Розширені класи, розширені зв’язки та інтерфейси

Розширені класи, розширені зв’язки та інтерфейси

Зміст

1. Розширені класи
2. Розширені зв’язки
3. Інтерфейси, типи та ролі

Вступ

Класи – це найбільш важливі будівельні блоки любої об’єктно-орієнтовної системи. Але вони представляють собою лише одну із різновидностей більш загальних будівельних блоків UML – класифікаторів.
Найбільш важливий вид класифікаторів в UML – це клас, який представляє собою опис набору об’єктів з однаковими атрибутами (полями), операціями (методами), зв’язками і семантикою.

Розширені класи, розширені зв’язки та інтерфейси

Зміст

1. Розширені класи
2. Розширені зв’язки
3. Інтерфейси, типи та ролі

Розширені класи

Класифікатор – це механізм, що описує структурні властивості та властивості поведінки.
Інші види класифікаторів:
інтерфейс – набір операцій, який використовується для специфікації сервісу класа, чи компонента;
тип даних – тип, значення якого не змінні (числа, строки та інші);
асоціація – опис набору посилань, кожна з яких з'єднує ряд об'єктів;
сигнал – специфікація асинхронного повідомлення, яке передається між екземплярами;
компонент – модульна частина системи, яка скриває свою реалізацію за набором зовнішніх інтерфейсів;
вузол – фізичний елемент, який існує під час виконання і який представляє обчислювальний ресурс (має пам'ять та процесор);
варіант використання – опис послідовності дій (включаючи їх різновид), які здійснює система і які породжують значимий результат для певної діючої особи;
підсистема – компонент, який представляє важливу частину системи.

Класифікатори

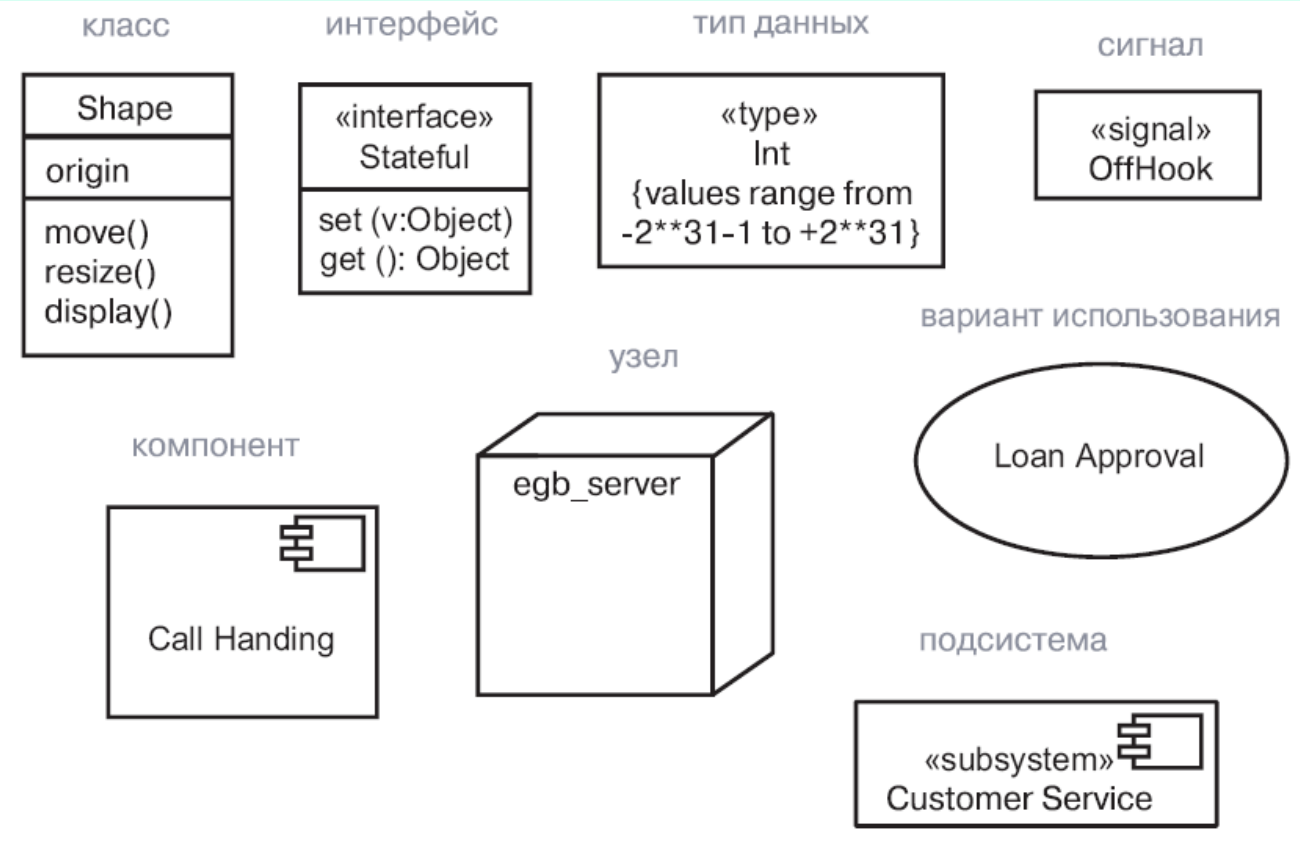


Рис. 1. Класифікатори.

В UML ряд класифікаторів має свої особисті піктограми, а для інших використовуються спеціальні ключові слова.

Видимість

Видимість (visibility) – це властивість, яка вказує на можливість використання даного атрибуту чи операції іншими класифікаторами.

public (відкритий) – Любий зовнішній класифікатор, якому видно даний, може використовувати цей атрибут чи операцію. Позначається символом “+”.
protected (захищений) – Любий спадкоємиць класифікатора може використовувати цей атрибут чи операцію. Позначається символом “#”.

private (закритий) – Цей атрибут чи операцію може використовувати тільки сам класифікатор. Позначається символом “–”.

package (пакетний) (в C# - internal) – Тільки класифікатори, які об’являються у тому ж пакеті, можуть використовувати цей атрибут чи операцію. Позначається символом “~” (тільда).

Видимість

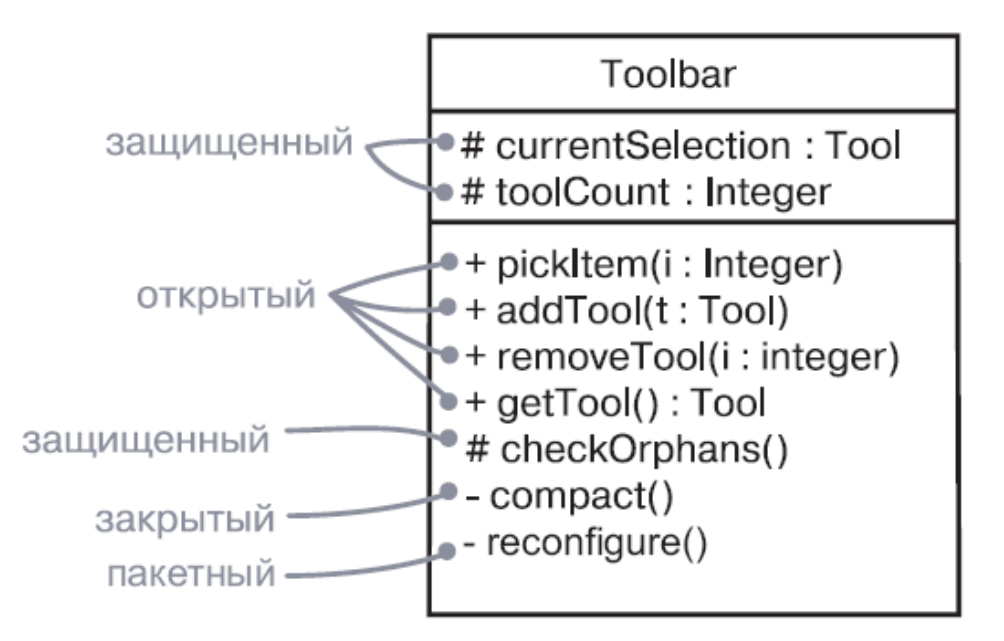


Рис. 2. Видимість.

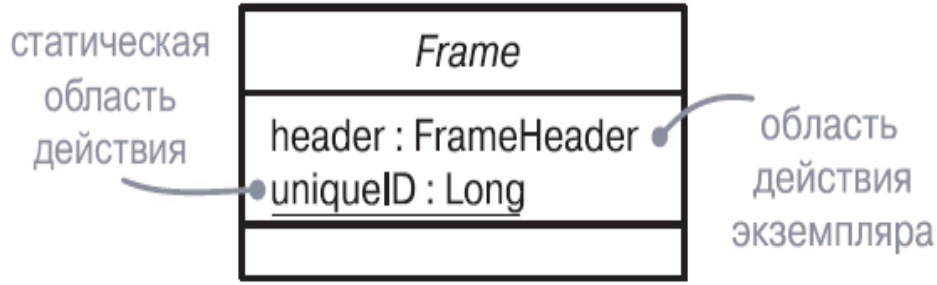
Видимість

Вказуючи видимість засобів класифікатора, ви зазвичай приховуєте деталі його реалізації та робите видимими тільки ті члени, які необхідні для виконання ним своїх обов’язків. Це основа приховування інформації, яка забезпечує побудову міцної стійкої системи.

Галузь дії екземпляра та статична галузь дії

В UML існує дві галузі дії (scope) (як в C++ та Java):
instance – галузь дії екземпляра. Кожний екземпляр класифікатора має власне значення властивості. Цей варіант прийнято за замовченням і не потребує додаткової нотації.

static – статична галузь дії, чи галузь дії класа (class scope). Передбачено тільки одне значення властивості для всіх екземплярів класифікатора. Позначається підкресленням.



Абстрактні, листові і поліморфні елементи.

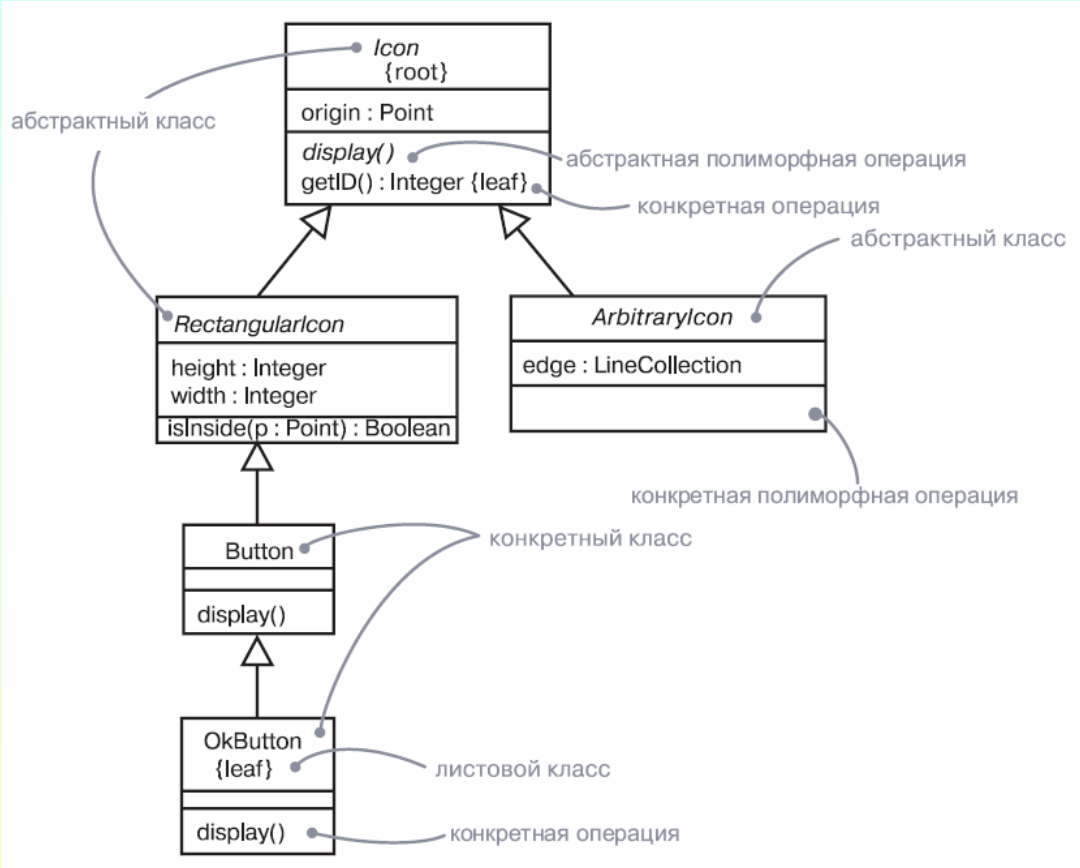
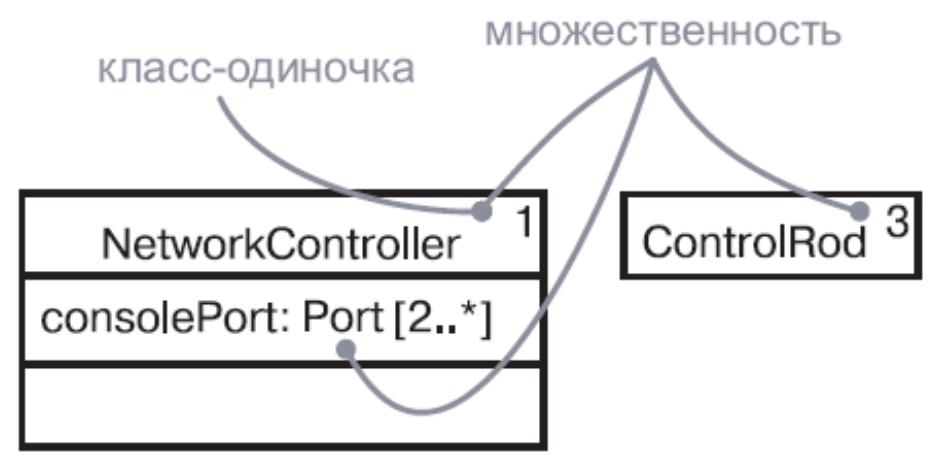


Рис. 4. Абстрактні і конкретні класи і операції.

Клас, який не повинен мати нащадків називається листовим.

Множинність



Можливе число екземплярів класу називається множинністю.
Множинність (multiplicity) – це діапазон допустимих значень кількості сутностей.
В UML ви можете вказати множинність класу у виразі, який знаходиться у правому верхньому кутку піктограми класу.

Атрибути

Синтаксис:
[видимість] ім’я [‘:’ тип] [‘[’ множинність ‘]’] [‘=’ початкове значення] [рядок властивостей {, рядок властивостей}]

Приклади:

origin	тільки ім’я
+ origin	видимість і ім’я
origin : Point	ім’я і тип
name : String [0..1]	ім’я, тип і множинність
origin : Point = (0, 0)	ім’я, тип і початкове значення
id : Integer {readonly}	ім’я і властивість

Операції

Операція – це сервіс, який можна запросити у будь-якому об’єкті класа для реалізації поведінки.
Кожній неабстрактній операції класу повинен бути зіставлений метод, який містить виконуваний алгоритм у вигляді тіла класу (зазвичай реалізованого на деякій мові програмування або у вигляді структурованого тексту).

Синтаксис:
[видимість] ім’я [‘(’ список параметрів ‘)’] [‘:’ тип повертаємого значення] [рядок властивостей {, рядок властивостей}]

Приклади:

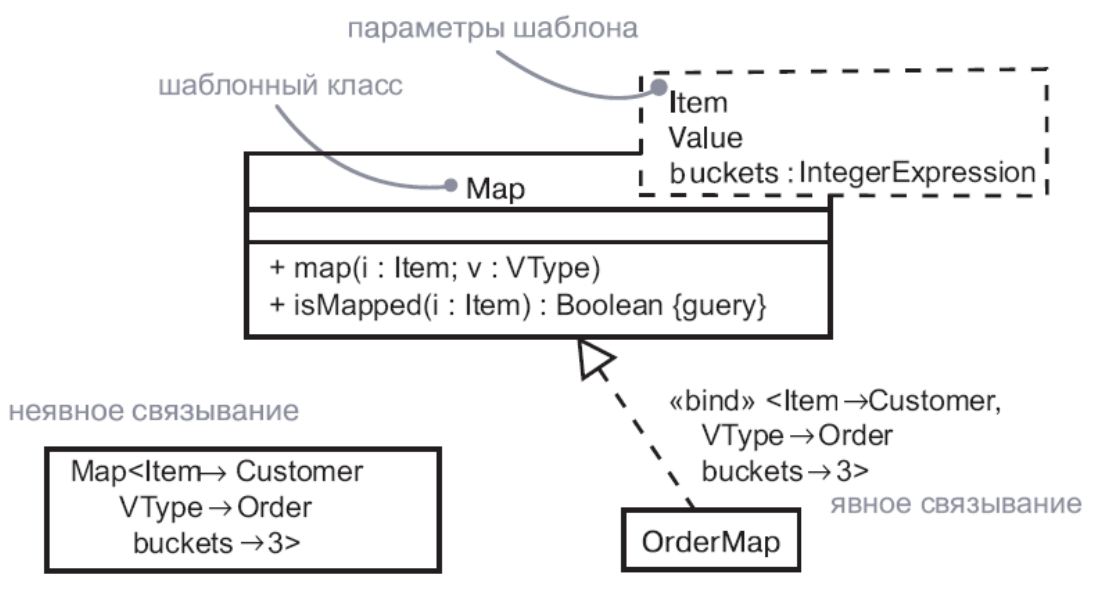
display	тільки ім’я
+ display	видимість і ім’я
set (n : Name, s : String)	ім’я і параметри
getID() : Integer	ім’я і тип повертаємого значення
restart() {static}	ім’я і властивість

Параметри

В сигнатурі операції параметри.
Синтаксис:
[direction] name : type [=default-value]

direction:
in – вхідний пареметр, не може бути модифіковано.
out – вихідний параметр, може бути модифіковано для передачі інформації коду, що викликає.
inout – вхідний параметр, може бути модифіковано для передачі інформації коду, що викликає.

Шаблонні класи



В таких мовах, як C++ і Ada можна створювати шаблонні класи.

Стандартні елементи

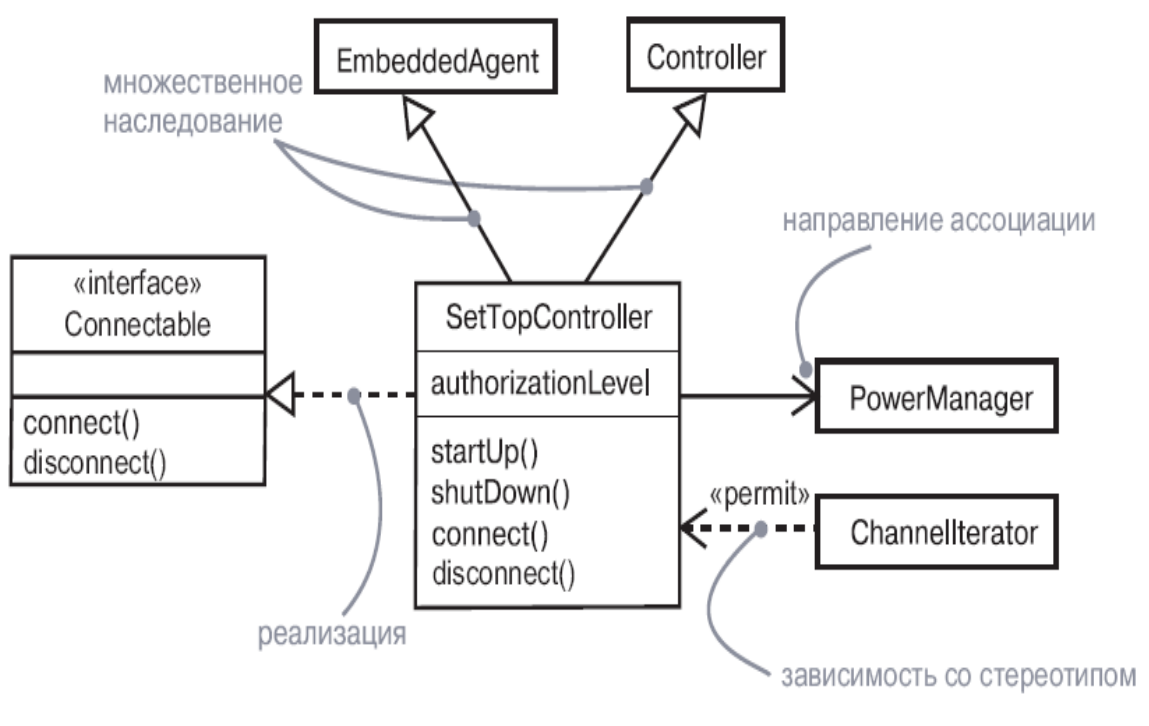
UML визначає такі стандартні стереотипи, які застосовуються до класів:
metaclass – описує класифікатор, об’єктами якого є всі класи;
powertype – описує класифікатор, об’єктами якого є дочірні класи;
stereotype – вказує на те, що класифікатор представляє собою стереотип;
utility – описує клас, атрибути і операції якого мають статичну галузь дії;

Розширені класи, розширені зв’язки та інтерфейси

Зміст

1. Розширені класи
2. Розширені зв’язки
3. Інтерфейси, типи та ролі

Розширені зв’язки

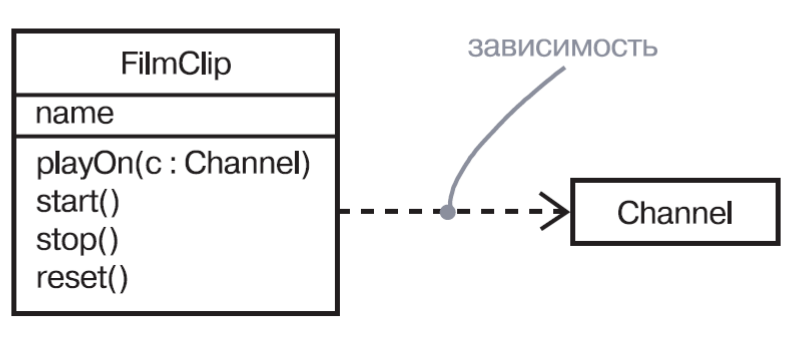


Залежність, узагальнення, асоціації – найбільш важливі будівельні блоки зв’язків в UML.

В UML існує четвертий вид зв’язку – реалізація.

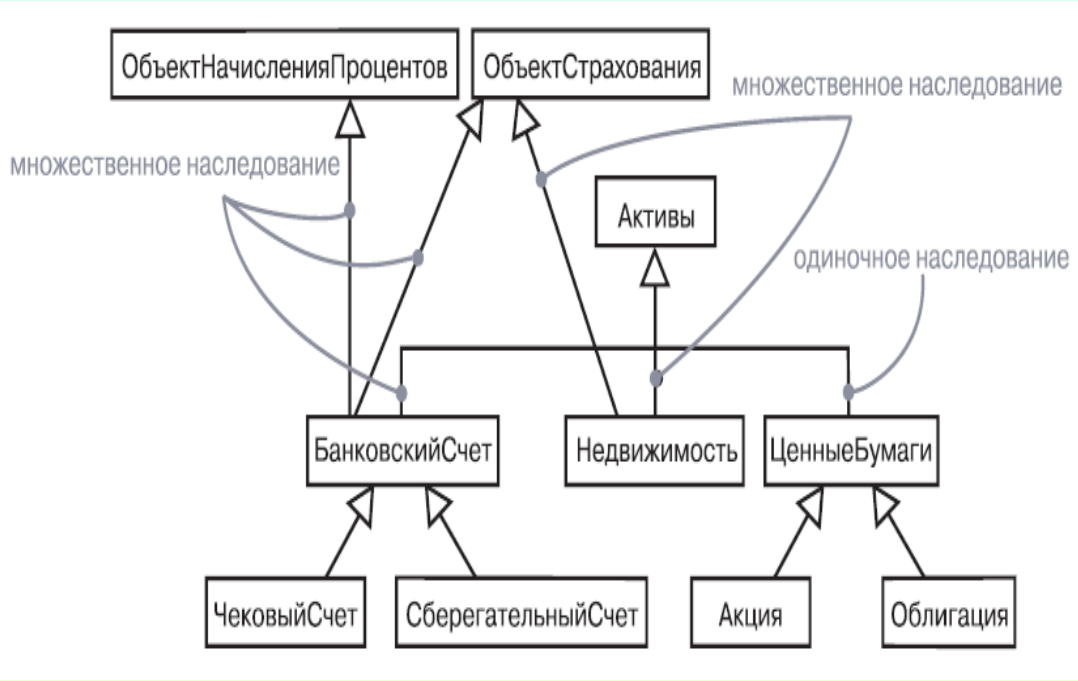
За допомогою реалізації можна моделювати з’єднання між інтерфейсом і класом, або компонентом, між варіантом використання і кооперацією.

Розширені зв’язки



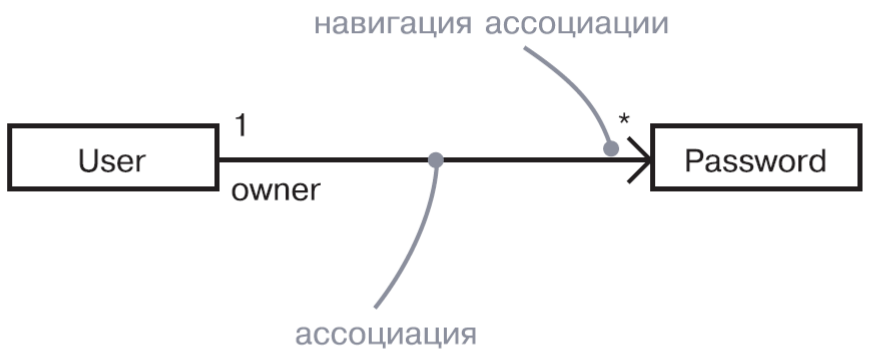
Залежність – це зв'язок використання, який вказує, що зміна специфікації однієї сутності може вплинути на інші сутності, котрі використовують її, але не навпаки.

Розширені зв’язки



Узагальнення – це зв'язок загального класифікатора, який називається суперкласом або батьком, і більш конкретного класифікатора, що називається підкласом чи дочірнім класом.

Розширені зв’язки

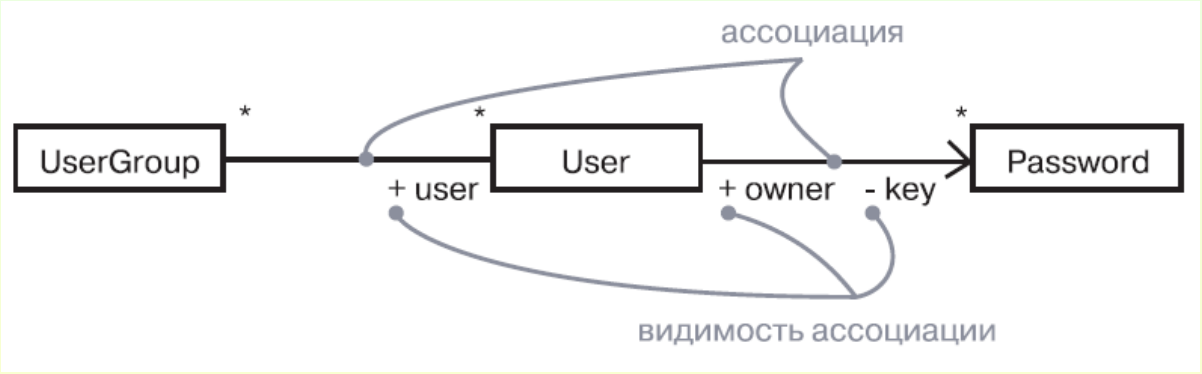


Асоціація – це структурний зв'язок, який показує, що об’єкти однієї сутності з’єднані з об’єктами іншої сутності.

Навігація.
Маючи просту асоціацію між двома класами, ви можете здійснити навігацію від об’єктів одного виду до об’єктів іншого. Якщо не вказано інше, навігація по асоціації двонаправлена. Іноді існує потреба обмежити її одним направленням.

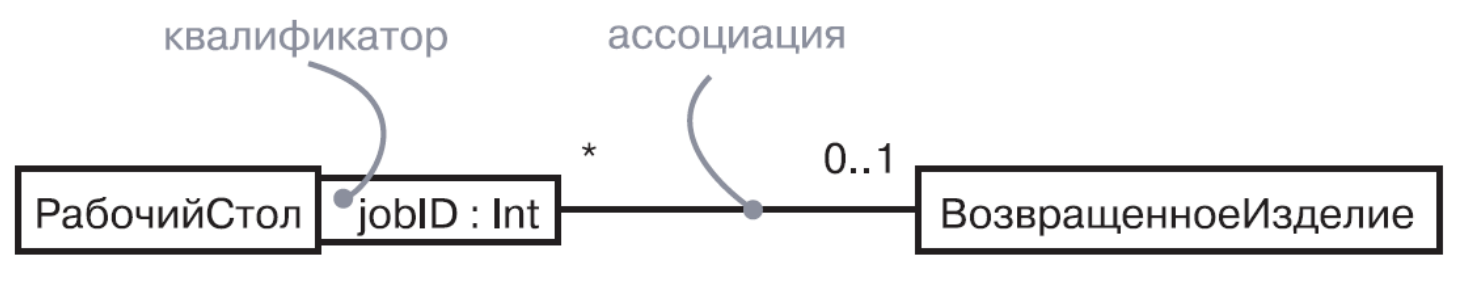
Розширені зв’язки

Видимість.
При наявності асоціації між двома класами об’єкти одного класу можуть “бачити” об’єкти іншого та допускати навігацію до них, якщо тільки вона не обмежена явним чином.
Проте, іноді необхідно обмежити видимість одних об’єктів по відношенню до інших в межах асоціації.



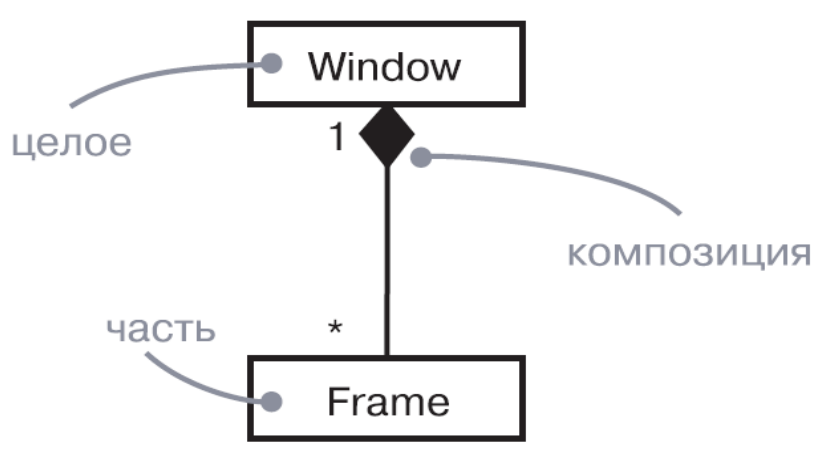
Розширені зв’язки

Кваліфікація.
Стосовно асоціацій, одна з ідіом моделювання, що найбільш часто використовується – проблема пошуку.
В UML дану ідіому необхідно моделювати за допомогою кваліфікаторів, що є атрибутами асоціацій, значення яких ідентифікує підмножену об’єктів, зв’язаних з іншим об’єктом по асоціації.

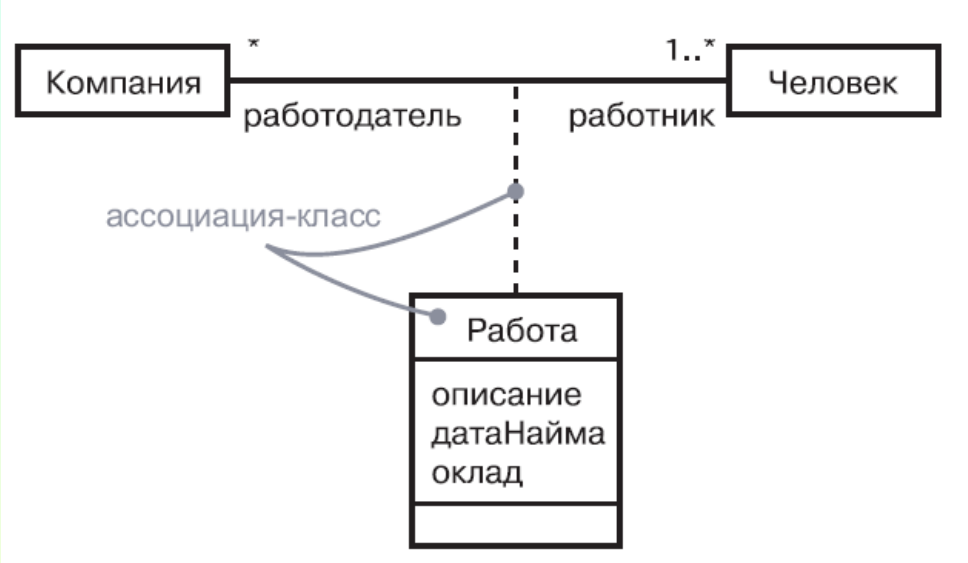


Розширені зв’язки

Композиція.
Проста агрегація виключно концептуальна і не заявляє нічого крім відміни цілого від частини. Вона не змінює смисла навігації по асоціації між цілим і його частинами і нічого не говорить про життєвий цикл зв’язку цілого з його частинами. Однак існує особа різновидність простої агрегації – композиція, яка додає семантику.
Композиція – це форма агрегації з чітко вираженими відношеннями володіння і збігами часу життя частин і цілого. Частини з нефіксованою множиною можуть бути створені після самого композиту, але одного разу створені, живуть і умирають разом з ним. Крім того, такі частини можуть бути видалені перед “смертю” композита.



Розширені зв’язки

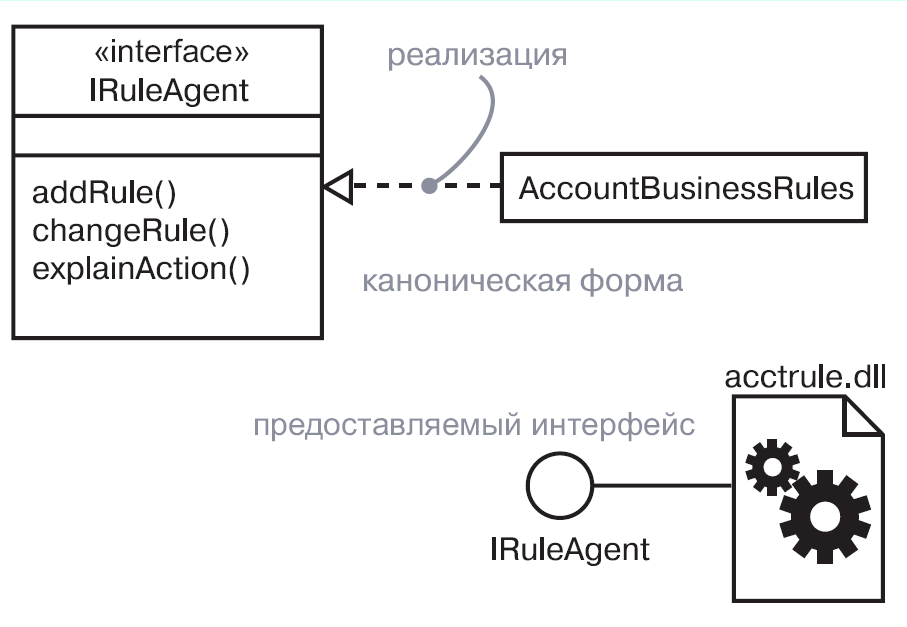


Асоціації-класи

Розширені зв’язки

Обмеження.
ordered – набір об’єктів на кінці асоціації повинен бути явно упорядкованим;
set – об’єкти унікальні без дублікатів;
bag – об’єкти не унікальні допускають дублікати;
odered set – об’єкти унікальні і впорядковані;
list чи sequence – об’єкти упорядковані, можуть бути дублікати;
readonly – посилання не може бути модифіковано, чи видалено.

Розширені зв’язки



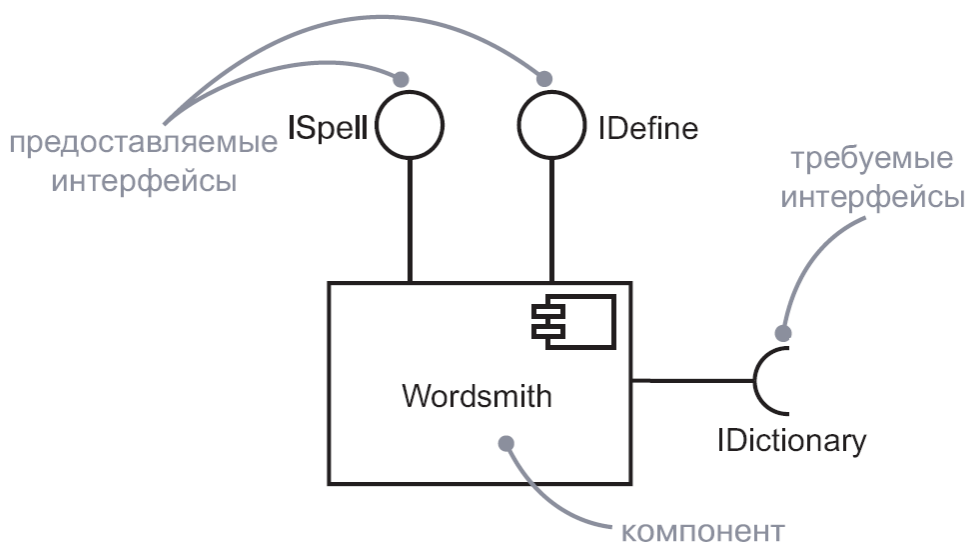
Реалізація.
Реалізація – це семантичний зв’язок між класифікаторами, один з яких специфікує деякий контракт, а інший зобов’язується його виконати.
Інтерфейс – це набір операцій, які використовуються для специфікації сервісу класу чи компонента.

Розширені класи, розширені зв’язки та інтерфейси

Зміст

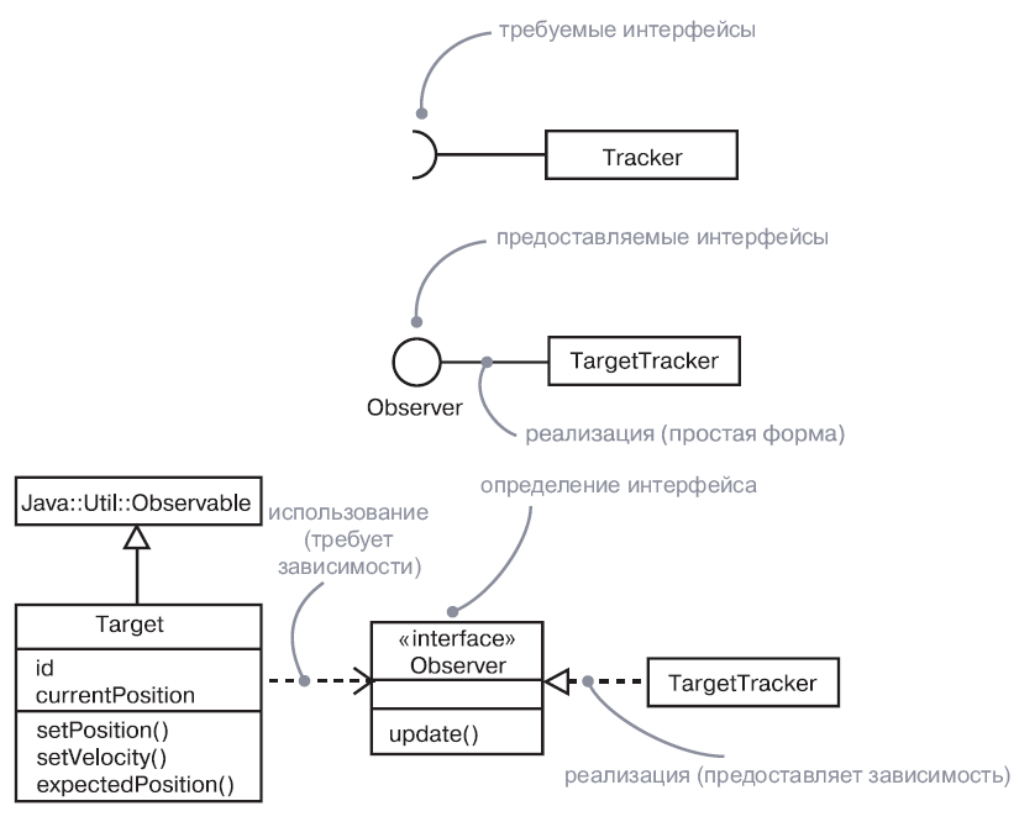
1. Розширені класи
2. Розширені зв’язки
3. Інтерфейси, типи та ролі

Інтерфейси, типи та ролі



Інтерфейс – це набір операцій, який використовується для опису сервісу класа чи компонента.
Інтерфейси використовуються для візуалізації, специфікування, конструювання та документування з'єднань всередині системи.

Інтерфейси, типи та ролі



Найбільш загальне призначення інтерфейсів – моделювати з'єднання в системах, які складаються з програмних компонентів.

Тип – це стереотип класа, що використовується для специфікації домена об'єктів разом з дозволеними для них операціями.
Роль – це поведінка сутності в певному контексті.

Заключна частина

Менш формальна модель - не означає менш точна. Просто вона менш детальна. При виборі рівня формальності ви повинні знати міру, забезпечуючи деталізацію, достатню для того, щоб зробити модель цілком зрозумілою.

Розширені класи, розширені зв’язки та інтерфейси

Дякую за увагу