромб з боку об'єкта-цілого.



Найпоширеніший вид залежності – це зв'язок класу, який використовує інший клас в якості параметра своєї операції. Щоб змоделювати цей зв'язок використання, необхідно створити залежність, спрямовану від класу з операцією до класу, що використовується в якості її параметра.

Наприклад:



Розглянемо приклади діаграм класів.

Приклад 1.



Приклад 2.



Приклад 3.



Приклад 4.



Приклад 5.



**Запитання для самоперевірки**

1. Для чого використовується нотація UML?

2. Що таке клас?

3. Що таке атрибути? Для чого вони використовуються?

4. Що таке операції? Для чого вони використовуються?

5. Що таке обов'язок? Для чого він використовуються?

6. Які три види зв'язків між класами ви знаєте?

7. Коли використовується кожен вид зв’яку?