Київський національний університет будівництва і архітектури

ТЕХНОЛОГІЇ КОМП'ЮТЕРНОГО ПРОЕКТУВАННЯ

Методичні вказівки
до виконання лабораторних робіт для студентів спеціальностей

Загальні положення

Лабораторні роботи є логічним доповненням лекційного курсу з дисципліни "Технології комп'ютерного проектування" і призначені для закріплення теоретичних знань, набуття практичних навичок та вмінь по використанню технологій комп'ютерного проектування.

Тематика і зміст лабораторних робіт обумовлені основними розділами робочої навчальної програми дисципліни "Технології комп'ютерного проектування" і орієнтовані на засвоєння матеріалу

Метою лабораторних робіт є формування навичок самостійного практичного застосування сучасних методів і засобів проектування ПО інформаційних систем для конкретної предметної області за допомогою Case засобу з використанням объектно-ориентированного підходу до проектування (на основі мови UML).

Лабораторна робота №1. Вивчення інтерфейсу системи PowerDesigner

Основні можливості Sybase PowerDesigner

Sybase PowerDesigner – повнофункціональний CASE-інструментарій для створення бізнес-застосувань, що включає в себе засоби моделювання бізнес-процесів і архітектури підприємства, можливості концептуального і фізичного проектування баз даних, а також можливості моделювання з використанням UML.

PowerDesigner підтримує провідні підходи до моделювання і управління метаданими, дозволяє працювати з різними типами моделей в єдиному інтегрованому середовищі, реалізує унікальну технологію з'єднання і синхронізації моделей, а також надає користувачам централізований репозиторій для зберігання моделей і проектів. Репозиторій забезпечує наступні можливості: роботу над проектом віддалених користувачів, пошук моделей і рольовий доступ до них, контроль версій і управління конфігураціями, об'єднання моделей, генерацію повідомлень та звітів про внесені зміни.

Середовище PowerDesigner забезпечує побудову користувачем різних типів моделей, основними з них є:

- ***Enterprise Architecture Model*** (EAM) - модель архітектури підприємства;
- ***Business Process Model*** (BPM) - модель бізнес-процесів;
- ***Conceptual Data Model*** (CDM) - концептуальна модель даних предметної області;
- ***Logical Data Model*** (LDM) - логічна модель даних, незалежна від конкретної СУБД;
- ***Physical Data Model*** (PDM) - фізична модель даних для конкретної СУБД;
- ***Object Oriented Model*** (OOM) - об'єктно-орієнтована модель в нотації UML;
- ***XML Model*** (XSM) – модель структури XML-файлів;

- ***Requirements Model*** (RQM) - опис вимог до системи;
- ***Information Liquidity Model*** (ILM) – модель реплікації даних між серверами баз даних;
- ***Free Model*** (FEM) – довільна графічна модель.

Взаємозв'язок моделей

Можливість внутрішньої синхронізації даних між моделями дозволяє організовувати роботу з великою кількістю моделей в PowerDesigner. Взаємозв'язки визначаються набором правил, залежних від типів моделей, що зв'язуються. В загальному вигляді ці правила вказано на рис. 30.

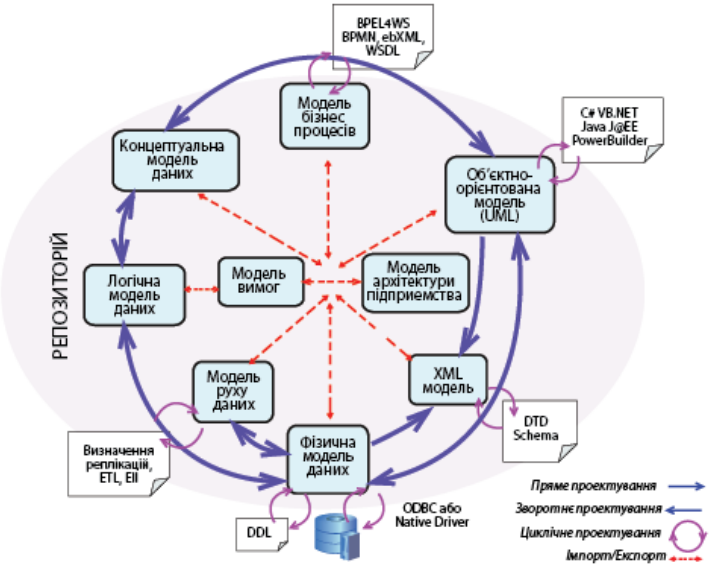


Рис. 30. Правила взаємозв'язків моделей в PowerDesigner

Потовщені суцільні стрілки показують можливість генерації однієї моделі з іншої. Наприклад, із фізичної моделі даних можна згенерувати XML-модель, з концептуальної моделі даних можна згенерувати фізичну і навпаки.

Пунктирними стрілками зв'язані моделі, для яких можливі процеси експорту і імпорту об'єктів. Наприклад, із моделі бізнес-процесів можна експортувати інформацію про дані в концептуальну модель даних.

Також PowerDesigner забезпечує синхронізацію окремих моделей з файлами зовнішніх форматів, які можуть бути текстами програмного коду в середовищі програмування, скриптами для створення або зміни бази даних і т.п. Наприклад, об'єктно-орієнтована модель може синхронізуватися з кодом на різних мовах програмування (C#, C++, VB.NET, Java, J2EE, PowerBuilder та інші). Як зображено на рис. 1, зв'язок з файлами зовнішніх форматів двосторонній. Це означає, що можна як згенерувати файл зовнішнього формату з існуючої моделі, так і створити нову модель шляхом процедури зворотного проектування (Reverse Engineering - реінжинірингу) з відповідного файлу. Зокрема, фізична модель бази даних відновлюється з реального DDL-скрипту для створення бази даних і такий же скрипт генерується з наявної моделі.

Для прямого та зворотного проектування баз даних в PowerDesigner використовуються ***концептуальна, логічна і фізична*** моделі даних. Всі ці моделі мають спільне призначення (графічне зображення структури даних з використанням нотації діаграм "сутність-зв'язок", перевірка коректності результатів проектування даних, взаємне перетворення моделей), але суттєво відрізняються за рівнем абстракції та місцем в загальному процесі проектування інформаційних систем (ІС).

Зазвичай, розробку бази даних розпочинають з створення ***концептуальної моделі*** (Conceptual Data Model - CDM), яка в найбільш загальному вигляді відображає бізнес-уявлення про дані предметної області та зв'язки між ними в незалежному від особливостей фізичного зберігання даних вигляді. Основними елементами моделі є сутності, їх атрибути (навіть з невизначеним типом) та ідентифікатори, а також зв'язки між сутностями, включно з успадкуванням. Крім традиційних для реляційних моделей даних зв'язків типу "один до одного" та "один до багатьох", в CDM використовується і зв'язок "багато до багатьох". Ця модель не є реляційною, а тому може бути перетворена в різні типи логічних моделей (ієрархічні, об'єктно-орієнтовані та реляційні), а також слугувати основою для генерації фізичної моделі.

Але на практиці з CDM, як правило, спочатку генерують ***логічну модель*** даних (Logical Data Model - LDM), яка більш детально відображає деталі фізичного зберігання даних і не залежить від типів полів та інших особливостей конкретної СКБД. LDM вже є реляційною моделлю, в якій зв'язки "один до багатьох" перетворюються в ідентифікатори "первинний ключ-зовнішній ключ" і замість зв'язків "багато до багатьох" використовуються проміжні сутності.

На основі розробленої LDM генерується одна чи декілька ***фізичних моделей*** даних (Physical Data Model - PDM) для конкретних СКБД. В PDM сутності перетворюються в таблиці, атрибути – в стовпчики таблиць. Створюються ключі, індекси, перегляди, тригери та збережені процедури. З фізичної моделі генерують DDL-скрипти для створення чи модифікації БД або безпосередньо створюють базу через ODBC-драйвер.

Розробку CDM і LDM зазвичай ведуть національною мовою, а фізичної моделі - мовою обраної СКБД (як правило, англійською). Кожен елемент моделі даних в PowerDesigner має ім'я (назву національною мовою) і код (ідентифікатор) англійською мовою.

Інтерфейс PowerDesigner

PowerDesigner являє собою унікальне інтегроване середовище, в якому можна одночасно працювати з моделями всіх типів в єдиному користувацькому інтерфейсі.

PowerDesigner складається з: вікна перегляду (навігатор), робочої області, панелі інструментів і вікна для виводу даних. Ви можете налаштувати робочий простір так, як вам буде зручно.

Інтерфейс PowerDesigner (рис. 31). складається з традиційного меню, вікна перегляду (навігатора), плаваючих панелей інструментів, робочої області та вікна виводу.

Інформація про всі об'єкти проекту представлена у навігаторі з використанням зручної для сприйняття деревоподібної структури. Можна використовувати вікно перегляду для навігації по моделях, що відображаються у вікні редагування діаграм. Об'єкт у вікні редагування діаграм PowerDesigner вибирається простим кліком миші на об'єкті у вікні перегляду. Навіть якщо діаграма включає забагато об'єктів, вікно перегляду надає швидкий і зручний спосіб для переміщення до потрібного об'єкта і вибору будь-якого іншого об'єкта в діаграмі.

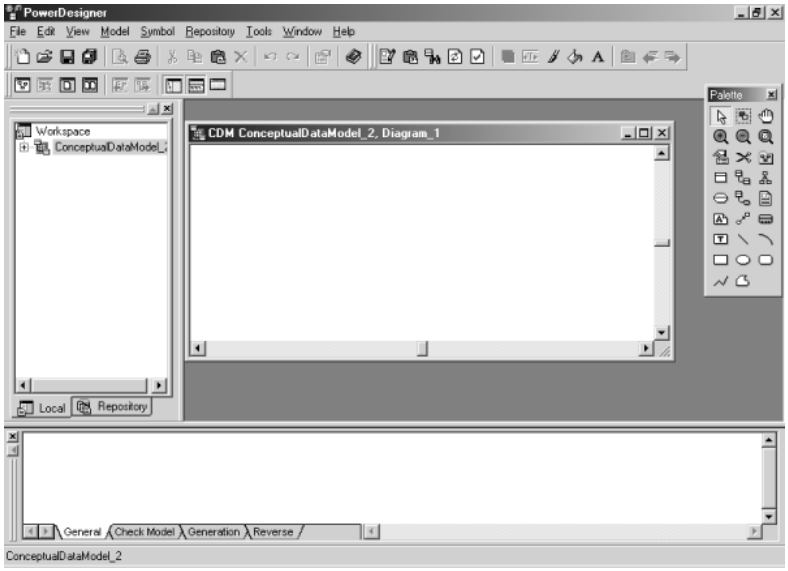
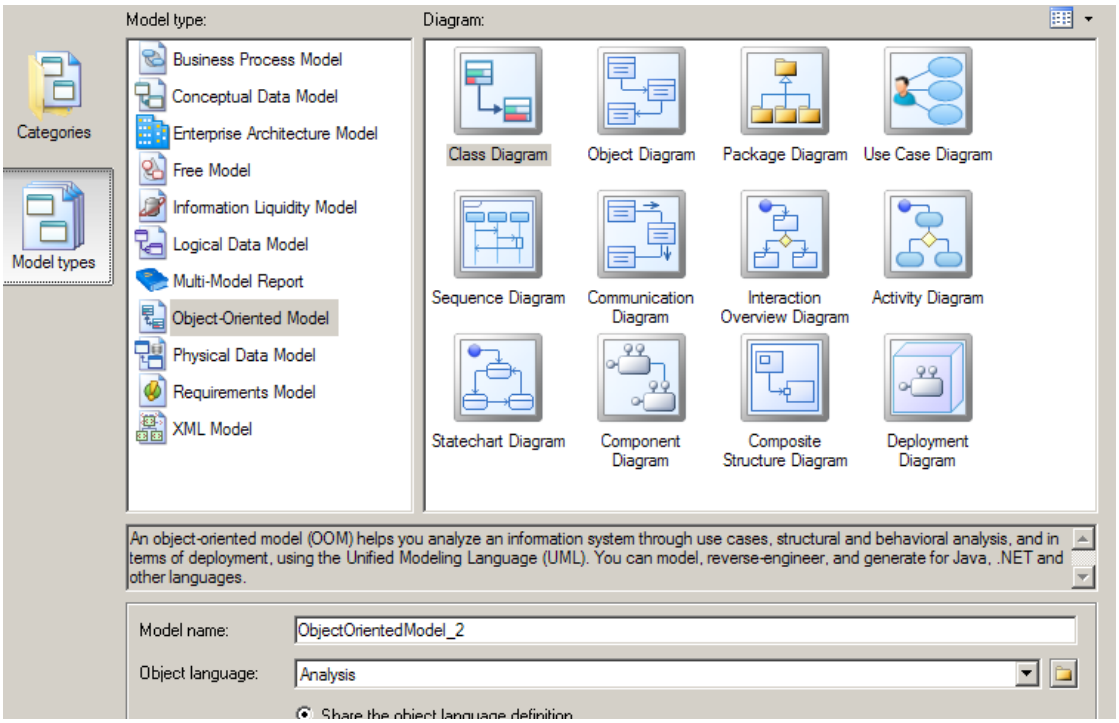


Рис. 31. Головне вікно PowerDesigner

Вікно вивода розміщується в нижній частині форми і призначено для виводу різноманітних діагностичних повідомлень.

Щоб створити новий проект, необхідно відкрити: **File->New Project** і задати необхідні властивості проекту. Далі можна створювати моделі командою **File->New Model**. В діалоговому вікні **New Model** на закладці **Model type** вибрати необхідну діаграму. Потім задати ім’я моделі **Model name** – наприклад, My_Model і натиснути OK.



Керування і настроювання тулбарів

Настроїти їхній уміст можна через меню **Customize Menus and Tools** (тут же, до речі, можна настроїти і вміст приладової панелі для кожного виду моделі). Усі ці можливості доступні через контекстне меню тулбарів.

Коротко, загальні настроювання підрозділяються на наступні групи:

- ***Display Preferences*** - опції відображення об'єктів діаграми (зберігаються в моделі й у registry) Контролюють колір, зовнішній вигляд, розміри, склад і розташування відображуваної інформації для різних символів діаграми.
- ***Model Options*** - опції моделі (зберігаються в моделі). Контролюють угоди про найменування об'єктів, використовувану нотацію, значення за замовчуванням, чутливість до регістра і т.п. Конкретний набір опцій залежить від типу моделі.
- ***General Options*** - загальні характеристики (зберігаються в registry). Контролюють зовнішній вигляд і поводження інтерфейсу PowerDesigner, наприклад настроювання діалогів, перемінні оточення, шрифти і т.п.
- ***Check Model options*** - опції перевірки моделі (зберігаються в моделі). Контролюють набір параметрів моделі, що перевіряються, і рівень реакції на невідповідність (помилка, попередження). Набір правил перевірки залежить від типу моделі.
- ***Інші опції*** (зберігаються в registry). Сюди відносяться такі речі як розташування тулбарів і вікон (**Organizing Views**), улюблені закладки для властивостей об'єкта, набір відображуваних за замовчуванням стовпчиків для списків об'єктів і т.п.

Можна заздалегідь задати різні набори цих настроювань і застосовувати ту конфігурацію, що найбільше підходить для цілей поточного проекту. Робиться це через можливість, що зветься Профілі користувачів чи **User profiles.**

Список наявних профілів можна подивитися через меню **Tools->Resourses->User Profiles**. Тут є список профілів за замовчуванням, для кожного можна подивитися і відредагувати його параметри. Тут же можна створити свій власний профіль, узявши за основу один із уже наявних, або (що

дуже корисно), поточні настроювання моделі чи інформацію про настроювання із системного реєстру Windows (registry). Застосувати створений чи вхідний у постачанні профіль можна через меню **Tools->Apply User Profile.** При цьому варто мати на увазі, що якщо новий профіль застосовується до моделі, що вже знаходиться в розробці, то параметри відображення створених діаграм залишаться незмінними, - нові настроювання будуть застосовуватися тільки до нових діаграм. Тому має сенс застосовувати потрібний профіль на самому початку роботи з моделлю.

Можна кликнути мишкою на назву об'єкта чи на кожнім з відображуваних елементів його колекції і відредагувати його прямо на діаграмі, без заходу в список властивостей об'єкта. Рядок з обраним елементом підсвічується, можна переміщатися нагору і вниз за списком відповідними клавішами, можна перетягнути чи скопіювати елемент (наприклад, колонку таблиці) з одного об'єкта в інший мишкою, змінити його місце розташування в списку, виділити кілька елементів і т.п. ; можна відкрити властивості елемента, двічі клацнувши по ньому мишкою, можна не тільки задати будь-які атрибути і колекції об'єкта, які потрібно відобразити, але також і описати їхнє місце розташування.

Вибір і розташування об'єктів було задано через меню ***Tools->Display Preferences***, об'єкт **Entity**. На закладці **Content** для цього об'єкта є кнопка **Advanced**..., що відкриває діалог **Customize Content**.

Є можливість задати іконки для відображення різних значень додаткових атрибутів об'єкта, і відображати на символі діаграми іконку, що відповідає обраному в даний момент значенню. Список іконок, як і додатковий атрибут, задається в розширенні до PowerDesigner (**Extended Definition**). Для нових моделей опції підтримуються автоматично, для старих - їх потрібно включити в меню ***Tools->Display Prefernces***, ***General Settings***: опції ***Show bridges at intersections і Automatic link routing.***

У PowerDesigner існує можливість використання об'єктів з однієї моделі в іншій моделі. Один зі способів зробити це - скористатися механізмом посилання (**shortcut**). Це дозволяло створювати цілі бібліотеки стандартних об'єктів для того, щоб використовувати їхній згодом в інших моделях.

Метою даної лабораторної роботи є ознайомлення з основними можливостями PowerDesigner, вивчення інтерфейсу системи й одержання навичок роботи при створенні діаграм UML. Звіт по роботі - короткий опис функціональних можливостей PowerDesigner з використанням PrintScreen - картинок, отриманих студентами при вивченні системи.

Лабораторна робота №2. Побудова функціональної моделі застосування у вигляді діаграм потоків даних

Мета роботи

Засвоєння основних правил побудови діаграм потоків даних

Порядок виконання роботи

1. Провести ґрунтовний аналіз функцій системи, що проектується, для виявлення елементів діаграми потоків даних – графічного подання моделі РАМ.
2. Створити та визначити головний процес системи.
3. Створити та визначити сутності.
4. Створити елементи даних.

5. Провести декомпозицію головного процесу.
6. Закріпити елементи даних за різними потоками в моделі.
7. Створити та визначити сховища даних, де буде зберігатися інформація.
8. Створити та зберегти файл звіту з розробленою діаграмою потоків даних.

Методичні вказівки

Функціональна модель системи визначається як ієрархія діаграм потоків даних. Діаграми верхніх рівнів ієрархії (контекстні діаграми) визначають основні процеси. Вони деталізуються за допомогою діаграм нижчого рівня, така деталізація продовжується доти, доки не буде досягнутий такий рівень декомпозиції, на якому процес стає елементарним і деталізувати його далі недоцільно.

Основними компонентами діаграм потоків даних є: зовнішні сутності, системи/підсистеми, процеси, сховища даних (таблиці баз даних, окремі файли) і потоки даних. Джерела інформації (зовнішні сутності) породжують інформаційні потоки (потоки даних), які переносять інформацію до підсистеми або процесу. Ті, в свою чергу, перетворюють інформацію і породжують нові потоки, які переносять інформацію до інших процесів або підсистем, сховищ даних або зовнішніх сутностей - споживачів інформації. Процес - це “чорний ящик”, який приймає дані, трансформує їх та видає результат.

Для відносно простих інформаційних систем будується єдина контекстна діаграма зіркоподібної топології, центром якої є так званий головний процес, з’єднаний з споживачами і джерелами інформації.

Наприклад: розробити базу даних для обліку земельних ділянок та їхніх власників і користувачів.

Визначимо основні сутності: **земельні ділянки** (територіальний код, площа, адреса, форма власності, власник, користувач, тип використання), **власники ділянок** (ПІБ, адреса, телефон) і **користувачі ділянок- юридичні та фізичні особи** (найменування, представник, адреса, телефон).

Запити: пошук ділянок за територіальним кодом, за власником, за формою власності, за значенням площі. Отже, головний процес – надання землі в оренду:

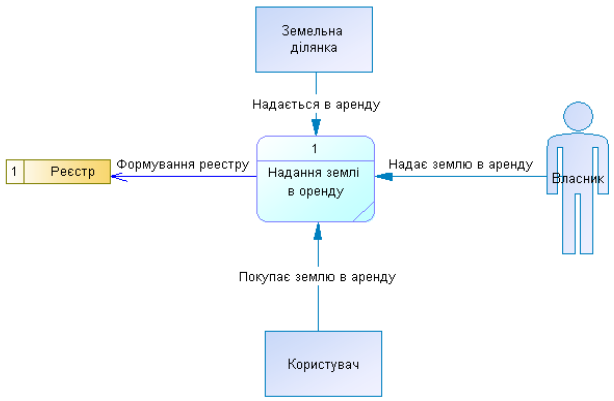
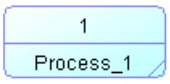


Рис. 32. Діаграма потоків даних для обліку земельних ділянок та їхніх

власників і користувачів

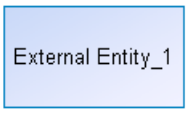
Для **створення моделі** необхідно відкрити: File->New Model-> В діалоговому вікні New Model на закладці Model type вибрати необхідну діаграму: Business Process Model (BPM) - модель для побудови бізнес-процесів і мову процесу Data Flow Diagram.

Для **створення процесу** необхідно:

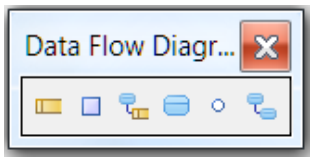
1) На панелі інструментів Palette вибрати значок процесу ;
2) Вставити значок процесу 

у вікно моделі.
3) Для того, щоб змінити номер та назву процесу необхідно натиснути на нього два рази лівою кнопкою мишки або правою викликати контекстне меню і вибрати Properties. У вікні Process Properties ввести ім'я та код (=) процесу. При необхідності номер процесу можна змінити в полі Number ID. Натиснути ***ОК***

Для **визначення зовнішньої сутності** потрібно:

1) На панелі інструментів Palette вибрати значок сутності ;
2) Вставити значок сутності 

у вікно моделі;
3) Двічі натиснути на зображенні сутності;
4) У вікні ***Properties*** *(властивості)* ввести ім'я та код (=) і натиснути ***ОК***.

Для створення потоку даних між об'єктами використовується панель інструментів Data Flow Diagram 

де Resource Flow – потік між ресурсом і процесом, а Flow – потік між процесами, процесами та сутностями. На панелі інструментів Palette потік Flow визначається автоматично, в залежності від того , між якими об'єктами він створюється.

Для **створення потоку даних** між об'єктами треба:

1) На панелі інструментів вибрати значок потоку даних , , або ;
2) Провести лінію потоку від одного об'єкту до іншого;
3) Двічі натиснути на зображенні потоку даних;
4) У вікні ***Properties*** *(Властивості)* ввести ім'я та код (=) ;

Для **декомпозиції** головного процесу необхідно:

1) Вибрати головний процес;
2) Натиснути на зображенні процесу правою кнопкою мишки, в контекстному меню, що з'явиться, вибрати ***Decompose Process*** *(Декомпозиція процесу);*

Для **створення елементів набору даних** та доменів необхідно вибрати ***Model→Data*** *(Модель→ Дані)* та у віконці ***List of Data*** ввести імена і типи полів.

Для **зв'язування набору даних з потоком даних** треба повернутися до головного процесу і виконати такі дії:

1) ***Model→Flows*** *(Модель→ Потоки)*. На ній відображаються лише потоки головного процесу. Зовнішні сутності і сховища даних показуються у списку об'єктів на всіх рівнях процесу (глобальні об'єкти). Потоки і процеси показуються у списку об'єктів на деталізованих рівнях (локальні об'єкти);

2) Вибрати потік і натиснути ***Properties*** *(Властивості)*.

3) Вибрати закладку ***Data*** *(Дані),* натиснути Add Objects (Додати об'єкти) і вибрати ті дані, що відносяться до потоку.

Для **визначення сховища даних** потрібно**:**

1) Вибрати значок сховища даних на панелі інструментів DataStore (сховище даних);

2) Двічі натиснути на зображенні сховища даних;

3) У вікні Data store ***Properties*** *(Властивості)* ввести ім'я та код (**=**) і натиснути ***OK***.

Для **перевірки коректності моделі** необхідно вибрати ***Tools→Check Model*** *(F4) (Інструменти→Перевірити Модель).*

Зберегти файл My_ BPD.bpm з розробленою діаграмою потоків даних.

Завдання:

1. **Інтернет-магазин**. Повинні бути реалізовані сценарії: покупка товару, пошук товару, додавання нового товару в базу даних магазина, перегляд і обробка замовлень покупателей , реєстрація нового покупця.
2. **Книжковий каталог**. Повинні бути реалізовані сценарії: додавання нової книги, пошук книги по декількох полях, бронювання книги, списання старих книг, реєстрація пользователей каталогу.
3. **Адресна книга.** Повинні бути реалізовані сценарії: додавання нового абонента, додавання категорії абонентів, пошук абонентів по декількох полях, додавання адміністраторі каталогу (пользователей , що мають право редагувати дані адресной книги), редагування даних абонента.
4. **Розклад занять.** Повинні бути реалізовані сценарії: додавання нової групи, додавання заняття (із указівкою назви предмета, часу, аудиторії, групи, тижня, викладача, типу занять), перегляд списку занять на обрану дату, додавання списку преподавателей , пошук занять по декількох полях (предмету, викладача, групі, часу, типі занять).
5. **База студентів.** Повинні бути реалізовані сценарії: додавання нової групи, додавання нового студента, пошук студента по різних полях, додавання інформації про оцінки по різних предметах, відрахування студента.
6. **Прайс-лист фірми.** Повинні бути реалізовані сценарії: додавання нової категорії товарів, додавання нового товару, пошук товару по різних полях, додавання адміністратора, прайс-листа (пользователей , що мають право редагувати прайс-лист), переміщення товару з одної категорії в іншу.

7. **База складу фірми.** Повинні бути реалізовані сценарії: додавання нового товару на склад, списання товару, видача товару, пошук товару по різних полях, зміна місця розташування товару на складі.
8. **Аптечна база.** Повинні бути реалізовані сценарії: прийом замовлення від клієнта на виготовлення розчину, продаж ліків, списання прострочених ліків, додавання нових ліків у базу даних, пошук замовлень по різних полях.

Контрольні питання

1. Мета проведення об'єктного аналізу.
2. Призначення діаграми потоків даних.
3. Основні елементи діаграми потоків даних.

Лабораторна робота №3. Створення концептуальної та логічної моделей предметної області з використанням діаграм «сутність-зв'язок»

Мета роботи

Набути навики побудови концептуальної моделі даних у вигляді діаграм “сутність-зв’язок” (ER - діаграм).

Порядок виконання роботи

1. Визначитися з основними об'єктами моделі, виходячи із проблематики завдання.
2. Створити новий файл моделі.
3. Визначити та створити домени, елементи даних, сутності, атрибути сутностей, зв’язки та успадкування.
4. Перевірити коректність створеної CDM.
5. Генерувати і зберегти файл звіту.

Методичні вказівки

Концептуальна модель - це систематизований змістовний опис модельованої системи (або проблемної ситуації) на неформалізованій мові. До об’єктів CDM належать: домен (**Domain**) - набір значень, в яких елемент даних дійсний; елемент даних (**Data item**) - елементарна частка інформації; сутність (**Entity**) - людина, місце, предмет або концепція, яка характеризує систему; атрибут сутності (**Entity attribute**) - елементарна інформація, яка характеризує сутність; зв’язок (**Relationship**) - з’єднання або асоціація між сутностями; успадкування (Inheritance line) - спеціальний зв’язок, що відображає відношення сутності-нащадка та сутності-предка.

Концептуальна модель даних (CDM) призначена для:

– представлення логічної організації даних у графічному вигляді;

– перевірки коректності результатів проектування даних;

– використання в якості основи для генерації фізичної моделі даних (PDM).

Для **створення моделі** необхідно відкрити: **File->NewModel->** В діалоговому вікні New Model на закладці Model type вибрати необхідну діаграму: Conceptual Data Model(CDM): Концептуальна модель даних. Ввести *Name* ***(Ім’я),*** *Code* ***(Код).***

Для **визначення опцій та властивостей CDM** необхідно правою кнопкої мишки натиснути на модель **CDM** на панелі **Browser.** Вибрати ***Model options*** *(Опції Моделі)* та ***Model Properties*** *(Властивості Моделі)*.

Побудову CDM доцільно починати з визначення доменів, яке допомагає ідентифікувати типи даних проекту. Застосування доменів до елементів даних полегшує уніфікацію характеристик даних атрибутів в різних сутностях.

В доменах можливо поєднувати інформацію про тип даних, їх довжину і точність, допустимі значення параметрів. Для **створення доменів** необхідно:

1) Вибрати ***Model→Domains*** *(Модель→ Домени);*

2) У вікні *List of Domains вибрати* *(Додати рядок)*

3) Ввести ім'я та код домену, натиснути (**=**);

4) Натиснути *Properties* ***(Властивості),*** в закладці General вибрати тип даних, в закладці Standard Checks *(Стандартні Перевірки)*

5) В діалоговому вікні (рис. 33) ввести параметри і натиснути ***OK***.

Для **визначення елементів даних** потрібно:

1) Вибрати ***Model→ Data Items*** *(Модель→ Елементи Даних);*

2) *У вікні* ***List of Data Items натиснути*** *(Додати рядок);*

3) Ввести ім'я та код (**=**) ;

4) Вибрати домен із списку ;

5) Вибрати тип даних;

6) Натиснути *Properties* ***(Властивості),*** в закладці General вибрати тип даних, в закладці Standard Checks *(Стандартні Перевірки)*, ввести параметри і натиснути ***OK***.

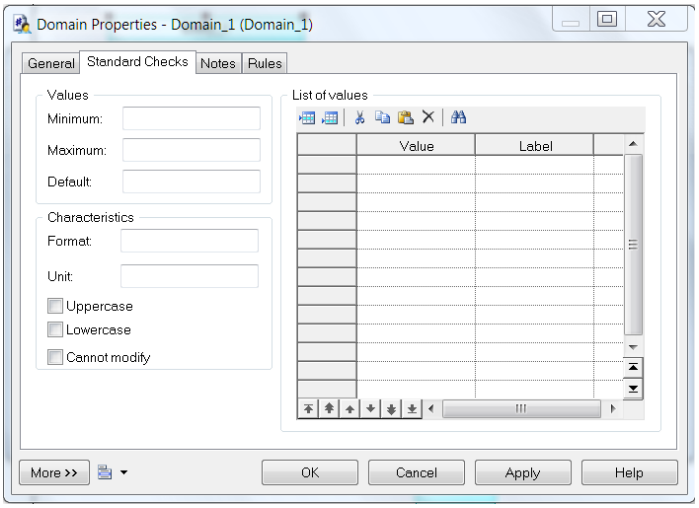
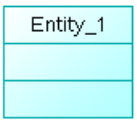


Рис. 33. Вікно параметрів контролю

Існують два шляхи **створення сутності**:

- вставити символ ***Entity*** *(Сутність)* в модель, для чого:

1) Вибрати на панелі інструментів значок ;

2) Вставити будь-де у вікні моделі зображення сутності 

;

3) Вибрати на панелі інструментів ***Pointer*** *(Покажчик)* ;

4) Двічі натиснути на зображенні створеної сутності в моделі;

5) Ввести ім'я та код (**=**), натиснути ***OK***.

- додати нову сутність в список сутностей, для чого:

1)Вибрати ***Model→ Entities*** *(Модель→ Сутності)*, натиснути кнопку ***Add a row*** *(Додати рядок);*

2)Ввести ім'я та код (=), натиснути ***OK***.

Існують чотири шляхи **створення атрибутів сутності**:

- використати атрибут даних як атрибут сутності, для чого:

1) Двічі натиснути на зображенні створеної сутності в моделі;

2) Натиснути вкладку ***Attributes*** *(Атрибути)*, натиснути ***Add*** *(Додати)**;***

3) Встановивши вибрати із списку ***Data Items*** *(Елементи Даних)* потрібні елементи даних, натиснути ***OK***.

- дублювати елемент даних як атрибут сутності:

1) Двічі натиснути на зображенні створеної сутності в моделі;

2) Натиснути кнопку ***Attributes*** *(Атрибути)*, натиснути ***Add Data Item*** *(Додати);*

3) Встановлюючи вибрати із списку ***Data Items*** *(Елементи Даних)* потрібні елементи даних;

4) Вибрати ***Duplicate*** *(Дублювати)*, натиснути ***OK***.

- повторно використати елемент даних в якості атрибуту даних.

Якщо було встановлено ***Allow Reuse*** в опціях елементу даних, то такий елемент даних може бути атрибутом для більш ніж однієї сутності. Повторне використання елементу даних не створює новий елемент даних. Елемент даних і відповідні атрибути сутності мають однаковий код. Для цього:

1) Двічі натиснути на зображенні створеної сутності в моделі;

2) Натиснути кнопку ***Attributes*** *(Атрибути)*, натиснути ***Reuse data Items****(Додати);*

3) Встановлюючи вибрати із списку ***Data Items*** *(Елементи Даних)* потрібні елементи даних;

4) Натиснути ***OK***.

- створити атрибут сутності безпосередньо в списку атрибутів:

1) Двічі натиснути на зображенні створеної сутності в моделі;

2) Натиснути кнопку ***Attributes*** *(Атрибути)*, натиснути Insert a row *(Вставити)*;

4) Ввести ім'я та код (=), натиснути ***OK***.

Для **закріплення атрибуту сутності за доменом** (тип даних домену заміщає тип даних інформації, який був попередньо визначений у атрибуту і автоматично з'являється в стовпчику Data Type) потрібно:

1) Двічі натиснути на зображенні створеної сутності в моделі;

2) Вибрати закладку ***Attributes*** *(Атрибути)*, вибрати атрибут;

3) Вибрати домен із списку, натиснути ***OK***.

Для **вибору типу даних атрибуту** треба:

1) Двічі натиснути на зображенні створеної сутності в моделі;

2) Вибрати закладку ***Attributes*** *(Атрибути);*

3) Натиснути ***…*** в стовпчику Data Type (Тип Даних), вибрати потрібний тип і натиснути ***OK.***

Порядок **визначення зв'язків**:

1) Вибрати на панелі інструментів значок зв'язку ;

2) Провести лінію від однієї сутності до іншої;

3) Двічі натиснути на зображенні створеного зв'язку, ввести ім'я та код (=), інші характеристики зв'язку (ролі, потужність тощо) і натиснути ***OK*** (рис. 34).

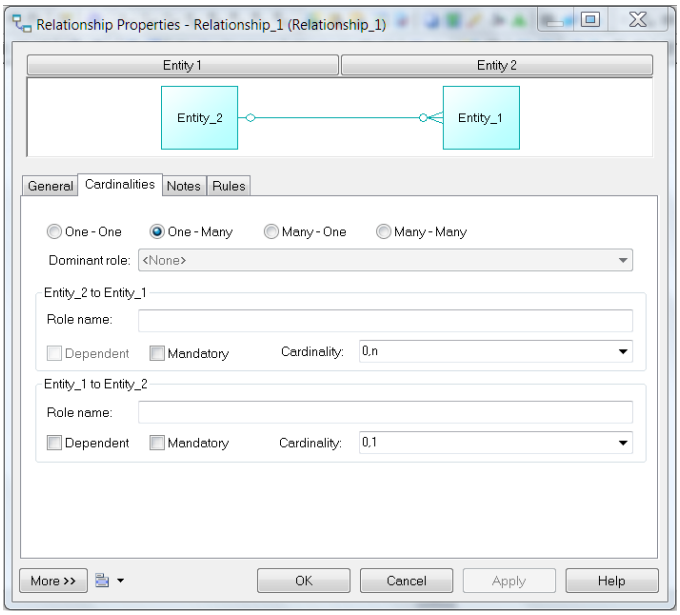


Рис.34. Вікно властивостей зв'язку

Особливості **визначення залежного зв'язку**. В залежному зв'язку одна сутність частково ідентифікується іншою, тобто ідентифікатор незалежної сутності стає первинним/зовнішнім ключем в таблиці, що генерується для залежної сутності. Для визначення залежного зв'язку потрібно:

1) Двічі натиснути на зображенні створеного зв'язку;

2) Ввімкнути або вимкнути ***Dependent*** *(Залежна)* і натиснути ***OK***.

Перевірка CDM:

1) ***Tools→Check Model*** *(Інструменти→Перевірити Модель);*

2) Вибрати об'єкти контролю: Package (Набір), Domain (Домен), Data items (Елементи даних), Entities (Сутності), Entities Attribute (Атрибути Сутностей), Relationships (Зв'язки); тип діагностичних повідомлень: Errors (Помилки) або Warning (Попередження); натиснути ***OK***.

Завдання:

1. Розробити модель **бібліотеки** (актори - керівник, бібліотекар, читач)
2. Розробити модель **провайдера Інтернет** (актори - керівник, співробітник по роботі з клієнтами, клієнт)
3. Розробити модель **системи охорони підприємства** (актори - керівник підприємства, співробітник підприємства, охоронець)
4. Розробити модель роботи **університету** (актори - ректор, декан, студент)
5. Розробити модель **салону з продажу мобільних телефонів** (актори - директор, продавець-консультант, клієнт)
6. Розробити модель **керування програмними проектами** (актори - керівник фірми, розроблювач, менеджер проектів)
7. Розробити модель **роботи комунального підприємства** (актори - директор, диспетчер, житель)
8. Розробити модель **ремонтної майстерні** (актори - директор, майстер, клієнт)
9. Розробити модель **міських електричних мереж** (актори - начальник, електрик, диспетчер)
10. Розробити модель **автосервісу** (актори - керівник, автомеханік, клієнт)

Лабораторна робота №4. Побудова діаграм варіантів використання Use Case diagram

Мета роботи

Знайомство з основними структурними компонентами, представленням варіантів використання. Придбання навичок:

- формування функціональних вимог до системи;
- виділення варіантів використання і їхніх описів за допомогою потоків подій;
- визначення діючих осіб;
- побудови діаграми варіантів використання.

Методичні вказівки

Діаграма варіантів використання відображає взаємодію між варіантами використання, що представляють функції системи, і діючими особами, що представляють людей чи системи, що одержують чи передають інформацію в дану систему.

Даний тип діаграм призначений для створення списку операцій, що виконує система, тому його іноді називають діаграмою функцій. Будь-яка система має свою множину варіантів використання і множину діючих осіб. Кожен варіант використання описує елемент своєї системою функціонування. Множина варіантів використання описує усю функціональність системи на деякому рівні абстракції. Абстракція (**abstraction**) - зосередження на найважливіших аспектах застосування й ігнорування всіх інших. Використання абстракції дозволяє зберегти волю прийняття рішень як можна довше завдяки тому, що деталі не фіксуються завчасно . Кожна діюча особа являє собою один вид об'єктів, для яких система може виконувати деяку поведінку.

Мова UML передбачає систему графічних позначень для варіантів використання (рис. 35).

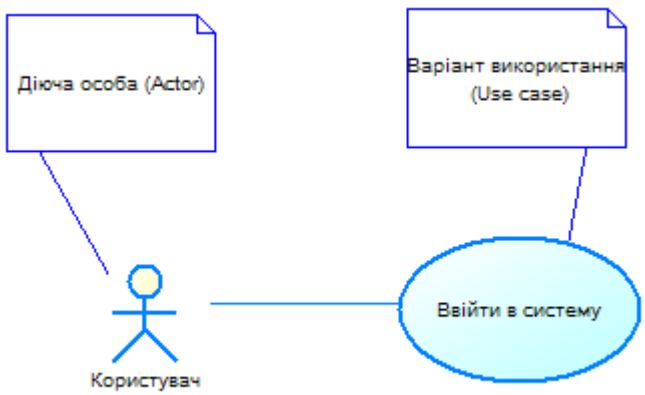


Рис. 35 – Графічні позначення діаграми варіантів використання

Діюча особа (**actor**) - це безпосередній зовнішній користувач системи. Це об'єкт чи множина об'єктів, безпосередньо взаємодіючих із системою. Кожна

діюча особа є узагальненням групи об'єктів, що ведуть себе певним чином стосовно системи. Діючими особами можуть бути люди, пристрої й інші системи - усе, що взаємодіє з цікавлячої нас системою безпосередньо.

Діюча особа повинна мати одну чітко визначену мету. Моделювання діючих осіб допомагає визначити границі системи, тобто ідентифікувати об'єкти, що знаходяться усередині системи, і об'єкти, що лежать на її границі.

Різні взаємодії діючих осіб із системою групуються у варіанти використання. Варіант використання (**use case**) - це зв'язний елемент функціональності, що представляється системою при взаємодії з діючими особами (рис. 36) .

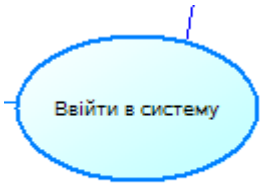


Рис. 36 – Варіант використання (**use case**)

У кожнім варіанті використання беруть участь одна чи кілька діючих осіб і система. Варіант використання поєднує всі поведінки, що мають відношення до елемента функціональності системи: нормальна поведінка, варіації нормальної поведінки, виняткові ситуації, збійні ситуації і скасування запитів.

Любой варіант використання повинний мати короткий опис, що пояснює дії в цьому варіанті, але в нього необхідно включити зведення про різні типи користувачів, що виконують даний варіант використання, і очікуваний результат. Під час роботи (особливо якщо проект складний) ці описи будуть нагадувати членам команди, чому той чи інший варіант використання був включений у проект і що він повинний робити. Чітко документуючи, таким чином, мети кожного варіанта використання, можна зменшити плутанину, що виникає серед розроблювачів.

Розробляючи діаграми варіантів використання, необхідно дотримуватись наступних правил:

- зв'язки між діючими особами не моделюють. По визначенню діючі особи знаходяться поза сферою дії системи. Це означає, що зв'язки між ними також не відносяться до її компетенції;
- два варіанти використання не з'єднують безпосередньо стрілкою. Для відображення порядку виконання варіантів використання застосовують діаграми діяльностей;
- кожен варіант використання повинний бути ініційований діючою особою. Це означає, що завжди є стрілка, що починається на діючій особі і закінчується на варіанті використання.

У мові UML мається кілька стандартних видів відносин між діючими особами і варіантами використання:

- відношення асоціації;
- відношення залежності;
- відношення узагальнення.

Асоціація - структурне відношення, що описує сукупність зв'язків, представлених з'єднаннями між об'єктами моделі. Виділяють різновид асоціації, агрегирование, передбачене для вираза відносин між цілим і його частинами (рис. 37).

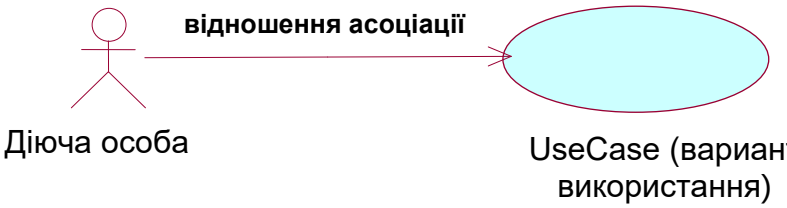


Рис. 37 – Приклад реалізації відносини асоціації

На діаграмі варіантів використання, так само як і на інших діаграмах, відношення асоціації позначається суцільною лінією між актором і варіантом використання. Ця лінія може мати додаткові умовні позначки, такі, наприклад, як ім'я і кратність

Кратність (**multiplicity**) асоціації вказується поруч з позначенням компонента діаграми, що є учасником даної асоціації. Кратність характеризує загальну кількість конкретних екземплярів даного компонента, що можуть виступати як елементи даної асоціації. Стосовно до діаграм варіантів використання кратність має спеціальне позначення у формі однієї чи декількох цифр і, можливо, спеціального символу «*» (зірочка).

Для варіантів використання і діючих осіб у мові UML підтримується кілька типів зв'язків:

- *зв'язки комунікації* (**communication**) - зв'язок між варіантом використання і діючую особою, напрямок стрілки показує, хто ініціює комунікацію;
- *узагальнення* (**generalization**) - показують, що кілька діючих осіб мають загальні риси;
- *включення* (**include**) - застосовується в тих ситуаціях, коли мається який-небудь фрагмент поведінки системи, що повторюється більш ніж в одному варіанті використання;
- *розширення* (**extend**) - застосовується при описі змін у нормальної поведінки системи, тобто можливості іншого варіанта - застосовуються лише при необхідності.

Узагальнення - відношення, при якому об'єкт спеціалізованого елемента може бути підставлений і використаний замість об'єкта узагальненого елемента. Варто підкреслити, що нащадок успадковує усі властивості і поведінку свого батька, а також може бути доповнений новими властивостями й особливостями поведінки. Графічно дане відношення позначається суцільною лінією зі стрілкою у формі незафарбованого трикутника, що вказує на батьківський варіант використання (рис. 38).

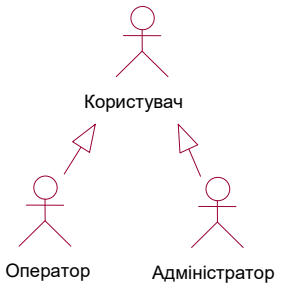


Рис. 38. Приклад графічного зображення відносин узагальнення між діючими особами

Відношення залежності між варіантами використання позначається пунктирною лінією зі стрілкою, спрямованої від того варіанта використання, що є розширенням для вихідного варіанта використання.

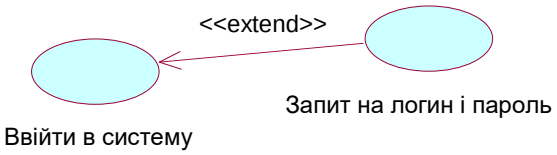


Рис. 39. Приклад графічного зображення відносини залежності (розширення) між варіантами використання

Конкретні деталі варіанта описуються в документі, називаному "потік подій". У ньому докладно описується, що роблять користувачі системи і що сама система. Опис потоку подій не повинний залежати від реалізації. Ціль - описати те, що робить система, а не як вона це робить.

Звичайно потік подій містить у собі:

- короткий опис;
- предумови (умови, що повинні бути виконани, перш ніж варіант використання почне виконуватися сам.);
- основний потік подій;
- альтернативний потік подій;
- Постумови (умови, що завжди повинні бути виконані після завершення варіанта використання, вони маються не в кожного варіанта).

Завдання:

Для побудови діаграми варіантів використання варіанти опису предметної області взяти из **лабораторної роботи №7.**

Контрольні питання

1. Що таке мова UML?
2. Які переваги дає застосування CASE-систем при розробці програмних систем?
3. Які UML діаграми доступні в Power Designer?
4. Для чого використовується діаграма Use Case?
5. Як створити нову діаграму?
6. Які значки знаходяться в рядку інструментів діаграми Use Case і яке їхнє призначення
7. Які типи зв'язку існують між елементами діаграми варіантів використання?

Лабораторна робота №5. Побудова діаграми класів Class diagram

Мета роботи

Придбати навики роботи для побудови діаграми класів (Class Diagram).

Порядок виконання роботи

1. Визначитися з основними об'єктами моделі, виходячи із проблематики завдання.
2. Створити новий файл моделі.

3. Визначити та створити класи, атрибути та асоціації
4. Перевірити коректність створеної ООМ і згенерувати скелетні коди застосування на обраній мові програмування.

Методичні вказівки

Діаграма класів (**Class Diagram**) - статична структурна діаграма, що описує структуру системи, вона демонструє класи системи, їх атрибути, методи і залежності між класами.

Діаграма класів визначає типи класів системи і різного роду статичні зв'язки, що існують між ними. На діаграмах класів зображуються також атрибути класів, операції класів і обмеження, що накладаються на зв'язку між класами. Вид і інтерпретація діаграми класів істотно залежить від точки зору (рівня абстракції): класи можуть представляти сутності предметної області (у процесі аналізу) чи елементи програмної системи (у процесах проектування і реалізації).

Діаграма класів представляє набір:

- класів
- типів даних
- інтерфейсів і
- відносин між ними.

Діаграма об'єктів представляє набір екземплярів класів і типів даних, найбільш типовим її використанням є представлення прикладів структур даних. Оскільки діаграма класів може включати у свій склад об'єкти, те окремого виду діаграми об'єктів не існує, це фактично підмножина діаграми класів.

Діаграми класів є відправною точкою процесу розробки. Вони допомагають при аналізі, даючи аналітику спілкуватися з клієнтом в «звичних» йому термінах і стимулюючи процес виявлення важливих деталей у проблемі, яку потрібно вирішити.

Клас - це категорія або група речей, яка має подібні атрибути та загальні властивості. Атрибути описують перелік значень, в рамках яких вказуються властивості об'єктів цього класу. Операція - це те, що може виконувати клас, або те, що ви можете виконувати над даними класом.

Імена класів найчастіше включають кілька слів. Кожне слово в імені класу починається з великої літери, пробіли між словами відсутні. Імена атрибутів і операцій будуються за тими ж правилами, але перша буква є рядковою.

На стадії аналізу використовуються діаграми класів для виділення загальної ролі й обов'язків сутностей, що забезпечують необхідне поведінки системи. На стадії проектування діаграмою класів користаються з метою передачі структури класів, що формують архітектуру системи.

У потоках подій варіанта використання виявляються класи трьох типів:

- граничні класи (**Boundary**) - служать посередниками при взаємодії зовнішніх об'єктів із системою. Як правило, для кожної пари "діюча особа - варіант використання" визначається один граничний клас. Типи граничних класів: користувальницький інтерфейс (обмін інформацією з користувачем, без деталей інтерфейсу - кнопок, списків, вікон), системний інтерфейс і апаратний інтерфейс (використовувані протоколи, без деталей їхньої реалізації);
- класу-сутності (**Entity**) - являють собою ключові абстракції (поняття) розроблювальної системи. Джерела виявлення класів-сутностей: ключові абстракції, створені в процесі архітектурного аналізу, глоссарий, опис потоків подій варіантів використання;

- керуючі класи (**Control**) - забезпечують координацію поводження об'єктів у системі. Можуть отсутствовать у деяких варіантах використання, що обмежуються простими маніпуляціями зі збереженими даними. Як правило, для кожного варіанта використання визначається один керуючий клас. Приклади керуючих класів: менеджер транзакций, координатор ресурсів, оброблювач помилок.

Для объектно-ориентированного стилю концептуальною базою є об'єктна модель. Вона має чотири головних елементи:

" абстрагування;
" інкапсуляція;
" модульность;
" ієрархія.

Абстракція виділяє істотні характеристики деякого об'єкта, що відрізняють його від всіх інших видів об'єктів і, таким чином, чітко визначає його концептуальні границі з погляду спостерігача.

Абстрагування зосереджує увагу на зовнішніх особливостях об'єкта і дозволяє відокремити самі істотні особливості поводження від несуттєвих

Вибір правильного набору абстракцій для заданої предметної області являє собою головну задачу объектно-ориентированного проектування.

Інкапсуляція - це процес відділення друг від друга елементів об'єкта, що визначають його пристрій і поводження; інкапсуляція служить для того, щоб ізолювати контрактні зобов'язання абстракції від їхньої реалізації

Абстракція й інкапсуляція доповнюють один одного: абстрагування спрямоване на поводження об'єкта, що спостерігається, а інкапсуляція займається внутрішнім пристроєм. Звичайно ховаються і внутрішня структура об'єкта і реалізація його методів

В объектно-ориентированном програмуванні необхідно фізично розділити класи й об'єкти, що складають логічну структуру проекту і скласти з них модулі найбільш ефективним образом. Приведемо кілька визначень модульности.

Модульность - це поділ програми на фрагменти, що компілюються по окремості, але можуть установлювати зв'язку з іншими модулями. Модульность - це властивість системи, що була розкладена на внутрішньо зв'язкові, але слабко зв'язані між собою модулі.

Таким чином, принципи абстрагування, інкапсуляції і модульности є взаємодоповнюючими. Об'єкт логічно визначає границі визначеної абстракції, а інкапсуляція і модульность роблять їх фізично непорушними.

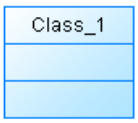
Значне спрощення в розумінні складних задач досягається за рахунок утворення з абстракцій ієрархічної структури. **Ієрархія** - це упорядкування абстракцій, розташування їх по рівнях.

Створення діаграми класів Class Diagram

Виберіть ***File> New Model*****,** щоб відкрити діалогове моделі. Виберіть тип моделі **ObjectOrientedModel**(Об'єктно-орієнтованої моделі). Тип діаграми - Діаграма класів (**Class Diagram**).

Створення класу

1) На панелі інструментів Pallette вибираємо (**Class**) Клас.

2) В робочій області створюємо клас 

.

3) Двічі натискаємо на зображення класу. У вікні властивості (**Class Properties**) вказується:

•Вкладка General: Назва класу і назва в програмному коді в полях Ім'я і код.

• *Вкладка Attributes* - вносяться атрибути класу. Кожен атрибут має свої властивості, їх можна вказати у властивостях атрибута (для цього натиснути на введений атрибут два рази лівою кнопкою миші - відкриється форма Властивості атрибутів). У властивостях атрибута заповнюються поля ім'я та код, вибирається тип для кожного значення атрибута (тип даних рядок). Для того щоб вибрати значення атрибута за замовчуванням, необхідно перейти на вкладку Детально (**Detail**) і в поле Початкове значення (**Initial value**) вписати задане значення.

• Вкладка Operation - вносяться операції, які може виконувати цей клас. Кожна операція має свої властивості, їх можна вказати у властивостях операції (для цього натиснути на створену операцію два рази лівою кнопкою миші - відкриється форма Властивості операції). У властивостях операції заповнюються поля ім'я та код, вибирається тип значення, що повертається (поле **Return type**). На вкладці Параметри (**Parameters**) можна вказати параметри операції та їх типи.

Якщо список атрибутів і / або операцій занадто великий, можна уточнити інформацію за допомогою ключового слова (стереотипу), поле стереотип.

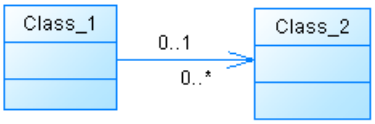
Також додаткову інформацію можна внести в поле коментаря Note (Примітка).

Створення асоціації

Якщо класи взаємодіють один з одним, то така взаємодія називається асоціацією .

1) На панелі інструментів Pallet вибираємо  асоціацію (**Association**)

2) Створюємо асоціацію між двома класами:



3) Два рази натискаємо на зображення асоціації, у вікні властивості асоціації (Association Properties) вказується:

• Вкладка General: назва асоціації і назва в програмному коді (поля ім'я та код).

• Якщо один клас асоціюється з іншим, кожен з них грає свою роль в цій асоціації. Ролі вказуються в закладці Detail (Детально) в полях Role name (Ім'я ролі) для Class1 і Class2.

Асоціації можуть працювати в різних напрямках, а також можуть бути більш складними - з одним класом може асоціюватися кілька інших. Подібно класам, асоціація може мати атрибути та операції. Для відображення класу асоціації в PowerDesigner використовується той же інструмент асоціації, але з'єднуються не два класи а асоціація і клас. А клас асоціації сам може бути пов'язаний з іншими класами.

Кратність - кількість об'єктів одного класу, які можуть бути пов'язані з певною кількістю об'єктів іншого класу, рис 40. (В PowerDesigner - відкрити у

властивостях асоціації вкладку Подробиці і встановити вказати кратність в поле кратності). Кратність може бути: 1:1, 1: *, *: 1, * (де * - до багатьох):

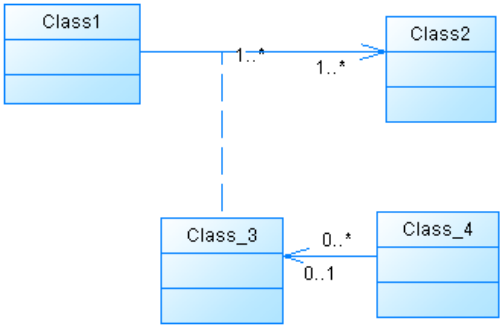


Рис. 40. Приклад з'єднання асоціації і класу

Клас може асоціюватися сам з собою (рефлекторна асоціація), рис. 41.

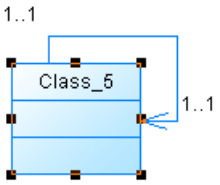


Рис. 41. Приклад рефлекторної асоціації

Спадкування: клас може успадковувати атрибути і операції іншого класу. Успадкований клас є дочірнім по відношенню до батьківського, від якого він успадковується. В PowerDesigner для зображення успадкування використовується інструмент узагальнення (Generalization).

Абстрактні класи призначені тільки для використання в якості базових для наслідування і не породжують своїх об'єктів.

Залежність: взаємозв'язок при використанні одного класу іншим називається залежністю. Найбільш загальний випадок залежності - це використання одного класу в сигнатурі операції іншого класу. Для зображення залежно застосовується інструмент залежностей (Dependency).

Агрегація - взаємозв'язок, при якій клас складається з деякої кількості класів-компонентів. Компоненти і клас, який вони складають, знаходяться в асоціації частина-ціле. Агрегацію можна представити у вигляді дерева, коренем якого є «ціле», а листям - його компоненти. В PowerDesigner застосовується інструмент агрегації . Наприклад:

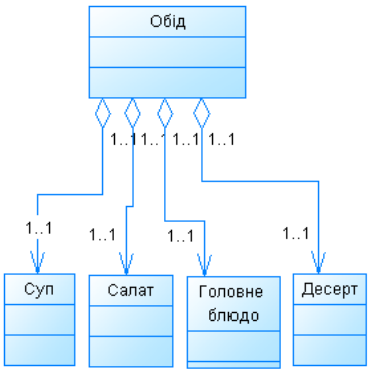


Рис. 42. Приклад агрегації класів

Композиція - це строгий тип агрегації, що характеризується тим, що кожен елемент може належати тільки одному цілому. Наприклад, компоненти стіл - стільниця і ніжки - складають композит. Для відображення в PowerDesigner використовується інструмент Композиція (Composition) .

Інтерфейси і реалізація:

Інтерфейс - це набір операцій, які задають деякі аспекти поведінки класу і представляють його для інших класів .

Є два варіанти створення інтерфейсу В PowerDesigner. Перший варіант - додати за допомогою інструменту Інтерфейс (Interface) (властивості заповнюються аналогічно властивостям класу), другий варіант - додати автоматично - натиснути правою кнопкою миші на створений клас і в списку вибрати *Create Interface* (Створити інтерфейс), в цьому випадку операції, прописані в створеному класі, додадуться автоматично.

Взаємозв'язок між класом і його інтерфейсом називається відношенням реалізації - інструмент реалізації (Realization).

Видимість: даний термін застосовується по відношенню до атрибутів та операцій. Виділяють три області видимості:

• Відкрита - можуть використовувати інші класи;
• Захищена - можуть використовувати тільки спадкоємці даного класу;
• Закрита - використовуються тільки самими класами.

В PowerDesigner видимість зазначається у властивостях у вкладці General в полі видимості (Visibility).

Для генерації скелетних кодів застосування на обраній мові програмування необхідно при відкритій об’єктній моделі скористатися командами ***Language->Change Current Object Language***… та ***Language->Generate*** *<обрана мова програмування>.*

Завдання:

Підготуйте діаграму класів для кожної групи класів. Додайте на діаграму не менш 10 відносин, атрибути і методи. При необхідності використовуйте імена асоціацій. Використовуйте також кваліфіковані асоціації, указуйте кратність і коментарі. При підготовці діаграм можна додавати додаткові класи.

1) школа, директор, клас, книга, учень, вчитель, комп'ютер, вчительська;
2) автомобіль, двигун, колесо, гальмо, двері, акумулятор, глушитель;
3) замок, рів, звідний міст, вежа, сходи, підземелля, поверх, коридор, кімната;
4) комп'ютерна програма, константа, перемінна, функція, список аргументів, арифметичний оператор, оператор, оператор відносин, вирази;
5) файлова система, файл, двоичный файл, текстовий файл, каталог, диск, сектор, доріжка;
6) шахова фігура, поле, шахівниця, горизонталь, вертикаль, хід;
7) аеропорт, літак, пілот, пасажир, посадковий талон, стюардеса;
8) словник, слово, частина мови, синонім, антонім, значення слова;
9) комп'ютер, користувач, дисплей, клавіатура, миша, принтер, CD-ROM, твердий диск, пам'ять, дисковод.

Лабораторна робота №6. Розробка діаграм станів Statechart diagram

Мета роботи

Придбання навиків побудови діаграм станів в середовищі PowerDesigner .

Методичні вказівки

Кожен об'єкт системи, що володіє визначеной поведінкою, може знаходиться у визначених станах, переходити зі стану в стан, роблячи визначені дії в процесі реалізації сценарію поведінки об'єкта. Поведінка більшості об'єктів реальних систем відбивається в його станах, і даний тип діаграм дозволяє відобразити це графічно. Для цього використовується два види діаграм: **Statechart diagram** (діаграма станів) і **Activity diagram** (діаграма активності).

Діаграми станів (**statechart diagrams**) визначають усі можливі стани, у яких може знаходитися конкретний об'єкт, а також процес зміни станів об'єкта в результаті настання деяких подій. Діаграми станів створюються для опису об'єктів з високим рівнем динамічного поводження. Якщо об'єкт класу може існувати в декількох станах і в кожнім з них поводиться по-різному, для нього може знадобитися діаграма станів.

Кожна діаграма станів у UML описує всі можливі стани одного екземпляра визначеного класу і можливі послідовності його переходів з одного стану в інше, тобто моделює всі зміни станів об'єкта як його реакцію на зовнішні впливи (рис..42).

Діаграма станів є графом спеціального виду, що представляє деякий автомат. Вершинами графа є можливі стани автомата, зображувані відповідними графічними символами, а дуги позначають його переходи зі стану в стан.

Діаграми станів можуть бути вкладені друг у друга для більш детального представлення окремих елементів моделі.

Виділимо ряд визначень:

Автомат (State machine) -це опис послідовності станів, через які проходить об'єкт протягом свого життєвого циклу, реагуючи на події, - у тому числі опис реакцій на ці події.

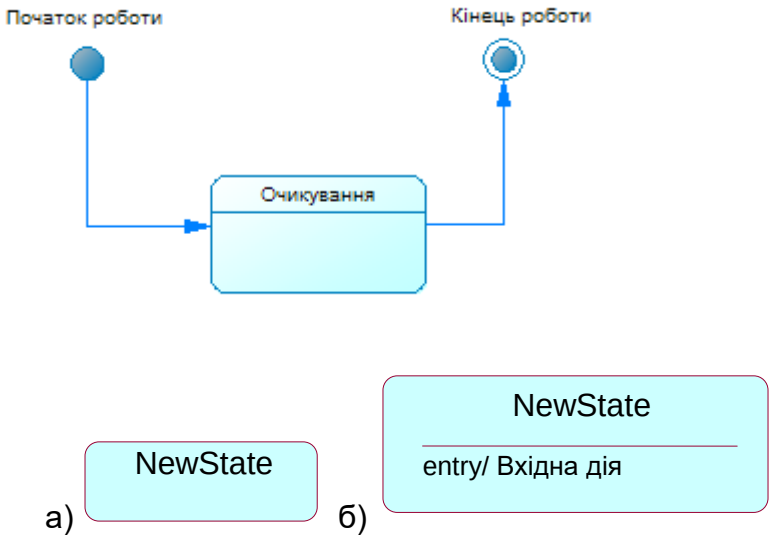
Стан (State) - це ситуація в житті об'єкта, протягом якого він задовольняє деякій умові, здійснює визначену діяльність чи очікує якоїсь події.

Подія (Event) - це специфікація істотного факту, що відбувається в часі і просторі. У контексті автоматів подія - це стимул, здатний викликати спрацьовування переходу.

Перехід (Transition) - це відношення між двома станами, що показує, що об'єкт, що знаходиться в першому стані, повинний виконати деякі дії і перейти в другий стан, як тільки відбудеться визначена подія і будуть виконані задані умови. . На діаграмі всі переходи зображують у виді стрілки, що починається на первісному стані і закінчується на наступному. Переходи можуть бути рефлексивним. Рефлексивні переходи зображують у виді стрілки, що починається і завершується на тому самому стані. У переходу існує кілька специфікацій. Вони включають події, аргументи, що обгороджують умови, дії і посылаемые події.

Дія (Action) - це атомарне обчислення, що приводить до зміни стану чи поверненню значення.

Графічно стан відображається у виді прямокутника з закругленими вершинами.



а) стан без вказівки в ньому дій чи переходів

б) стан із указівкою дій чи переходів

Рис. 42. Приклади діаграми станів

На діаграмі маються два спеціальних стани - початковий і кінцевий. Початковий стан на діаграмі станів може бути тільки один і позначається чорною крапкою. Кінцевих станів може бути стільки, скільки буде потрібно, чи їх може не бути взагалі, вони позначаються чорною крапкою в білому кружку.
Коли об'єкт знаходиться в якомусь конкретному стані, можуть виконуватися різні процеси. Процеси, що відбуваються в момент, коли об'єкт знаходиться у визначеному стані, називаються діями (**actions**).

Зі станом можна зв'язувати наступні дані: *діяльність, вхідна дія, вихідна дія і подія.*

Діяльність (activity) - це поведінка, реалізована об'єктом, поки він знаходиться в даному стані. Це поведінка, що переривається. Діяльність зображують усередині самого стану, їй повинне передувати слово **do** (виконувати) і **двокрапка**.

Вхідна дія (entry action) - це поведінка, що виконується, коли об'єкт переходить у даний стан. Вхідну дію також показують усередині стану, йому передує слово **entry** (вхід) і **двокрапка**.

Вихідна дія (exit action) подібна вхідної. Однак вона здійснюється як складова частина процесу виходу з даного стану. Вихідну дію зображують усередині стану, їй передує слово **exit** (вихід) і **двокрапка**.

Поведінка об'єкта під час діяльності, при вхідних і вихідних діях, може включати відправлення події іншому об'єкту. У результаті одержання об'єктом деякої події, також може виконуватися діяльність.

Подію розміщають на діаграмі уздовж лінії переходу. Для відображення події можна використовувати як ім'я операції, так і звичайну фразу. Обмежуючі умови зображуються на діаграмі уздовж лінії переходу після імені події і полягають у квадратні дужки. Дію зображують уздовж лінії переходу після імені події, йому передує коса риса.

Завдання:

Намалюйте діаграму станів

1) **розсувних сходів.** Розсувні сходи постачені мотузкою, шківом і засувкою для підйому, опускання і фіксування. Коли засувка зафіксована, можна лізти по сходам нагору. Щоб відчепити засувку, потрібно злегка підняти розсувний сегмент мотузкою. Після цього його можна вільно піднімати чи опускати. Засувка клацає при проходженні по ступінях сходів. Засувка може бути зафіксована при підйомі сходів зміною напрямку відразу після того, як вона пройшла чергову ступінь.

2) **годинника.** Найпростіший цифровий годинник складається з дисплея і двох кнопок A и B. Годинник може працювати в 2-х режимах: відображення і настроювання. У режимі відображення годинник показує години та хвилини, між якими мигає ':'. Режим настроювання складається з 2-х підрежимів: настроювання годин і настроювання хвилин. Кнопка A дозволяє вибрати режим. Щораз при її натисканні відбувається перехід до чергового режиму в послідовності: відображення, установка годин, установка хвилин, відображення і т.д. Кнопка B дозволяє збільшувати значення годин чи хвилин на одиницю при кожнім натисканні в одному з режимів установки. Щоб кнопка могла породити нову подію, її необхідно відпустити.

3) **телефонного автовідповідача.** Автовідповідач детектирует вхідний дзвоник по першому ж сигналі і відповідає заздалегідь записаним повідомленням. Коли повідомлення завершується, автовідповідач записує повідомлення що дзвонить. Коли абонент вішає трубку, автовідповідач теж вішає трубку і відключається до наступного дзвоника.

4) **контролера диска.** Контролер передає ведучому вузлу сигнал про кожен новий доступний байт. Дані повинні бути прочитані і збережені для того, щоб контролер міг перейти до наступного байту. Коли контролер виявляє, що дані були прочитані, він повідомляє про відсутність даних доти, поки не підготує наступний байт. Якщо байт не буде прочитаний до того, як контролер підготує наступний, контролер видає сигнал утрати даних доти, поки не одержить сигнал скидання.

5) для виділення і перетаскування об'єктів у **редакторі діаграм.** Курсор керується двухкнопочной мишею. При натисканні лівої кнопки в той момент, коли курсор знаходиться над об'єктом, об'єкт виділяється (при цьому виділення знімається з кожного раніше виділеного об'єкта). Якщо ліва кнопка натискається в той момент, коли курсор не знаходиться над об'єктом, виділення знімається з усіх раніше виділених об'єктів. Переміщення миші з натиснутою лівою кнопкою приводить до перетаскування виділеного об'єкта.

6) **копіювального апарата.** У початковому стані копіювальний апарат виключений. Включення живлення переводить апарат в основний стан: одна копія, автоматичне настроювання контрасту, нормальний розмір. У процесі прогріву апарат мигає індикатором готовності. Коли самоперевірка автомата завершується, індикатор готовності перестає мигати і починає горіти безупинно. Після цей автомат вважається готовим до роботи. Можна змінити будь-який параметр (кількість копій, настроювання контрасту, розмір), поки апарат знаходиться в стані готовності до роботи. Після натискання кнопки "Пуск" апарат починає копіювання. Копіювання продовжується доти, поки не буде зроблене потрібна кількість копій.

7) **системи охоронної сигналізації.** Система починає своє життя з ініціалізації, потім переходить у стан очікування. У цьому стані через кожні 10 секунд виконується самоперевірка системи. При настанні події "Тривога" реалізуються дії, зв'язані з блокуванням периметра охоронюваного об'єкта, і

здійснюється перехід в активний стан. В активному стані через кожні 5 секунд виконується перевірка надходження команди "Скидання". Якщо команда отримана, система повертається в стан очікування. У процесі повернення розблокується периметр охоронюваного об'єкта.

8) **торгового автомата.** Споконвічно автомат знаходиться в бездіяльності. Коли людина кидає монету в автомат, той додає її ціну до загального балансу. Поклавши монети в автомат, людина може вибрати товар. Якщо даний товар закінчився чи грошей для нього недостатньо, автомат чекає, поки людина вибере що-небудь інше. У противному випадку автомат видає товар і, якщо необхідно, повертає здачу.

Лабораторна робота №7. Побудова діаграм діяльності Activity diagram

Мета роботи

Придбання навиків моделювання поведінки системи за допомогою діаграм діяльності.

Методичні вказівки

Діаграми діяльностей або діаграми активності (**activity diagrams**) призначені для моделювання поведінки системи в рамках різних варіантів використання чи моделювання діяльностей. Вони особливо корисні в описі поведінки, що включає велику кількість рівнобіжних процесів.

Це подальший розвиток діаграми станів. Фактично даний тип діаграм може використовуватися і для відображення станів моделируемого об'єкта, однак, основне призначення **Activity diagram** у тім, щоб відбивати бізнес-процеси об'єкта. Цей тип діаграм дозволяє показати не тільки послідовність процесів, але і розгалуження і навіть синхронізацію процесів.

Цей тип діаграм дозволяє проектувати алгоритми поводження об'єктів будь-якої складності, у тому числі може використовуватися для складання блок-схем.

Діаграми діяльності корисно використовувати в наступних ситуаціях:

- аналіз варіанта використання. На цій стадії цікавить не зв'язок між діями й об'єктами, а те, які дії повинні мати місце, і які залежності в поведінки системи. Зв'язування методів і об'єктів виконується пізніше за допомогою діаграм взаємодії;
- аналіз потоків робіт (workflow) у різних варіантах використання. Коли варіанти використання взаємодіють один з одним, діаграми діяльностей є могутнім засобом представлення й аналізу їхньої поведінки.На рис.43. приведено приклад діаграми діяльності для варіанта запису на прийом до лікаря інформаційної системи поліклініки.

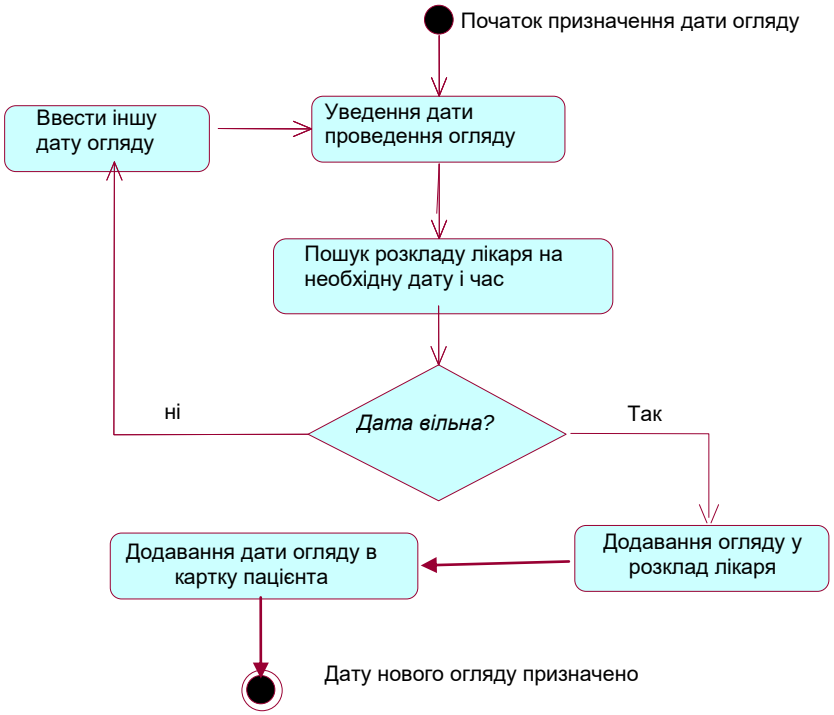


Рис. 43. Приклад діаграми діяльності для варіанта запису на прийом до лікаря інформаційної системи поліклініки.

Завдання:

Намалюйте діаграму дій для

1) опису обчислення рахунка в ресторані. За кожне блюдо варто узяти визначену плату. Крім того, із загальної суми стягується податок, а також 18% надбавка за обслуговування груп більш 6 чоловік. Для груп меншої чисельності варто залишити поле для запису чайових.

2) наступного процесу: користувач вирішує модернізувати свій ПК і купити DVD-ROM. Він починає з дзвоника у відділ продажів виробника, відкіля його відправляють у службу підтримки. Він дзвонить у службу підтримки, де його переводять у режим очікування на час розмови з інженерами. Нарешті, служба підтримки повідомляє йому про можливі моделі DVD-ROM. Користувач вибирає модель і замовляє доставку поштою. Він одержує покупку, успішно встановлює DVD-ROM у ПК, після чого відправляє поштою оплачений рахунок.

3) наступного процесу: компанія робить новий продукт і повинна координувати роботу декількох відділів. Продукт починає своє існування з маркетингової ідеї, що передається в проектний відділ. Проектний відділ моделює функції продукту і підготовляє проект. Відділ виробництва вивчає проект і коректує його, приводячи у відповідність з наявним устаткуванням. Проектний відділ приймає зміни, після чого проект вивчає служба підтримки. Проектний відділ приймає пропозиції служби підтримки і перевіряє, чи задовольняє проект після коректувань вимогам, пред'явленим до цільової функціональності.

4) надання премій частим клієнтам авіакомпаній. За кожен політ нараховується 750 км преміальних. Власники золотих і червоних карт за кожен політ одержують 1000 км преміальних. Золота карта видається при накопичуванні 5000 км. Червона карта видається при накопичуванні 10000 км.

Знижка для власника золотої карти складає 25%, для власника червоної карти - 50 %.

5) обробки замовлення брокерською системою на покупку акцій. Брокерська система починає з перевірки замовлення. Потім замовлення реалізується на фондовій біржі. Якщо виконання замовлення проходить успішно, система виконує три дії одночасно: відправляє клієнту підтвердження, обновляє портфель цінних паперів з урахуванням результатів угоди і закінчує угоду з другою стороною, знімаючи засоби з рахунка клієнта і перелічуючи наявні. Коли всі три дії закінчені, система закриває замовлення. Якщо ж виконання замовлення виявляється невдалим, система відправляє повідомлення про це клієнту і теж закриває замовлення.

6) процедури зняття грошей у банкомату. Клієнт уставляє карту в банкомат і вводить PIN код. Банкомат зчитує карту і PIN код, і відправляє цю інформацію в банк для перевірки. У базі даних банку проходить перевірка. Якщо перевірка пройшла успішно, банкомат видає гроші клієнту, інакше видає інформацію про неправильний PIN код і видає карту клієнту. Після видачі грошей банкомат запитує клієнта, про потребу друкувати чек. При позитивній відповіді чек друкується.

7) процедури покупки квитка на концерт. Клієнт дзвонить у касу. Касир запитує, який концерт його цікавить. Клієнт повідомляє цікавлячий його концерт. Касир повідомляє клієнту про вільні місця і ціни на квитки. Клієнт вибирає місце і ціну. Касир резервує квиток і запитує про спосіб оплати. Клієнт повідомляє необхідну інформацію. Касир відправляє квиток поштою .

8) процесу готування кава. Людина заливає воду в резервуар для води кавоварки, насипає каву у фільтр і установлює фільтр у кавоварку. Після цього він включає машину і на ній загоряється лампочка. Машина варить каву і, коли кава готова, лампочка стухає. Людина виключає машину і наливає каву в чашку.

9) процедури входу в систему з введенням імені користувача і пароля.

Лабораторна робота №8. Побудова діаграм послідовностей Sequence diagram

Мета роботи

Здобуття навиків до побудови діаграми послідовності

Методичні вказівки

Існують два види діаграм взаємодії: діаграми послідовності (**sequence diagrams**) і кооперативні діаграми (**collaboration diagrams**).

Діаграми послідовності відбивають тимчасову послідовність подій, що відбуваються в рамках варіанта використання, а кооперативні діаграми зосереджують увагу на зв'язках між об'єктами. Як правило, діаграма взаємодії охоплює поведінки об'єктів у рамках тільки одного варіанта використання. На такій діаграмі відображається ряд об'єктів і ті повідомлення, якими вони обмінюються між собою. Усі діючі особи показані у верхній частині діаграми. Стрілки відповідають повідомленням, переданим між діючую особою і об'єктом

чи між об'єктами для виконання необхідних функцій. Діаграма будується як графік і має два виміри. По вертикалі відкладається час, що може бути схематичним чи може мати реальний масштаб, по горизонталі відображаються об'єкти (рис.44).

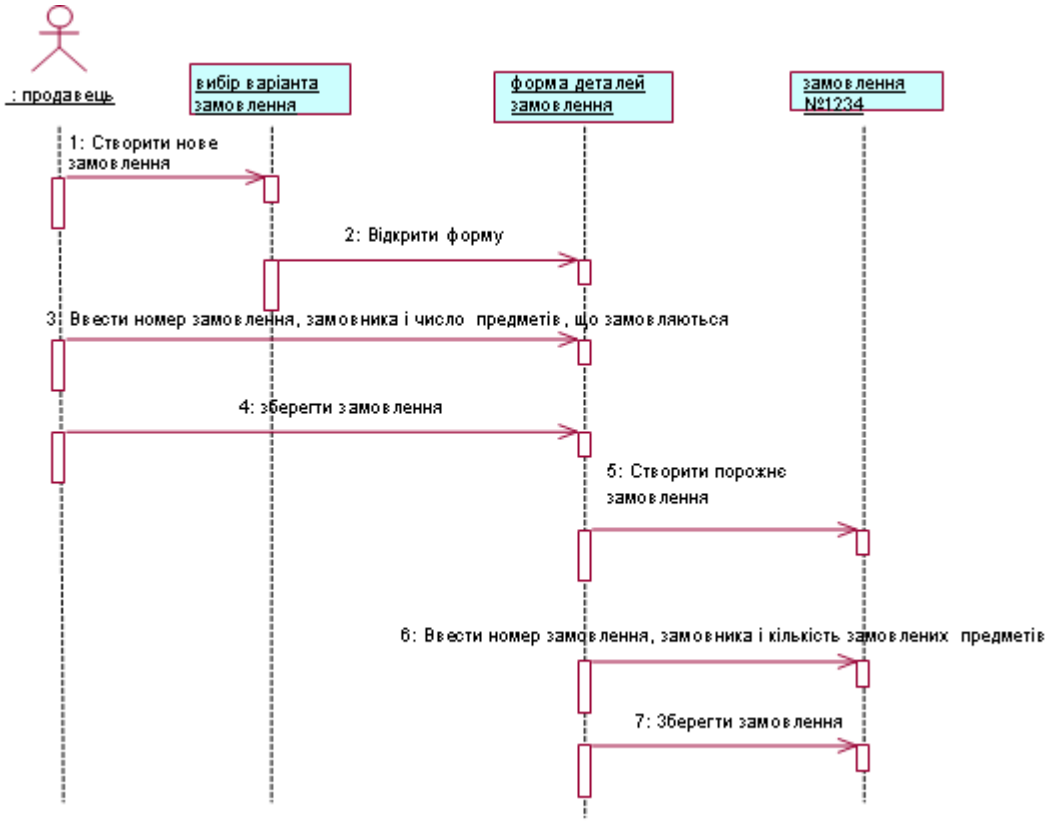
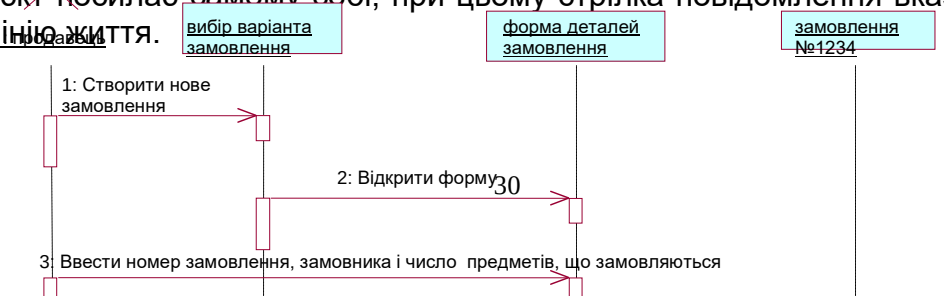


Рис. 44. Приклад діаграми послідовностей.

Діаграма складається з наступних елементів:

- об'єкт, позначається прямокутником із записаним у ньому ім'ям об'єкта;
- лінія життя об'єкта, штрих - пунктирна лінія, що виходить з об'єкта і розташована уздовж осі часу, позначає час життя об'єкта;
- активація, тонкий вертикальний прямокутник, розташований уздовж осі часу об'єкта, що позначає період активного життя об'єкта, або виходить з об'єкта;
- виклик методу поводження об'єкта (повідомлення), позначається стрілкою між активаціями об'єктів з ім'ям дії, напрямок стрілки задає напрямок передачі даних;
- текстові мітки (оцінки часу, опис дій і т.п.)

Повідомлення з'являються в тім порядку, як вони показані на сторінці зверху вниз. Кожне повідомлення позначається як мінімум ім'ям повідомлення. При бажанні можна додати також аргументи і деяку керуючу інформацію. Можна показати самоделегування (self-delegation) - повідомлення, що об'єкт посилає самому собі, при цьому стрілка повідомлення вказує на ту ж саму лінію життя.



Завдання:

Намалюйте діаграму послідовностей для одного зі сценаріїв:

1. комп'ютерна система електронної пошти
2. Інтернет-система бронювання авіаквитків
3. комп'ютерна система, що забезпечує видачу книг у бібліотеці
4. програмне забезпечення, що керує колекцією музичних треків
5. покупка бензину на автозаправній станції
6. підйому на ліфті
7. обіду в ресторані
8. одержання грошей у банкомату
9. здачі іспиту

Лабораторна робота №9. Розробка діаграм співробітництва Collaboration diagram

Мета роботи

Придбання навиків при розробці диаграммы кооперации

Методичні вказівки

Діаграми кооперації **(collaboration diagram**) загострюють увагу на зв'язках між об'єктами. На діаграмі кооперації представлена вся та інформація, що є і на діаграмі послідовності, але кооперативна діаграма по-іншому описує потік подій. З неї легше зрозуміти зв'язки між об'єктами, однак, сутужніше усвідомити послідовність подій.

На кооперативній діаграмі так само, як і на діаграмі послідовності, стрілки позначають повідомлення, обмін якими здійснюється в рамках даного варіанта використання. Їхня тимчасова послідовність указується шляхом нумерації повідомлень.

Як правило, діаграма взаємодії охоплює поводження об'єктів у рамках тільки одного варіанта використання. На такій діаграмі відображається ряд об'єктів і ті повідомлення, якими вони обмінюються між собою.

Повідомлення (**message**) ~ засіб, за допомогою якого об'єкт-відправник запитує в об'єкта-одержувача виконання однієї з його операцій.

Інформаційне (**informative**) повідомлення - повідомлення, що постачає об'єкт-одержувач інформацією для відновлення його стану.

Повідомлення-запит (**interrogative**) - повідомлення, що запитує видачу інформації про об'єкт-одержувача.

Імперативне (**imperative**) повідомлення - повідомлення, що запитує в об'єкта-одержувача виконання дій.

На кооперативній діаграмі так само, як і на діаграмі послідовності, стрілки позначають повідомлення, обмін якими здійснюється в рамках даного варіанта використання. Їхня тимчасова послідовність, однак, указується шляхом нумерації повідомлень.

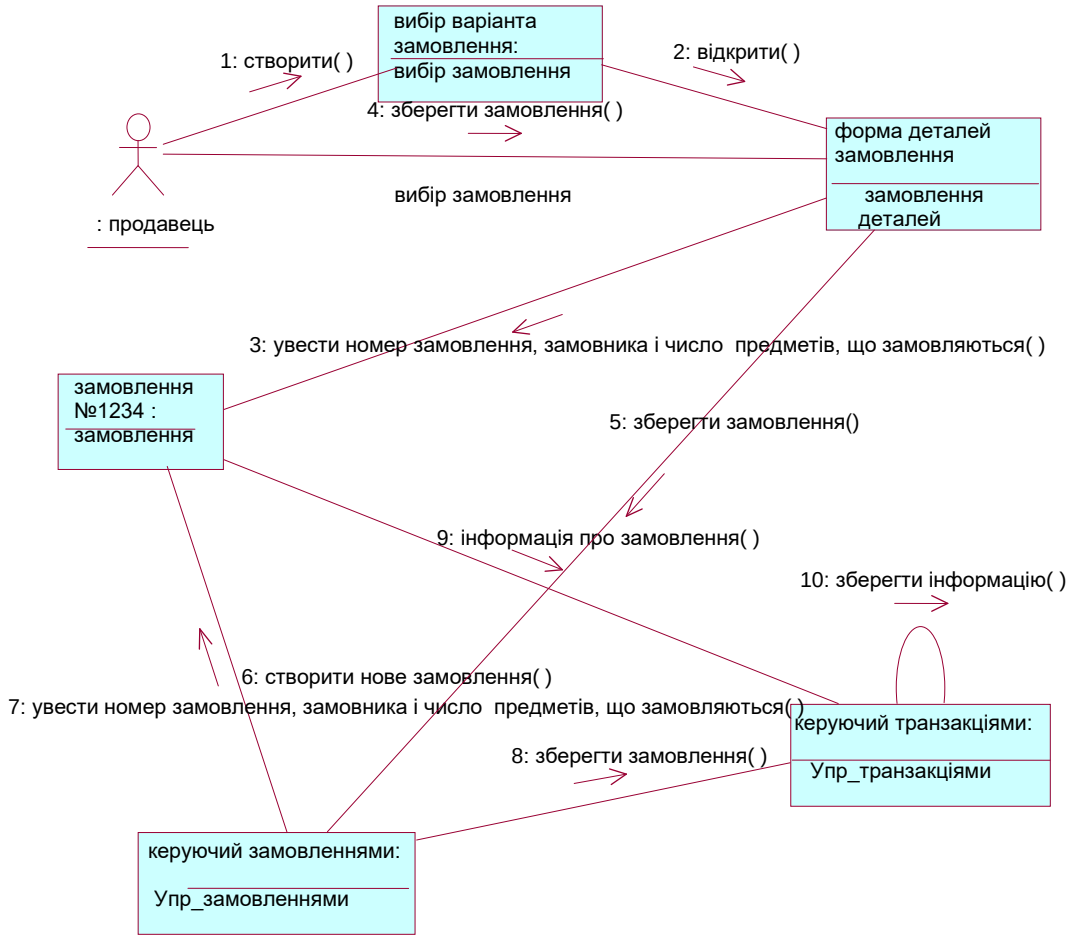


Рис. 45. Приклад кооперативної діаграми.

Завдання:

Переробіть діаграму послідовностей з попереднього завдання в діаграму кооперації

Лабораторна робота №10. Побудова діаграм компонентів Component diagram

Мета роботи

Представлення компонентів та придбання навичок моделювання фізичної архітектури системи і моделювання ієрархії компонентів (підсистем) системи.

Методичні вказівки

Повний проект програмної системи являє собою сукупність моделей логічного і фізичного рівнів, що повинні бути погоджені між собою. У мові UML для фізичного представлення моделей систем використовуються діаграми

реалізації (implementation diagrams), що містять у собі діаграму компонентів і діаграму розміщення (розгортання).

Діаграми компонентів (**component diagrams**) створюються для моделювання ієрархії компонентів (підсистем) системи і показують, як виглядає модель на фізичному рівні. На них зображені компоненти програмного забезпечення і зв'язки між ними. При цьому на такій діаграмі виділяють два типи компонентів: компоненти, що виконуються, і бібліотеки коду. Кожен клас моделі (чи підсистема) перетвориться в компонент вихідного коду. Між окремими компонентами зображують залежності, що відповідають залежностям на етапі компіляції чи виконання програми.

Основними графічними елементами діаграми компонентів є компоненти, інтерфейси і залежності між ними.

Діаграма компонентів розробляється для наступних цілей:

- візуалізації загальної структури вихідного коду програмної системи;
- специфікації варіанта програмної системи, що виконується;
- забезпечення багаторазового використання окремих фрагментів програмного коду;
- представлення концептуальної і фізичної схем баз даних.

Компонент реалізує деякий набір інтерфейсів і служить для загального позначення елементів фізичного представлення моделі. Для графічного представлення компонента використовується спеціальний символ - прямокутник із уставленими ліворуч двома більш дрібними прямокутниками. Усередині великого прямокутника записується ім'я компонента і, при необхідності, деяка додаткова інформація. Зображення цього символу може незначно варіюватися в залежності від характеру ассоциируемой з компонентом інформації (рис. 46).

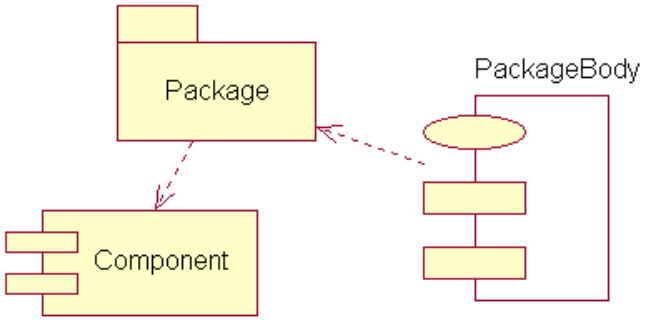


Рис.46. Диаграма компонентів.

На такій діаграмі показана відповідність класів реалізованим компонентам. Діаграма компонентів забезпечує погоджений перехід від логічного представлення до конкретної реалізації проекту у формі програмного коду. Одні компоненти можуть існувати тільки на етапі компіляції програмного коду, інші на етапі його виконання. Діаграма компонентів відбиває загальні залежності між компонентами, розглядаючи останні як класифікатори. При проектуванні великих систем може виявитися, що система повинна бути розкладена на кілька сотень чи навіть тисяч компонентів, і цей тип діаграм дозволяє не втратитися в достатку модулів.

Єдиний можливий тип зв'язків між компонентами - це залежність, що означає, що один компонент залежить від іншого.

Залежність зображується штриховою лінією від компонента, що використовує, до використовуваного. **Композиція** (чи включення) зображується розміщенням компонента, що включається, усередині включаючого. Компоненти можуть мати інтерфейси, через які виражаються

залежності. Інтерфейсами можуть бути, наприклад, імена викликуваних підпрограм. **Інтерфейси** зображуються ***окружностями***, з'єднаними з компонентом лінією без напрямку, поруч записується ім'я інтерфейсу.

Залежності зв'язані також із проблемами керування системою. . За допомогою діаграми компонентів персонал супроводу системи може оцінити наслідки будь-яких внесених змін. Якщо компонент залежить від більшого числа інших компонентів, велика імовірність того, що його торкнуться зміни в системі.

Нарешті, залежності дають можливість зрозуміти, які частини системи можна використовувати повторно, а які не можна. . Чим від меншого числа компонентів залежить даний, тим легше його буде використовувати повторно.

Завдання:

Намалюйте діаграму компонентів для діаграми класів з завдання **лабораторної роботи №9.** Зв'яжіть компоненти на діаграмі компонентів із класами діаграми класів і згенеруйте програмний код.

Контрольні питання

1. Яке призначення діаграми класів?
2. Якими способами можна створити діаграму?
3. Які інструменти доступні для діаграми?
4. Які команди надає контекстне меню класу?
5. Як настроїти властивості атрибутів класу?
6. Як настроїти властивості методів класу?
7. Які типи відносин класів ви знаєте?

Лабораторна робота №11. UML-моделювання представлення розгортання (Deployment diagram) та кількісна оцінка розроблених діаграм

Мета роботи

Придбання навиків побудови діаграм розміщення. Навчитися виділяти особливості функціональної поведінки проектованої системи.

Методичні вказівки

Діаграма розміщення (**deployment diagrams**) показує фізичні взаємозв'язки між програмними й апаратними компонентами системи, тобто моделює фізичну архітектуру системи, розміщення об'єктів і компонентів у розподіленій системі. На відміну від діаграм логічного представлення, діаграма розміщення є єдиної для системи в цілому, оскільки повинна цілком відбивати особливості її реалізації. При розробці діаграми розміщення переслідують наступні цілі:

- визначити розподіл компонентів системи по її фізичних вузлах;
- показати фізичні зв'язки між усіма вузлами реалізації системи на етапі її виконання;
- виявити вузькі місця системи і реконфигурировать її топологію для досягнення необхідної продуктивності.

Фізичне представлення програмної системи не може бути повним, якщо отсутствует інформація про те, на якій платформі і на яких обчислювальних засобах вона реалізована. Якщо розробляється програма, що

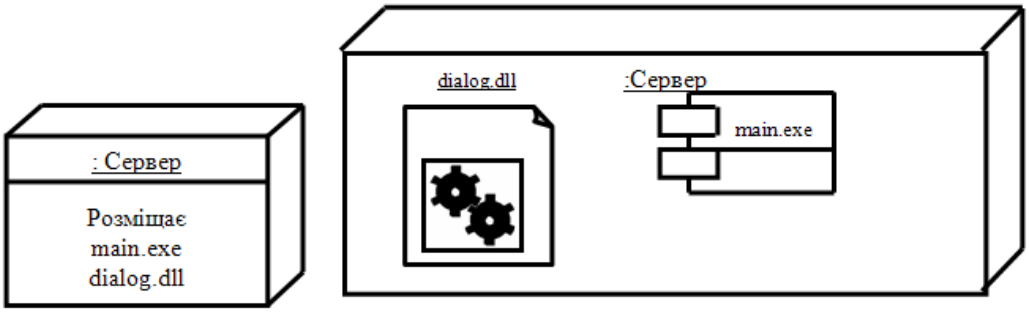
виконується локально на комп'ютері користувача і не використовує периферійних пристроїв і ресурсів, то в розробці додаткових діаграм немає необхідності. При розробці ж корпоративних застосувань наявність таких діаграм може бути вкрай корисною для рішення задач раціонального розміщення компонентів з метою ефективного використання розподілених обчислювальних і комунікаційних ресурсів мережі, забезпечення безпеки.

Основні елементи діаграми розміщення:

- вузол (**node**) - обчислювальний ресурс (процесор чи інший пристрій (дискова пам'ять, контролери різних пристроїв і т.д.). Для вузла можна задати процеси, що виконуються на ньому. Графічно на діаграмі розміщення вузол зображується у формі тривимірного куба. Вузол має власне ім'я, що вказується усередині цього графічного символу. Самі вузли можуть представлятися як типи, так і як екземпляри. У першому випадку ім'я вузла записується без підкреслення і починається з заголовної букви. В другому ім'я вузла-екземпляра записується у виді <ім'я вузла **':'** ім'я типу вузла>. Ім'я типу вузла вказує на деякий різновид вузлів, що є присутнім у моделі системи (рис. 47).

Рис.47. Приклад діаграми розміщення.

Зображення вузлів можуть розширюватися, щоб включити деяку додаткову інформацію про специфікацію вузла. Якщо додаткова інформація відноситься до імені вузла, то вона записується під цим ім'ям у формі позначеного значення. Якщо необхідно явно вказати компоненти, що розміщаються на окремому вузлі, то це можна зробити двома способами. Перший з них дозволяє розділити графічний символ вузла на двох секцій горизонтальною лінією. У верхній секції записують ім'я вузла, а в нижній секції - розміщені на цьому вузлі компонента, другий спосіб дозволяє показувати на діаграмі розміщення вузли з вкладеними зображеннями компонентів (мал.48, а, б). Важливо пам'ятати, що у якості таких вкладених компонентів можуть виступати тільки ті компоненти, що виконуються. Як доповнення до імені вузла можуть використовуватися різні стереотипи, що явно специфицируют призначення цього вузла. Хоча в мові UML стереотипи для вузлів не визначені, у літературі зустрічаються наступні їхні варіанти: "процесор", "датчик", "модем", "мережа", "консоль" і ін.;



(а) (б)

Рис. 48. Варіанти графічного зображення вузлів-екземплярів з розміщеними на них компонентами

- з'єднання (**connection**) - канал взаємодії вузлів (мережа). Крім зображень вузлів на діаграмі розміщення указуються відносини між ними. У якості відносин виступають фізичні з'єднання між вузлами і компонентами, зображення яких теж можуть бути присутнім на діаграмах розміщення.

З'єднання є різновидом асоціації і зображуються відрізками ліній без стрілок. Наявність такої лінії вказує на необхідність організації фізичного каналу для обміну інформацією між відповідними вузлами. Характер з'єднання може бути додатково специфікований приміткою, позначеною значенням чи обмеженням. Так, на представленому нижче фрагменті діаграми розміщення (рис. 49) явно визначені не тільки вимоги до швидкості передачі даних у локальній мережі за допомогою позначеного значення, але і рекомендації з технології фізичної реалізації з'єднань у формі примітки.

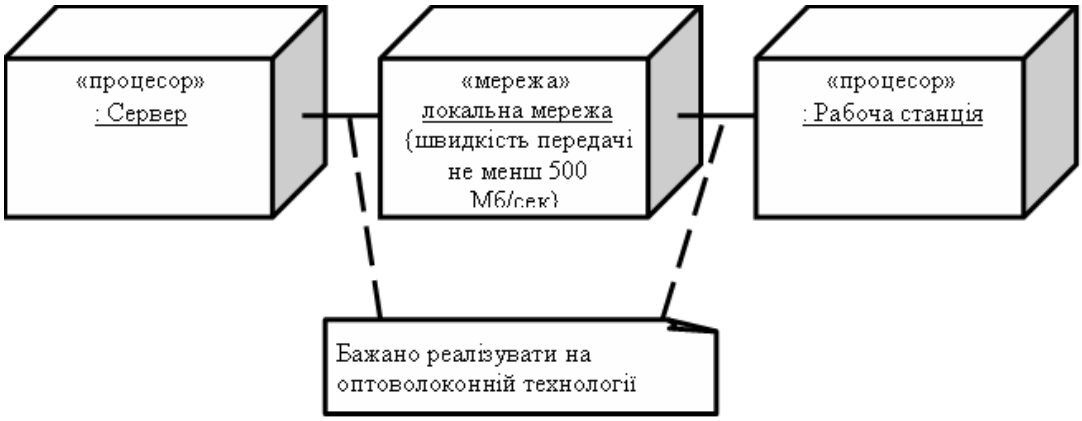


Рис. 49. Фрагмент діаграми розміщення зі з'єднаннями між вузлами.

Крім з'єднань на діаграмі розміщення можуть бути присутнім відносини залежності між вузлом і розміщеними на ньому компонентами. Подібний спосіб є альтернативою вкладеному зображенню компонентів усередині символу вузла, що не завжди зручно, оскільки робить цей символ зайво об'ємним. Тому при великій кількості розміщених на вузлі компонентів відповідну інформацію можна представити у формі відносини залежності (рис. 50).

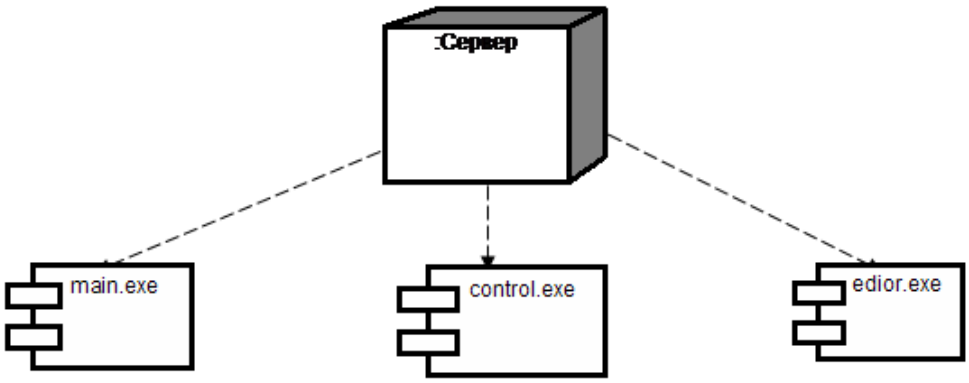


Рис. 50. Діаграма розміщення з відношенням залежності між вузлами.

Розробка діаграми розміщення, як правило, є останнім етапом специфікації моделі програмної системи. Діаграма розміщення використовується користувачами, менеджером проекту, архітектором системи й експлуатаційним персоналом для розуміння фізичного розміщення системи і розташування її окремих підсистем.

Кількісна оцінка розроблених діаграм

Кількісна оцінка і порівняння діаграм UML будується на присвоєнні елементам діаграм оцінок, що залежать від їхньої інформаційної цінності, а також від внесеної ними в діаграму додаткової складності. Цінність окремих елементів міняється в залежності від типу діаграми, на якій вони знаходяться.

Кількісну оцінку діаграми можна провести по наступній формулі:

$$S = \frac{\sum S_{Obj} + \sum S_{Lnk}}{1 + Obj + \sqrt{T_{Obj} + T_{Lnk}}}$$

S - оцінка діаграми

S_{Obj} - оцінки для елементів діаграми

S_{Lnk} - оцінки для зв'язків на діаграмі

Obj - число об'єктів на діаграмі

T_{Obj} - число типів об'єктів на діаграмі

T_{Lnk} - число типів зв'язків на діаграмі

Якщо діаграма містить велику кількість зв'язків одного типу, то число і тип зв'язків можна не враховувати і формула розрахунку приводиться до виду:

$$S = \frac{\sum S_{Obj}}{1 + Obj + \sqrt{T_{Obj}}}$$

Якщо на діаграмі показані атрибути й операції класів, можна врахувати їх при розрахунку, при цьому оцінка додається до оцінки відповідного класу:

$$S_{cls} = \frac{\sqrt{Op} + \sqrt{Art}}{0{,}3 * (Op + Art)}$$

S_{cls} - оцінка операцій і атрибутів для класу

Op - число операцій класу

Art - число атрибутів класу

Приведемо оцінки для різних типів елементів і зв'язків:

Основні елементи мови UML:

Тип елемента	**Оцінка для елементів**
Клас	5
Інтерфейс	4
Прецедент	2
Компонент	4
Вузол	3
Процесор	2
Взаємодія	6
Пакет	4
Стан	4
Примітка	2

Основні типи зв'язків мови UML:

Тип зв'язку	Оцінка для зв'язку
Залежність	2
Асоціації	1
Агрегирование	2
Композиція	3
Узагальнення	3
Реалізація	2

Інші типи зв'язків повинні розглядатися як асоціації.

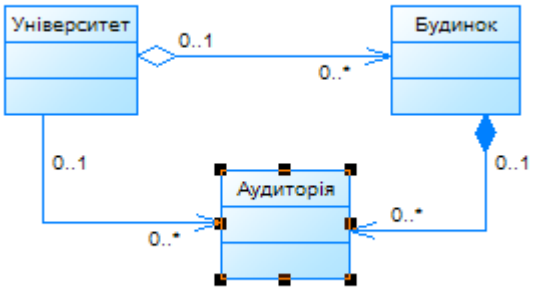
Недоліком діаграми є як занадто низька оцінка (при цьому діаграма недостатньо інформативна), так і занадто висока оцінка (при цьому діаграма звичайно занадто складна для розуміння).

Діапазони оцінок для діаграм UML:

Тип діаграми	**Діапазон оцінок**
Класів - з атрибутами й операціями	5 – 5,5
Класів - без атрибутів і операцій	3 – 3,5
Компонентів	3,5 – 4
Варіантів використання	2,5 – 3

Розгортання	2 – 2,5
Послідовності	3 – 3,5
Кооперативна	3,5 – 4
Пакетів	3,5 – 4
Станів	2,5 – 3

Приклад оцінки простої діаграми класів за даною методикою:



Діаграма містить три класи без операцій і атрибутів; отже T_{Obj} =1, $\sum S_{Obj}$ =15 и Obj =3. Як зв'язки використовується асоціація, агрегирование й узагальнення; отже $\sum S_{Lnk}$ =6 и T_{Lnk} =3.

$$S = \frac{\sum S_{Obj} + \sum S_{Lnk}}{1 + Obj + \sqrt{T_{Obj} + T_{Lnk}}} = \frac{15+6}{1+2+3} = 3{,}5$$

Тобто чисельна оцінка для даної діаграми дорівнює 3,5.

Завдання:

Намалювати діаграму розгортання

1) підрозділів КНУБА (ректорат, факультети, кафедри, канцелярія, бібліотека);

2) архітектури клієнт-сервер (декілька веб-клиентов, сервер бази даних, веб-сервер);

3) Провести кількісну оцінку діаграм