**Практична робота №8. Поглиблене знайомство з діаграмами класів**

**Мета: поглиблене вивчення діаграм класів та методів їх застосування засобами мови UMLв процесі проектування.**

**На період** карантину в дистанційній формі навчання на надані в кінці запитання потрібно надати письмові відповіді, надіславши їх на електронну адресу викладача. Файл надавати з іменем у форматі

**PI<Номер групи><Номер лекції / практичної / лабораторної>[-<Номер завдання>][літера позначення типу роботи L – лекція, P –практична, R – лабораторна]<Прізвищеанглійською>**. Наприклад, **PI3104L**buts.doc. Відповіді на запитання повинні бути не довгими і змістовними. Не копіюйте фрагментів з різних інформаційних джерел, подумайте і викладіть свою точку зору. При наявності відповідей-"близнюків" відповідь буде зараховуватися першому за часом надсилання.

**Строк виконання цієї роботи ІПЗ-31, ІПЗ-32 23.04.2020**

**Завдання:**

1. Опрацювати теоретичні відомості. Перевірити засвоєння Вами матеріалу на контрольних запитаннях.
2. Розробити діаграми класів для:

* для покупця книжок через Інтернет із визначенням атрибутів та операцій;
* для замовлення пацієнта поліклініки на виклик лікаря із визначенням атрибутів та операцій

1. Побудуйте діаграму класів для відношення узагальнення класу "Користувач" для відділу кадрів.

Приклади визначення класів та їх складу.

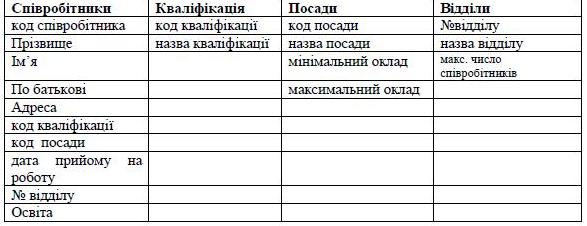
Покупка книжок через інтернет



Замовлення на виклик лікаря



Відділ кадрів



**Контрольні запитання**

1. Які види зв’язків між сутностями представлені на діаграмах UML?
2. Коротко охарактеризуйте специфікацію атрибута класу.
3. Коротко охарактеризуйте специфікацію операції класу.
4. Коротко охарактеризуйте можливі застосування відношення асоціації на діаграмах класів.
5. Що таке асоційований клас і багатополюсна асоціація?
6. Коротко охарактеризуйте можливі застосування відношення агрегації на діаграмах класів.
7. Коротко охарактеризуйте можливі застосування відношення композиції на діаграмах класів.
8. Коротко охарактеризуйте можливі застосування відношення узагальнення на діаграмах класів.

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

***Діаграма класів (class diagram) ‒ основний спосіб опису структури системи,*** оскільки UML в першу чергу об’єктно-орієнтована мова, і класи є основний "будівельний матеріал".

На діаграмах класів показано різноманітні класи, які утворюють систему і їх взаємозв’язки. Діаграми класів називають “статичними діаграмами”, оскільки на них показано класи разом з методами і атрибутами, а також статичний взаємозв’язок між ними: те, яким класам “відомо” про існування яких класів, і те, які класи “є частиною” інших класів, — але не показано методи, які при цьому викликаються. На рис.1 наведено показ в Umbrello UML Modeller діаграми класів.

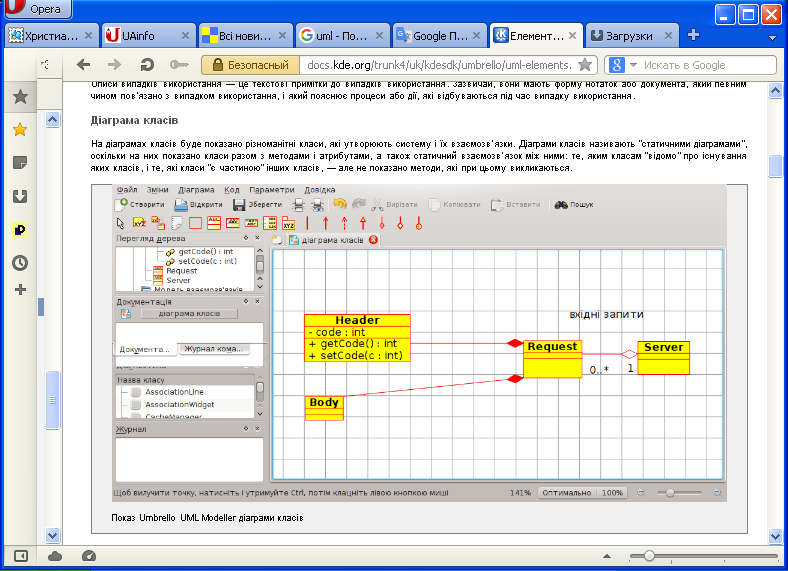


Рисунок 1. Показ в Umbrello UML Modeller діаграми класів

На діаграмі класів застосовується один основний тип сутностей:

класи (***1***) (включно з різними окремими випадками класів: інтерфейси, примітивні типи, класи-асоціації та багато інших),між якими встановлюються такі основні типи відношень:

асоціація між класами (***2)*** (с великою кількістю додаткових відомостей);

узагальнення між класами (***3)***;

залежності (різних типів) між класами (***4)*** та між класами та інтерфейсами.

Деякі елементи нотацій, застосовуваних на діаграмі класів, показано на рис.2.

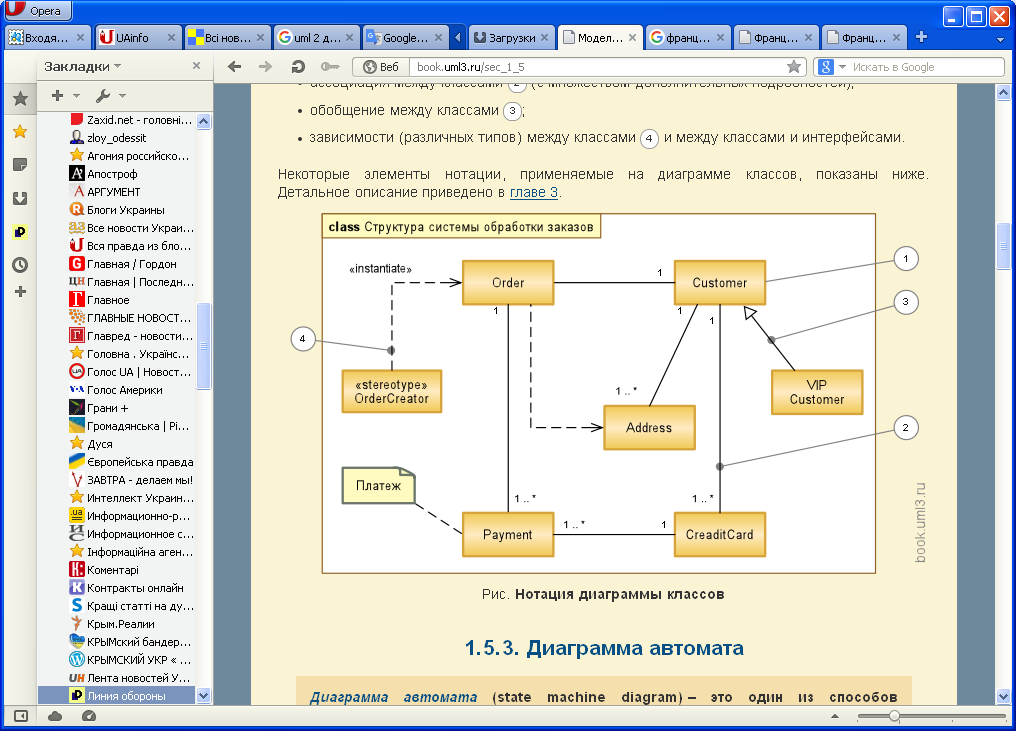
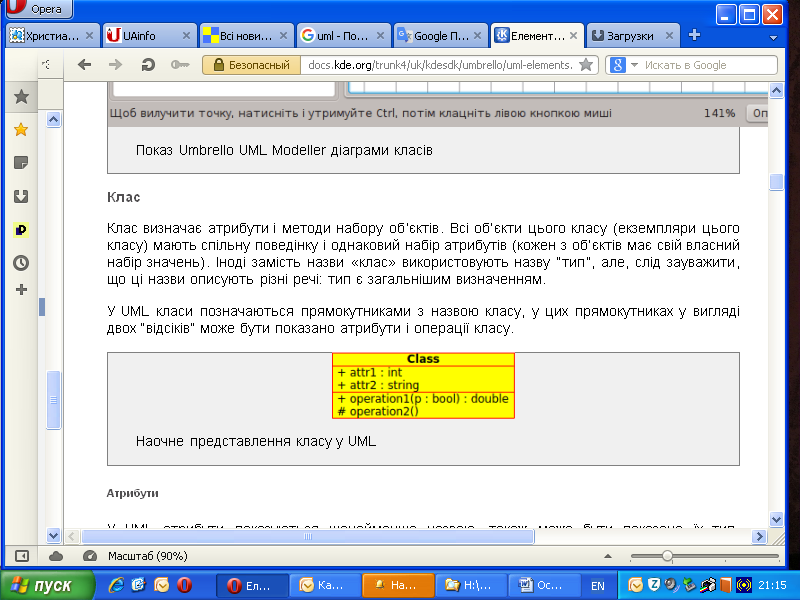


Рис.2. Нотація діаграми класів

У зображенні класу виокремлюють три секції, в яких послідовно записують *назву*, *атрибути* та *операції* (рис.3). Кожну секцію, крім першої, можна опустити . Назва класу має бути унікальною в межах пакета, в якому розташовано клас. Назви *абстрактних* класів в UML позначають курсивом.

Рисунок 3. Наочне представлення класу у UML

Перед іменем класу може стояти стереотип, який може визначатися так:

| Стереотип | Опис |
| --- | --- |
| «actor» | Актор, дійова особа |
| «auxiliary» | Допоміжний клас |
| «enumeration» | Тип даних - перелічення |
| «exception» | Виключення (тільки в UML 1) |
| «focus» | Основний клас |
| «implementationClass» | Реалізація класу |
| «interface» | Всі складові абстрактні |
| «metaclass» | Екземпляри є класи |
| «powertype» | Метаклас, екземплярами якого є всі нащадки даного класу (тільки в UML 1) |
| «process» | Активний клас |
| «thread» | Активний клас (тільки в UML 1) |
| «signal» | Клас, екземплярами якого є сигнали |
| «stereotype» | Новий елемент на основі існуючого |
| «type» | Тип даних |
| «dataType» | Тип даних |
| «utility» | Нема екземплярів, служба |

Зазвичай, клас може налічувати довільну кількість екземплярів. Однак виникають ситуації, за яких число екземплярів класу необхідно обмежити. Найчастіше потрібно задавати клас, у якого:

• немає жодного екземпляра – службовий клас (*Utility*);

• рівно один екземпляр – синглетний клас (*Singleton*);

• строго визначена кількість екземплярів;

• довільна кількість екземплярів – варіант за домовленістю.

Кількість екземплярів класу називають *кратністю* класу (*multiplicity*), яку задають у специфікації класу. Кратність класу, залежно від інструментального засобу, може відображатися за допомогою спеціальної позначки (малий пунктирний квадратик) або числа у правому верхньому куті прямокутника, який зображає клас.

*Атрибут* – це позначене місце (*слот*), в якому може/можуть зберігатися значення. Загальний формат атрибутів такий:

**[<видимість>] <назва> [<кратність>] [:<тип>] [=<початкове значення>][<властивості>]**

*Видимість* атрибута визначає рівень доступу до атрибута і найчастіше набуває таких значень:

• *відкритий* (*public*) – передбачає необмежений доступ до атрибута з боку інших класів і позначається символом “+”;

• *захищений* (*protected*) – передбачає доступ до атрибута лише класам, які його наслідують, і позначається символом “#”;

• *закритий* (*private*) – забороняє доступ до атрибута всім іншим класам і позначається символом “–”.

*Назва* атрибута має бути унікальною для певного класу. *Тип* атрибута бажано задавати відповідно до типів даних у мові програмування, на якій реалізовуватимуть модель. *Кратність* визначає *атрибут-масив* для зберігання множини значень (здебільшого, кратність обмежується квадратними дужками). Для атрибута також може бути вказано його *початкове значення*.

Можна задавати такі *властивості* атрибута:

• changeable (змінюваний) – за домовленістю;

• addOnly (тільки долучення) – можна долучати нові значення для *атрибута-масиву*, без можливості наступних змін;

• frozen (заморожений) – після ініціалізації значення атрибута не змінюється (відповідає слову const у С++).

*Статичні атрибути* (або *атрибути класу*) на відміну від звичайних атрибутів належать не окремим об’єктам, а класу загалом, і позначаються на діаграмах класів підкресленням.

Таким чином, атрибутом класу називають іменовану властивість класу, що описує безліч значень, які можуть приймати екземпляри цієї властивості. Клас може мати будь-яке число атрибутів (зокрема, не мати жодного атрибута). Властивість, що виражається атрибутом, є властивістю модельованої сутності, загальною для всіх об'єктів даного класу. Тобто атрибут є абстракцією стану об'єкта. Будь-який атрибут будь-якого об'єкта класу повинен мати деяке значення.

Імена атрибутів представляються у розділі класу, розташованому під ім'ям класу. Хоча UML НЕ накладає обмежень на способи створення імен атрибутів (ім'я атрибута може бути довільною текстової рядком), на практиці рекомендується використовувати короткі прикметники та іменники, що відображають смисл відповідної властивості класу. Перше слово в назві рекомендується писати з великої літери, а всі інші слова - з прописної.

Операції дають змогу закласти в клас та його об’єкти деяку поведінку. Загальний формат запису операцій наступний:

**[<видимість>]<назва>([список параметрів])[:<тип результату>]**

*Видимість* операції визначає рівень доступу до операції і позначається аналогічно до видимості атрибутів. *Назва* операції має бути унікальною для певного класу. *Тип результату* визначає тип значення, яке повертає операція, і повинен відповідати типам даних у мові програмування, яку обрано для реалізації моделі.

*Список параметрів* дає змогу специфікувати інтерфейс операції. Кількість параметрів у списку може бути довільною.

Кожен параметр у загальному випадку записують так:

**[<напрям>] <назва> [:<тип>][= <значення за домовленістю>]**

*Напрям* визначає характер використання параметра в операції і найчастіше набуває таких значень:

• in – параметр використовується як *вхідний*;

• out – параметр використовується як *вихідний*;

• inout – параметр використовується як вхідний і вихідний одночасно.

*Назва параметра* має бути унікальною для певної операції. *Тип* параметра варто визначати з огляду на типи даних в обраній мові програмування. *Значення за домовленістю* відображає значення параметра, яке буде прийняте в операції, якщо цей параметр буде опущено.

*Абстрактні операції* позначають на діаграмах класів курсивом. Якщо клас має хоча б одну абстрактну операцію, то його вважають абстрактним.

*Статичні операції* (або *операції класу*) належать цілому класу, працюють зі статичними атрибутами класу і позначаються на діаграмах підкресленням.

*Відповідальності* класу визначають його функції та роль в системі і формуються на початкових стадіях моделювання. Зазвичай відповідальності класів зазначають на загальних діаграмах, які містять лише концептуальні класи системи, і слугують для створення початкового уявлення про розподіл функціональності за класами. Кожен клас може мати одну або більше відповідальностей.

***Асоціації класів (Категорії зв'язків)***

Відношення *асоціації* на діаграмах класів трапляються найчастіше. Асоціацію позначають суцільною лінією (може завершуватися однією чи двома стрілками), що з’єднує класи. Асоціація означає, що екземпляри одного класу взаємодіють з екземплярами іншого класу під час виконання програми. Оскільки екземплярів класу може бути багато і кожен може взаємодіяти з декількома екземплярами іншого класу, то асоціація є *дескриптором*, що описує множину зв’язаних об’єктів (екземплярів асоціації).

Зв’язок між об’єктами у програмі може бути організований довільними способами. Можливість зв’язку означає, що об’єкт одного класу може надіслати повідомлення об’єкту іншого класу, зокрема, активізувати операцію або прочитати/змінити значення відкритого атрибута. Оскільки в об’єктно-орієнтованій програмі такі дії складають суть виконання програми, то виявлення асоціацій є однією з ключових задач під час її складання.

Базова нотація асоціації (суцільна лінія) засвідчує, що об’єкти асоційованих класів (*полюси асоціації*) взаємодіють між собою під час виконання програми. Для асоціацій в UML передбачено чимало *доповнень*. Доповнення не є обов’язковими: їх використовують за необхідності, у різних ситуаціях по-різному. Якщо використовувати усі доповнення одразу, то діаграма класів стає перевантаженою настільки, що її важко читати і розуміти. Розглянемо найважливіші доповнення:

• *Назва асоціації* (зазначається посередині над/під лінією) – ідентифікатор, який позначає асоціацію у моделі. Додаткового семантичного навантаження ця назва не несе, отож, зазвичай, не використовується (за винятком випадку, коли два класи зв’язані декількома асоціаціями). Задане ім'я зазвичай характеризує природу зв'язку.

• *Роль полюса асоціації* (або *специфікатор інтерфейсу*) – конкретизує асоціацію щодо певного класу. Роль зазначається поблизу лінії неподалік класу, набуває вигляду:

**[<видимість>] <назва ролі> [: <тип>]**

Використовується для ідентифікації полюсів у випадку само-асоціації (клас зв’язується сам з собою) або у випадку, коли два класи зв’язані декількома різними асоціаціями.

• *Кратність полюса асоціації* – визначає кількість об’єктів певного класу (з боку полюса), що беруть участь у зв’язку під час виконання програми. Кратність може задаватися конкретним числом, і тоді у кожному зв’язку з боку цього полюса беруть участь рівно стільки об’єктів, скільки задано. Найчастіше кратність задано у вигляді діапазону можливих значень:

**<нижня\_межа> .. <верхня\_ межа>**

Кратність полюса або верхню межу може зображати символ “\*”, який позначає довільну кількість екземплярів класу. При-клади допустимих діапазонів: 1..5, 1..\*, \*, 0..4. В наданій нижче таблиці надані приклади вираження кратності.

| Вираз кратності | Множина може мати |
| --- | --- |
| 0..\* або \* | Довільну кількість елементів |
| 1..\* | Один або більше елементів |
| 0..1 | Не більше одного елемента |
| 1..10 | Від одного до десяти елементів |
| 1..3, 5, 7..10 | Один, два, три, п’ять, сім, вісім, дев’ять або десять елементів |
| 5..3 | Некоректна кратність. Нижня границя більша за верхню |
| -1..3 | Некоректна кратність. Від’ємні числа недопустимі |

• *Сортування на полюсі асоціації* – визначає тип упорядкування об’єктів на полюсі з кратністю понад 1. Можливі значення: *упорядковано* (ordered або sorted), *неупорядковано* (unordered або unsorted). За домовленістю – *неупорядковано*.

• *Змінюваність на полюсі асоціації* (Is changeable) – визначає можливість/неможливість зміни кількості екземплярів класу з боку полюса під час роботи програми (у межах специфікованої кратності).

• *Напрям навігації на полюсі асоціації* (Is navigable) – визначає можливість/ неможливість доступу до екземплярів класу з боку полюса за допомогою певної асоціації. Ті полюси, через які навігація можлива, на діаграмах позначають стрілками. У деяких CASE-засобах відсутність стрілок означає можливість навігації у будь-якому напрямку. На рис.4 показано приклад подання асоціації.

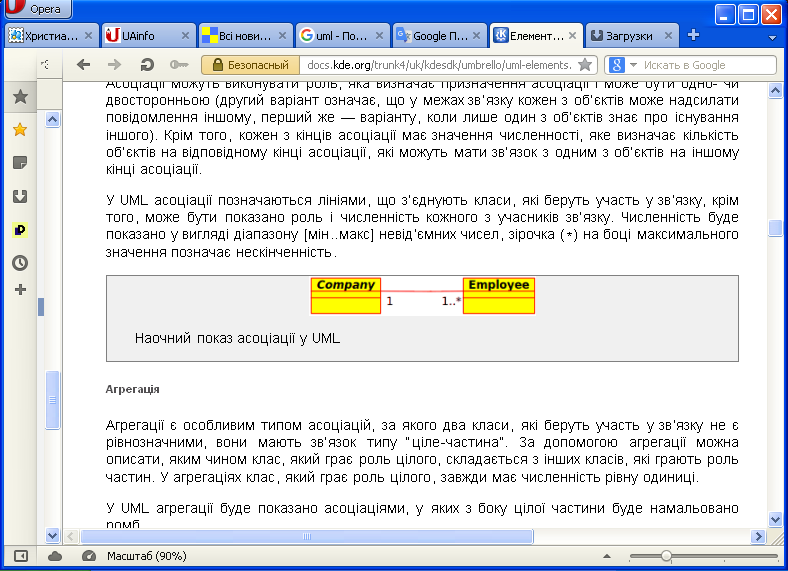


Рисунок 4. Наочний показ асоціації у UML

Найчастіше використовують *бінарні асоціації* (*Binary Asso-ciation*) – відношення між двома класами або рефлексивне відношення класу із самим собою. На рис. 5 зображено бінарну асоціацію з назвою *Містить* типу “один до багатьох” між класами *Авіарейс* і *Місце* (у літаку).

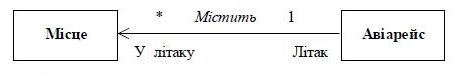


Рисунок 5. Приклад бінарної асоціації між класами

Можлива ситуація, коли один екземпляр класу асоціюється з декількома екземплярами того ж самого класу. Нехай один екземпляр класу *ЧлениКафедри* представляє завідувача кафедри, а інші екземпляри цього класу – викладачів. Завідувач кафедри *керує* викладача-ми. Зв’язок, який виникає між екземплярами одного і того ж класу, називають *самоасоціацією* (рис. 6).

Рисунок 6. Приклад самоасоціації

Здебільшого асоціація між класами *А* і *В* – це множина пар (*а*; *b*), де *а* – об’єкт класу *А*, а *b* – об’єкт класу *В*. Ці пари є *екземплярами* асоціації (зв’язками). Отже, асоціація є *класом*, який ще може володіти іншими додатковими властивостями, які залежать від асоціації.

Для визначення цих властивостей асоціації використовують *асоційований клас* (*аssociaton class*), який зображається, зазвичай, прямокутником і зв’язується з відповідною асоціацією штриховою лінією. Назви асоціації та асоційованого класу *мають збігатися*. Розглянемо, наприклад, постачання деталей для виготовлення деякого виробу (рис. 7).

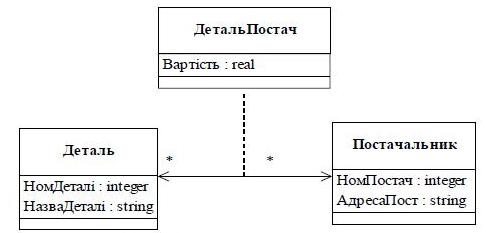


Рисунок 7. Приклад асоційованого класу

*Багатополюсна асоціація* (*N-аry Association*) – це зв’язок між трьома та більше класами. Такий зв’язок графічно зображають ромбом, до якого підходять лінії, що з’єднують ромб із класами. Клас, що описує зв’язок, приєднується до ромба за допомогою штрихової лінії. Приклад такої асоціації зображено на рис. 8.

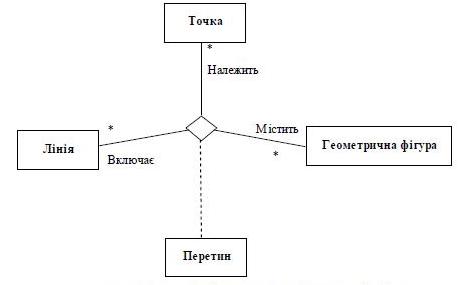


Рисунок 8. Приклад багатополюсної асоціації

**Агрегація та композиція між класами**

У мові UML використовують два часткові й дуже важливі випадки відношення асоціації – *агрегацію* та *композицію*. В обох випадках йдеться про моделювання відношення типу “частина – ціле”. Відношення такого типу є відношеннями асоціації, оскільки частини і ціле, зазвичай, взаємодіють між собою.

*Агрегація* (*Aggregation*) від класу *А* до класу *В* означає, що об’єкти (один чи декілька) класу *А* входять до складу об’єкта класу *В*. На діаграмі класу це відзначається за допомогою спеціального графічного *доповнення*: на полюсі асоціації з боку “цілого” (у нашому випадку клас *В*) зображається порожній ромбик.

При агрегації жодних додаткових обмежень не накладають: об’єкт класу *А* (“частина”) може бути зв’язаний відношеннями агрегації з іншими об’єктами (тобто брати участь у декількох агрегаціях), може створюватися і знищуватися незалежно від об’єкта класу *В* (“цілого”). За допомогою агрегації можна описати, яким чином клас, який грає роль цілого, складається з інших класів, які грають роль частин. У агрегаціях клас, який грає роль цілого, завжди має кратність рівну одиниці. У UML агрегації буде показано асоціаціями, у яких з боку цілої частини буде намальовано ромб. На рис. 9 зображено агрегацію між класом *Car – автомобіль* (“ціле”) і класом *Wheel – колесо* (“частина”).

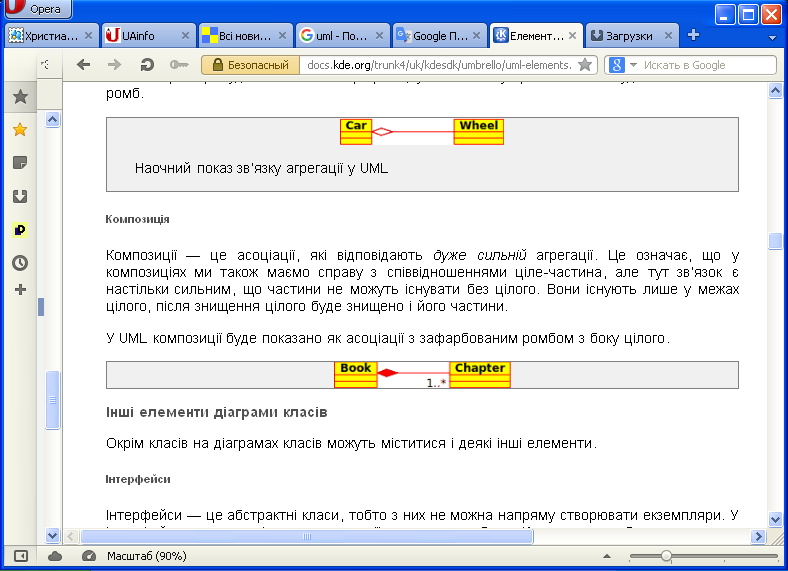


Рисунок 9. Наочний показ зв’язку агрегації у UML

*Композиція* (*Composition*) є посиленою формою агрегації і створюється на основі бінарної асоціації. *Композиція* накладає дещо сильніші обмеження: композиційно “частина” може входити тільки в одне “ціле”, “частина” існує доти, доки існує “ціле”, і припиняє своє існування разом з “цілим”. Графічно відношення композиції зображають зафарбованим ромбиком.

“Ціле” у композиції слугує *класом-власником*, а “частина” – підпорядкованим класом. Підпорядковані класи, що належать до композиції, не можуть водночас налічувати декілька класів-власників. *Об*’*єкт-власник* і його складові частини (*підпорядковані об*’*єкти*) не можуть існувати окремо. Усі підпорядковані об’єкти змінюються разом із об’єктом-власником. Графічно, композицію зображають як “дерево”, коренем якого є клас-власник, а листками – підпорядковані класи. На рис. 10 зображено частину композиції класу *Автомобіль*.

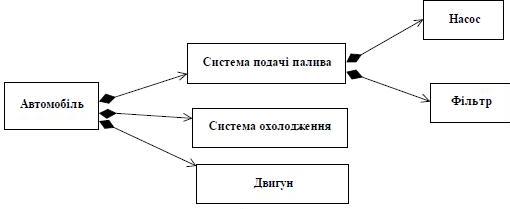


Рисунок 10. Приклад композиції

**Узагальнення та залежності між класами**

*Узагальнення* (*generalіzatіon*) – це відношення між двома сутностями, одна з яких є *частковим* (або *спеціалізованим*) випад-ком іншої. Відношення узагальнення передбачає виконання *принципу підстановки*:

якщо сутність *А* – *загальне* (або *батько* – parent, *предок*) є узагальненням сутності *В* – часткове (або *дитина* – chіld, *нащадок*), то *В* може бути підставлене замість *А*.

Графічно відношення узагальнення зображають лінією з незафарбованою стрілкою, яка вказує на предка (рис. 11).

Відношення *успадкування* між класами в об’єктно-орієнтованому програмуванні є типовим прикладом узагальнення, за якого об’єкт спеціалізованого класу (нащадок) може бути підставлений замість об’єкта узагальненого класу (батька, предка).

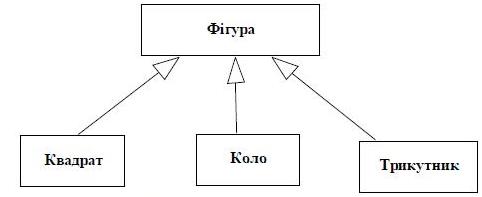


Рисунок 11. Приклад узагальнення між класами

Таким чином, зв'язком-узагальненням називається зв'язок між загальною сутністю, названою суперкласом, або батьком, і більш спеціалізованої різновидом цієї сутності, названим підкласом, або нащадком. Узагальнення іноді називають зв'язками «is a», маючи на увазі, що клас-нащадок є окремим випадком класу-предка. Клас-нащадок успадковує всі атрибути і операції класу-предка, але в ньому можуть бути визначені додаткові атрибути та операції.

Об'єкти класу-нащадка можуть використовуватися скрізь, де можуть використовуватися об'єкти класу-предка. Це властивість називають поліморфізмом по включенню, маючи на увазі, що об'єкти нащадка можна вважати включеними в безліч об'єктів класу-предка. Графічно узагальнення зображуються у вигляді суцільної лінії з великою незафарбовані стрілкою, спрямованої до суперкласу.

У UML зв‘язок Узагальнення між двома класами розташовує їх у вузлах ієрархії, яка відповідає концепції успадкування класу-нащадка від базового класу. У UML узагальнення буде показано у вигляді лінії, яка поєднує два класи, зі стрілкою, яку спрямовано від базового класу.

*Залежність* (*dependency*) – це найузагальненіше відношення між двома сутностями, яке вказує на те, що зміна *незалежної сутності* якось впливає на *залежну сутність*. Графічно відношення залежності зображають пунктирною стрілкою, спрямованою від залежної сутності до незалежної (рис. 12). Зазвичай, семантика конкретної залежності уточнюється в моделі за допомогою додаткової інформації.

На рис. 12 зображено залежність *Області малювання* від *Геометричної фігури*. Подібний зв’язок ніяк не регламентує тип відношення між об’єктами, а вказує лише на те, що залежний клас *Геометрична фігура* використовує якісь особливості реалізації іншого класу *Область малювання*.



Рисунок 12. Приклад залежності між класами

Під час моделювання класів залежності найчастіше використовують, щоб відобразити у сигнатурі операції той факт, що один клас використовує інший клас (незалежну сутність) як аргумент (рис. 13). У цьому випадку зміна одного класу знайде своє відображення при роботі іншого, оскільки незалежний клас може представити новий інтерфейс і/або поведінку.

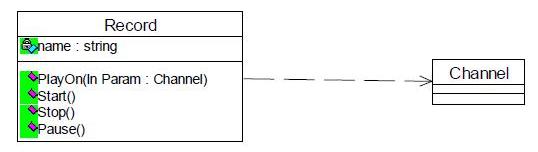


Рисунок 13. Виклик класу залежним класом

Залежності можуть мати назви, хоча їх рідко використовують – у випадках, коли модель має чимало залежностей і доводиться на них посилатися. Здебільшого, для ідентифікації залежностей використовують стереотипи.

Табл. **Стандартні стереотипи залежностей на діаграмі класів**

| Стереотип | Опис |
| --- | --- |
| «bind» | Підстановка параметрів в шаблон. Незалежною сутністю є шаблон (клас з параметрами), а залежною ‒ клас, який отримується з шаблона через завдання аргументів. |
| «call» | Вказує залежність між двома операціями: операція залежного класу викликає операцію незалежного класу. |
| «derive» | Буквально означає "може бути обчислений по". Залежність з цим стереотипом застосовується не тільки до класів, але й до інших елементів моделі: атрибутам, асоціаціям тощо. Суть полягає в тому, що залежний елемент може бути відновлений по інформації, яка міститься в незалежному елементі. Таким чином, ця залежність показує, що залежний елемент, є надлишковим і введений до моделі для зручності, наочності тощо. |
| «friend» | Призначає спеціальні права видимості. Залежний клас має доступ до складових незалежного класу, навіть якщо по загальним правилам видимості такі права у нього відсутні. |
| «instanceOf» | Вказує, що залежний об’єкт (або клас) є екземпляром незалежного класу (метакласу). |
| «instantiate» | Вказує, що операції незалежного класу створюють екземпляри залежного класу. |
| «powertype» | Вказує, що екземплярами залежного класу є підкласи незалежного класу. Таким чином, в цьому випадку залежний клас є метакласом. |
| «refine» | Вказує, що залежний клас уточнює (конкретизує) незалежний. Ця залежність показує, що зв’язані класи концептуально співпадають, але знаходяться на різних рівнях абстракції. |
| «use» | Залежність самого загального виду, яка показує, що залежний клас якимось чином використовує незалежний клас. |

**Інші елементи діаграми класів**

Окрім класів на діаграмах класів можуть міститися і деякі інші елементи.

**Інтерфейси**

Інтерфейси — це абстрактні класи, тобто з них не можна напряму створювати екземпляри. У інтерфейсах можуть міститися операції, але не атрибути. Класи можуть бути нащадками інтерфейсів (за допомогою асоціації реалізації), а з цих діаграм можна потім створювати сутності.

**Типи даних**

Типи даних — це базові елементи, з яких типово будується мова програмування. Типовими прикладами є цілі числа і булеві значення. Вони не можуть мати зв’язків з класами, але класи можуть мати зв’язки з ними.

**Переліки**

Переліки є простими списками значень. Типовим прикладом є перелік днів тижня. Пункти переліків називаються літералами переліків. Подібно до типів даних, переліки не можуть мати зв’язків з класами, але класи можуть мати зв’язки з переліками.

**Пакунки**

Пакункам відповідають простори назв у мовах програмування. На діаграмі пакунки використовуються для позначення частин системи, у яких міститься декілька класів, може навіть сотні класів.