АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ЛЕКЦІЇЇ 3-4. Основи UML. Umbrello UML Modeller

Umbrello UML Modeller це програма для роботи з діаграмами **UML**, яка допоможе вам у розробці програмного забезпечення, особливо під час стадій аналізу і розробки компонува-ння проекту. Umbrello UML Modeller допоможе вам створити високоякісний продукт. Крім того, UML можна використовувати для створення документації до компонування вашого програмного забезпечення, ця документація допоможе вам і вашим партнерам-розробникам.

Створення доброї моделі вашого програмного забезпечення є найкращим шляхом для обміну інформацією з іншими розробниками, які працюють над проектом, та користувачами прое-кту. Добра модель особливо важлива для проектів середнього та великого розмірів, але вона не буде зайвою і для невеличких проектів. Навіть якщо ви працюєте над маленьким прое-ктом, у якого лише один розробник, добра модель буде корисною, оскільки вона дасть вам змогу бачити загальну картину і писати код правильно з першого разу.

UML це мова створення діаграм, яка використовується для опису таких моделей. Ви можете відтворити свої ідеї у UML за допомогою діаграм різних типів. У Umbrello UML Modeller 2.11 передбачено підтримку таких діаграм:

- Діаграма класу
- Діаграма послідовностей
- Діаграма співпраці
- Діаграма випадків використання
- Діаграма станів
- Діаграма діяльності
- Діаграма компонентів
- Діаграма впровадження
- Діаграма взаємозв’язків сутностей

Докладніші відомості щодо UML можна знайти на вебсайті OMG, http://www.omg.org, компанії, яка створила стандарт UML.

Ми сподіваємося, що вам сподобається Umbrello UML Modeller, і що ця програма допомо-же вам створити високоякісне програмне забезпечення. Програма Umbrello UML Modeller є вільним програмним забезпеченням, отже доступна безкоштовно, розробники програми Umbrello UML Modeller лише просять вас повідомляти про будь-які вади, проблеми або про-позиціїthe за адресою електронної пошти umbrello-devel@kde.org або за допомогою мережі, http://bugs.kde.org.

2. Основи UML

2.1 Про UML

Цей розділ присвячено короткому огляду основ UML. Пам’ятайте, що це далеко не повний курс з UML, а скоріше короткий вступ до UML. Якщо ви бажаєте дізнатися більше про універсальну мову моделювання (Unified Modelling Language або UML) або отримати загальні відомості щодо аналізу і розробки програмного забезпечення, зверні-ться до однієї з багатьох книжок, присвячений цим темам. Крім того, у мережі Інтернет ви знайдете багато навчальних посібників, якими можна скористатися для вивчення основних відомостей.

Універсальна мова моделювання (Unified Modelling Language або UML) це мова позначень або побудови діаграм, призначена для визначення, візуалізації і документування моделей зорієнтованих на об'єкти систем програмного забезпечення. UML не є методом розробки, іншими словами, у конструкціях цієї мови не повідомляється про те, що робити першим, а що останнім, і не надається інструкцій щодо побудови вашої системи, але ця мова допомагає вам наочно переглядати компонування системи і полегшує співпрацю з іншими її розробниками.

Розробкою UML керує Object Management Group (OMG). Ця мова є загальноприйнятим стандартом графічного опису програмного забезпечення.

UML розроблено для розробки структури зорієнтованого на об’єкти програмного забезпечення, ця мова має дуже обмежену користь для програмування на основі інших парадигм.

Конструкції UML створюються з багатьох модельних елементів, які позначають різні частини системи програмного забезпечення. Елементи UML використовуються для побудови діаграм, які відповідають певній частині системи або точці зору на систему.

У Umbrello UML Modeller реалізовано підтримку таких типів діаграм:

- *Діаграма випадків* використання показує дієвих осіб (людей або інших користувачів си-стеми), випадки використання (сценарії використання системи) та їх взаємодію
- *Діаграми класів*, на яких буде показано класи та зв'язки між ними
- *Діаграми послідовності*, на яких показано об'єкти і послідовність методів, якими ці об'єкти викликають інші об'єкти.
- *Діаграми співпраці*, на яких буде показано об'єкти та їх взаємозв'язок з наголосом на об'єкти, які беруть участь у обміні повідомленнями
- *Діаграми стану*, на яких буде показано стани, зміну станів і події у об'єкті або частині системи
- *Діаграми діяльності*, на яких буде показано дії та зміни однієї дії іншою, які є наслідком подій, що сталися у певній частині системи
- *Діаграми компонентів*, на яких буде показано програмні компоненти високого рівня (на зразок KParts або Java Beans).

- *Діаграми впровадження*, на яких буде показано екземпляри компонентів та їх взаємодію.
- *Діаграми взаємозв'язку сутностей*, на яких буде показано дані, взаємозв'язки і умови обмеження зв'язків між даними.

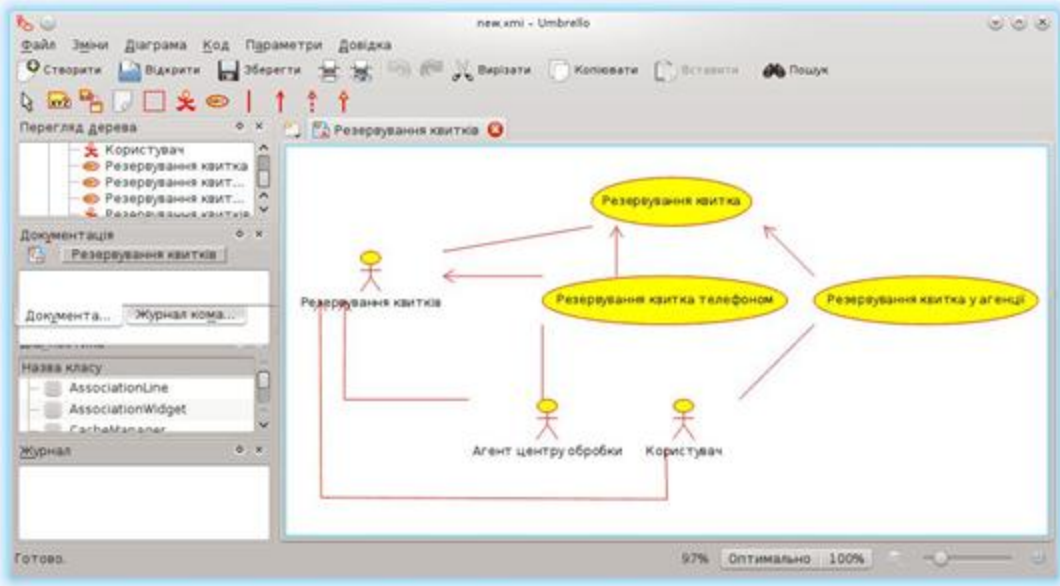
2.2 Елементи UML

2.2.1 **Діаграма випадків використання**

Діаграми випадків використання описують взаємозв’язки і залежності між групою випадків використання і акторами, що беруть участь у процесі.

Важливо зауважити, що діаграми випадків використання не призначено для показу компонування, вони не можуть описати внутрішню структуру системи. Діаграми випадків вико-ристання призначено для полегшення обміну інформацією між майбутніми користувачами системи і замовником, вони особливо корисні для визначення переліку можливостей, які по-винна мати система. За діаграмами випадків використання можна сказати, що система має

робити, але не те, як вона досягає потрібних результатів, для останнього ці діаграми просто не придатні.



Показ у Umbrello UML Modeller діаграми випадків використання

2.2.1.1 Випадок використання

Випадок використання визначає, з точки зору акторів (користувачів), групу дій у системі, які призводять до конкретного видимого результату.

Випадки використання є описом типових елементів взаємодії користувачів системи з самою системою. Вони відповідають зовнішньому інтерфейсу системи і визначають форму вимог

до того, що має робити система (зауважте, лише «що», а не «як»).

Під час роботи з випадками використання важливо пам’ятати декілька простих правил:

- Кожен випадок використання має бути пов’язано принаймні з одним актором

У кожного з випадків використання має бути ініціатор (тобто актор).

- Кожен з випадків використання має призводити до відповідного результату (результату з «комерційним значенням»)

- Кожен з випадків використання має призводити до відповідного результату (результату з «комерційним значенням»)

Випадки використання можуть мати зв’язки з іншими випадками використання. Ось три найпоширеніших зв’язки між випадками використання:

- «включення», яке вказує на те, що випадок використання відбувається всередині іншого випадку використання

- «розширення», яке означає, що у певних випадках або у певній точці (яку називають то-чкою розширення) випадок використання буде розширено іншим випадком використання

- Узагальнення, за якого випадок використання успадковує характеристики випадку вико-ристання «вищого рангу», при цьому можливе перевизначення деяких з характеристик у спосіб, подібний до успадкування між класами.

2.2.1.2 Актор

Актор це зовнішній чинник (поза межами системи), який взаємодіє з системою шляхом участі (і часто ініціювання) у випадку використання. Акторами, на практиці, можуть бути

звичайні люди (наприклад, користувачі системи), інші комп'ютерні системи або зовнішні події.

Акторам відповідають не реальні люди або системи, а лише їх ролі. Це означає, що коли особа у різний спосіб взаємодіє з системою (виконуючи різні ролі), їй відповідають декілька акторів.

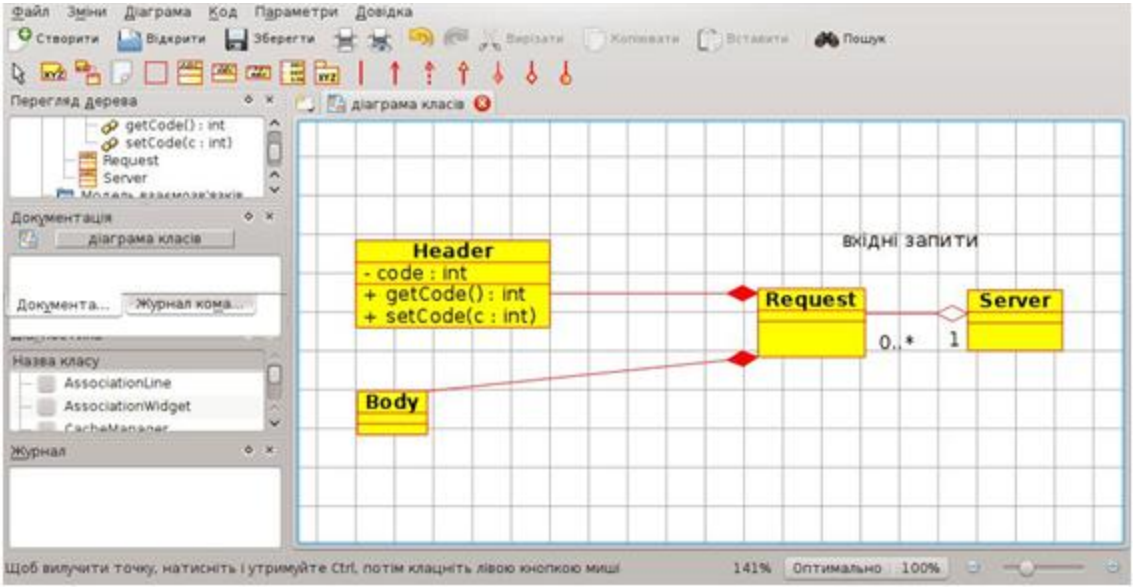
Наприклад, особа, яка виконує підтримку користувачів телефоном і приймає замовлення від користувачів до системи, може бути показано актором «Персонал служби підтримки» і актором «Відповідальний за продажі».

2.2.1.3 Опис випадків використання

Описи випадків використання це текстові примітки до випадків використання. Зазвичай, вони мають форму нотаток або документа, який певним чином пов’язано з випадком використання, і який пояснює процеси або дії, які відбуваються під час випадку використання.

2.2.2 Діаграма класів

На діаграмах класів буде 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.



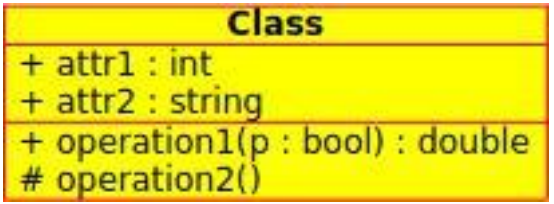
Показ Umbrello UML Modeller діаграми класів

2.2.2.1 Клас

Клас визначає атрибути і методи набору об’єктів. Всі об’єкти цього класу (екземпляри цьо-го класу) мають спільну поведінку і однаковий набір атрибутів (кожен з об’єктів має свій

власний набір значень). Іноді замість назви «клас» використовують назву «тип», але, слід зауважити, що ці назви описують різні речі: тип є загальнішим визначенням.

У UML класи позначаються прямокутниками з назвою класу, у цих прямокутниках у вигляді двох «відсіків» може бути показано атрибути і операції класу.



Наочне представлення класу у UML

2.2.2.1.1 Атрибути

У UML атрибути показуються щонайменше назвою, також може бути показано їх тип, по-чаткове значення і інші властивості. Крім того, атрибути може бути показано з областю видимості атрибута:

- + відповідає публічним (public) атрибутам
- # відповідає захищеним (protected) атрибутам
- - відповідає приватним (private) атрибутам

2.2.2.1.2 Операції

Операції (методи) також показуються принаймні назвою, крім того, може бути показано їх параметри і типи значень, які буде повернуто. Операції, як і атрибути, може бути показано з областю видимості:

- + відповідає публічним (public) операціям
- # відповідає захищеним (protected) операціям
- - відповідає приватним (private) операціям

2.2.2.1.3 Шаблони

Серед класів можуть бути шаблони, значення, які використовуються для невизначеного класу або типу. Тип шаблону визначається під час ініціалізації класу (тобто, під час створення об’єкта). Шаблони існують у сучасній мові програмування C++, їх буде введено у Java 1.5, де вони матимуть назву Generic.

2.2.2.2 Асоціації класів

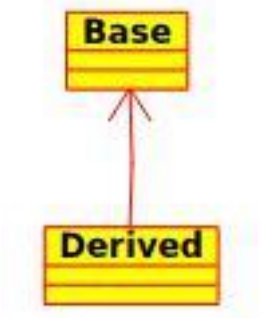
Класи можна співвіднести (пов’язати) один з одним у декілька способів:

2.2.2.2.1 Узагальнення

Наслідування є однією з фундаментальних основ об’єктно-орієнтованого програмування, у якому клас «отримує» всі атрибути і операції класу, нащадком якого він є, і може переви-значати або змінювати деякі з них, а також додавати власні атрибути і операції.

У UML пов’язування 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.



Наочний показ узагальнення у UML

2.2.2.2.2 Асоціації

Асоціація означає взаємозв’язок між класами, вона є базовим семантичним елементом і структурою для багатьох типів «з’єднань» між об’єктами.

Асоціації є тим механізмом, який надає об’єктам змогу обмінюватися даними між собою. Асоціація описує з’єднання між різними класами (з’єднання між дійсними об’єктами називається об’єктним з’єднанням, або зв’язком).

Асоціації можуть виконувати роль, яка визначає призначення асоціації і може бути одно- чи двосторонньою (другий варіант означає, що у межах зв'язку кожен з об'єктів може надсилати повідомлення іншому, перший же варіанту, коли лише один з об'єктів знає про існування іншого). Крім того, кожен з кінців асоціації має значення численності, яке визначає кількість об'єктів на відповідному кінці асоціації, які можуть мати зв'язок з одним з об'єктів на іншому кінці асоціації.

У UML асоціації позначаються лініями, що з'єднують класи, які беруть участь у зв'язку, крім того, може бути показано роль і численність кожного з учасників зв'язку. Численність буде показано у вигляді діапазону [мін..макс] невід'ємних чисел, зірочка (*) на боці максимального значення позначає нескінченність.

Наочний показ асоціації у UML

2.2.2.2.3 Агрегація

Агрегації є особливим типом асоціацій, за якого два класи, які беруть участь у зв’язку не

є рівнозначними, вони мають зв’язок типу «ціле-частина». 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 численність рівну одиниці.

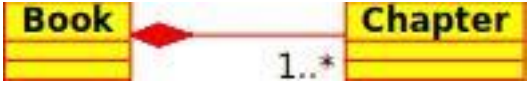
У UML агрегації буде показано асоціаціями, у яких з боку цілої частини буде намальовано ромб.

Наочний показ зв’язку агрегації у UML

2.2.2.2.4 Композиція

Композиції це асоціації, які відповідають дуже сильній агрегації. Це означає, що у компо-зиціях ми також маємо справу з співвідношеннями ціле-частина, але тут зв’язок є настільки сильним, що частини не можуть існувати без цілого. Вони існують лише у межах цілого, після знищення цілого буде знищено і його частини.

У UML композиції буде показано як асоціації з зафарбованим ромбом з боку цілого.



2.2.2.3 Інші елементи діаграми класів

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

2.2.2.3.1 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.

2.2.2.3.2 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.

2.2.2.3.3 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.

2.2.2.3.4 Пакунки

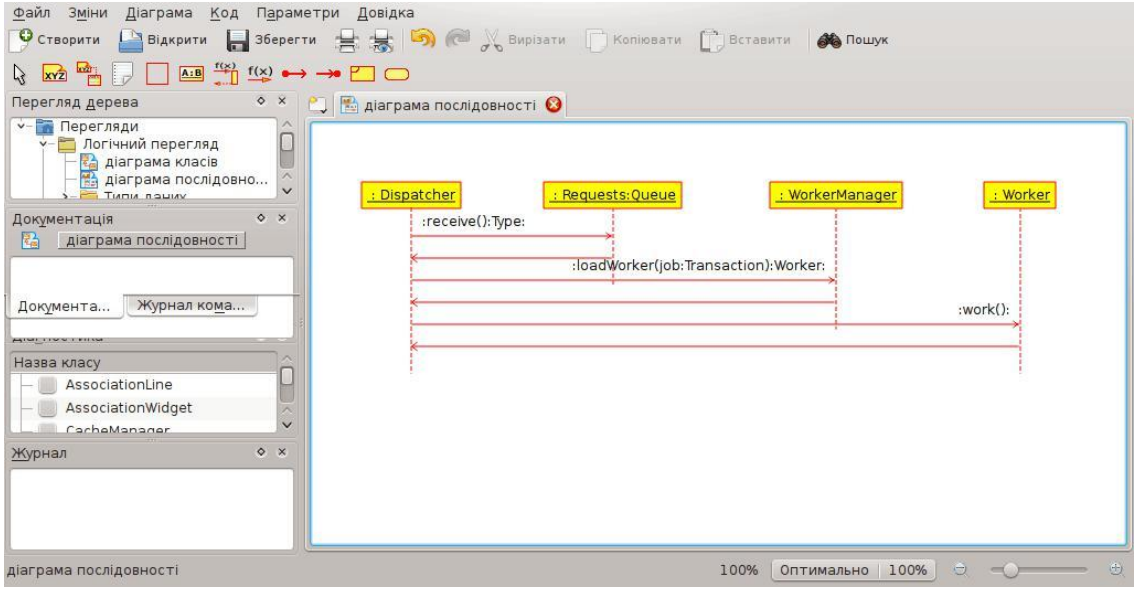
Пакункам відповідають простори назв у мовах програмування. На діаграмі пакунки використовуються для позначення частин системи, у яких міститься декілька класів, може навіть сотні класів.

2.2.3 Діаграми послідовностей

На діаграмах послідовностей буде показано обмін повідомленнями (тобто виклик методів)

між декількома об’єктами у окремій обмеженій часом ситуації. Об’єкти є екземплярами класів. Основний наголос на діаграмах послідовностей робиться на порядок і моментах часу, у які повідомлення надсилаються об’єктам.

На діаграмах послідовностей об’єкти буде показано вертикальними штриховими лініями з назвою об’єкта над ними. Вісь часу також має вертикальний напрямок, її спрямовано вниз, повідомлення, які надсилаються від одного об’єкта до іншого, буде позначено стрілками з назвами операції і параметрів.



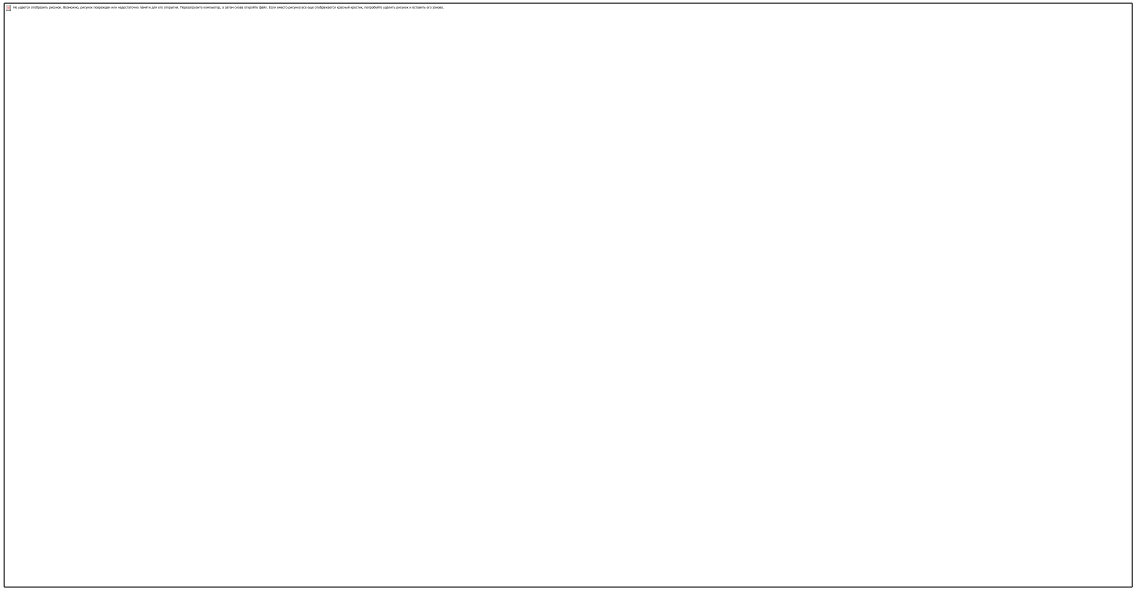
Показ діаграми послідовностей у Umbrello UML Modeller

Повідомлення можуть бути або синхронними, звичайного типу повідомленнями, за виклику яких керування передається викликаному об’єкту до завершення виконання методу, або асинхронними, за виклику яких керування передається назад напряму об’єкту, який здійснював виклик. За використання синхронного повідомлення збоку від викликаного об’єкта буде показано вертикальний блок, який показуватиме перебіг виконання програми.

2.2.4 Діаграми співпраці

На діаграмах співпраці показується взаємодія між об’єктами, які беруть участь у певній події. Ця інформація більшою чи меншою мірою подібна до інформації, показаної на діаграмі послідовностей, але там наголос робиться на часовій характеристиці взаємодії, а на діаграмах співпраці основний наголос робиться на взаємодії між об’єктами та її топології на передньому плані.

На діаграмах співпраці повідомлення надіслані від одного з об’єктів до іншого позначаються стрілочками, поряд з якими показано назву повідомлення, параметри і послідовність повідомлення. Діаграми співпраці найкраще пасують для показу специфічного перебігу виконання або ситуацій у програмі. Такі діаграми є найкращим засобом для швидкого показу і пояснення окремого процесу у програмній логіці.



Показ діаграми співпраці у Umbrello UML Modeller

2.2.5 Діаграма станів

На діаграмах станів зображають різні стани об’єкта під час його існування і стимули, які призводять до переходу об’єкта з одного стану у інший.

На діаграмах стану об’єкти розглядаються як машини станів або скінченні автомати, які можуть перебувати у одному зі станів скінченного набору станів, і які можуть змінювати цей стан через вплив одного зі стимулів зі скінченного набору стимулів.

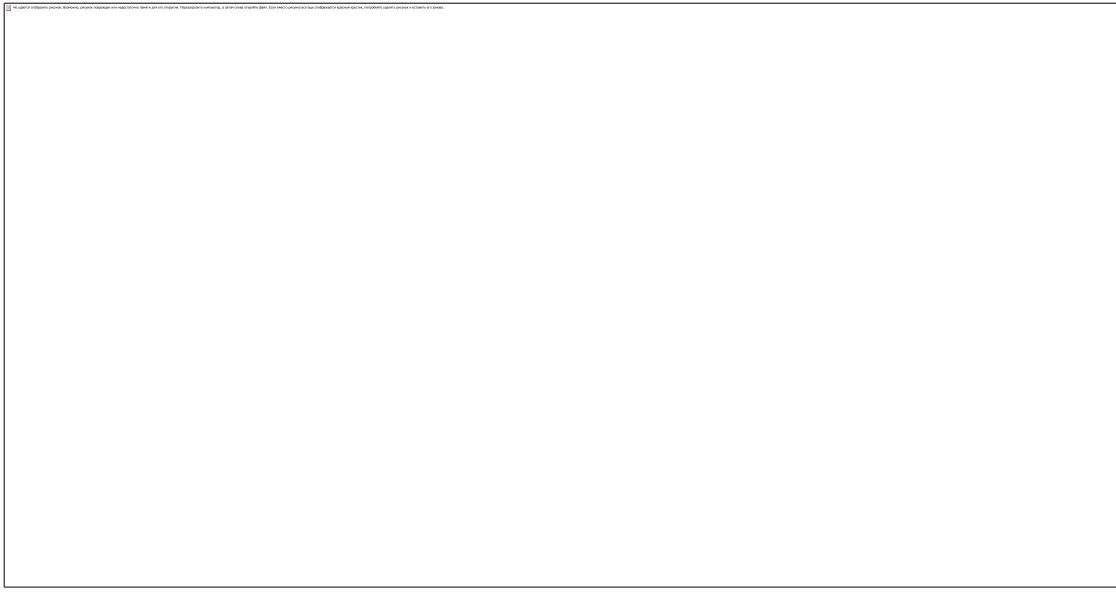
Наприклад, об’єкт типу Сервер мережі може перебувати у одному з таких станів протягом існування:

- Готовність
- Очікування
- Робота
- Зупинка

а подіями, які можуть спричинити зміну стану об’єкта можуть бути

- Створення об’єкта
- Об’єкт отримує повідомлення «очікувати»
- Клієнт надсилає запит на з’єднання мережею
- Клієнт перериває запит
- Запит виконано і перервано
- Об’єкт отримує повідомлення «зупинка»

тощо



Показ діаграми станів у Umbrello UML Modeller

2.2.5.1 Стан

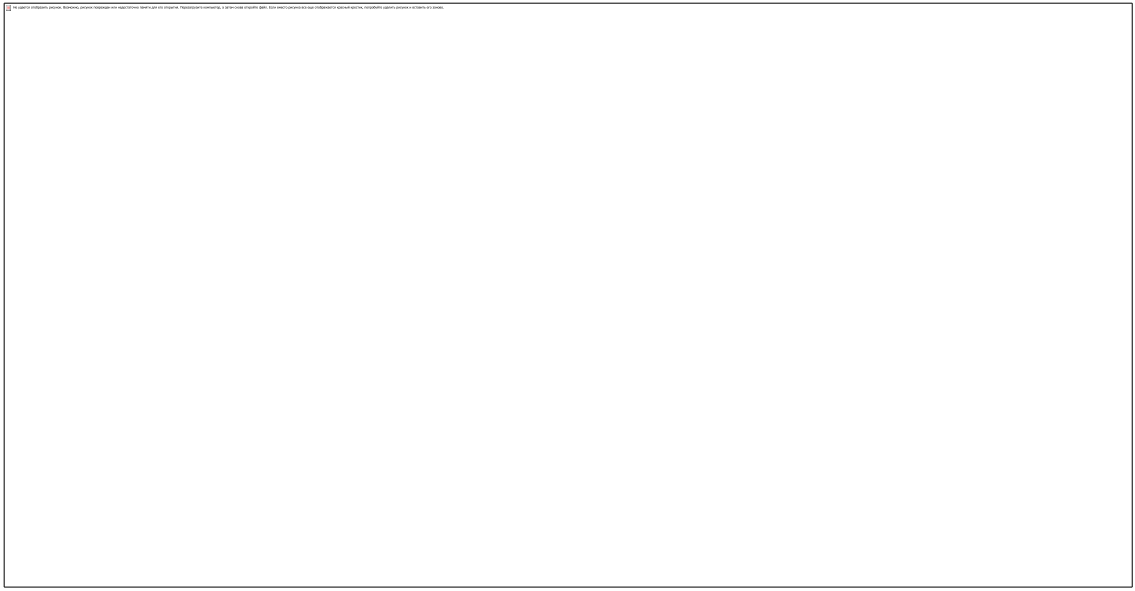
Будівельними цеглинками діаграм станів є стани. Стан належить лише одному класу і від-повідає переліку значень атрибутів, які може приймати клас. У UML стан описує внутрішній стан об’єкта одного з окремих класів

Зауважте, що не кожну зміну одного з атрибутів об’єкта має бути показано станом, станам відповідають лише ті зміни, які значно впливають на виконання об’єктом завдань.

Існує два особливих типи станів: початок і кінець. Їх особливість полягає у тому, що не існує жодної події, яка може спричинити повернення об’єкта до його початкового стану, так само, не існує жодної події, яка б могла повернути об’єкт зі стану кінця, тільки-но він його досягне.

2.2.6 Діаграма діяльності

На діаграмі діяльності буде показано послідовність актів дій системи на основі Діяльностей. Діаграми діяльності є особливою формою діаграм стану, на яких містяться лише (або головним чином) діяльності.



Показ діаграми діяльності у Umbrello UML Modeller

Діаграми діяльності подібні до процедурних діаграм потоку, але відрізняються від них тим, що діяльності точно прив’язано до об’єктів.

Діаграми діяльності завжди пов’язано з класом, операцією або випадком використання.

На діаграмах діяльності може бути показано як послідовні, так і паралельні діяльності. Па-ралельне виконання буде показано за допомогою піктограм Розділити/Чекати, для діяльностей, які виконуються паралельно, неважливим є порядок їх обробки (їх може бути виконано одночасно або одну за одною).

2.2.6.1 Діяльність

Діяльність є окремим кроком у процесі. Одній діяльності відповідає окремий стан у системі з внутрішньою діяльністю і, принаймні, одна вихідна транзакція. Крім того, діяльності можуть мати декілька вихідних транзакцій, якщо умови цих транзакцій є різними.

Діяльності можуть формувати ієрархічні структури, це означає, що діяльність може бути складено з декількох «менших» діяльностей, у цьому випадку вхідні і вихідні транзакції мають відповідати вхідним і вихідним транзакціям докладної діаграми.

2.2.7 Допоміжні елементи

У UML є декілька елементів, які не мають реального семантичного змісту для моделі, але допомагають прояснити частини діаграми. Цими елементами є

- Рядки тексту
- Текстові нотатки і якорі
- Блоки

Рядки тексту можуть знадобитися, якщо до діаграми слід додати коротку текстову інфор-мацію. Вони є довільно розташованим тестом і не мають значення для самої моделі.

Нотатками можна скористатися для додавання докладніших відомостей щодо об’єкта або певної ситуації. У них є велика перевага у тому, що нотатки можна пов’язати з елементами UML, щоб було видно, що нотатка «стосується» певного об’єкта або ситуації.

Блоки є довільно розташованими прямокутниками, які можна використовувати для групування об’єктів діаграми, яке зробить діаграму зрозумілішою. Вони не мають логічного навантаження у межах моделі.

2.2.8 Діаграми компонентів

На діаграмах компонентів буде показано компоненти програмного забезпечення (або техно-логії компонентів, такі як KParts, компоненти CORBA або Java Beans, або просто розділи системи, які чітко відрізняються один від одного), а також елементи, з яких вони складаються, такі як файли з початковими кодами, програмні бібліотеки або таблиці реляційних баз даних.

Компоненти можуть мати інтерфейси (тобто абстрактні класи з операціями), які надають змогу створювати асоціації між компонентами.

2.2.9 Діаграми впровадження

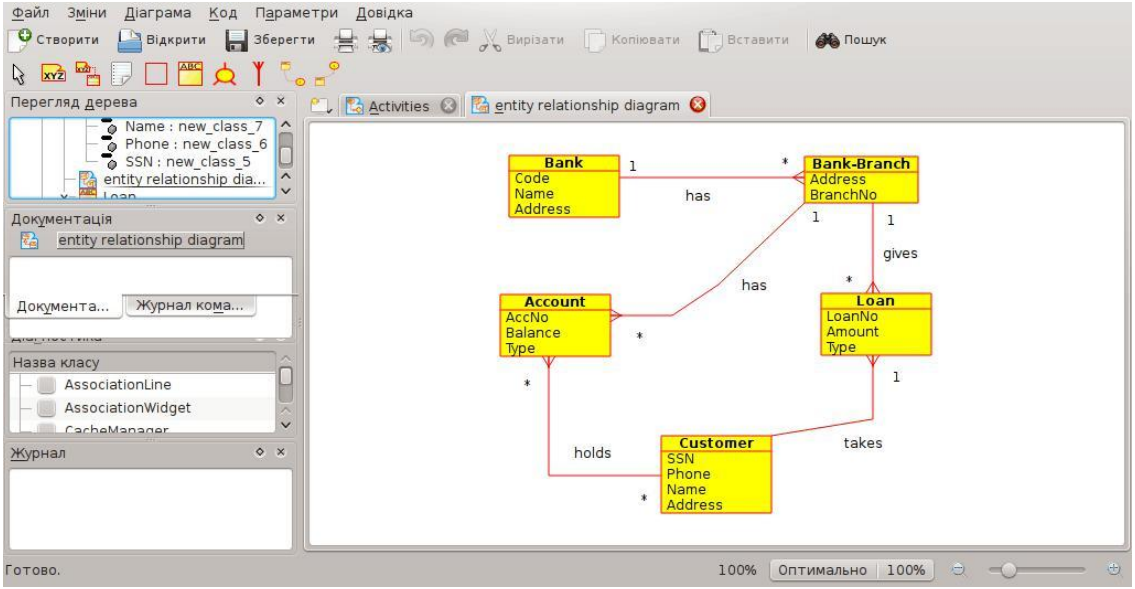
На діаграмах впровадження буде показано екземпляри компонентів та їх асоціації. На них буде показано вузли, які є фізичними ресурсами, типово, окремими комп’ютерами. Крім того, на них показують інтерфейси і об’єкти (екземпляри класів).

2.2.10 Діаграми взаємозв’язків сутностей

На діаграмах взаємозв’язку сутностей (діаграмах ВС (ER)) показують концептуальний дизайн програм для роботи з базами даних. На них показують різноманітні сутності (концепти)

у інформаційній системі і існуючі взаємозв’язки і обмеження між ними. Розширення діаграм взаємозв’язку сутностей називають «Розширеними діаграмами взаємозв’язку сутностей» або

«Покращеними діаграмами взаємозв’язку сутностей» (EER), їх використовують для інтеграції методик компонування орієнтованих на об’єкти у діаграми ВС.



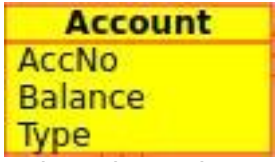
Показ діаграми взаємозв’язків сутностей у Umbrello UML Modeller

2.2.10.1 Сутність

Сутністю є будь-яке з понять реального світу, яке має окреме існування. Нею може бути об’єкт фізичної природи (наприклад, комп’ютер або робот), нею може бути і об’єкт зі концептуальним існуванням (університетський курс). Кожна з сутностей має набір атрибутів, які описують властивості сутності.

Зауваження: *не існує стандартів позначень діаграм ВС* (ER). У різних працях з цього питання використовують різні позначення. Поняття і позначення для діаграм у Umbrello UML Modeller запозичено з книги: Elmasri R. and Navathe S. (2004). Fundamentals of Database Systems 4th edn. Addison Wesley

На діаграмі ВС(ER) сутності позначаються прямокутниками з назвою у верхній частині, на них також може бути показано атрибути сутності у іншому «відсіку» прямокутника.



Наочний показ сутності на діаграмі взаємозв'язку сутностей

2.2.10.1.1 Атрибути сутності

На діаграмах взаємозв’язку сутностей атрибути сутностей показуються назвами у окремій ділянці сутності, якій вони належать.

2.2.10.1.2 Обмеження

Обмеження на діаграмах взаємозв’язку сутностей визначають обмеження на дані у інфор-маційній схемі.

У Umbrello UML Modeller підтримуються чотири типи обмежень:

- Головний ключ: Набір атрибутів, оголошених як головний ключ є унікальним для сутності. У сутності має існувати лише один головний ключ, а жоден з складових атрибутів цього ключа не повинен дорівнювати NULL.

- Унікальний ключ: Набір атрибутів, оголошених як унікальний ключ є унікальним для сутності. У сутності може бути декілька унікальних обмежень. Складові атрибути ключа можуть приймати значення NULL. Унікальні ключі і головні ключі однозначно визначають рядок у таблиці (сутність).

- Сторонній ключ: Сторонній ключ є довідковим обмеженням між двома таблицями. За стороннім ключем визначається стовпчик або набір стовпчиків у одній (тій, для якої потрібна довідка) таблиці), яка стосується стовпчика або набору стовпчиків у іншій (еталонній) таблиці. Стовпчики у еталонній таблиці повинні мати форму головного ключа і унікального ключа.

- Обмеження перевірки: Обмеження перевірки (також відоме як обмеження перевірки таблиці) є умовою, яка визначає коректність даних під час додавання або оновлення запису у таблиці реляційної бази даних. Обмеження перевірки застосовується до кожного з рядків таблиці. Обмеження має бути предикативним. Воно може стосуватися одного або декількох стовпчиків таблиці.

  Приклад: вартість >= 0

2.2.11 Концепції розширеної діаграми взаємозв’язку сутностей

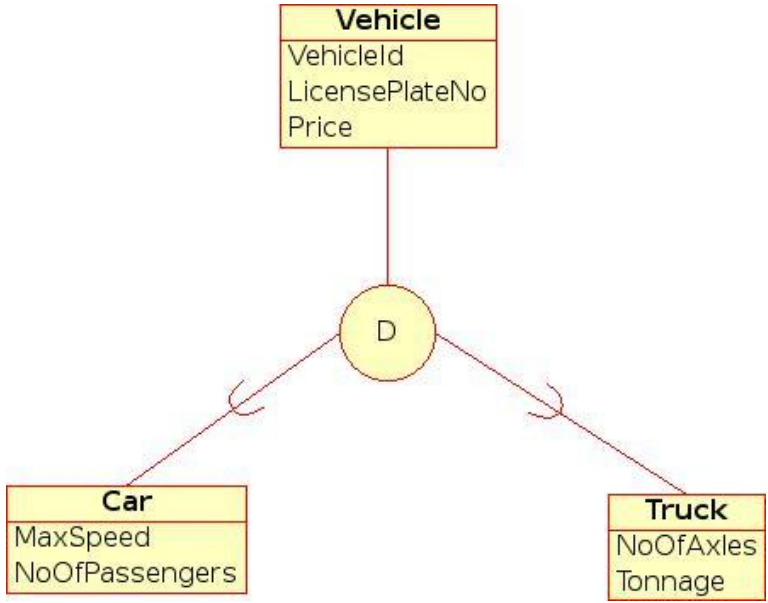
2.2.11.1 Спеціалізація

Спеціалізація це один зі способів формування нових сутностей за допомогою сутностей, які вже було визначено. Нові сутності, відомі як сутності-нащадки, успадковують атрибути сутностей, які вже існували, і які називають базовими. Спеціалізація призначена для спрощення повторного використання даних з невеликою модифікацією або взагалі без модифікації.

У Umbrello UML Modeller можна визначати несумісну і перекриту спеціалізацію

2.2.11.1.1 Несумісна спеціалізація

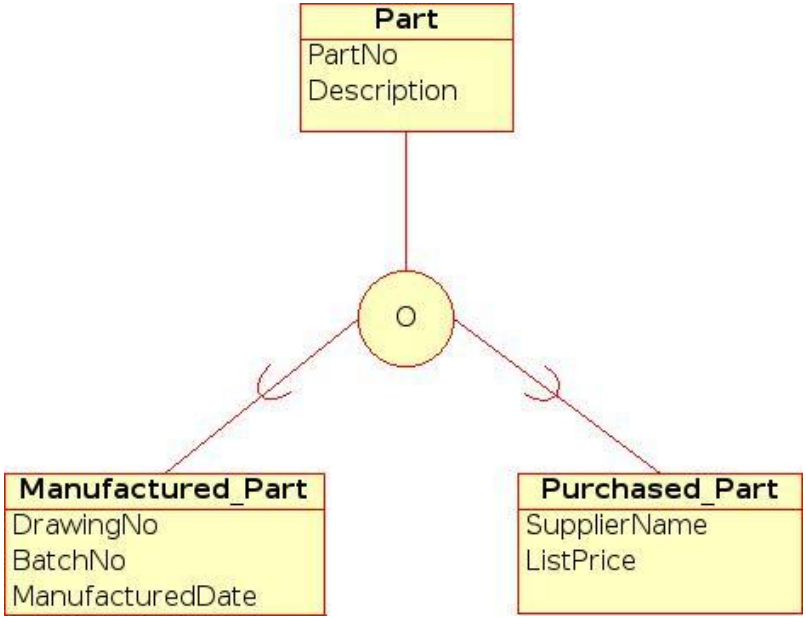
Несумісна спеціалізація вказує, що підкласи спеціалізації не повинні перетинатися. Це означає, що сутність може бути членом не більше, ніж однієї з сутностей-нащадків спеціалізації.



Наочний показ несумісної спеціалізації на діаграмі EER

2.2.11.1.2 Перекрита спеціалізація

Якщо сутності-нащадки не обмежено вимогою відсутності перетинів, такий набір сутностей належить до перекритої спеціалізації. Це означає, що однакові сутності реального світу можуть бути членами декількох сутностей-нащадків спеціалізації/

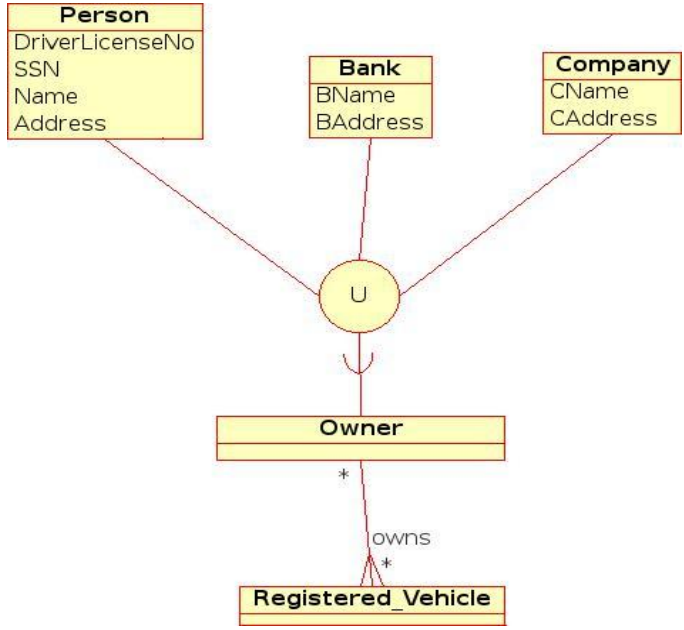


Наочний показ перекритої спеціалізації на діаграмі EER

2.2.11.1.3 Категорія

Говорять, що сутність-нащадок є Категорією, якщо їй відповідає збірка об’єктів, яка є підмножиною об’єднання різних типів сутностей. Категорії беруть участь у моделюванні, якщо

виникає потреба у окремому співвідношенні суперкласі/підкласі з декількома суперкласами, де суперкласу відповідають різні типи сутностей. (Дуже схоже на кратне успадкування у об’єктно-орієнтованому програмуванні).



Наочний показ категорії на діаграмі EER

Наступні заняття:

ЛЕКЦІЇ 5-6. Як працювати з Umbrello UML Modeller

Домашнє завдання: встановити систему **Umbrello** та спробувати самостійно візобразити в системі приклади з лекцій 3-4.