

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ БОГДАНА ХМЕЛЬНИЦЬКОГО**

ГАЛИНА ЛУЦЕНКО

**UML -МОДЕЛЮВАННЯ
У СЕРЕДОВИЩІ VISUAL PARADIGM**

ЛАБОРАТОРНИЙ ПРАКТИКУМ

**ЧЕРКАСИ
2019**

УДК 378.1

Л86

Рекомендовано до друку кафедрою автоматизації та комп'ютерно-інтегрованих технологій
Черкаського національного університету імені Богдана Хмельницького.
Протокол №4 від 28 листопада 2019 року

Л86 Луценко Г.В. UML-моделювання у середовищі Visual Paradigm: лабораторний практикум для студентів закладів вищої освіти / автор та укладач Г.В. Луценко. – Черкаси: видавець Чабаненко Ю.А., 2019. – 60 с.

Рецензенти:

Гриценко Валерій Григорович

доктор педагогічних наук, доцент кафедри автоматизації та комп'ютерно-інтегрованих технологій Черкаського національного університету імені Богдана Хмельницького

Луценко Григорій Васильович

доктор педагогічних наук, завідувач кафедрою педагогіки та менеджменту освіти Глухівського національного педагогічного університету імені Олександра Довженка

У лабораторному практикумі розглянуто теоретичні та практичні аспекти створення основних діаграм уніфікованої мови моделювання UML у середовищі Visual Paradigm. Матеріали посібника можуть бути використані в освітньому процесі при підготовці студентів інженерних і природничо-математичних спеціальностей. Навчально-методичний посібник адресований науковцям, викладачам, докторантам, аспірантам, студентам закладів вищої освіти.

© Г.В. Луценко, 2019.

ЗМІСТ

Лабораторна робота 1 Створення діаграми прецедентів	4
Контрольні запитання до лабораторної роботи 1	23
Лабораторна робота 2 Створення діаграми класів	24
Контрольні запитання до лабораторної роботи 2	33
Лабораторна робота 3 Створення діаграм послідовності та комунікації.....	34
Контрольні запитання до лабораторної роботи 3	42
Лабораторна робота 4 Створення діаграми діяльності.....	43
Контрольні запитання до лабораторної роботи 4	51
Лабораторна робота 5 Створення діаграми станів	52
Контрольні запитання до лабораторної роботи 5	58
Список використаних джерел.....	59

Лабораторна робота 1

Створення діаграми прецедентів

Завдання:

1. Створити головну діаграму прецедентів, задавши на ній варіанти використання акторів.
2. Додати відношення між акторами та варіантами використання.
3. Створити додаткову діаграму прецедентів.
4. Додати опис до акторів та варіантів використання.
5. Для кожного варіанту використання задати потік подій у вигляді окремого файлу та прикріпити його до варіанту використання.

Теоретичні відомості

Візуальним моделюванням називається спосіб опису ідей та проблем реального світу за допомогою моделей. У загальному розумінні, модель – це абстракція, опис сутності складної проблеми чи структури, при створенні якої нехтують несуттєвими деталями. При побудові моделі використовують нотацію – набір загальноприйнятих позначень. Як правило, мова йде про складні системи, для яких виконується морфологічний і функціональний опис, що приводить до потреби поділу складної системи на декілька абстрактних нотацій.

Важливим етапом розвитку нотацій стала поява в 1995 році першої загальнодоступної версії UML (Unified Modelling Language) – уніфікованої мови моделювання для візуального моделювання систем з використанням об'єктно-орієнтованої парадигми. UML – це стандартизована мова моделювання, що складається з інтегрованого набору діаграм, створених для допомоги розробниками систем та програмного забезпечення. У 1997 році організація Object Management Group визнала UML стандартною мовою моделювання. Фактично з того ж часу вивчення UML стає складником освітніх програм підготовки ІТ фахівців і майбутніх інженерів.

У сучасних працях, присвячених методології створення програмних систем різних рівнів складності, визнається, що створення системи не є питанням вибору методології проектування, програмування чи технічної реалізації. Ключовим завданням на першому етапі створення системи є пошук консолідованого бачення майбутнього результату, визначення якого відбувається шляхом взаємодії замовників і розробників. Сформоване узгоджене бачення може мати формальне або напівформальне представлення, але має включати аналіз бізнес-вимог, доступних ресурсів, ідеї та побажання замовників. На наступному кроці (на етапі розробки специфікації вимог) відбувається деталізація й формалізація опису. Використання графічних нотацій дозволяє орієнтуватися в структурі системи як кваліфікованим розробникам, так і замовникам, що не мають відповідної кваліфікації.

Моделювання інформаційної системи можна представити як спуск за рівнями: від концептуальної моделі до логічної, а потім до фізичної моделі створюваної системи. Важливим елементом концептуальної моделі є діаграма варіантів використання. Діаграми варіантів використання саме і є основою для досягнення порозуміння між програмістами, що розробляють проект системи, і замовниками проекту.

Діаграма варіантів використання або діаграма прецедентів (Use Case Diagram), що належить до класу діаграм поведінки UML, є основною формою, що використовується для опису вимог до створюваної інформаційної системи. Варіанти

Діаграми прецедентів зазвичай мають дуже просту структуру, на них не відображаються деталі системи. Такі діаграми:

- 1) узагальнюють деякі взаємозв'язки між прецедентами, акторами та системами;
- 2) не показують порядок виконання кроків для досягнення цілей кожного з прецедентів.

Діаграми варіантів використання розробляють, як правило, на початкових етапах створення системи. Серед цілей їх створення є: визначення контексту системи, окреслення вимог до системи, перевірка архітектури системи, упровадження та розробка тестових випадків, організація спільної роботи аналітиків та експертів з певної предметної області.

Приклад типової діаграми варіантів використання наведено на рис. 1, а в таблиці 1 описано сутність складових такої діаграми.

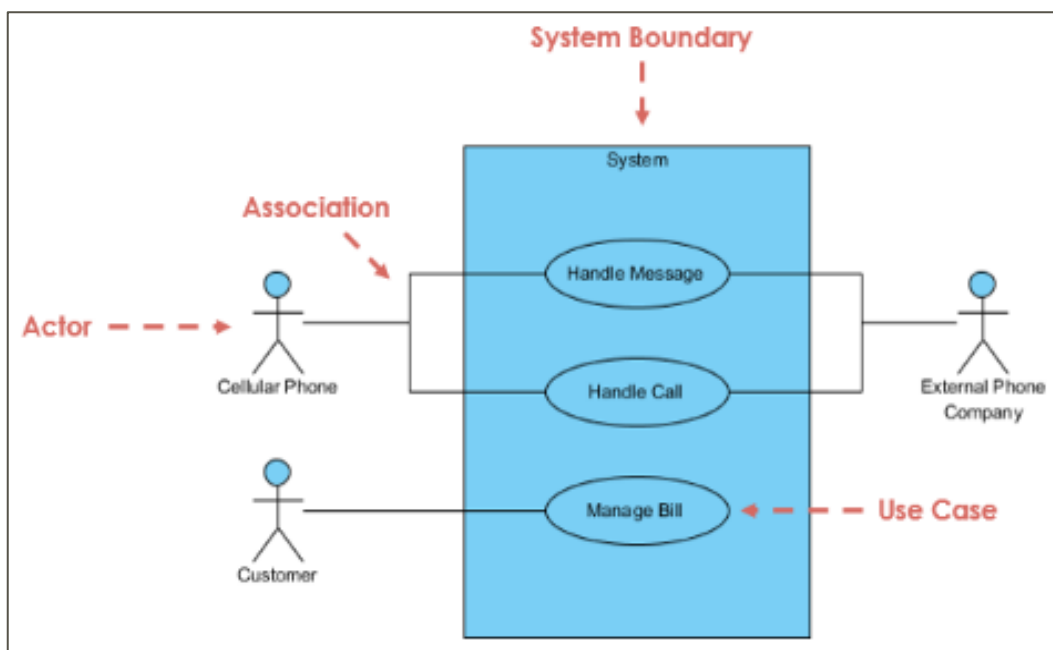
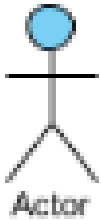


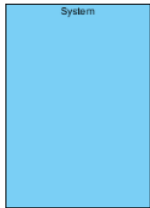


Рис. 1. Діаграма варіантів використання та її основні складові

Таблиця 1. Основні елементи діаграми прецедентів

Опис позначення	Візуальна репрезентація
<p>Актор Об'єкт, що взаємодіє з прецедентом (функціями системи). Описується іменником. Актор грає певну роль для функціонування системи. Акторі подібний до користувача, але користувач може грати різні ролі. Наприклад: професор може бути викладачем, а також дослідником. Іншим прикладом є об'єкт, що виконує різні ролі в двох системах. Актор є тригером для прецедентів (запускає їх). Актор виконує дії по відношенню до системи (на вхід) та актор має очікування від системи (на виході).</p>	 <p>Actor</p>

Продовження таблиці 1. Основні елементи діаграми прецедентів

Опис позначення	Візуальна репрезентація
<p>Варіант використання (прецедент) Функція системи (процес є автоматизованим або ручним). У назві використовується дієслово + іменник (або іменниковий зворот), наприклад, «робити щось». Кожен актор повинен бути пов'язаним із прецедентом, але деякі прецеденти можуть не бути пов'язані з акторами.</p>	
<p>Комунікаційний зв'язок (link) Участь актора у прецедентів відображається підключенням актора до прецеденту суцільною лінією. Актори можуть бути підключені до прецедентів з використанням відношення асоціації, що вказує на те, що актор та прецедент спілкуються між собою за допомогою повідомлень.</p>	
<p>Границі системи Границею системи потенційно є вся система, визначена в документації. Для великих та складних систем кожен модуль може бути границею системи. Складна система може охоплювати всі модулі (наприклад, персонал, оплата праці, облік тощо).</p>	

Для створення нового проекту потрібно обрати пункт New на панелі меню. Після натискання відкриється вікно діалогу (рис. 2). У ньому слід вказати назву нового проекту та тип діаграм, які використовуватимуться (UML).

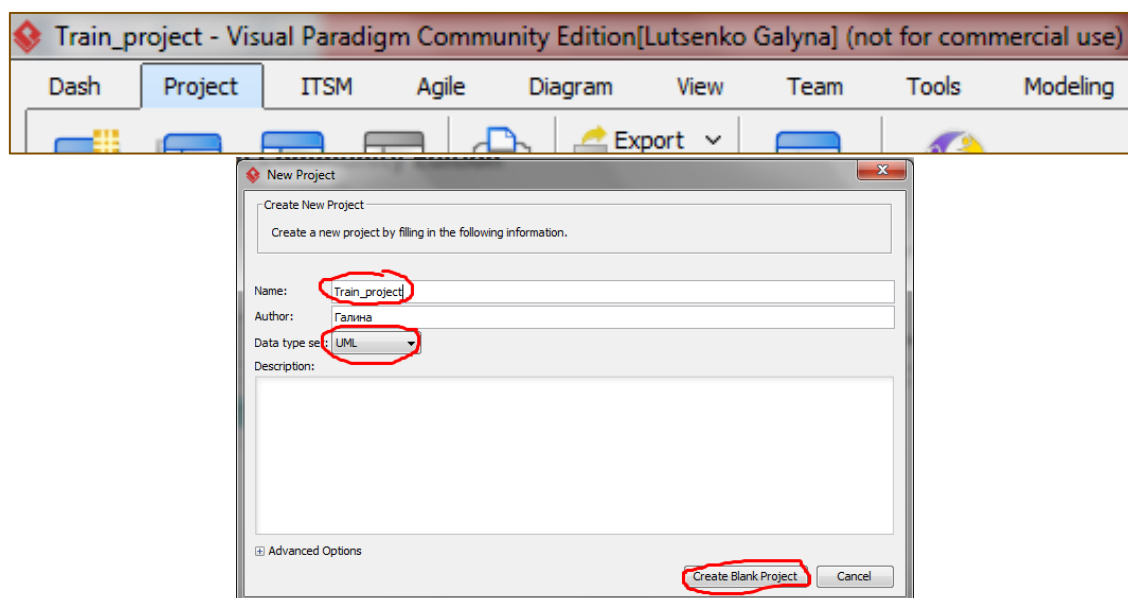


Рис. 2. Вікно діалогу для створення нового проекту

Створення діаграм UML для проекту та навігація між ними здійснюються за допомогою панелі Diagram Navigator (рис. 3). Якщо ця панель не відобразилася відразу для нового проекту, перейдіть до пункту меню View > Panel > Diagram Navigator (рис. 4).

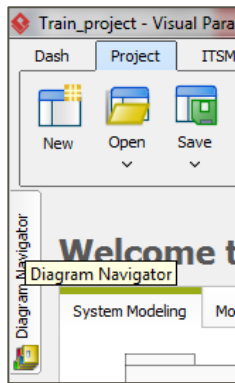


Рис. 3. Звернення до Diagram Navigator

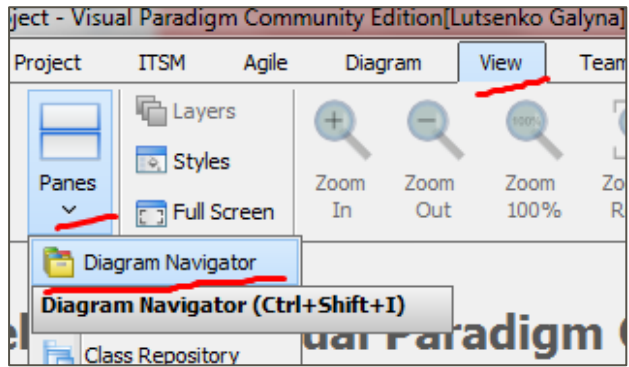


Рис. 4. Увімкнення панелі Diagram Navigator

Прийоми роботи для створення прецедентів та акторів

Створення діаграм проекту розпочинається зі створення діаграми прецедентів (інша назва – діаграма варіантів використання). Для її створення необхідно перейти до Diagram Navigator та обрати New Use Case Diagram. Відповідна команда та результат її виконання наведені на рис. 5 і 6, відповідно.

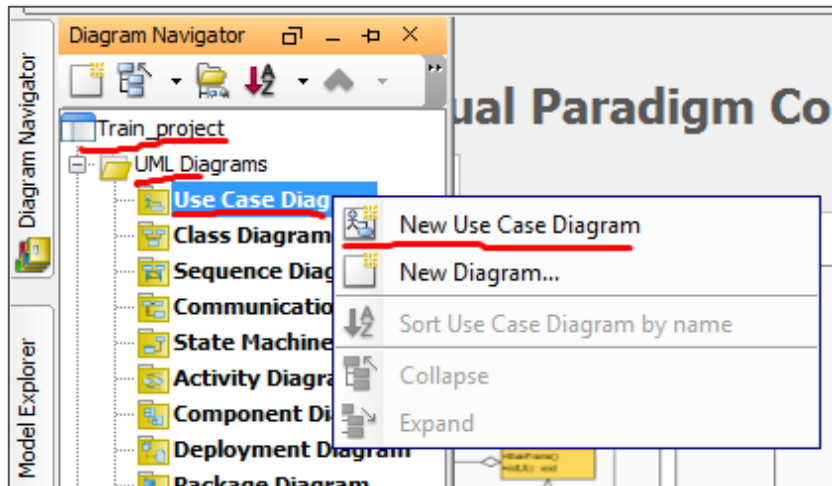


Рис. 5. Створення нової діаграми прецедентів

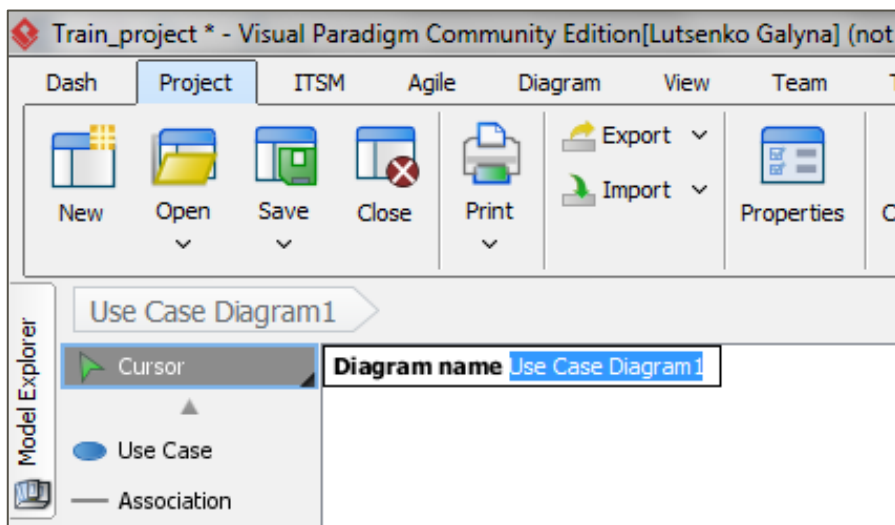


Рис. 6. Вікно введення назви створеної діаграми

Ключовими елементами діаграми прецедентів є власне прецеденти та актори, що виконують ці прецеденти. Для створення нового прецеденту потрібно вибрати елемент Use Case з інструментів діаграми й перетягнути на діаграму (рис. 7).

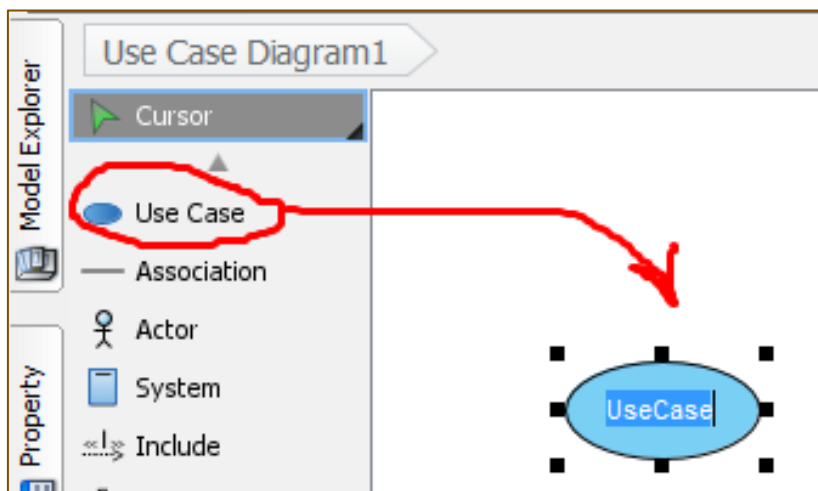


Рис. 7. Створення прецеденту

За замовчуванням прецедент має назву Use Case. Ви можете змінити її відразу. В іншому випадку, для того щоб ввести назву прецеденту потрібно двічі клацнути по діаграмі лівою кнопкою миші (ЛКМ) та натиснути Enter для підтвердження (рис. 8).



Рис. 8. Введення назви прецеденту

Розглянемо, що означають додаткові значки біля створеного прецеденту. У лівому нижньому куті розміщено команду, що дозволяє додавати посилання на матеріали різних типів (файли, папки, URL-посилання тощо) (рис. 9).

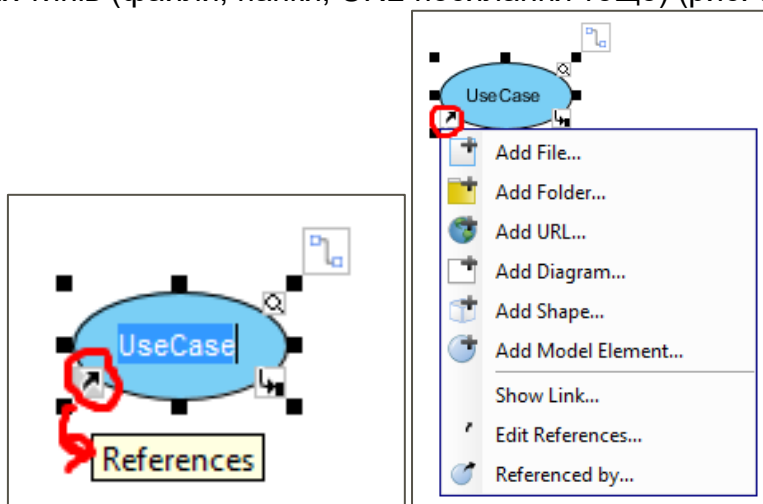


Рис. 9. Додавання супровідних файлів

У правому нижньому куті розміщено команду для додавання допоміжних діаграм (під-діаграм) (рис. 10).

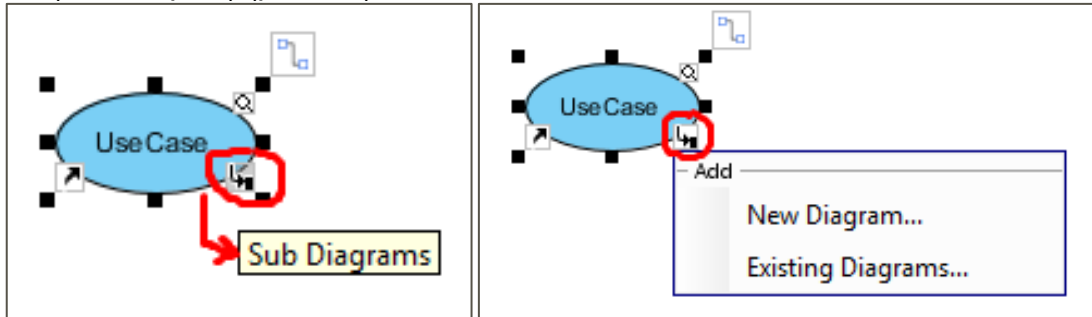


Рис. 10. Додавання під-діаграм

У правому верхньому куті розміщено команду, за допомогою якої відкривається вікно введення детального опису прецедентів (специфікації прецеденту) (рис. 11).

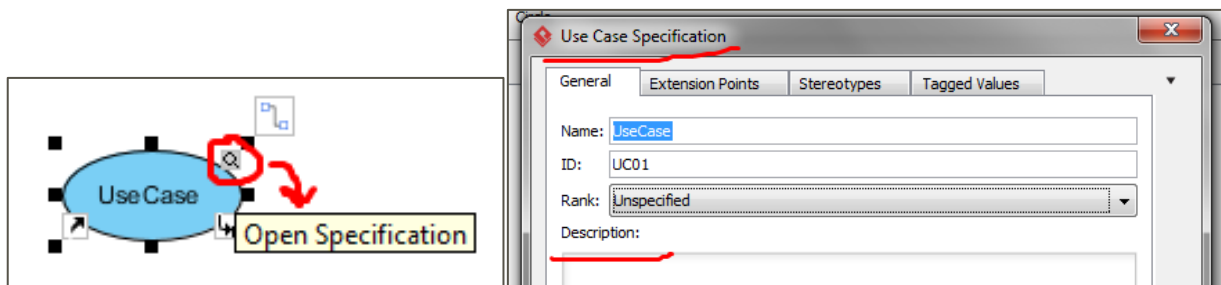


Рис. 11. Відкриття вікна специфікації прецеденту

Вище від цієї команди знаходиться кнопка для швидкого відкриття каталогу ресурсів.

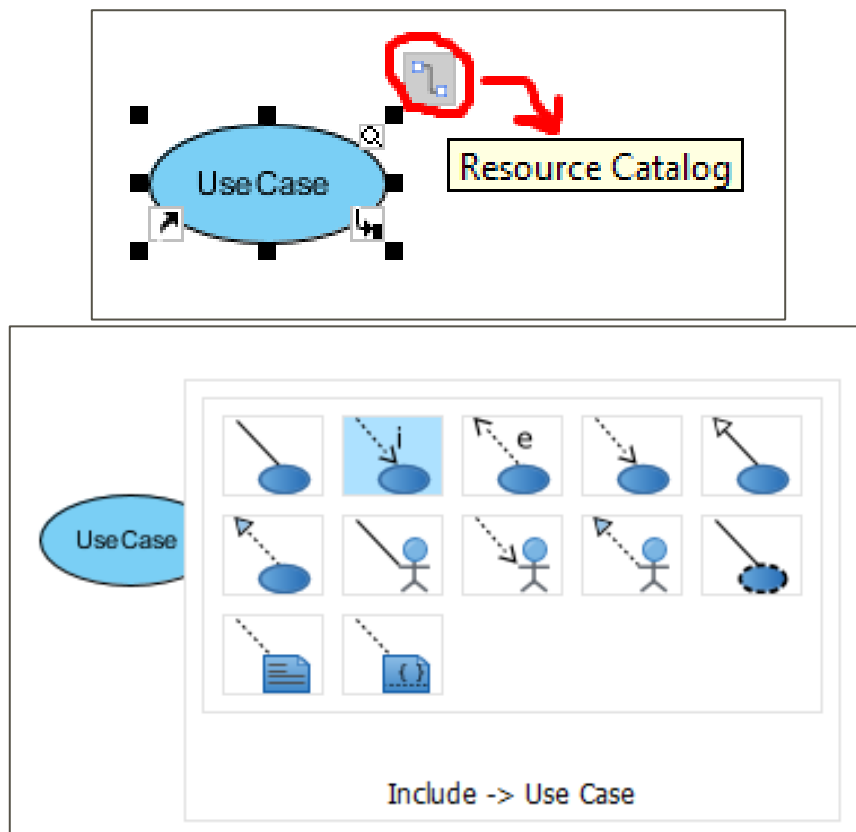


Рис. 12. Відкриття каталогу ресурсів (основних з'єднань)

13). Створення актора відбувається аналогічно до створення прецеденту (рис. 13).

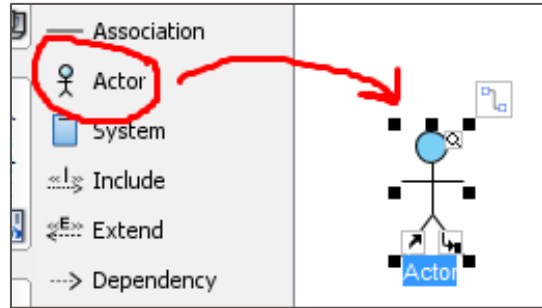


Рис. 13. Створення актора в Visual Paradigm

Натиснувши праву кнопку миші й обравши пункт open Specification можна перейти до вікна діалогу в якому задається детальна інформація про актора (рис. 14). Аналогічно відкривається вікно специфікації для прецеденту (рис. 15).

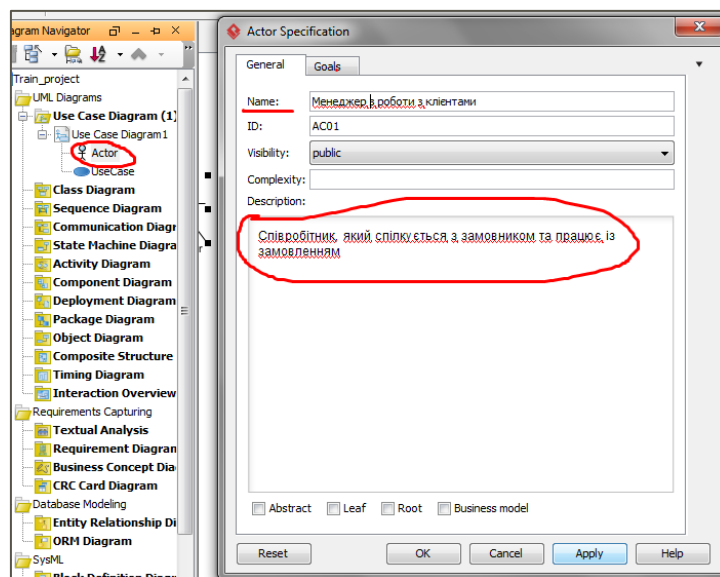


Рис. 14. Вікно специфікації актора

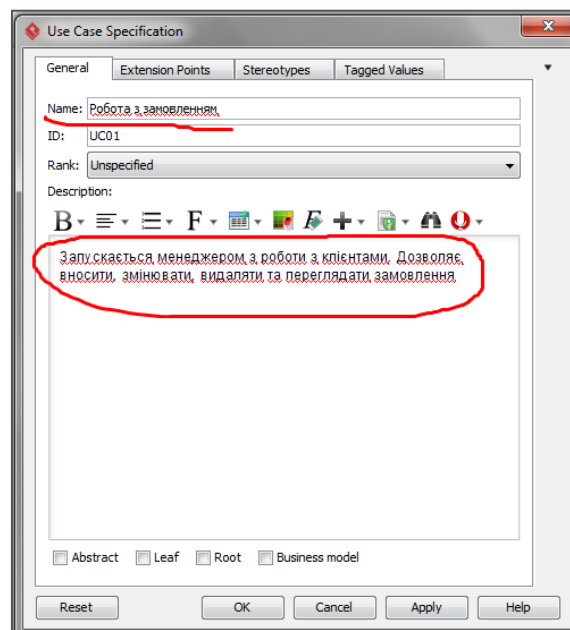


Рис. 15. Вікно специфікації прецеденту

Два елементи діаграми прецедентів можуть бути пов'язані за допомогою значків, що оточують кожний елемент. Наприклад, якщо необхідно побудувати відношення асоціації між актором та прецедентом, слід навести мишку на актора, відкрити каталог ресурсів й у ньому обрати потрібне відношення.

На рис. 16а обрано відношення асоціації, на яке потрібно натиснути ЛКМ. У вікні, що з'явиться (рис. 16б) стерти рядок <New Use Case>. Тоді, система запропонує вибір із доступних діаграм прецедентів (рис. 16в). У ньому слід обрати потрібну діаграму (рис. 16г).

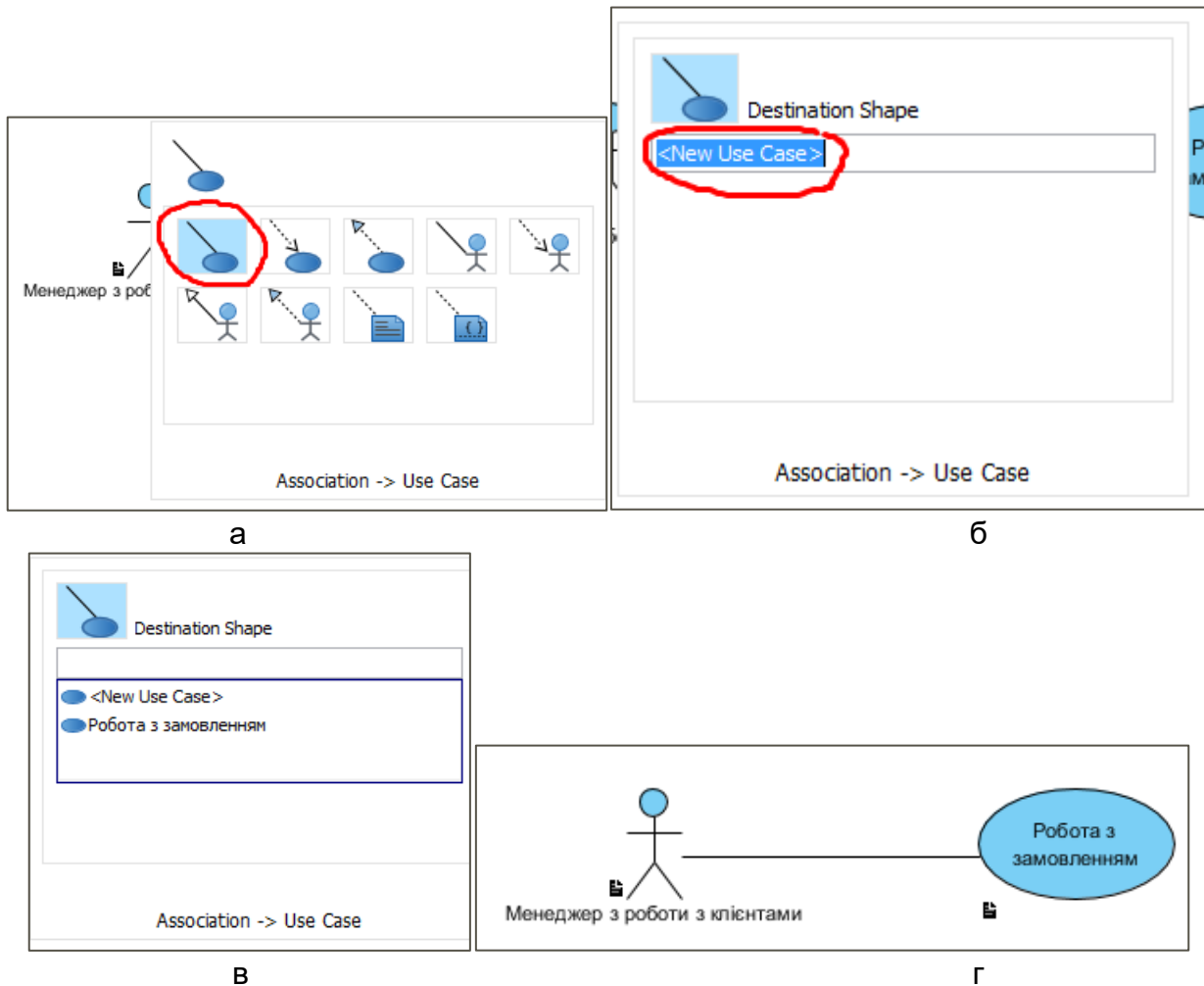


Рис. 16. Асоціація між актором та прецедентом

Для полегшення роботи з прецедентами доцільно використовувати елемент System, який знаходиться в Diagram Navigator (рис. 17). Надалі, всіх акторів потрібно розташовувати за межами елемента System, а всі прецеденти – на полі системи.

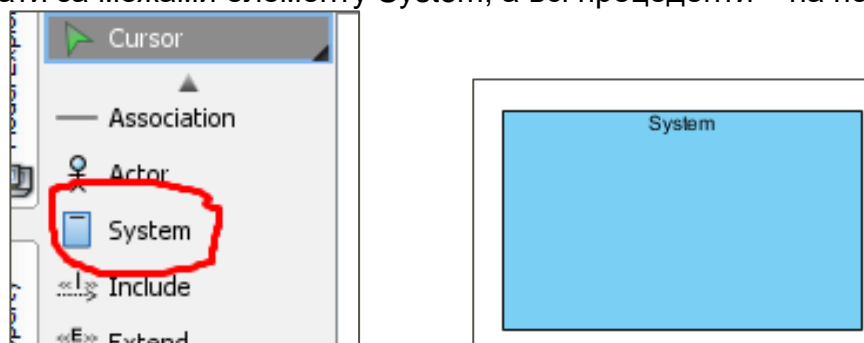


Рис. 17. Елемент System

Відношення включення.

Застосовується, коли один із прецедентів використовує інший. Позначається як відношення залежності, що спрямовано від базового прецеденту до того, який використовується, із вказівкою стереотипу - include.

Відношення розширення.

Застосовується для відображення:

- додаткових режимів;
- режимів, які запускаються тільки за певних умов;
- альтернативних потоків, які запускаються на вибір користувача.

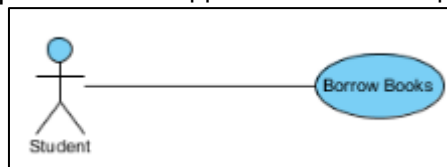
Позначається як відношення залежності, що спрямовано від додаткового прецеденту до базового, із вказівкою стереотипу - extend.

Відношення узагальнення.

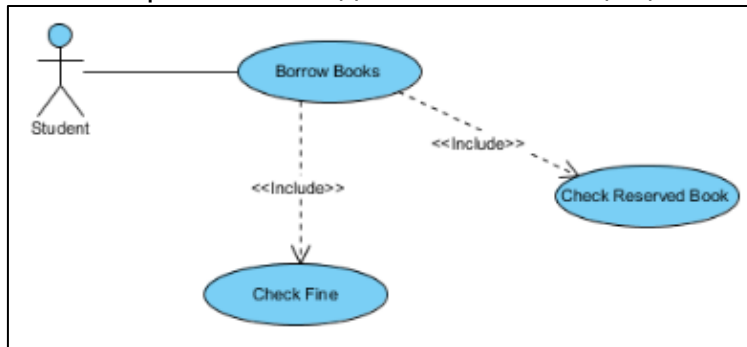
Відношення узагальнення служить для вказівки того факту, що деякий варіант використання А може бути узагальнений до варіанта використання В. У цьому випадку варіант А буде спеціалізацією варіанта В. Фактично цим відношенням показується спадкування. При цьому В називається предком або батьком по відношенню А, а варіант А - нащадком стосовно варіанта використання В. Нащадок успадковує всі властивості й поведінку свого батька, а також може бути доповнений новими властивостями й особливостями поведінки.

Приклади діаграм варіантів використання з різними типами відношень

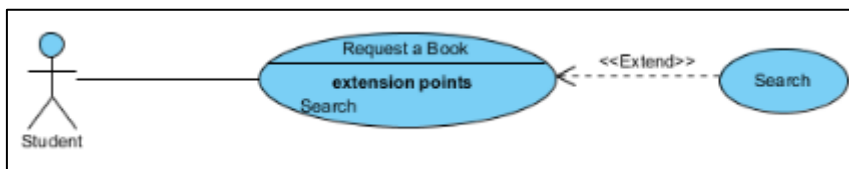
1. Діаграма варіантів використання з відношенням асоціації.



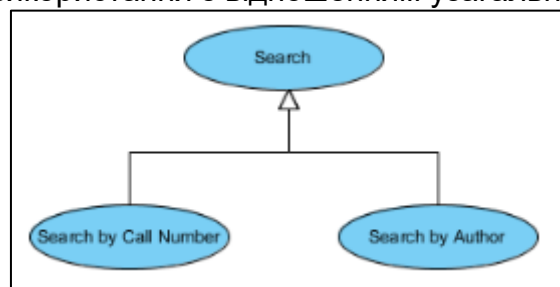
2. Діаграма варіантів використання з відношеннями асоціації та включення.



3. Діаграма варіантів використання з відношеннями асоціації, включення і розширення.



4. Діаграма варіантів використання з відношенням узагальнення.



Хід виконання лабораторної роботи

Проектується „Підприємство зі збору та продажі комп'ютерів”. Для нашої предметної області вводимо акторів, наведених у Таблиці 2.

Таблиця 2. Опис акторів системи

Актор	Стислий опис
Менеджер з роботи з клієнтами	Співробітник, який спілкується з замовником та працює з замовленням
Менеджер з постачання	Співробітник, який займається закупкою необхідних комплектуючих
Інженер зі збору настільних комп'ютерів	Співробітник, який займається збором і налаштуванням настільних комп'ютерів
Інженер зі збору ноутбуків	Співробітник, який займається налаштуванням ноутбуків
Інженер з тестування	Співробітник, який займається тестуванням зібраних комп'ютерів
Завскладом	Співробітник, що керує складом комплектуючих

Опишемо, які можливості повинна надавати система:

- ❖ Актор Менеджер з роботи з клієнтами використовує систему для оформлення, редагування замовлень та управління інформацією про клієнтів підприємства;
- ❖ Актор Менеджер з постачання використовує систему для перегляду переліку необхідних для закупки комплектуючих та укладання інформації про постачання;
- ❖ Актор Інженер зі збору настільних комп'ютерів використовує систему для перегляду нарядів на збір ПК, для замовлення комплектуючих зі складу та відмітки про хід виконання роботи;
- ❖ Актор Інженер зі збору ноутбуків використовує систему для перегляду нарядів на збір ноутбуків, для замовлення комплектуючих зі складу та відмітки про хід виконання роботи;
- ❖ Актор Інженер з тестування використовує систему для перегляду нарядів на тестування зібраної продукції та відмітки про хід виконання роботи;
- ❖ Актор Завскладу використовує систему для обліку надходжень та видачі комплектуючих.

На основі описаних можливостей визначаємо прецеденти, наведені нижче в Таблиці 3.

Таблиця 2. Опис прецедентів системи

Прецедент	Стислий опис
Робота з замовленням	Запускається менеджером з роботи з клієнтами. Дозволяє вносити, змінювати, видаляти та переглядати замовлення
Управління інформацією про клієнта	Запускається менеджером з роботи з клієнтами. Дозволяє додавати, змінювати чи видаляти клієнтів, а також переглядати інформацію про клієнтів.
Управління інформацією про поставників	Запускається менеджером з постачання. Дозволяє додавати, змінювати чи видаляти постачальників, а також переглядати інформацію про постачальників.

Управління інформацією про комплектуючі	Запускається менеджером з постачання. Дозволяє переглядати інформацію про комплектуючі, здійснювати аналіз їх використання, прогнозувати необхідну їх кількість та робити замовлення.
Збір комп'ютерів	Запускається інженером зі збору . Дозволяє переглядати наряди на збір комп'ютерів та робити відмітки про хід виконання роботи.
Замовлення комплектуючих	Запускається інженером зі збору. Призначається для замовлення потрібних комплектуючих зі складу.
Тестування комп'ютерів	Запускається інженером з тестування. Дозволяє переглянути список комп'ютерів, які потрібно тестувати та робити відмітки про виконану роботу.
Облік надходження та видачі комплектуючих	Запускається завскладом. Дозволяє вести облік надходження та видачі запчастин та комплектуючих.

Розглянемо відношення між акторами та прецедентами (рис. 18). На мові UML можливий лише один тип відношення між актором та прецедентом – відношення комунікації. Тому всіх акторів ми зв'язуємо з прецедентами відношенням *Unidirectional Association*. Оскільки інший тип відношень тут задати не можна, то стереотип *communicate* можна не вказувати.

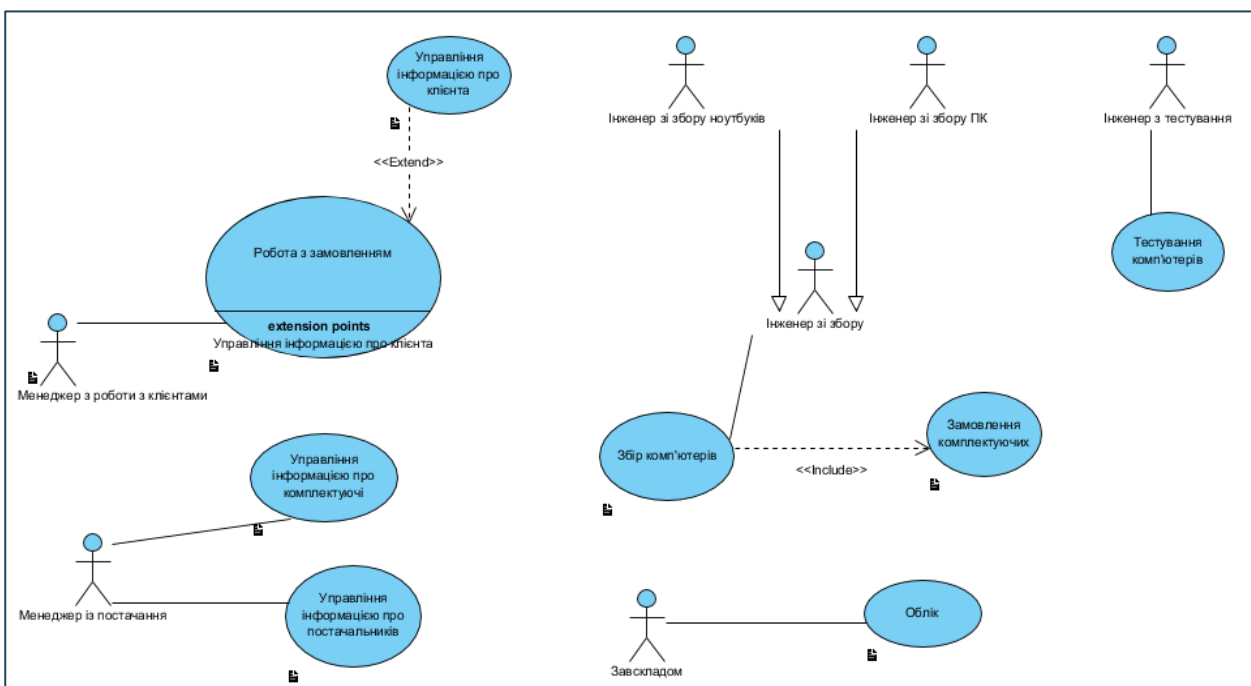


Рис. 18. Основна діаграма Варіантів Використання

Для прецеденту **Збір комп'ютерів** не має значення, який саме актор буде з ним взаємодіяти – **Інженер зі збору настільних комп'ютерів** чи **Інженера зі збору ноутбуків**. Тому ми вводимо ще одного актора – **Інженер зі збору**, з яким ми зв'язали перших двох акторів відношенням узагальнення *Generalization*.

Відношення між прецедентами **Робота з замовленням** та **Управління інформацією про клієнта** – відношення розширення, оскільки коли актор **Менеджер з роботи з клієнтами** працює з замовленням (оформляє, змінює і т.д.), то не завжди при цьому він управляє інформацією про клієнтів.

Відношення між прецедентами Збір комп'ютерів та Замовлення комплектуючих – відношення включання, оскільки для збору комп'ютерів обов'язково потрібно замовляти необхідні комплектуючі зі складу.

Із використанням елементу System діаграма прецедентів набуває вигляду наведеного на рис. 19.

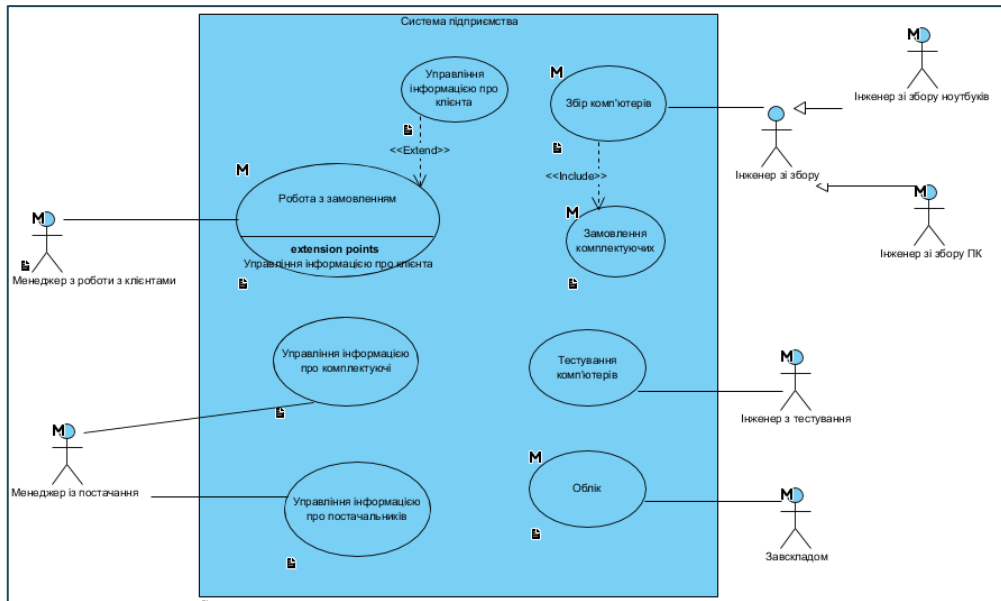


Рис. 19. Система підприємства

Створення додаткової діаграми прецедентів

Як можна бачити з опису потоку подій для всіх прецедентів кожен з них включає перевірку користувача. Перевірка здійснюється однотипно для будь-якого прецеденту. Тому її можна представити у вигляді окремого прецеденту Аутентифікація користувача, пов'язаного відношенням включання з усіма іншими. Результат створення діаграми приведено на рис. 20.

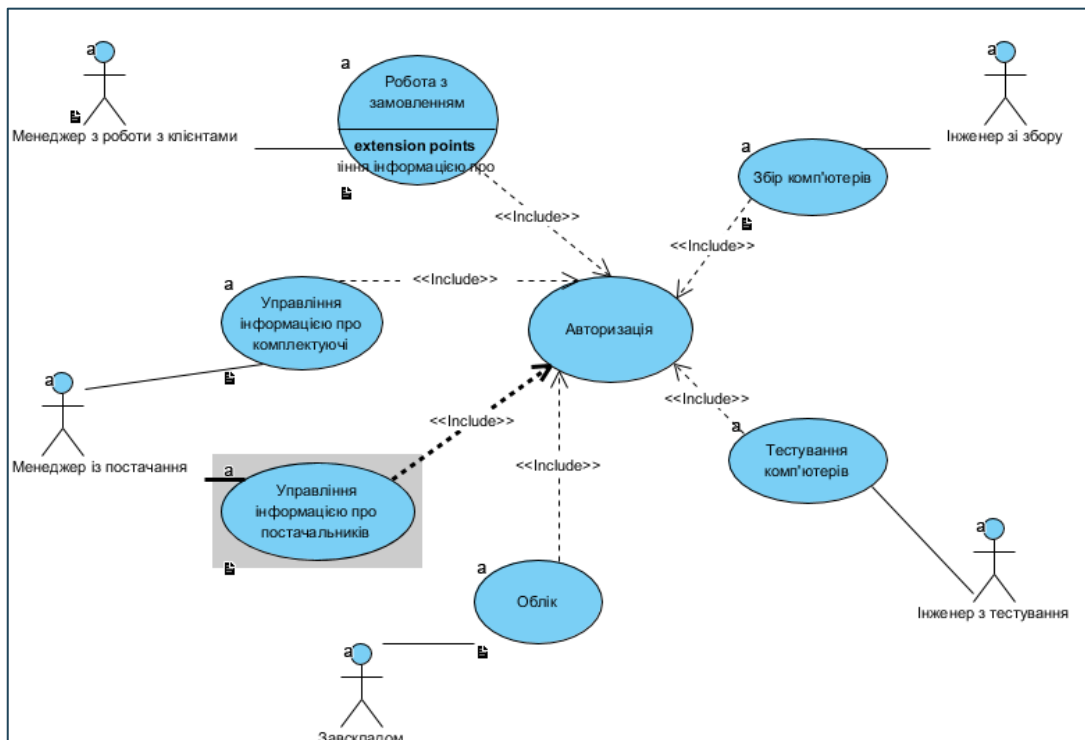


Рис. 20. Додаткова діаграма Варіантів Використання

Потік подій для прецедентів головної діаграми прецедентів

Потік подій для прецедентів будемо описувати за наступним шаблоном:

- X.1 передумови;
- X.2 головний потік;
- X.3 підпотоки;
- X.4 альтернативні потоки;
- X.5 постумови.

где X – число від 1 до кількості прецедентів

Потік подій для прецеденту „Робота з замовленням”

1.1. Передумови

Якщо замовлення оформляється для нового клієнта, то під-потік *Додати нового клієнта (Add a New Client)* прецеденту **Управління інформацією про клієнта** повинен виконатися перед його початком.

1.2. Головний потік

Прецедент починає виконуватися, коли менеджер підключається до системи і вводить своє ім'я та пароль. Система перевіряє правильність пароля (E-1) та виводить можливі варіанти дій: додати (Add), змінити (Change), видалити (Delete), переглянути (View), вийти (Exit).

Якщо вибрана операція додати (Add), S-1: виконується під-потік додати нове замовлення (Add a New Order).

Якщо вибрана операція змінити (Change), S-2: виконується під-потік змінити замовлення (Change Order).

Якщо вибрана операція видалити (Delete), S-3: виконується під-потік видалити замовлення (Delete Order).

Якщо вибрана операція переглянути (View), S-4: виконується під-потік переглянути замовлення (View Order).

Якщо вибрана операція вийти (Exit), прецедент завершується.

1.3. Під-потоки

S-1: Додати нове замовлення

Система відображає вікно діалогу, яке містить поле, у якому менеджер повинен вибрати тип комп'ютера (настільний чи ноутбук). Користувач вибирає необхідний тип. Система відображає поле для вибору клієнта і список можливих комплектуючих для вибраного типу комп'ютера, у якому менеджер відмічає комплектуючі, вибрані клієнтом. Менеджер заповнює поля (E-2). Система запам'ятовує введені дані та роздруковує рахунок для оплати. Потім прецедент розпочинається спочатку.

S-2: Змінити замовлення

Система відображає вікно діалогу, яке містить список замовлень та поле для вводу номеру замовлення. Менеджер вибирає потрібне замовлення зі списку чи вводить номер замовлення у поле (E-3). Система відображає інформацію про дане замовлення. Менеджер вносить потрібні зміни (E-2). Система запам'ятовує введені дані. Потім прецедент починається спочатку.

S-3: Видалити замовлення

Система відображає вікно діалогу, яке містить список замовлень і поле для вводу номеру замовлення. Менеджер вибирає потрібне замовлення зі списку чи вводить номер замовлення у поле (E-3). Система видаляє вибране замовлення (E-4). Потім прецедент починається спочатку.

S-4: переглянути замовлення

Система відображає вікно діалогу, яке містить список замовлень і поле для вводу номера замовлення. Менеджер вибирає потрібне замовлення із списку чи вводить номер замовлення у поле (E-3). Система відображає інформацію про вибране замовлення. Коли менеджер переглядає інформацію, прецедент розпочинається спочатку.

1.4. Альтернативні потоки

E-1: введено неправильне ім'я чи пароль. Користувач повинен повторити введення чи завершити прецедент.

E-2: вибрані не всі комплектуючі, необхідні для збору комп'ютера чи комплектуючих немає у наявності. Менеджер повинен змінити склад комп'ютера чи завершити прецедент.

E-3: введено неправильний номер замовлення. Менеджер повинен повторити введення чи завершити прецедент.

E-4: система не може видалити замовлення. Інформація зберігається, система видалить замовлення пізніше. Виконання прецеденту продовжується.

Потік подій для прецеденту „Управління інформацією по клієнта”

2.1. Передумови

2.2. Головний потік

Прецедент починає виконуватися, коли менеджер підключається до системи і вводить своє ім'я і пароль. Система перевіряє правильність пароля (E-1) і виводить можливі варіанти дій: додати, змінити, видалити, переглянути чи вийти.

Якщо вибрана операція додати (Add), S-1: виконується потік додати нового клієнта (Add a New Client).

Якщо вибрана операція змінити (Change), S-2: виконується потік змінити дані про клієнта (Change Client Data).

Якщо вибрана операція видалити (Delete), S-3: виконується потік видалити клієнта (Delete Client).

Якщо вибрана операція переглянути (View), S-4: виконується потік переглянути дані про клієнта (View Client Data).

Якщо вибрана операція вийти (Exit), прецедент завершується.

2.3. Під-потоки

S-1: додати нового клієнта

Система відображає вікно діалогу, яке містить поля вводу даних про нового клієнта. Користувач заповнює поля (E-2). Система запам'ятовує введені дані. Потім прецедент розпочинається спочатку.

S-2: змінити дані про клієнта

Система відображає вікно діалогу, яке містить список клієнтів і поле для вводу номера клієнта. Менеджер вибирає необхідного клієнта зі списку чи вводить його номер у поле (E-3). Система відображає інформацію про даного клієнта. Менеджер вносить потрібні зміни (E-2). Система запам'ятовує введені дані. Потім прецедент розпочинається спочатку.

S-3: видалити клієнта

Система відображає вікно діалогу, яке містить список клієнтів і поле для вводу номера клієнта. Менеджер вибирає потрібного клієнта зі списку чи вводить його номер у поле (E-2). Система видаляє вибраного клієнта (E-4). Потім прецедент розпочинається спочатку.

S-4: переглянути дані про клієнта

Система відображає вікно діалогу, що містить список клієнтів і поле для вводу номера клієнта. Менеджер вибирає необхідного клієнта зі списку чи вводить його номер у поле (E-3). Система відображає інформацію про клієнта. Коли менеджер перегляне інформацію, прецедент розпочнеться спочатку.

2.4. Альтернативні потоки

E-1: введено неправильне ім'я чи пароль. Користувач повинен повторити введення чи завершити прецедент.

E-2: заповнені не всі поля. Менеджер повинен заповнити пусті поля чи завершити прецедент.

E-3: введено неправильний номер клієнта. Менеджер повинен повторити введення чи завершити прецедент.

E-4: система не може видалити клієнта. Інформація зберігається, система видалить клієнта пізніше. Виконання прецеденту продовжується.

Потік подій для прецеденту „Облік надходження та видачі комплектуючих”

3.1. Передумови

3.2. Головний потік

Прецедент починає виконуватися, коли завскладом підключається до системи та вводить своє ім'я і пароль. Система перевіряє правильність пароля (E-1) та виводить можливі варіанти дій: додати (Add), відмітити (Mark) чи вийти (Exit).

Якщо вибрана операція додати, S-1: виконується потік занести комплектуючі, що надійшли (add a New Components).

Якщо вибрана операція відмітити (Mark), S-2: виконується потік зробити відмітку про видачу комплектуючих (Mark Components).

Якщо вибрана операція вийти (Exit), прецедент завершується.

3.3. Під-потоки

S-1: Занести комплектуючі, що надійшли

Система відображає вікно діалогу, яке містить поля для вводу назв комплектуючих, їх кількості, поставника. Завскладом заповнює вказані поля (E-2). Система запам'ятовує введені дані. Потім прецедент розпочинається спочатку.

S-2: зробити відмітку про видачу комплектуючих

Система відображає список комплектуючих, що знаходяться на складі. Завскладом, біля потрібних комплектуючих вводить кількість виданих (E-3). Система запам'ятовує введені дані. Потім прецедент повторюється спочатку.

3.4. Альтернативні потоки

E-1: введено неправильні ім'я чи пароль. Користувач повинен повторити введення чи завершити прецедент.

E-2: заповнені не всі поля. Користувач повинен заповнити пропущені поля чи завершити прецедент.

E-3: вказано таку кількість комплектуючих, яка перевищує їх кількість на складі. Користувач має повторити введення чи завершити прецедент.

Потік подій для прецеденту „Збір комп'ютерів”

4.1. Передумови.

4.2. Головний потік.

Прецедент починає виконуватися, коли інженер зі збору комп'ютерів підключається до системи і вводить своє ім'я та пароль. Система перевіряє правильність пароля (E-1) та виводить можливі варіанти дій: переглянути (View), відмітити (Mark) чи вийти (Exit).

Якщо вибрана операція переглянути S-1: виконується потік Переглянути наряд на збір комп'ютера (View an Make Computer Order).

Якщо вибрана операція відмітити S-2: виконується потік зробити відмітку про статус комп'ютера, що збирається (Mark Computer).

Якщо вибрана операція вийти, то прецедент завершується.

4.3. Під-потоки

S-1: переглянути наряд на збір комп'ютера

Система відображає вікно діалогу, яке містить список нарядів та поле для вводу номеру наряду. Інженер вибирає потрібний наряд зі списку чи вводить його номер у поле (E-2). Система відображає інформацію про вибраний наряд. Коли інженер перегляне інформацію, прецедент розпочнеться спочатку.

S-2: зробити відмітку про статус комп'ютера, який збирається (Mark Computer)

Система відображає вікно діалогу, яке містить список нарядів. Біля потрібного наряду інженер робить відмітку про статус комп'ютера з даного наряду. Інженер зберігає зміни. Потім прецедент розпочинається спочатку.

4.4. Альтернативні потоки

E-1: введено неправильні ім'я чи пароль. Користувач повинен повторити введення чи завершити прецедент.

E-2: заповнені не всі поля. Користувач повинен заповнити пропущені поля чи завершити прецедент.

E-3: вказано неправильний номер наряду. Користувач має повторити введення чи завершити прецедент.

Потік подій для прецеденту „Запит на необхідні комплектуючі”

5.1. Передумови.

5.2. Головний потік.

Прецедент починає виконуватися, коли інженер зі зборки підключається до системи та вводить своє ім'я і пароль. Система перевіряє правильність пароля (E-1) та виводить можливі варіанти дій: переглянути (View), замовити (Order) чи вийти (Exit). Якщо вибрана операція переглянути, S-1: виконується потік переглянути замовлені комплектуючі на складі (*View Ordered Components on Warehouse*).

Якщо вибрана операція замовити (Order), S-2: виконується потік замовити необхідні комплектуючі на складі (*Order Required Components on Warehouse*).

Якщо вибрана операція вийти, то прецедент завершується.

5.3 Під-потоки.

S-1: Переглянути всі замовлені комплектуючі на складі (*View Ordered Components on Warehouse*)

Система відображає наступну інформацію про всі зроблені замовлення даним інженером зі збору: дата запити, назви комплектуючих, їх кількість, виконано замовлення чи ні. Коли інженер зі збору переглянув список, він повідомляє систему. Прецедент починається спочатку.

S-2: Замовити необхідні комплектуючі на складі (*Order Required Components on Warehouse*)

Система відображає вікно діалогу, яке містить поля для вводу списку необхідних комплектуючих та їх кількості. Інженер зі зборки заповнює його. Система запам'ятовує введені дані. Потім прецедент розпочинається спочатку.

5.4. Альтернативні потоки

E-1: введено неправильне ім'я чи пароль. Користувач має повторити введення.

Опис потоків подій для прецедентів Управління інформацією про поставників та Управління інформацією про комплектуючих аналогічне опису для прецеденту Управління інформацією про клієнта; для прецеденту Тестування комп'ютерів – прецеденту Збір комп'ютерів.

Для введення інформації про потоки для всіх прецедентів, потрібно виконати наступні дії. Для необхідного прецеденту натисніть ПКМ й оберіть Open Use case Details (рис. 21). Далі відкриється вікно опису прецеденту (рис. 22).

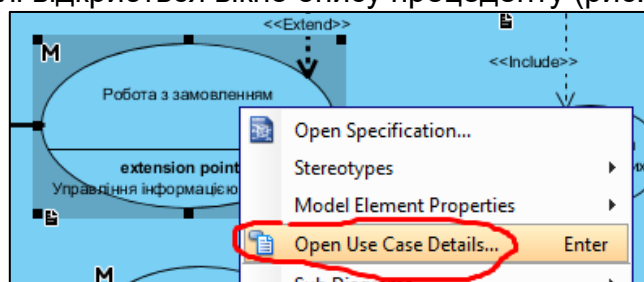


Рис. 21. Вікно Open Use Case Details

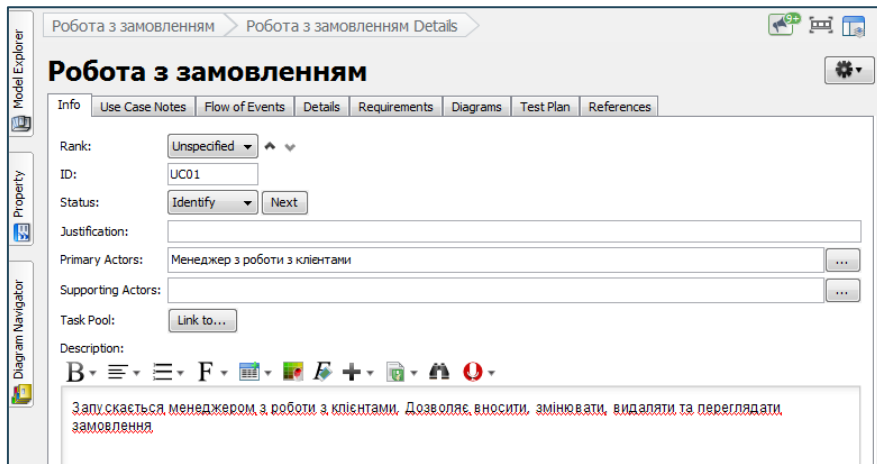
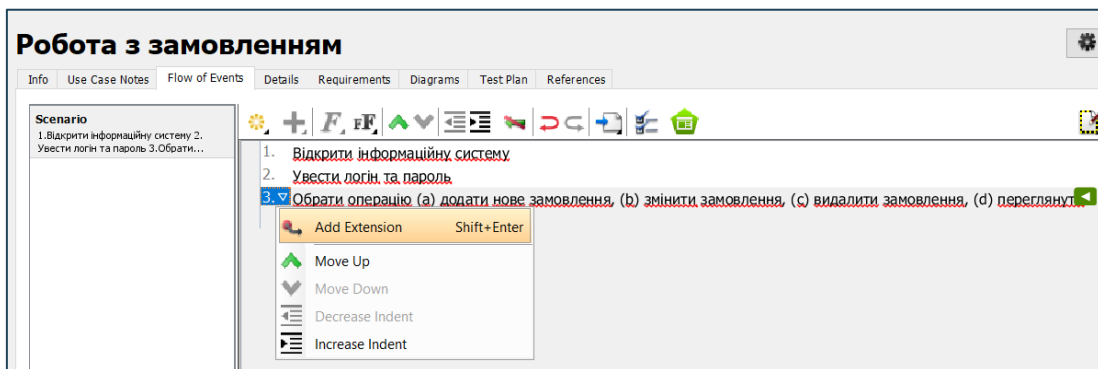


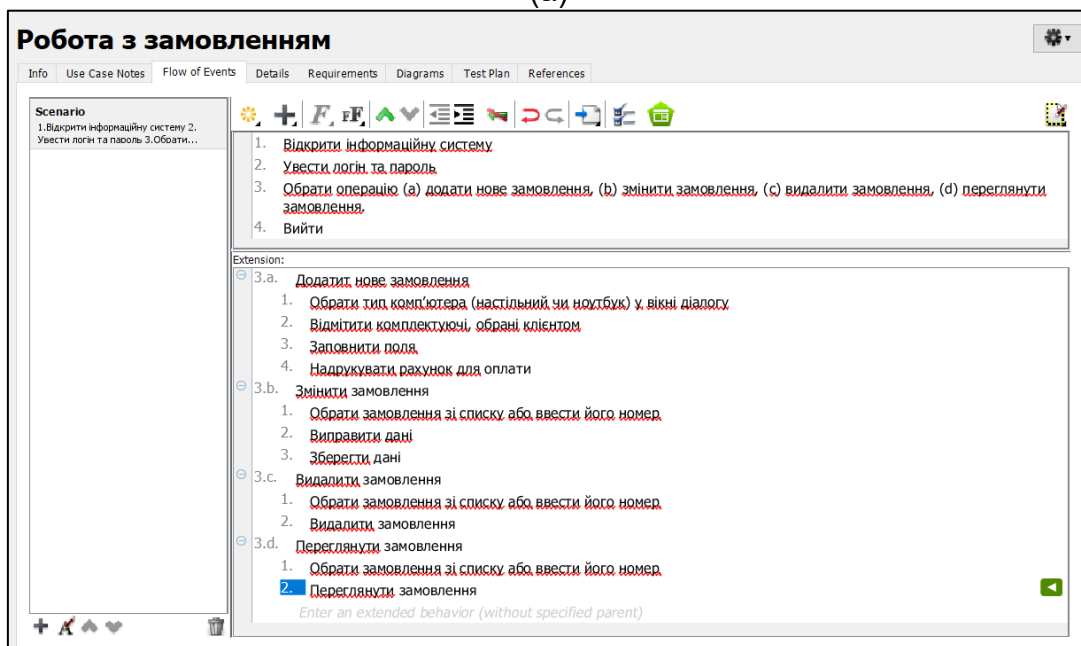
Рис. 22. Опис прецеденту "Робота із замовленням"

Базова інформація включає дані про використання прецеденту. Ранг прецеденту та його статус визначаються розробниками. Пункт Primary Actors уключає акторів, які беруть участь у прецедентів. Supporting Actors – це актори, що зазнають впливу прецеденту.

Для введення потоку подій обираєте закладку Flow of Events (рис. 23) і вводите короткий опис.



(a)



(б)

Рис. 23. Заповнення закладки потік подій

Далі потрібно заповнити інші елементи опису. Оскільки потік для прецеденту "Робота із замовленням" може виконуватися лише для клієнта, що вже внесений до бази системи, потрібно додати цю вимогу до опису. Оберіть закладку Requirements (рис. 24).

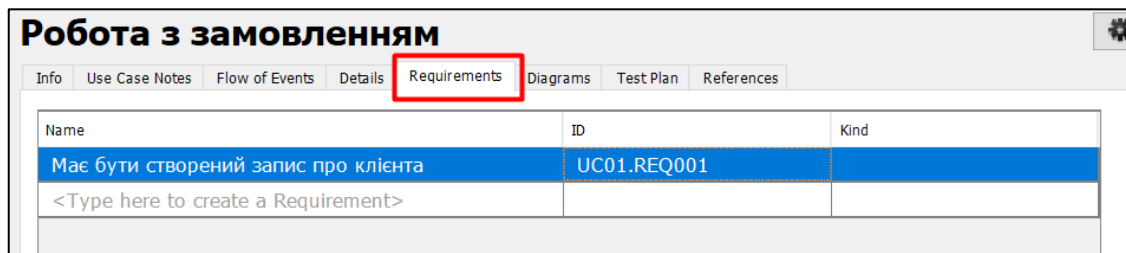


Рис. 24. Внесення даних про вимоги до прецеденту

Після цього перейдіть до закладки Details (рис. 25). За допомогою кнопок Insert Requirements і Insert Use Case, додайте інформацію про попередні умови для даного прецеденту.

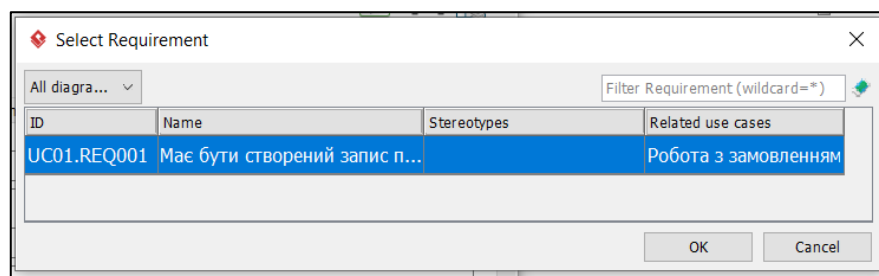
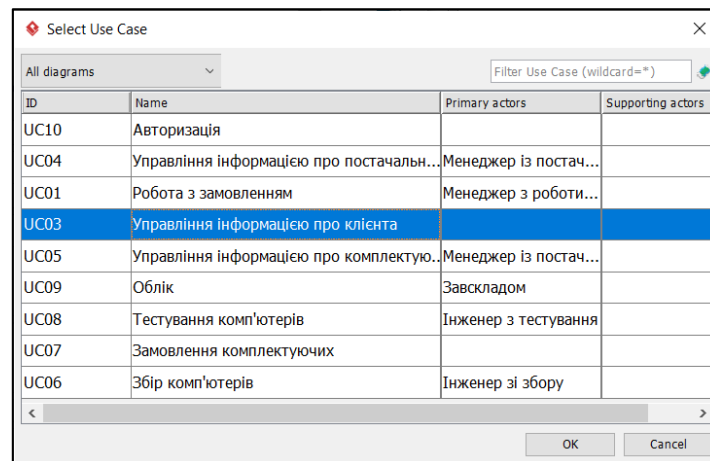
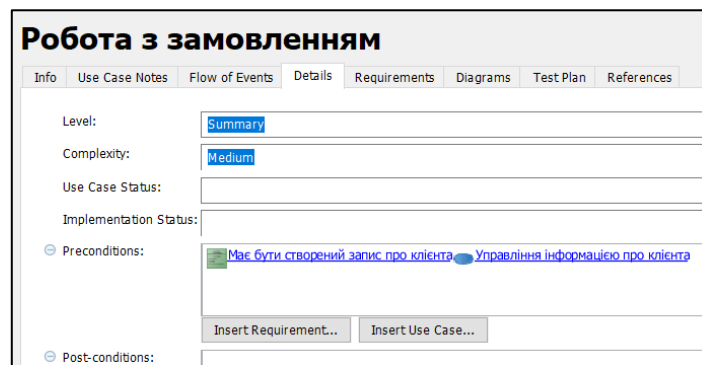


Рис. 25. Опис передумов прецеденту
Діючи за зразком, створіть опис для інших прецедентів.

Контрольні запитання до лабораторної роботи 1

1. У чому полягає різниця між відношеннями агрегації та узагальнення? Наведіть конкретні приклади цих типів відношень.
2. Що є спільного і у чому полягає різниця між відношеннями агрегації та композиції? Наведіть приклади цих типів відношень.
3. Для чого у процесі проектування ПЗ використовують моделі прецедентів? Що має відобразити така модель ?
4. На підставі якої інформації будується модель прецедентів?
5. Що таке «актор» та «прецедент»? Які типи відношень можуть бути визначені між ними?
6. Який тип відношень між акторами і прецедентами є найбільш поширеним? Навести приклади.
7. У чому полягає спільність і у чому є різниця між відношенням включення та розширення на діаграмі прецедентів? Навести приклади.
8. Для чого використовується відношення узагальнення? Навести приклади.

Лабораторна робота 2

Створення діаграми класів

Завдання:

1. Створити діаграму класів для одного зі сценаріїв діаграми прецедентів, створеної в попередній лабораторній роботі. Для кожного класу необхідно задати атрибути й операції. Кожен клас повинен бути докладно задокументований.
2. Створити пакет для групування класів, створених у пункті 1.
3. Згрупувати класи з пункту 1 у пакет.
4. Розробити головну діаграму класів.

Теоретичні відомості

Діаграми класів (class diagram) використовуються для моделювання статичного вигляду системи з точки зору проектування.

Клас (це опис сукупності об'єктів які мають спільний набір властивостей і володіють однаковою поведінкою) Об'єктом називають екземпляр відповідного класу. Об'єктом називається концепція, абстракція чи річ з чітко визначеними границями і значенням для системи. Кожен об'єкт характеризується трьома показниками стан, поведінка, індивідуальність.

Стан – це одна із умов, в яких може перебувати об'єкт. Стан системи зазвичай змінюється з часом і визначається набором властивостей, які називаються атрибутами. Поведінка визначає як об'єкт реагує на запити інших об'єктів і що він може робити сам. Поведінка реалізується за допомогою операцій для об'єкта. Індивідуальність означає, що кожен об'єкт унікальний, навіть якщо його стан ідентичний стану іншого об'єкта.

Діаграма класів – це UML-діаграма, на якій показано набір класів, інтерфейсів, кооперацій і відносин між ними. Діаграми класів використовуються:

- для моделювання словника системи, що дозволяє прийняття рішення про те, які абстракції є частиною системи, а які – ні.
- для моделювання простих кооперацій. Кооперація – це множина класів, інтерфейсів й інших елементів, що діють спільно для забезпечення деякої поведінки.
- для моделювання логічної схеми бази даних.

Існують три різні погляди на побудову діаграм класів або будь-якої іншої моделі:

- *концептуальний погляд* - діаграми класів слугують для представлення понять досліджуваної предметної області. Ці поняття будуть відповідати класам, що їх реалізують, але пряма відповідність може бути відсутньою. Концептуальна модель може бути слабо зв'язана або взагалі не мати ніякого відношення до програмного забезпечення, що її реалізує, тому її можна розглядати без прив'язки до конкретної мови програмування;
- *погляді специфікації* - розглядається програмна система, при цьому розглядаються тільки її інтерфейси, але не реалізація;
- *погляд реалізації* - класи діаграми відповідають класам програмної системи.

У Visual Paradigm позначення класів складаються з трьох частин:

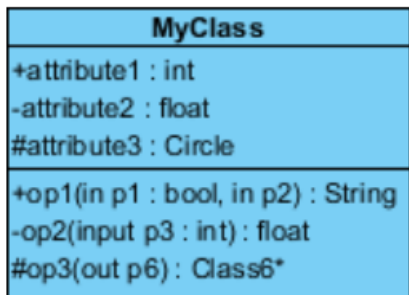
1. Ім'я класу.

2. Атрибути класу, що визначають дані для опису екземпляру класу та його стану. Типовий опис атрибута : *видимість ім'я : тип = значення_за_замовчуванням*. Тип атрибута (цілочисельний int, дійсний float, текстовий String) відображається

після двокрапки. Видимість може бути: - *закритий (приватний)*; + *відкритий (публічний)*; # *захищений*.

3. Операції класу визначають дії, які можна застосувати до класу або його об'єкту. Операція описується рівнем доступу (видимості), списком параметрів та типом результату: *видимість ім'я(список_параметрів) : тип*. Можна опустити усі частини специфікації операції, за винятком імені та круглих дужок.

Графічне зображення класу MyClass наведено нижче.



My Class має 3 атрибути та 3 операції.
 Вхідний параметр p3 для операції op2 є цілочисельним
 Операція op2 повертає дійсне значення
 Операція op3 повертає покажчик Class6

Клас може взаємодіяти з іншими класами. Приклади можливих відношень між класами наведено у Таблиці 1.

Таблиця 1. Типи відношень між класами

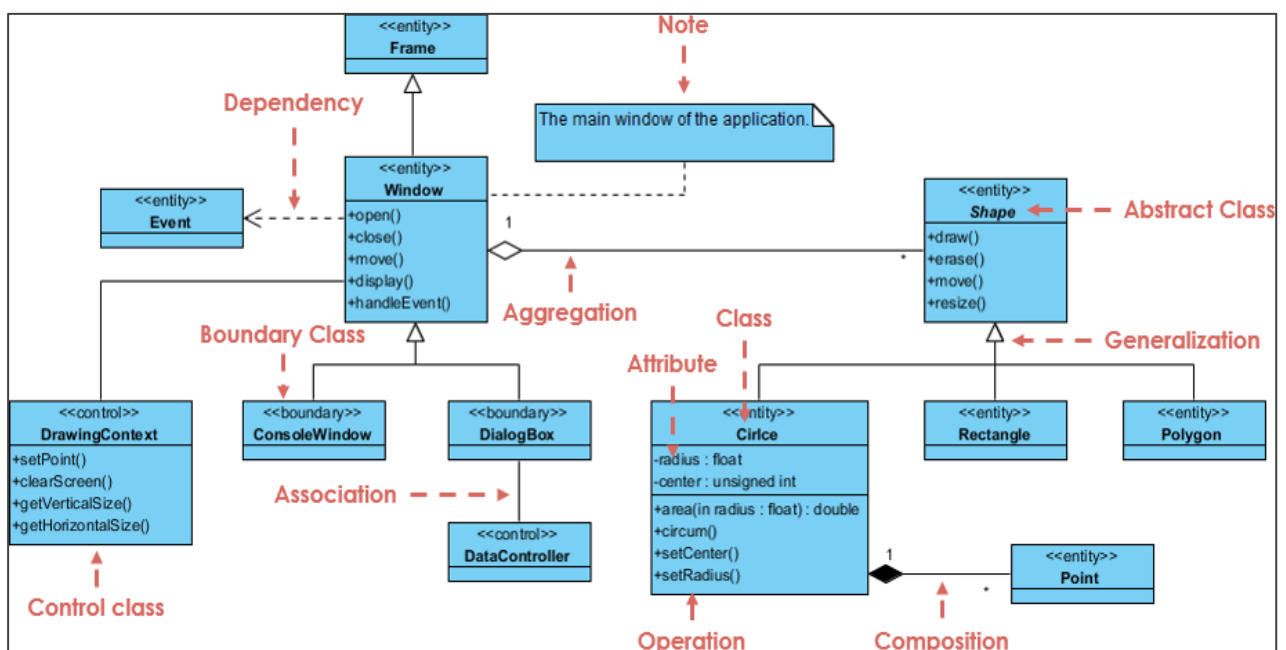
	<p>Відношення узагальнення (generalization relationship), що відображає зв'язок між загальним елементом (батьківський чи предок) та частинними (нащадок). Прикладом такого зв'язку є: клас "Геометрична фігура" – класи "Прямокутник", "Трикутник", "Коло" тощо.</p>
	<p>Відношення асоціації (association relationship). Проста асоціація зв'язує лише два класи. Асоціація може бути спрямованою або неспрямованою.</p>
	<p>Відношення агрегації (aggregation relationship) відображає зв'язок між декількома класами за типом "частина-ціле". У прикладі відношення вказує, що Class2 є частиною Class1. Знак * показує, що багато екземплярів Class2 асоціюються з Class 1.</p>
	<p>Відношення композиції (composition relationship) є частинним випадком відношення агрегації, з урахуванням того, що з видаленням цілого, видаляються і всі складові.</p>
	<p>Відношення залежності (dependence relationship) існує між двома класами. Зміни у визначенні одного класу можуть спричинити зміни для іншого.</p>

Клас сутність (entity class) використовується для моделювання даних і поведінки з тривалим життєвим циклом. Даний тип класів може представляти сутності реального світу або внутрішні елементи системи. Як правило такі класи не залежать від оточення, тобто нечутливі до взаємодії навколишнього середовища та системи. Зазвичай, класи сутності це ті класи, які потрібні системі для виконання певних обов'язків.

Граничні класи (boundary class) забезпечують взаємодію між навколишнім середовищем та внутрішніми елементами системи. Фактично, вони описують інтерфейс для користувача чи іншої системи (тобто, актора). Граничні класи використовуються для моделювання інтерфейсів системи. Для їх визначення вивчаються пари актор/сценаріїв.

Управляючі класи (control class) використовуються для моделювання послідовної поведінки одного чи декількох прецедентів та координації подій, що реалізують закладену в них поведінку. Такі класи можна представити як класи, що «реалізують» прецедент і визначають його динаміку.

На рисунку наведено приклад діаграми класів, для якої використано всі типи залежностей і класів.



Хід виконання лабораторної роботи

1. Створення діаграми класів для сценарію "Додати нове замовлення" прецеденту "Робота із замовленням"

Створимо в Логічному представленні браузера нову діаграму класів і назвемо її "Add New Order". У поле документації напишемо для неї наступний текст: "Діаграма класів для сценарію "Додати нове замовлення" прецеденту "Робота із замовленням"". Заповнення діаграми почнемо з визначення класів-сутностей. Розглянутий сценарій складається з: самого замовлення; клієнта, що робить замовлення; комплектуючих виробів, які входять у замовлення.

Створимо класи-сутності *Order* (Замовлення), *Client* (Клієнт) і *ComponentPart* (Комплектуючий виріб). Оскільки в одне замовлення може входити багато різних комплектуючих виробів, і один комплектуючий виріб може входити в багато замовлень, то введемо ще один клас-сутність *OrderItem* (Склад замовлення). Опишемо класи.

Клас Client:

Параметр	Значення
Коментар	Клас, що представляє клієнта фірми
Атрибути	name : String - найменування клієнта address : String - адреса клієнта phone : String - телефон клієнта Всі атрибути мають модифікатор доступу - private
Операції	AddClient() - додавання нового клієнта RemoveClient() - видалення існуючого клієнта GetInfo() - одержати інформацію про клієнта Всі операції мають модифікатор доступу - public

Клас Order:

Параметр	Значення
Коментар	Клас, що представляє собою замовлення, що робить клієнт
Атрибути	orderNumber : Integer - номер замовлення orderDate : Date - дата оформлення замовлення orderComplete : Date - дата виконання замовлення Всі атрибути мають модифікатор доступу - private
Операції	Create() - створення нового замовлення SetInfo() - занести інформацію про замовлення GetInfo() - одержати інформацію про замовлення Всі операції мають модифікатор доступу - public

Клас OrderItem:

ПАРАМЕТР	ЗНАЧЕННЯ
Коментар	Клас, що представляє собою пункт замовлення, що робить клієнт
Атрибути	itemNumber : Integer - номер пункту замовлення quantity : Integer - кількість комплектуючих виробів price : Double - ціна за одиницю Всі атрибути мають модифікатор доступу - private
Операції	Create() - створення нового рядка замовлення SetInfo() - занести інформацію про рядок замовлення GetInfo() - одержати інформацію про рядок замовлення Всі операції мають модифікатор доступу - public

Клас ComponentPart:

Параметр	Значення
Коментар	Клас, що представляє собою комплектуючі вироби
Атрибути	name : String - найменування manufacturer : String - виробник price : Double - ціна за одиницю description - опис Всі атрибути мають модифікатор доступу - private
Операції	AddComponent() - додавання нового комплектуючого виробу RemoveComponent() - видалення комплектуючого виробу GetInfo() - одержати інформацію про комплектуючий виріб Всі операції мають модифікатор доступу - public

Для створення класів у Visual Paradigm у Diagram Navigator обираєте Class Diagram (рис. 1). В утвореній діаграмі перетягуєте на поле елемент Class.

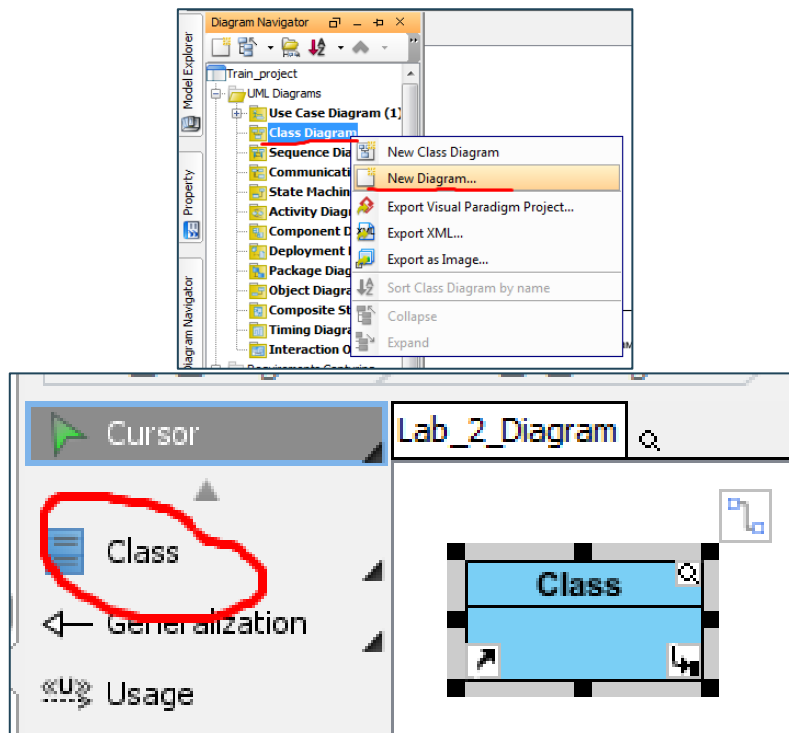


Рис. 1. Створення діаграми класів

Властивості класу задаємо через вікно Open Specification (рис. 2).

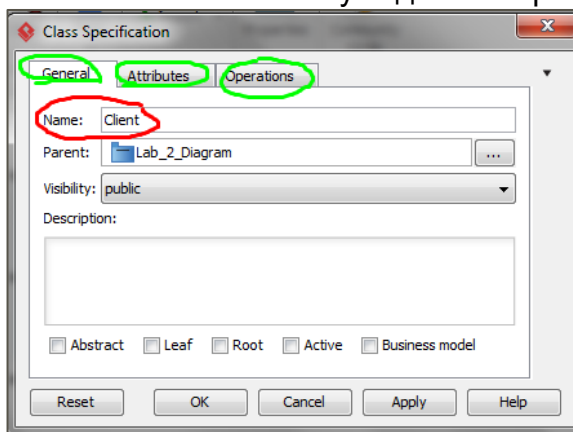


Рис. 2. Опис властивостей класу

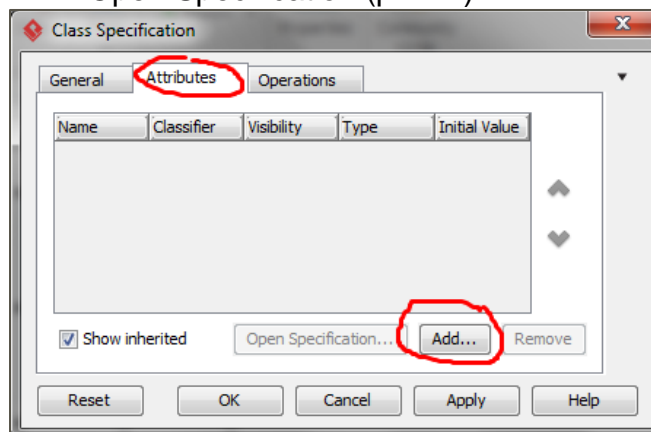


Рис. 3. Створення атрибутів

Для створення атрибутів обираєте закладку Attributes і натискаєте Add (рис. 4).

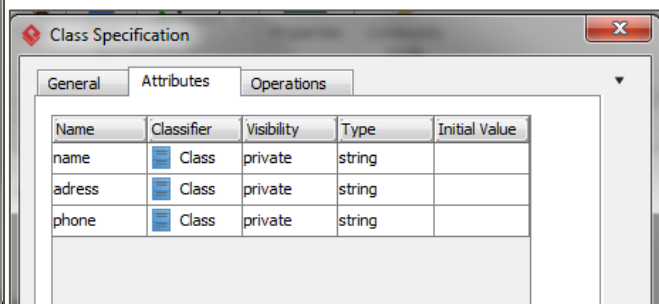
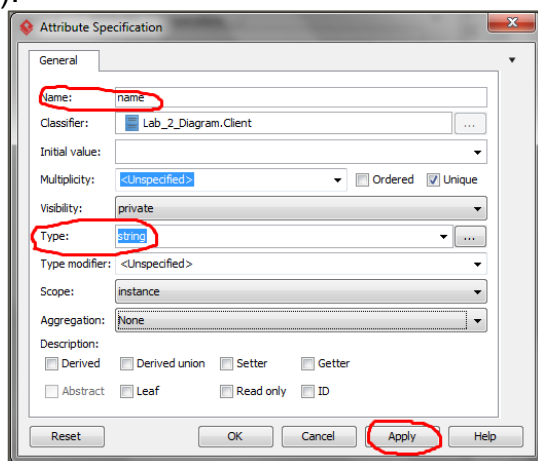


Рис. 4. Заповнення даних про атрибути класу

Аналогічним чином створюються Операції, властиві даному класу (рис. 5).

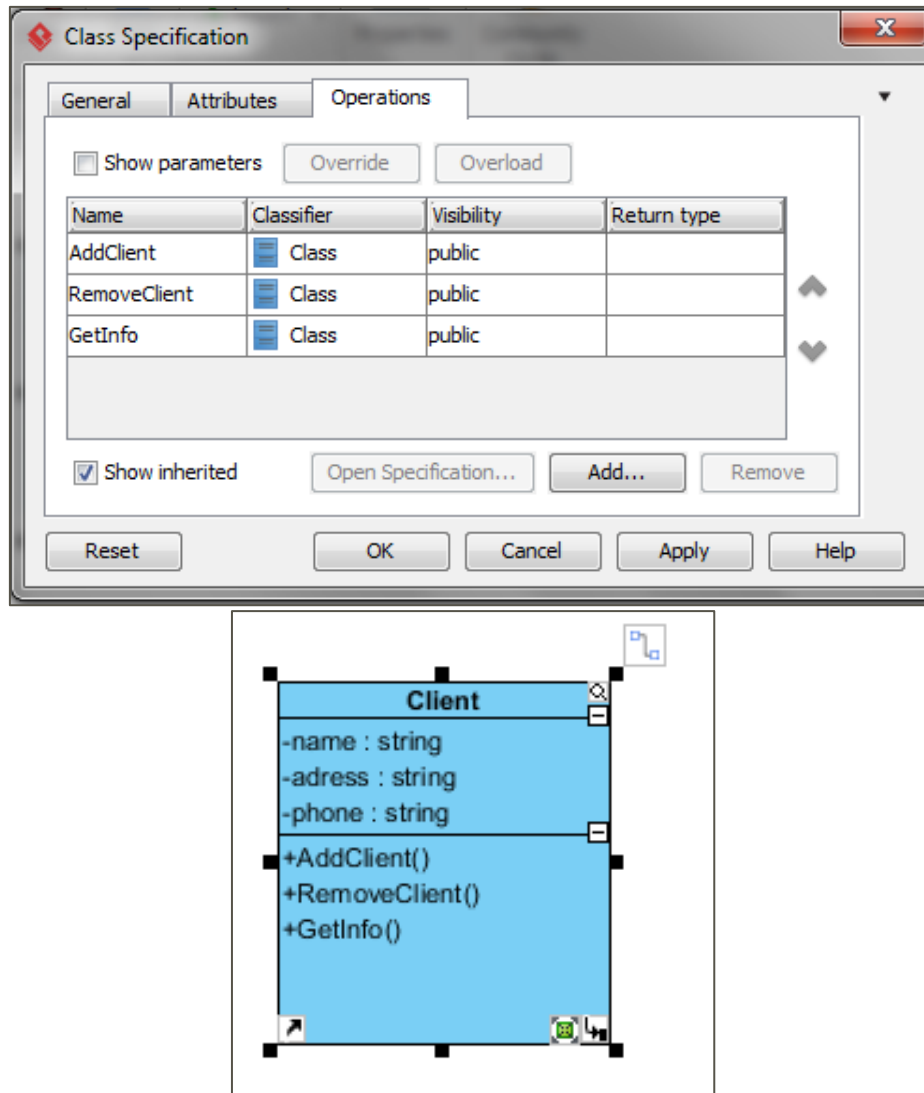


Рис. 5. Заповнення даних про операціїй результат виконання дій

Додамо відношення між класами:

- класи *Client* й *Order* - відношення асоціації, оскільки дані два класи просто зв'язані один з одним і ніякі інші типи зв'язків тут застосувати не можна. Один клієнт може зробити декілька замовлень, кожне замовлення надходить тільки від одного клієнта, тому кратність зв'язку з боку класу *Client* - 1, з боку *Order* - 1..*;
- класи *Order* й *OrderItem* - відношення композиції, оскільки рядок замовлення є частиною замовлення, і без нього існувати не може. В одне замовлення може входити декілька рядків замовлення, рядок замовлення ставиться тільки до одного замовлення, тому кратність зв'язку з боку *Order* - 1, з боку *OrderItem* - 1..n;
- класи *OrderItem* й *ComponentPart* - відношення агрегації, оскільки комплектуючі вироби є частинами рядка замовлення, але й ті, й інші, є самостійними класами. Один комплектуючий виріб може входити в багато рядків замовлення, в один рядок замовлення входить тільки один комплектуючий виріб, тому кратність зв'язку з боку *ComponentPart* - 1, з боку *OrderItem* - 1..n.

Задати їх можна або клацнувши двічі ЛКМ по точках з'єднання лінії асоціації з першим і другим класом, відповідно (рис. 6).

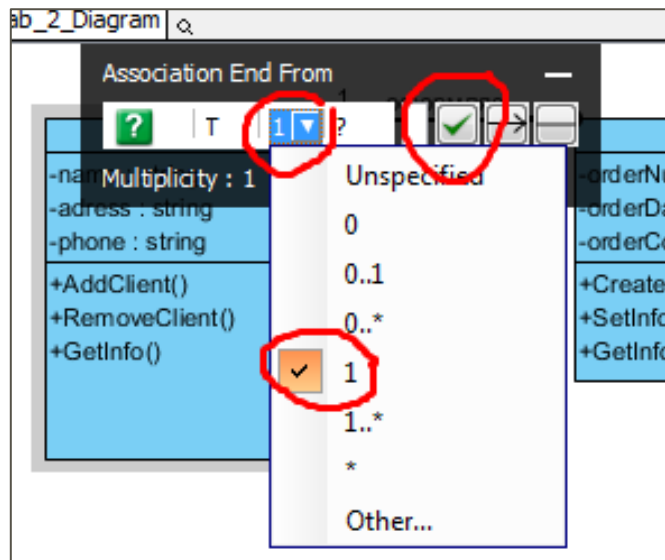


Рис. 6. Відношення між класами

В іншому випадку можна клацнути ПКМ по лінії з'єднання і відкрити вікно з доступними функціями (рис. 7).

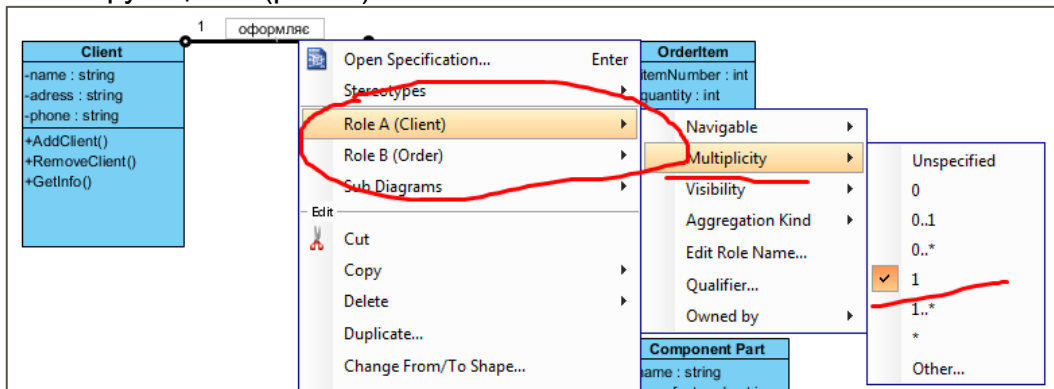


Рис. 7. Встановлення параметру Multiplicity

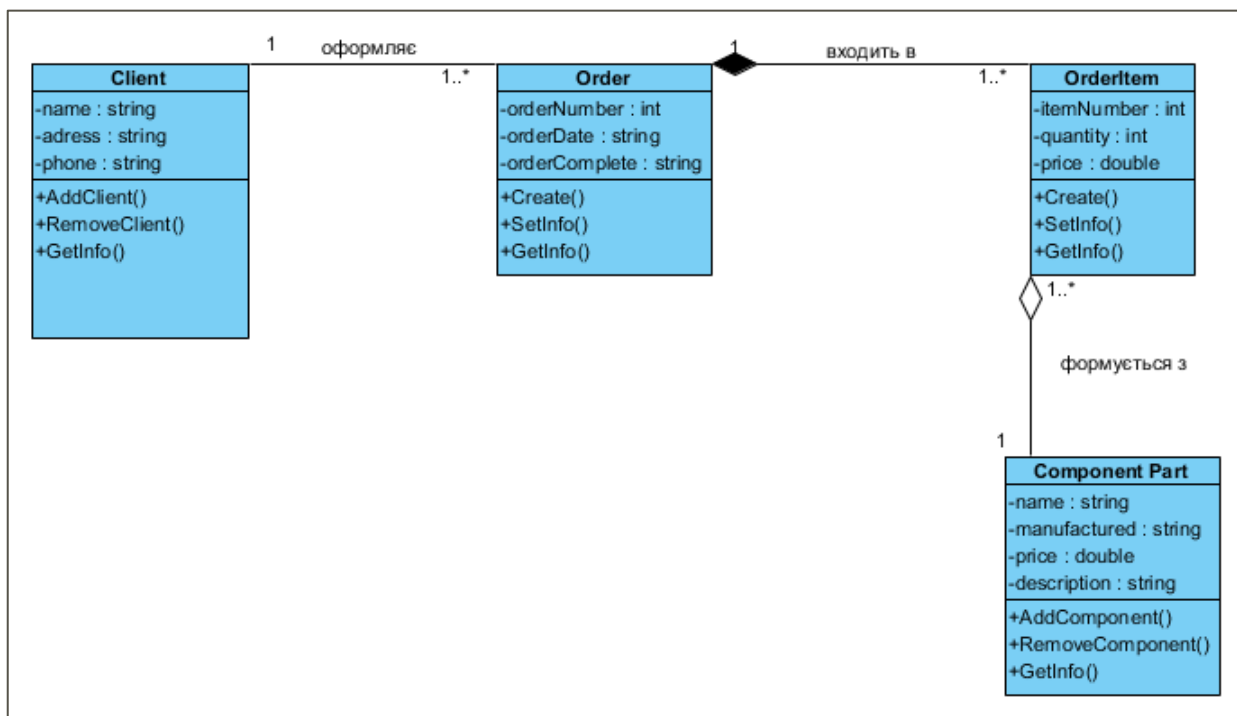


Рис. 8. Класи-сущності й відносини між ними

Створіть копію діаграми, що містить класи-сутності. У одному з файлі потрібно буде створити головну діаграму класів. Додайте на діаграму клас AddNewOrder та встановіть для нього стереотип boundary (рис. 9).

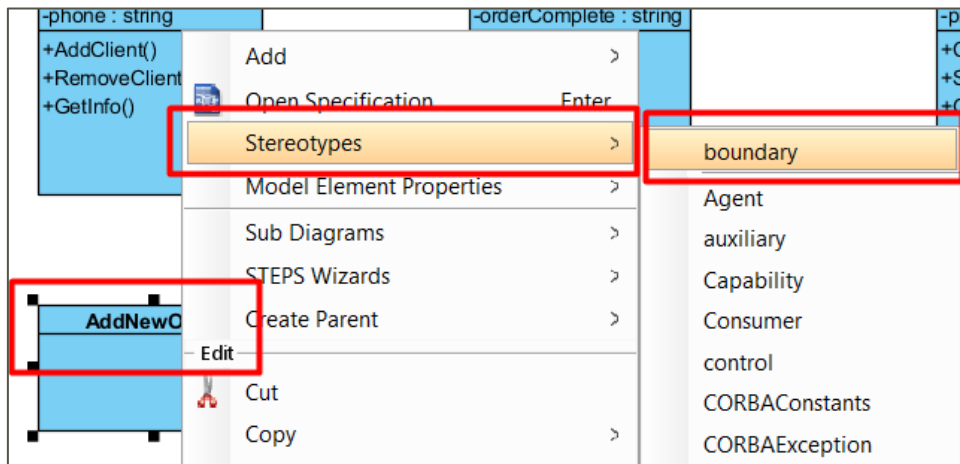


Рис. 9. Створення граничного класу

Дадайте на головну діаграму класи OrderOptions (стереотип – boundary) та OrderManager (стереотип – control) (рис. 10).

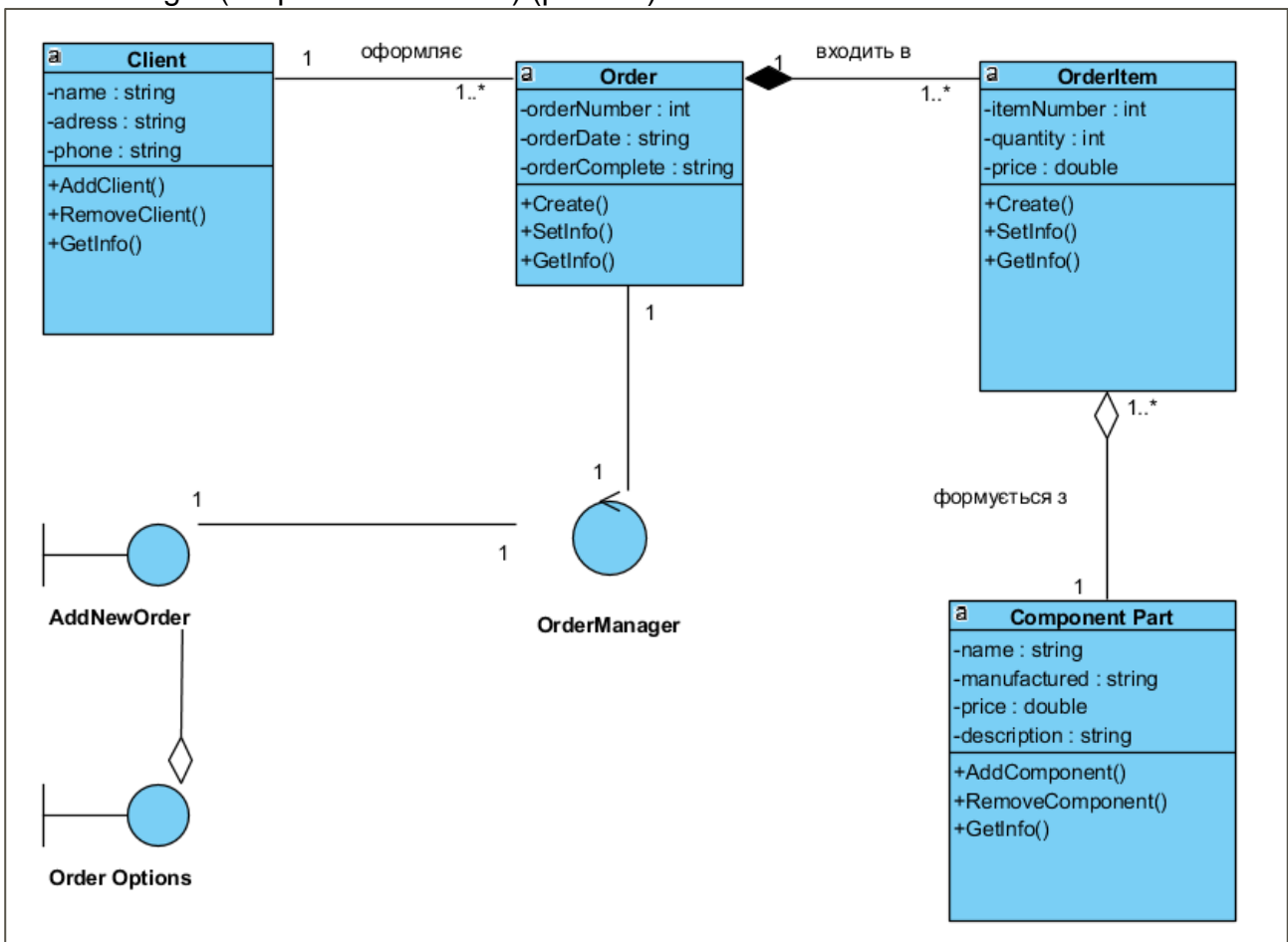


Рис. 10. Головна діаграма класів

Пакети призначаються для групування елементів за певними критеріями. У найпростішому випадку класи можна групувати за їх стереотипами. Створимо пакет *Entities* (класи-сутності) (рис. 11).

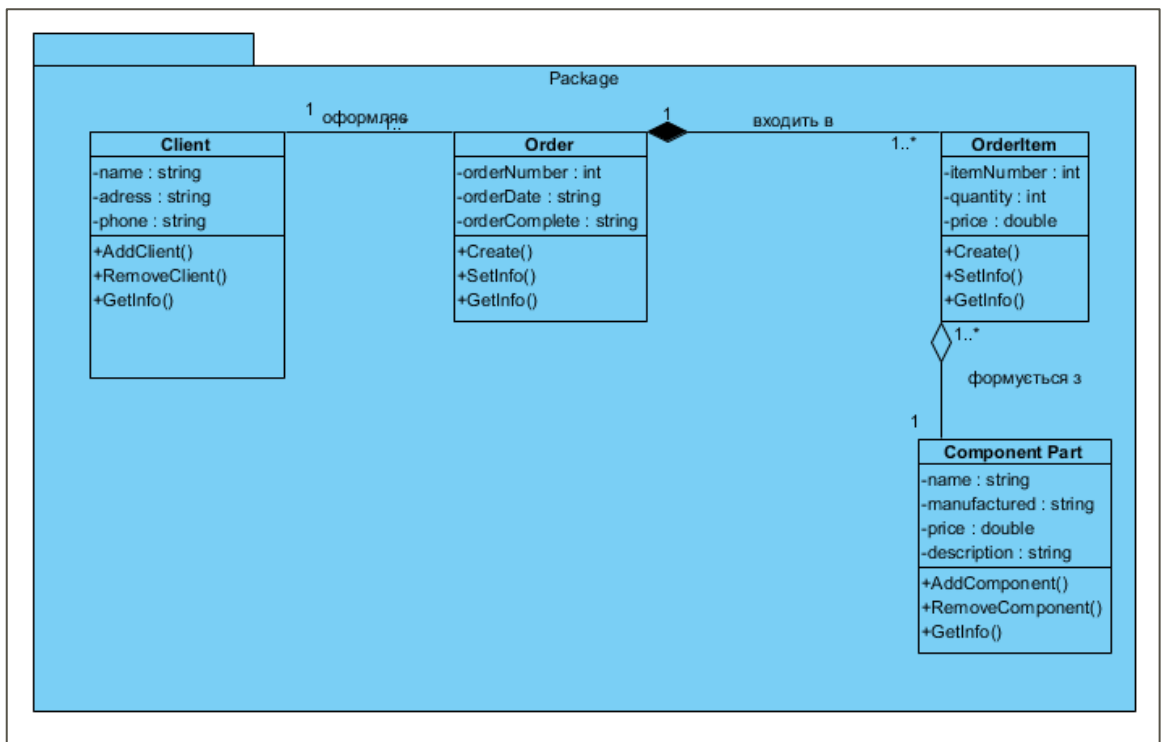


Рис. 11. Пакет класів-сутностей

Контрольні запитання до лабораторної роботи 2

1. Чи дозволяє діаграма класів UML відобразити відношення: а) між класами об'єктів; б) між екземплярами класів?
2. Які значення може мати атрибут видимості класів та що вони означають?
3. Які відношення позначаються в діаграмі класів UML спеціальними графічними символами?
4. Наведіть типи відношень для UML–діаграми класів і надайте їх стислу характеристику.
5. У чому полягає семантична різниця між відношенням залежності та відношенням асоціації? Наведіть конкретні приклади цих відношень.
6. У чому полягає семантична різниця між відношенням агрегації та відношенням узагальнення? Наведіть конкретні приклади цих відношень.
7. Що означає відношення композиції? Чим воно відрізняється від відношення агрегації?
8. Що таке інтерфейс? Для чого він може бути застосований при розробці діаграми класів?
9. У чому полягає різниця між діаграмою класів та діаграмою об'єктів? Для чого доцільно використання діаграм об'єктів?
10. Для чого призначаються пакети? За якими параметрами можна групувати класи?

Лабораторна робота 3

Створення діаграм послідовності та комунікації

Завдання:

1. Створити діаграму послідовності та діаграму комунікації для одного зі сценаріїв будь-якого прецеденту, створеного в лабораторній роботі № 1.
2. Створити діаграму комунікації.

Теоретичні відомості та хід виконання роботи

У складних системах об'єкти взаємодіють один з одним, обмінюючись повідомленнями. Ключові поняття обміну повідомленнями:

- Взаємодія – це поведінка, яка виражається в обміні повідомленнями між об'єктами в заданому контексті, унаслідок чого досягається певна мета.
- Повідомлення – це специфікація обміну даними між об'єктами, за якого відбувається передача інформації і передбачається, що у відповідь відбудеться певна дія.

Взаємодії моделюють потоки управління за часовою упорядкованістю повідомлень або за структурною організацією об'єктів, що обмінюються повідомленнями.

Відповідно, виділяють два типи моделей:

- Діаграми послідовності (sequence diagram) відображають часову упорядкованість повідомлень.
- Діаграми комунікації (communication diagram) відображають структурну організацію об'єктів, які обмінюються повідомленнями.

Діаграми послідовності та діаграми комунікації є ізоморфними, тобто можуть вільно трансформуватися між собою. На діаграмі послідовності неявно присутня вісь часу, що допомагає візуалізувати відносини між переданими повідомленнями.

Головними елементами, з яких утворюються діаграми послідовності та комунікації, є об'єкти – окремі екземпляри класів, які створюються на етапі реалізації моделі (виконання програми). Об'єкти, які надсилають повідомлення, називаються клієнтами а об'єкти, які їх обробляють серверами.

Діаграми послідовності та діаграми комунікації зображають взаємодію елементів моделі в контексті реалізації окремих прецедентів.

Для створення діаграми послідовності у середовищі Visual Paradigm, у Diagram Navigator обираєте Sequence Diagram (рис. 1) та вводите її назву.

Діаграма послідовності призначена для відображення часових залежностей, що виникають у процесі взаємодії об'єктів. На верхньому краю діаграми горизонтально розміщують об'єкти (прямокутники із назвами об'єктів). З кожного об'єкта виходить вертикальна штрих пунктирна лінія (лінія життя об'єкта), на якій умовно відкладено час.

Зліва на діаграмі зображають актора (або об'єкт), який є ініціатором взаємодії. Справа від нього зображають інший об'єкт, який безпосередньо взаємодіє з актором чи першим об'єктом і т.д. Лінія життя об'єкта (object lifeline) – це вертикальна лінія на діаграмі послідовності, яка представляє існування об'єкта протягом певного періоду часу. Якщо об'єкт існує в системі постійно, то і його лінія життя будується по всій робочій області діаграми послідовності (від верхньої до нижньої частини).

Для створення елементів діаграми послідовності, не виходячи зі вже створеної діаграми, через Diagram Navigator зверніться до діаграми прецедентів. Оберіть актора Менеджер з роботи з клієнтами та перетягніть значок у поле діаграми (рис. 1).

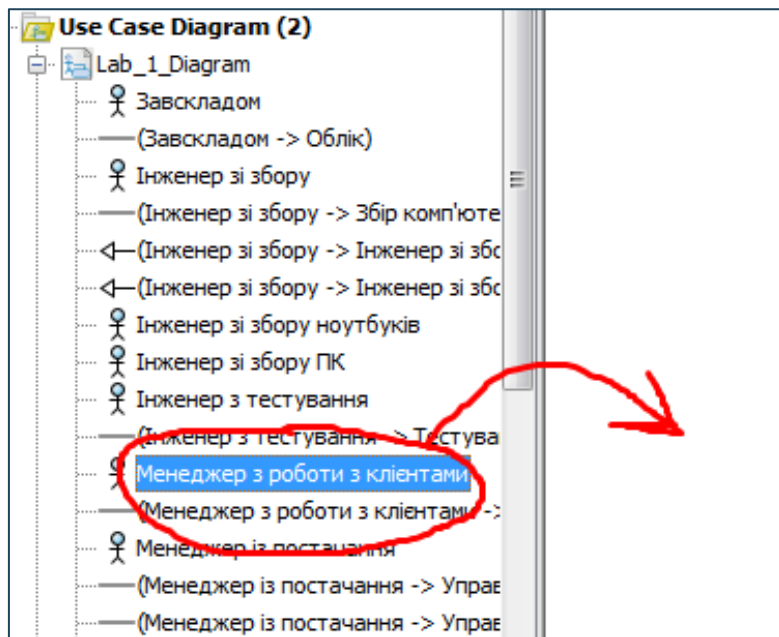


Рис. 1. Створення актора на діаграмі послідовності

Далі обираєте елемент LifeLine – об'єкт, з якими актор обмінюється повідомленнями у процесі функціонування системи (рис. 2). Об'єкти, що відповідають створюваній системі, наведені на рис. 3

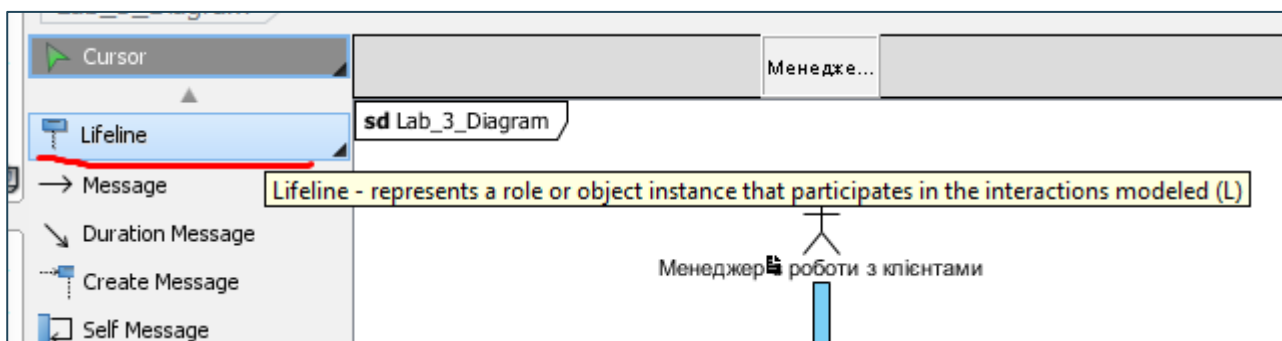


Рис. 2. Створення об'єкту (LifeLine), з яким актор обмінюється повідомленням

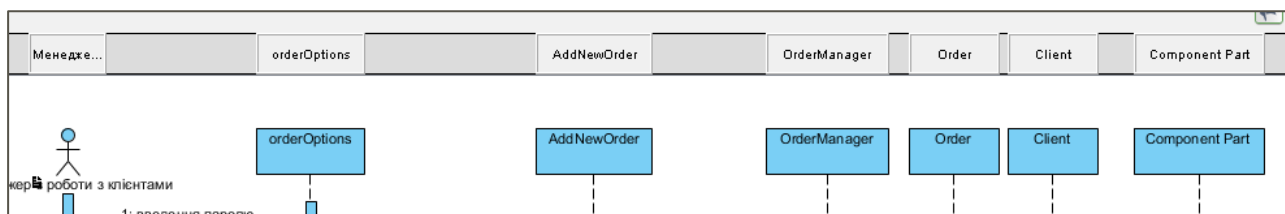



Рис. 3. Верхня частина діаграми послідовності

Потім на діаграмі потрібно розмістити повідомлення, якими будуть обмінюватися об'єкти (табл. 1).

Таблиця 1

Номер повідомлення	Об'єкт - відправник повідомлення	Об'єкт - одержувач повідомлення	Назва
1	Менеджер по роботі з клієнтами	OrderOptions	введення паролю
2	OrderOptions	OrderOptions	перевірка паролю
3	Менеджер по роботі із клієнтами	OrderOptions	вибір операції "додати"
4	OrderOptions	AddNewOrder	відображення полів введення
5	Менеджер по роботі із клієнтами	AddNewOrder	вибір типу комп'ютера
6	AddNewOrder	OrderManager	одержання списку клієнтів
7	OrderManager	Client	одержання списку клієнтів
8	Client	AddNewOrder	список клієнтів
9	AddNewOrder	AddNewOrder	відображення списку клієнтів
10	Менеджер по роботі із клієнтами	AddNewOrder	вибір клієнта
11	AddNewOrder	OrderManager	одержання списку комплектуючих
12	OrderManager	ComponentPart	одержання списку комплектуючих
13	ComponentPart	AddNewOrder	список комплектуючих
14	AddNewOrder	AddNewOrder	відображення списку комплектуючих
15	Менеджер по роботі із клієнтами	AddNewOrder	вибір необхідних комплектуючих
16	Менеджер по роботі із клієнтами	AddNewOrder	зберегти замовлення
17	AddNewOrder	OrderManager	передача управління
18	OrderManager	Order	зберегти

Для цього клацаєте ЛКМ по значку  й обираєте відповідний тип повідомлення (рис. 4, 5).

Найуживаніші типи повідомлень:

- Procedure Call – виклик процедур, виконання операцій або перехід на вкладені потоки керування. Початок стрілки з'єднаний з фокусом керування об'єкта, який ініціює повідомлення.
- Object Message – виклик методів об'єкта.
- Return Message – повернення з виклику процедури/методу (наприклад, повідомлення про завершення обчислень без виведення результату).
- Asynchronous Message – просте асинхронне повідомлення, яке передається в довільний момент.
- Message to Self – повідомлення об'єкта самому собі (рефлексивне повідомлення). Якщо в результаті рефлексивного повідомлення створюється новий підпроцес або гілка керування, то говорять про рекурсивний або вкладений фокус керування.

Особливість діаграми послідовності можливість візуалізувати розгалуження процесу. Для зображення розгалуження використовуються дві або більше стрілки, що виходять з однієї точки фокусу керування. Поруч із кожною стрілкою зазначається умова гілки (у формі булевого виразу (виразу алгебри логіки)).

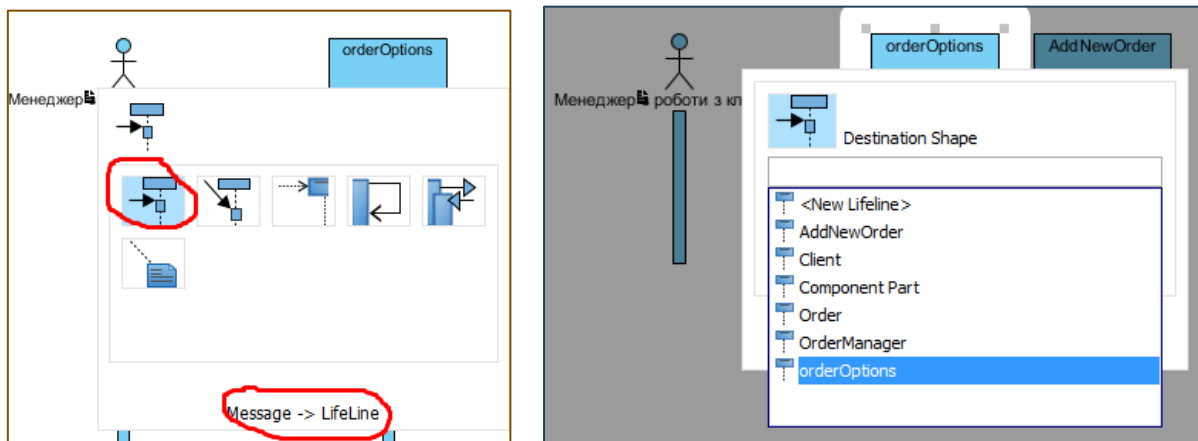


Рис. 4. Пряме повідомлення

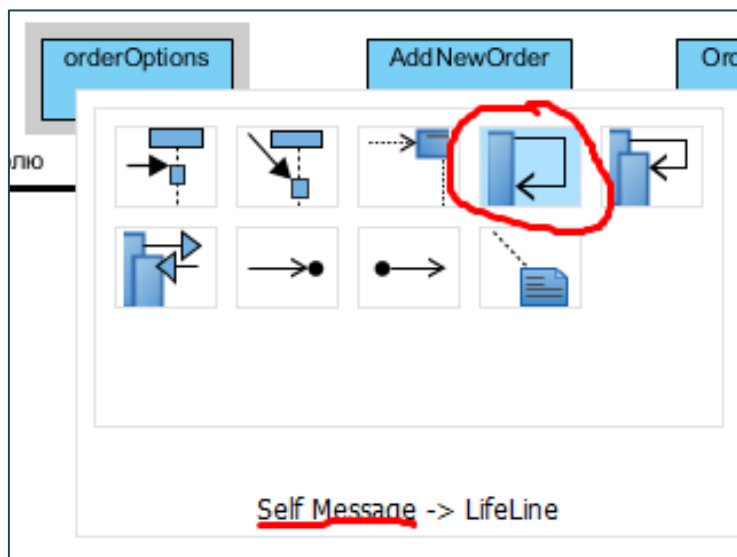


Рис. 5. Повідомлення типу Self-Message

Уведені повідомлення отримують номери. Порядок присвоєння визначається тим, яка операція знаходиться на діаграмі вище, а яка нижче. При підключенні до одного блоку, нумерація стає двозначною (рис. 6).



Рис. 6. Приклад нумерації повідомлень

Підсумкова діаграма послідовності наведена на рис. 7.

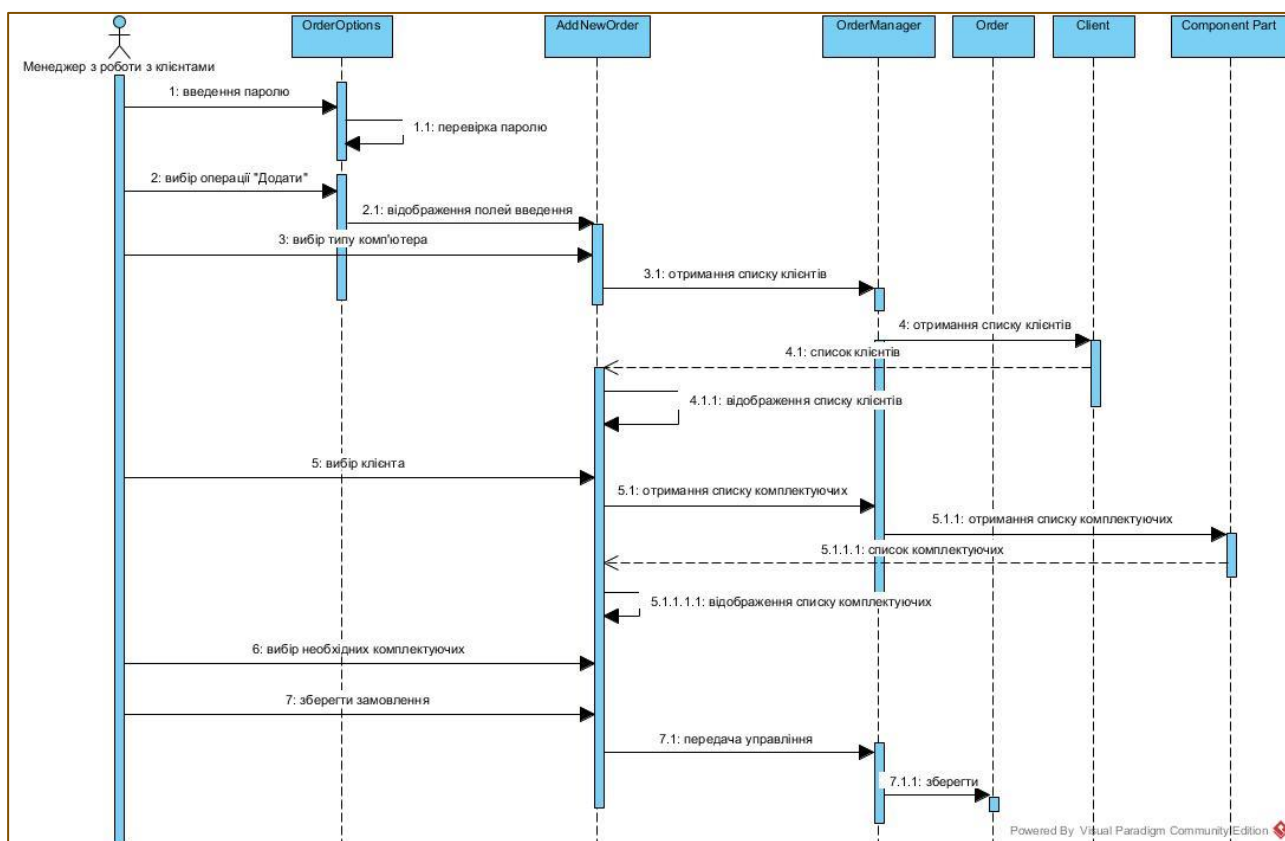
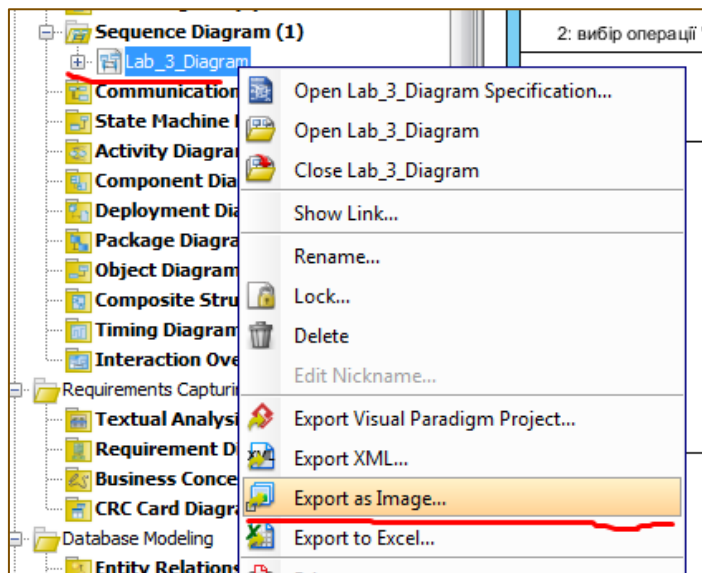


Рис. 7. Підсумкова діаграма послідовності

Для того, щоб сформуванати зображення для звіту до лабораторної роботи, доцільно використати функції експорту, що пропонує Visual Paradigm. Для цього, у навігаторі діаграм обираєте потрібну діаграму, клацаєте правою кнопкою миші й у вікні діалогу обираєте Export as Image.



Діаграма комунікації UML описує статичну структуру об'єктів, що реалізують поведінку підсистеми. Мета комунікації полягає у тому, щоб визначити особливості реалізації прецедентів і найважливіших операцій у системі. Комунікація визначає структуру поведінки системи у термінах взаємодії учасників і може зображатися на рівні специфікацій або на рівні прикладів.

Діаграма комунікації на рівні специфікацій містить класи, асоціації та взаємодії (зображуються пунктирним еліпсом). Такі діаграми комунікації відображають особливості реалізації прецедентів у системі.

Діаграми комунікації на рівні прикладів містять об'єкти (екземпляри класів), зв'язки (екземпляри асоціацій) і повідомлення (зв'язки доповнюються стрілками повідомлень). На цьому рівні ілюструються тільки ті зв'язки й об'єкти, які мають безпосереднє відношення до реалізації певної комунікації.

Для створення діаграми комунікації у середовищі Visual Paradigm, у Diagram Navigator обираєте Communication Diagram та вводите її назву. Побудова діаграми кооперації починається з розміщення на ній об'єктів, які будуть обмінюватися повідомленнями. Перелік об'єктів на даній діаграмі такий же, як і на попередній. Ці елементи можна додати, звернувшись до вже створеної діаграми послідовності (рис. 8). При додаванні класів, слід встановити відповідні стереотипи.

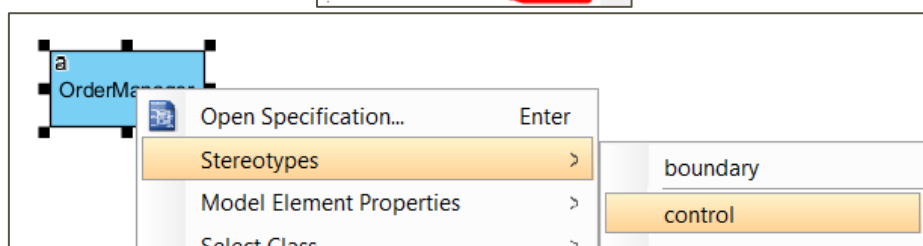
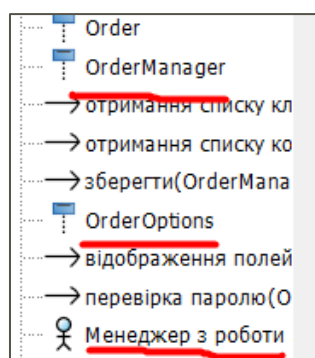


Рис. 8. Додавання елементів на діаграму комунікації

Далі необхідно додати на діаграму зв'язку між об'єктами, які обмінюються повідомленнями і додати повідомлення:

- Менеджер по роботі із клієнтами й AddNewOrder
- Менеджер по роботі із клієнтами й OrderOptions
- AddNewOrder й OrderOptions
- AddNewOrder й OrderManager
- AddNewOrder й Client
- AddNewOrder й ComponentPart
- OrderManager й Client
- OrderManager й ComponentPart
- OrderManager й Order

Спочатку потрібно зв'язати елементи, використовуючи тип зв'язку Link. Для цього, натискаєте ЛКМ на Resource Catalog і, утримуючи мишку, клацаєте по іншому елементу (рис. 10). Далі, обираєте елемент Message і клацаєте по лінії зв'язку.

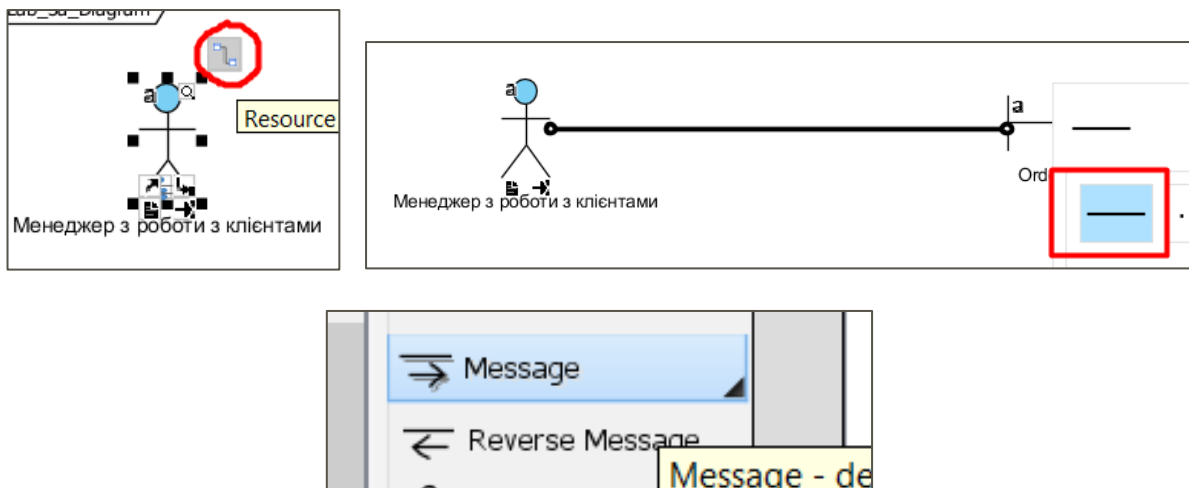


Рис. 10. Створення зв'язку та додавання повідомлення між елементами

У окремих випадках, потрібно створювати елемент, що дозволяє надсилати повідомлення об'єкту від самого себе. Такий тип зв'язку теж є в каталозі ресурсів – Self-message (рис. 11).

Повідомлення додається аналогічно.

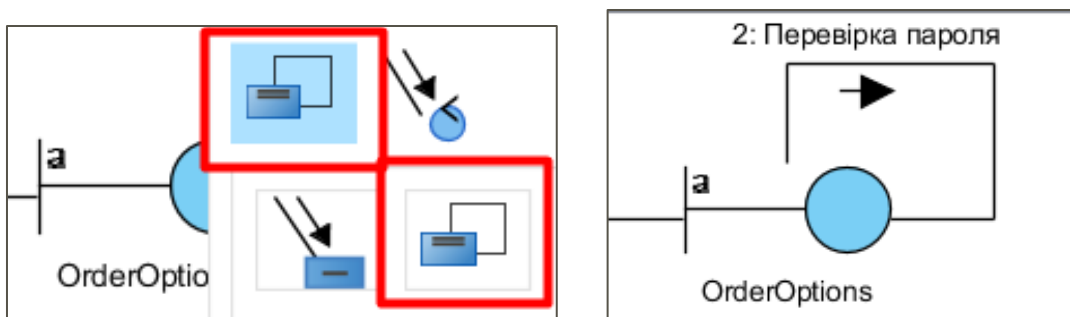


Рис. 11. Надсилання Self-message для елемента OrderOptions

На рис. 12 наведено елемент діаграми комунікації з доданими повідомленнями. Для того, щоб порядок повідомлень відповідав потребам системи, уводьте їх послідовно.

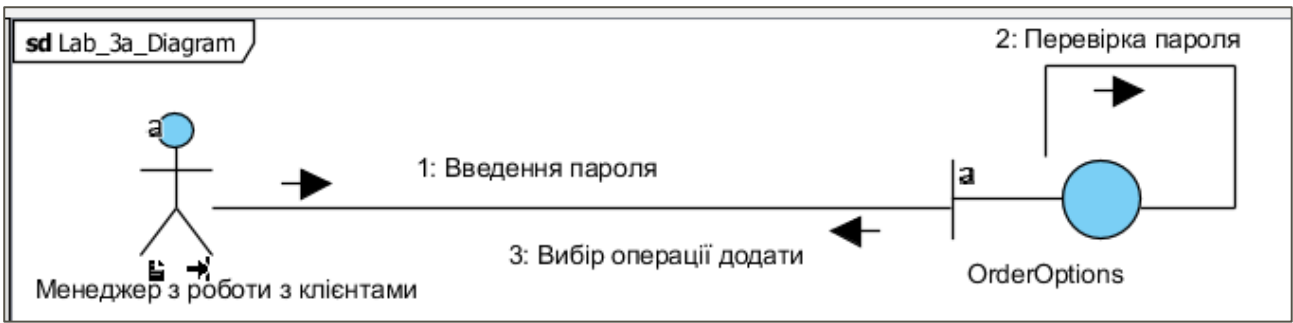


Рис. 12. Елемент діаграми комунікації

Підсумкова діаграма комунікації наведена на рис. 13.

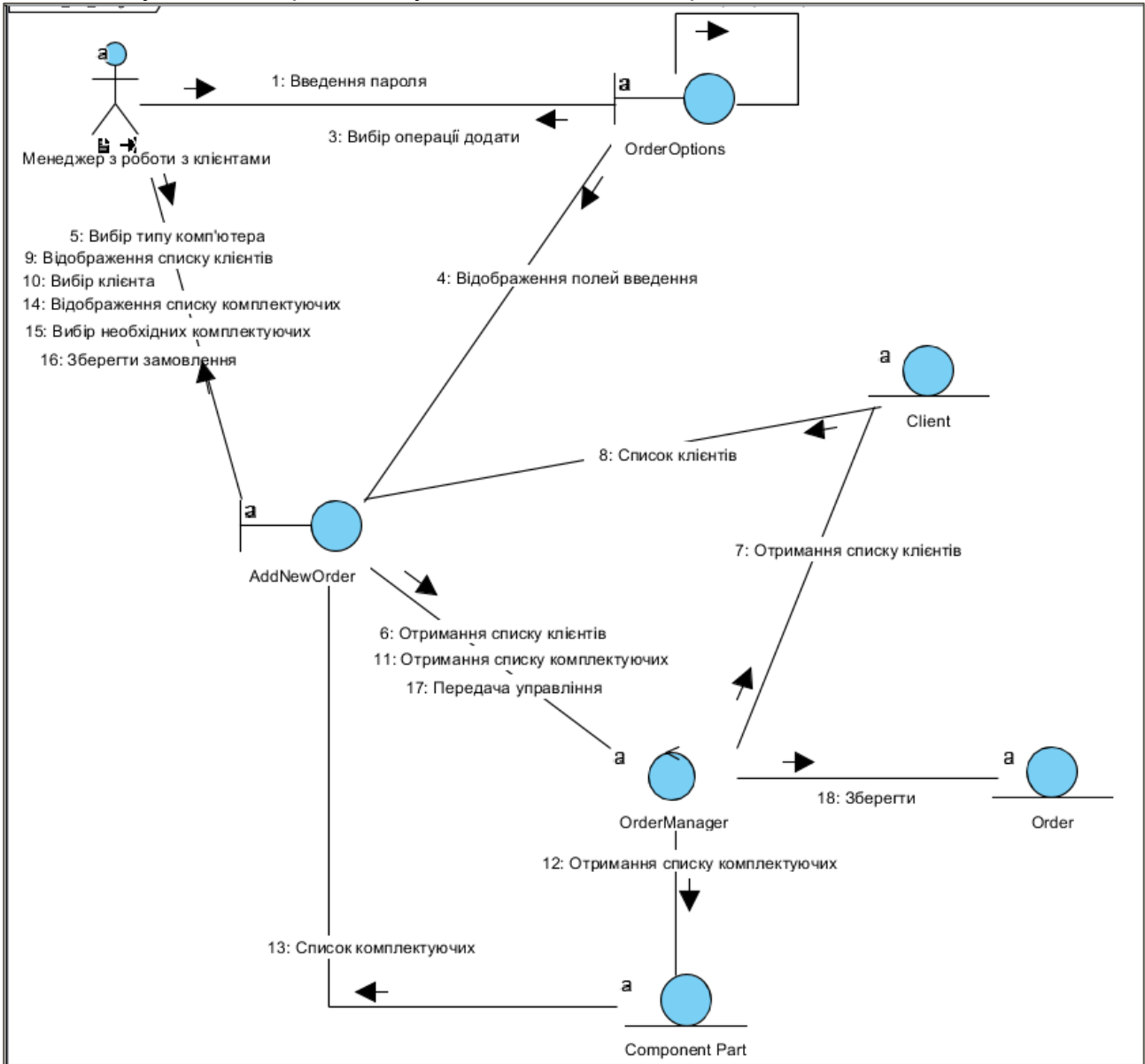


Рис. 13. Підсумкова діаграма комунікації

Контрольні запитання до лабораторної роботи 3

1. Для чого призначена діаграма послідовності?
2. Дайте визначення об'єктів діаграми послідовності, перерахуйте основні з них.
3. Дайте визначення повідомлень діаграми послідовності, перерахуйте основні типи повідомлень.
4. Для чого використовуються прямі, зворотні та рефлексивні повідомлення?
5. Яким чином на діаграмі послідовності використовуються актори?
6. Що таке лінія життя об'єкта? Чим визначається початковий і кінцевий момент періоду життя кожного об'єкта?
7. Для чого призначена діаграма послідовності?
8. Визначте відмінності між діаграмами послідовності та кооперації?

Лабораторна робота 4

Створення діаграми діяльності

Завдання:

1. Створити діаграму діяльності, що описує один з бізнесів-процесів обраної предметної області.
2. Створити діаграму діяльності, що описує потік подій для одного з варіантів використання, створеного в лабораторній роботі № 1.

Теоретичні відомості та хід виконання роботи

Створення діаграми діяльності для бізнесу-процесу підприємства зі зборки комп'ютерів

Розглянемо, в цілому, що відбувається на підприємстві від моменту оформлення замовлення на зборку комп'ютера до видачі готового комп'ютера. Після оформлення замовлення менеджер з роботи з клієнтами передає його менеджеру зі зборки, який, перш ніж почати зборку, замовляє необхідні комплектуючі зі складу.

На складі завідувач підбирає потрібні комплектуючі (у випадку їх відсутності замовляючи їх у менеджера з постачання) і передає їх інженеру зі зборки. Після отримання комплектуючих менеджер зі зборки здійснює зборку комп'ютера й передає його інженерові з тестування. Якщо комп'ютер не пройшов тестування, він повертається для повторної зборки.

При успішному завершенні тестування комп'ютер передається на склад на зберігання. Зі складу комп'ютер за вимогою передається інженеру з роботи з клієнтами, що оформляє на нього документи й видає клієнтові.

Діаграми діяльності UML використовуються для моделювання динамічних аспектів систем, дозволяючи моделювати послідовності бізнес-процесів або дій, реалізованих методами класів. Фактично, діаграми діяльності є аналогом блок-схеми будь-якого алгоритму чи процесу обробки даних. Діаграми діяльності виражаються у вигляді орієнтованого графу, вершини якого є дії, а ребра – переходи між діями.

Діаграми діяльності доцільно використовувати для аналізу:

- Змісту сценаріїв застосування системи, що проектується;
- Взаємодії потоків робіт різних сценаріїв;
- Виконання сценаріїв у багатопроцесорних обчислюваних середовищах.

Діаграма діяльності дозволяє відображати паралельні процеси й розгалуження потоку керування. Також, діаграми діяльності відображають динаміку проекту та візуалізують схеми потоків управління в системі від дії до дії.

Основними елементами діаграм активності є:

Дія (action),

Вузол діяльності (activity node),

Переходи між діями (transition),

Точки прийняття рішення або розгалуження (decision),

Вертикальна/горизонтальна смуги синхронізації (synchronization),

Доріжки (swimlines).

Діяльність (activity) - структурований опис поточної поведінки. Він складається з набору окремих Дій (action), кожна з яких може змінювати стан системи або передавати повідомлення. Назва дії складається з дієслова та декількох пояснюючих слів. Дії не можуть піддаватися декомпозиції. Вони є атомарні (події

можуть відбуватися, але внутрішня поведінка дії невідома). Не можна виконати частину дії: вона або виконується цілком, або не виконується взагалі.

Вузол діяльності (activity node) – це організаційна одиниця діяльності, що є вкладеною групою дій або інших вузлів. Вузли діяльності мають видиму підструктуру й вимагають певного часу на завершення.

Коли деяка дія чи вузол діяльності звершує виконання, потік керування переходить до наступної дії чи вузла діяльності. Потік зображується за допомогою стрілок, що показують його шлях від однієї дії або вузла до іншої.

В UML для позначення використовується проста стрілка, спрямована від попередника до наступника.

Оскільки потік керування має початок і завершення, на діаграмі діяльності використовуються спеціальні позначення для ініціалізації події (початкового стану) і завершення (кінцевого стану).

Окрім послідовних потоків управління в діаграмі можна включати розгалуження, що описують альтернативні шляхи. Розгалуження може мати один вхідний потік і декілька вихідних. На кожному вихідному потоці міститься булевий вираз умови, яка обчислюється на вході з розгалуження.

Умови для вибору потоків не повинні перекриватися, але при цьому повинні враховуватися всі можливі варіанти.

Розгалуження позначаються ромбом.

В UML для опису поділу і з'єднання паралельних потоків керування застосовують лінію синхронізації (synchronization bar). Вона зображується у вигляді жирної горизонтальної або вертикальної лінії.

Поділ (forking) є розщепленням одного потоку керування на паралельні. У цьому процесі присутні один вхідний і ряд незалежних вихідних потоків керування. Після поділу діяльність, асоційована з кожним з них, виконується паралельно.

З'єднання (joining) є синхронізацією кількох паралельних потоків керування. З'єднання може мати ряд вхідних потоків і один вихідний.

При моделюванні бізнес-процесів може виявитися, що на діаграмах діяльності іноді необхідно розбивати стан діяльності на групи, кожна з яких представляє відділ, що займається певною діяльністю. В UML такі групи називаються «плаваючими доріжками» (swimlanes), де кожна група відділяється від сусідніх вертикальною лінією.

Доріжки визначають набори діяльностей, яким притаманна деяка загальна організаційна властивість. Кожній доріжці присвоюється унікальне ім'я.

Для створення діаграми діяльності у Visual Paradigm у Diagram Navigator обираєте Activity Diagram (рис. 1) та вводите назву Lab_4_Diagram_main.

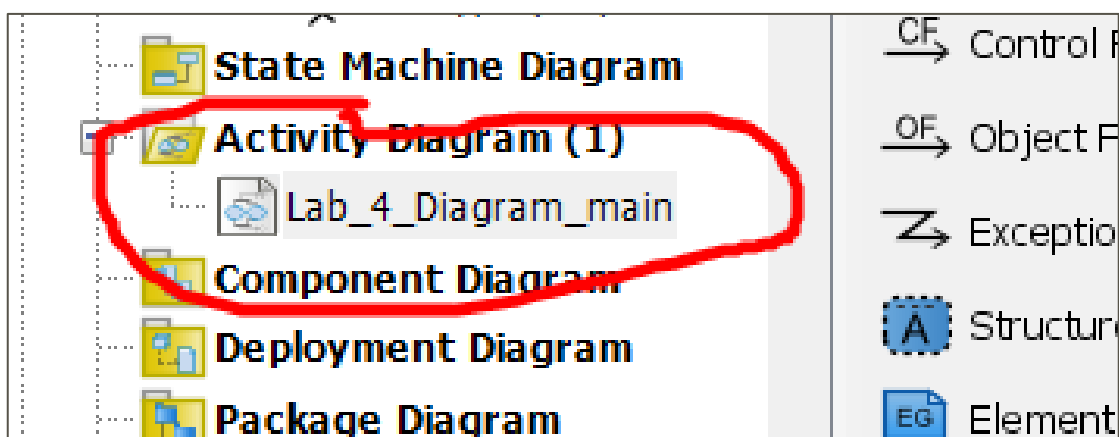


Рис. 1. Створення діаграми діяльності

Обираєте і додаєте елемент Vertical Swimlane, за допомогою якого розподіляються види діяльності між акторами (рис. 2).

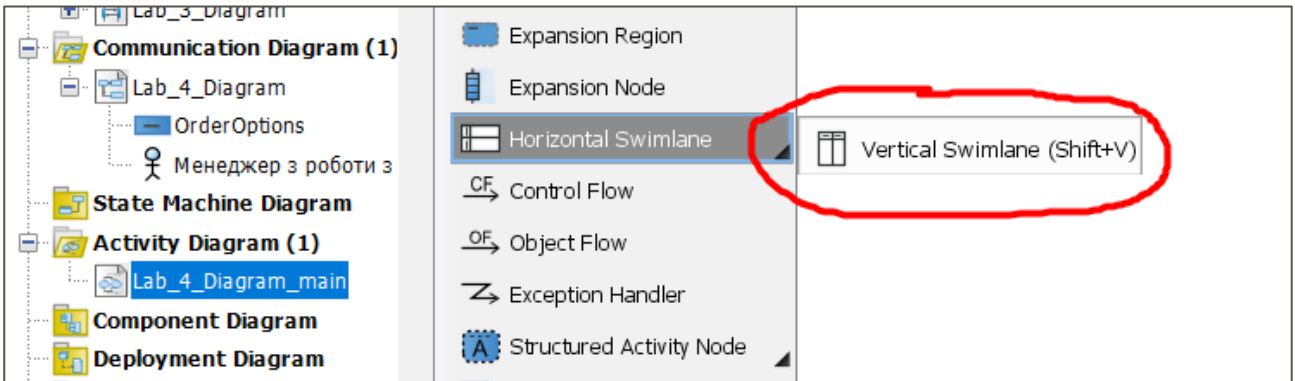


Рис. 2. Вибір елемента Vertical Swimlane

Щоб додати стовпці, клацніть ПКМ і оберіть Add Vertical Partition (рис. 3). Усього має бути чотири стовпці, які потрібно назвати: Менеджер з роботи з клієнтами, Інженер зі збору, Завскладом, Інженер з тестування.

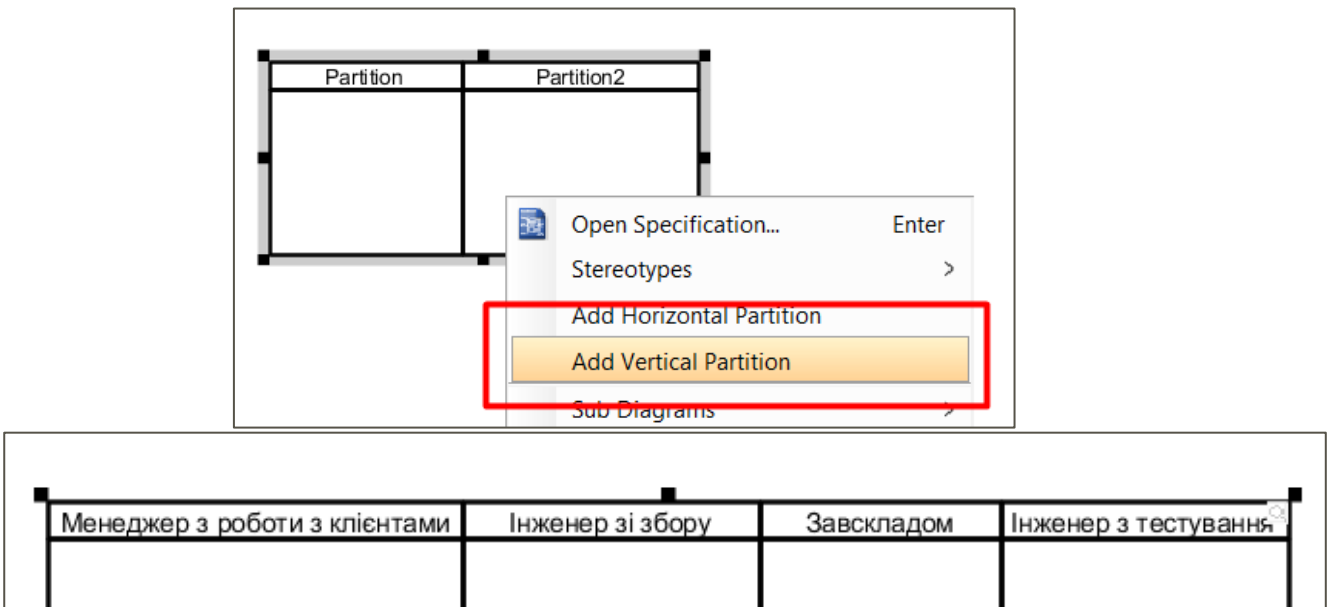
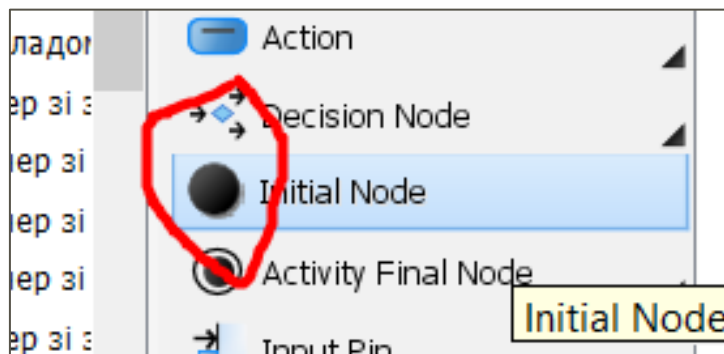


Рис. 3. Стовпці діаграми

Перший елемент діаграми діяльності описує початок процесу – Initial Node. Діяльність розпочинається з дій Менеджера з роботи з клієнтами (рис. 4). Далі, вставляєте елемент Action та перейменовуєте.



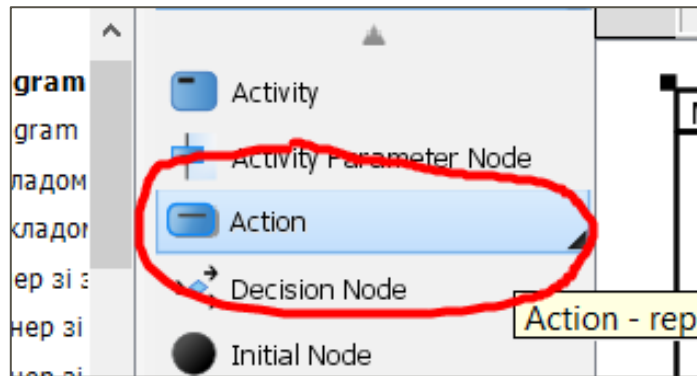


Рис. 4. Основні елементи діаграми

Для того щоб зв'язати елементи ЛКМ натисніть Resource Catalog елемента Initial Node і оберіть тип зв'язку Control Flow (рис. 5).



Рис. 5. Вибір типу зв'язку

На рис. 6 наведено вибір елементів вибору Decision Node та завершення процесу Activity Final Node.

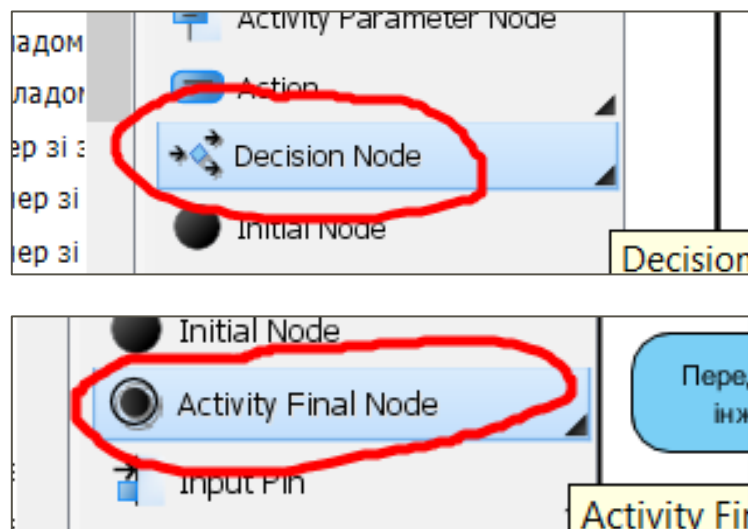


Рис. 6. Елементи Decision Node та Activity Final Node

Результат побудови діаграми показаний на рис. 7.

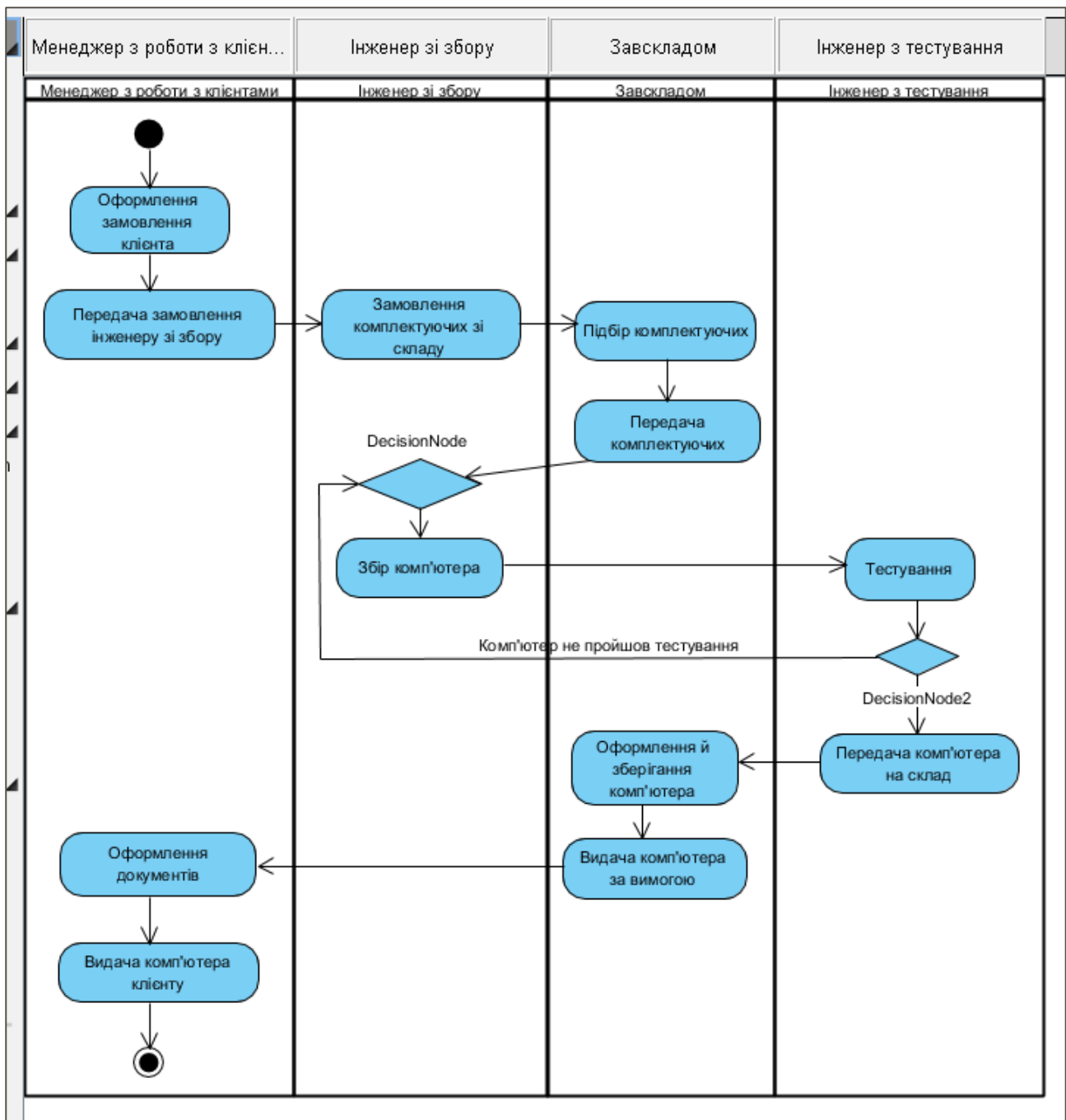


Рис. 7. Діаграма діяльності бізнесу-процесу

Створення діаграми діяльності потоку події для варіанту використання "Робота із замовленням".

Потік подій Варіанту Використання "Робота із замовленням" складається з головного потоку, під-потоків й альтернативних потоків. Щоб не перевантажувати діаграму покажемо потік подій на декількох діаграмах діяльності.

На першій з них (умовно назвемо її головною) покажемо дії для основного потоку й пов'язаний з ним альтернативний потік. Під-потоки можна буде показати шляхом декомпозиції відповідної дії головної діаграми.

Для створення додаткової діаграми діяльності потрібно обрати елемент Робота з замовленням і, клацнувши ПКМ, обрати Sub Diagrams та New Diagram (рис. 8). Далі обираєте тип діаграми та пустий лист для її створення (рис. 9).

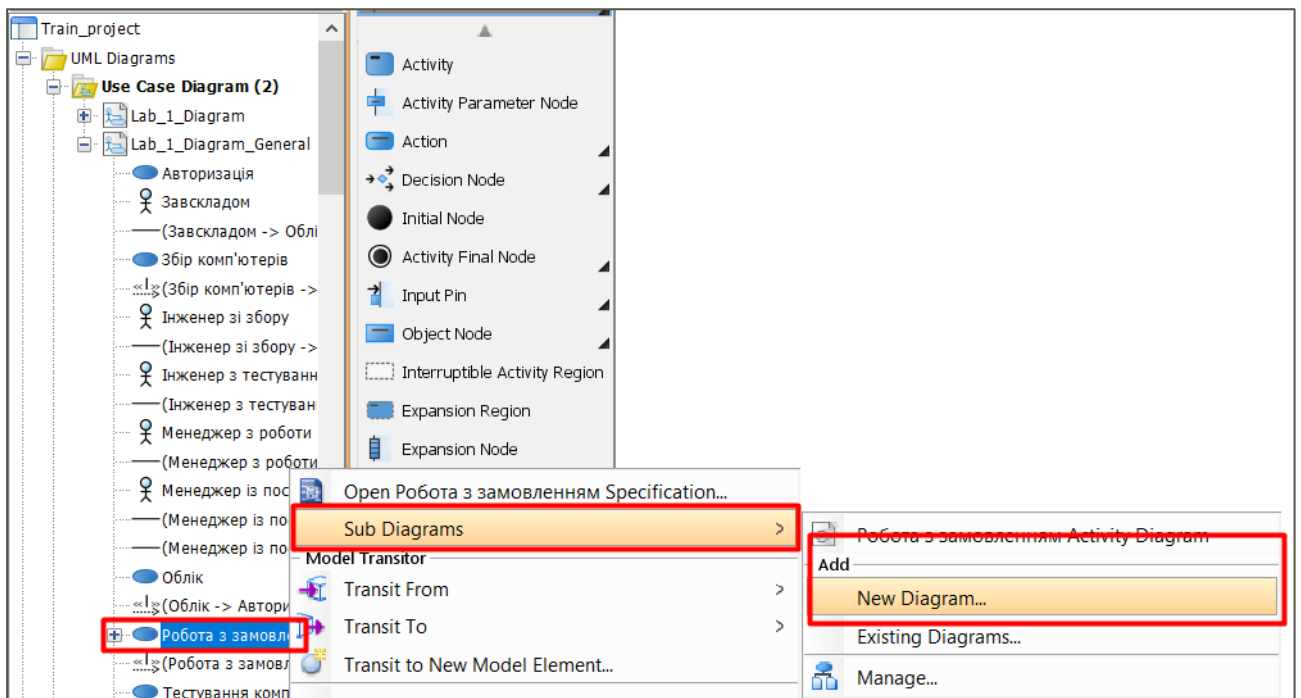


Рис. 8. Створення під-потому

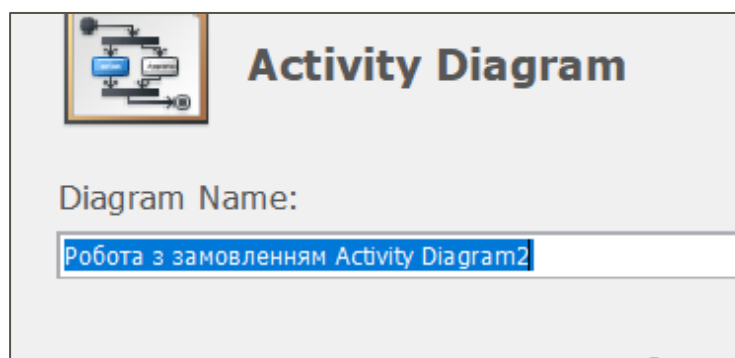
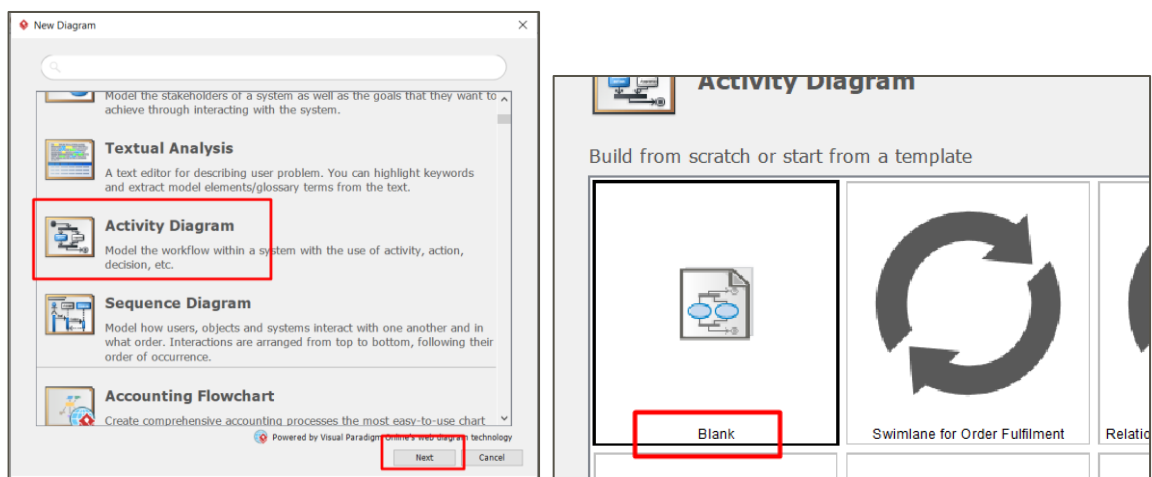


Рис. 9. Створення додаткової діаграми діяльності

На рис. 10 показана діаграма діяльності для потоку подій прецеденту "Робота із замовленням".

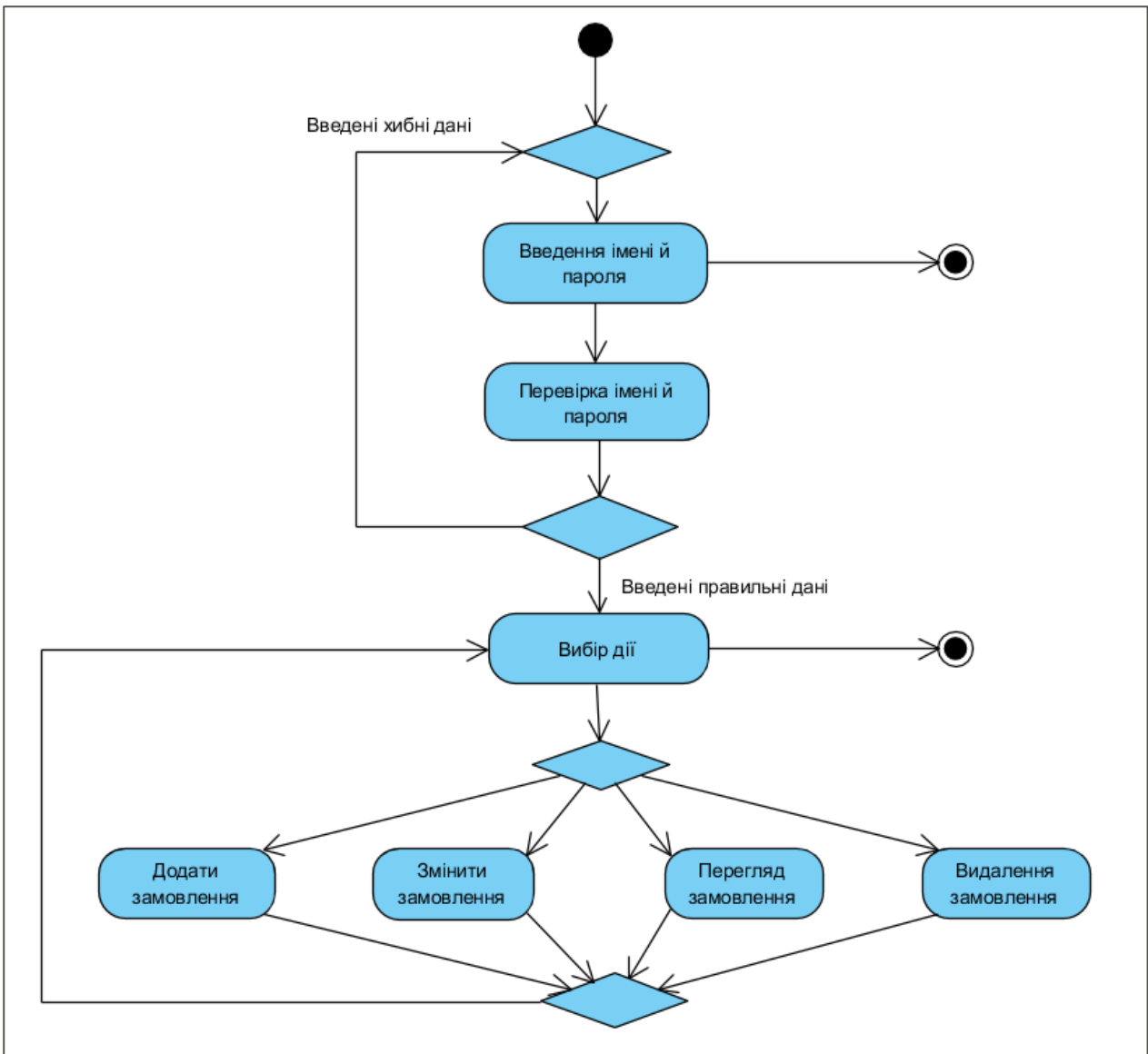


Рис. 10. Діаграма діяльності для потоку подій прецеденту "Робота із замовленням"

Аналогічно, створюєте додаткову діаграму діяльності для елемента Додати замовлення (рис. 11).

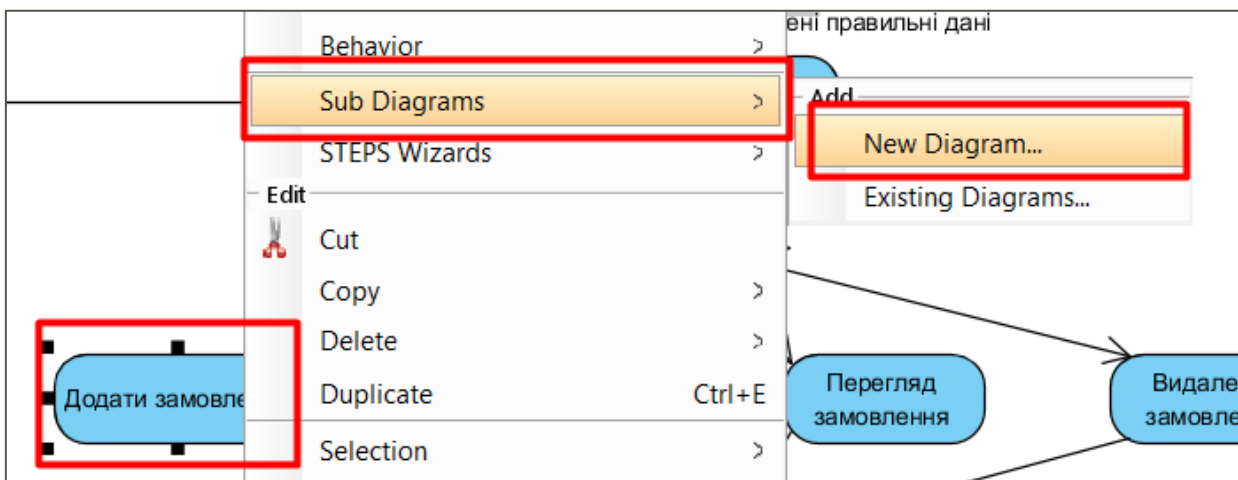


Рис. 11. Створення додаткової діаграми для під-потоку

На рис. 12 показана діаграма діяльності для під-поточку "Додати замовлення", що є декомпозицією дії "Додати замовлення" головної діаграми діяльності.

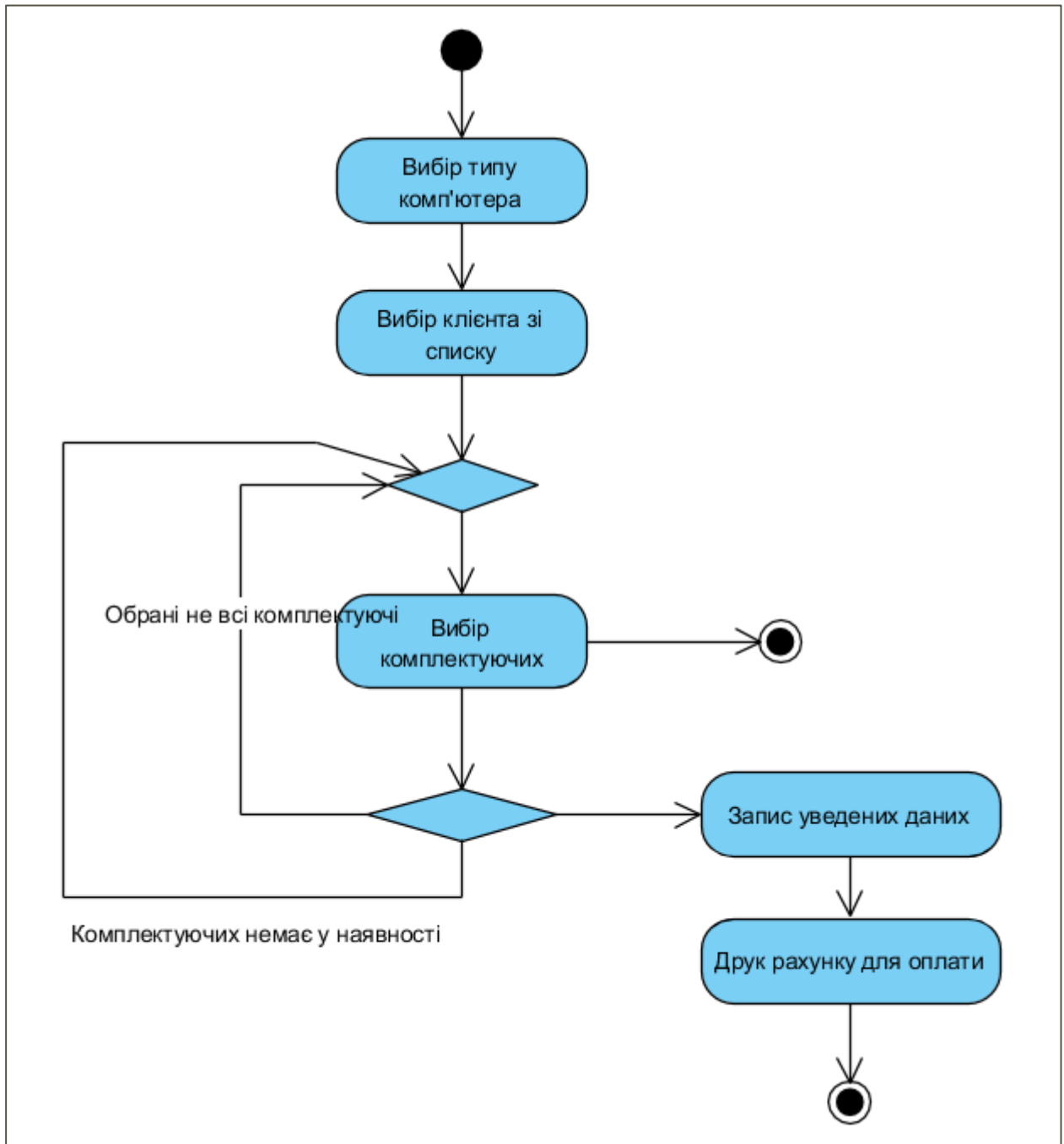


Рис. 12. Діаграма діяльності для дії "Додати замовлення"

Контрольні запитання до лабораторної роботи 4

1. У яких випадках використовується діаграма діяльності?
2. Як позначаються стани дій на діаграмі діяльності?
3. Для чого використовуються стани дій на діаграмі діяльності?
4. Як позначаються переходи на діаграмі діяльності?
5. Для чого використовуються переходи на діаграмі діяльності?
6. Для чого використовуються доріжки на діаграмі діяльності?

Лабораторна робота 5

Створення діаграми станів

Завдання:

1. Створити діаграму станів для одного з раніше розроблених класів або прецедентів.

Теоретичні відомості та хід виконання роботи

Діаграма станів (statechart diagram) призначена для опису можливих послідовностей станів і переходів, які в сукупності характеризують поведінку системи, що моделюється протягом усього життєвого циклу.

Діаграма станів представляє динамічну поведінку сутностей, на основі визначення й опису їх реакції й сприйняття деяких конкретних подій. Системи, які реагують на зовнішні дії інших систем або користувачів, називаються реактивними. Якщо такі дії ініціюються в довільні випадкові моменти часу, то говорять про асинхронну поведінку моделі.

Діаграми станів застосовуються, як правило, для моделювання поведінки класів, прецедентів або системи в цілому. Діаграма станів описує процес зміни станів тільки одного класу, а точніше – одного екземпляра визначеного класу, тобто моделює всі можливі зміни в стані конкретного об'єкта. При цьому зміна стану об'єкта може бути викликана впливами з боку інших об'єктів або зовнішніми впливами. Саме для опису реакції об'єкта на подібні зовнішні впливи і використовуються діаграми станів.

Стан (state) – умова або ситуація в ході життєвого циклу об'єкта, протягом якої він задовольняє логічній умові, виконує певну діяльність або очікує події. Стан може бути заданий у вигляді набору конкретних значень атрибутів класу або об'єкту, при цьому зміна їх окремих значень відобразатиме зміну стану модельованого класу або об'єкту.

Стан на діаграмі зображується прямокутником із заокругленими вершинами. Цей прямокутник може бути розділений на дві секції горизонтальною лінією.

Одна секція – ім'я стану

Дві секції – ім'я стану та список певних внутрішніх дій або переходів у даному стані.

Ім'я стану є рядком тексту, який розкриває змістовний сенс даного стану. Ім'я завжди починається із великої букви. Оскільки стан системи є складовою частиною процесу її функціонування, рекомендується як ім'я використовувати дієслова в теперішньому часі (дзвенить, друкує, чекає) або відповідні дієприкметники (зайнятий, вільний, передано, отримано).

Ім'я стану може бути відсутнім. У цьому випадку стан є анонімним (якщо на діаграмі станів їх декілька, то всі вони повинні розрізнятися між собою).

Внутрішня дія – певна атомарна (неподільна) операція, виконання якої приводить до зміни стану або поверненню деякого значення. Дія може бути реалізована за допомогою передачі повідомлення об'єкту, модифікацією зв'язку або значення атрибута.

Перелік міток дій:

- Вхідна діяльність (entry action) – дія, що виконується в момент переходу в даний стан. Ключове слово – entry.
- Вихідна діяльність (exit action) – дія, що виконується при виході з даного стану. Ключове слово – exit.

- Внутрішня діяльність (do activity) – виконання операцій або процедур, які вимагають певного часу. Ключове слово – do.

Псевдостан (pseudo-state) – вершина в кінцевому автоматі, що має форму стану, але не має поведінки.

Початковий стан (start state) – різновид псевдостану, що позначає початок виконання процесу зміни станів кінцевого автомату або знаходження об'єкта, що моделюється, в складеному стані.

Кінцевий стан (final state) – різновид псевдостану, що позначає припинення процесу зміни станів кінцевого автомату або знаходження об'єкта, що моделюється, в складеному стані.

Перехід (transition) – відношення між двома станами, яке вказує на те, що об'єкт у першому стані повинен виконати певні дії і перейти в другий стан.

Простим переходом (simple transition) є відношення між двома послідовними станами, яке вказує на факт зміни одного стану іншим. Перебування модельованого об'єкта в першому стані може супроводжуватися виконанням деяких дій, а перехід в другий стан буде можливий після завершення цих дій, а також після задоволення деяких додаткових умов. У цьому випадку кажуть, що перехід спрацьовує, або відбувається спрацювання переходу.

Перехід здійснюється при настанні деякої події: закінчення виконання діяльності (do activity), отриманні об'єктом повідомлення або прийомом сигналу. На переході вказується ім'я події.

На діаграмі станів перехід зображується суцільною лінією із стрілкою, яка направлена в цільовий стан.

Спрацювання переходу (fire) – виконання переходу з одного стану в інший. Спрацювання переходу може залежати не лише від настання деякої події, але і від виконання певної умови, званої сторожовою умовою

Сторожова умова (guard condition) - логічна умова, яка записується в дужках і є булевим виразом. Якщо сторожова умова набуває значення «істинно», то перехід спрацьовує й об'єкт переходить у цільовий стан. Якщо умова набуває значення «хибно», то перехід є неможливим і об'єкт залишається у початковому стані.

Складемо діаграму станів для класу *Order* (*Замовлення*), оскільки в нашій моделі він найчастіше буде змінювати свій стан. Замовлення може перебувати в декількох станах:

- при створенні замовлення він переходить у стан *Ініціалізація*, у якому виконуються деякі попередні дії;
- після завершення ініціалізації замовлення переходить у стан *Відкритий*, у якому до замовлення додаються нові пункти. Вихід із цього стану можливий або у випадку скасування замовлення, або у випадку заповнення всіх необхідних пунктів замовлення;
- якщо заповнені всі необхідні пункти замовлення, то він переходить у стан *Закритий*, у якому відбувається виписка рахунку. Вихід із цього стану відбудеться тільки після того, як рахунок буде виписаний;
- якщо замовлення скасоване, то зі стану *Відкритий* він переходить у стан *Скасований*. При виході із цього стану відбувається видалення всіх пунктів замовлення.

Для створення діаграми станів у Visual Paradigm клацніть ПКМ по елементу State Machine Diagram у Diagram Navigator й оберіть створити New State Machine Diagram (рис. 1).

Переименуйте створений файл. На створеній діаграмі відразу буде встановлений елемент Initial Pseudo State.

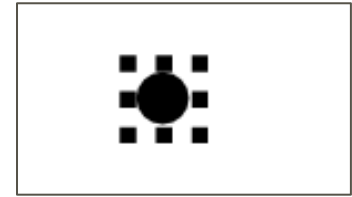
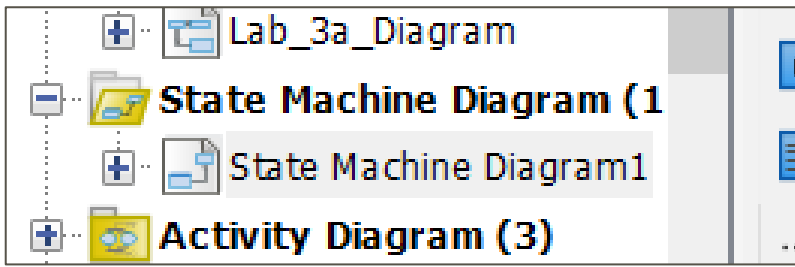


Рис. 1. Створення діаграми станів

Іншим елементом діаграми є стан – State. Стан – це якесь положення в житті об'єкта, при якому він задовольняє визначеній умові, виконує деяку дію або очікує події. Поточний стан об'єкта можна описати сукупністю поточних значень одного або декількох атрибутів класу та зв'язків об'єкта .

Перетягніть елемент State на поле діаграми та уведіть назву (рис. 2).

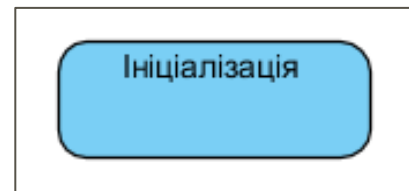
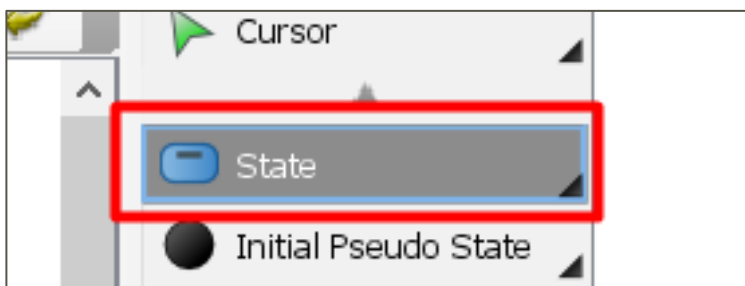


Рис. 2. Створення елемента State

Далі потрібно додати список деяких внутрішніх дій або переходів для даного стану. Клацніть по елементі ПКМ і оберіть опцію Open Specification (рис. 3).

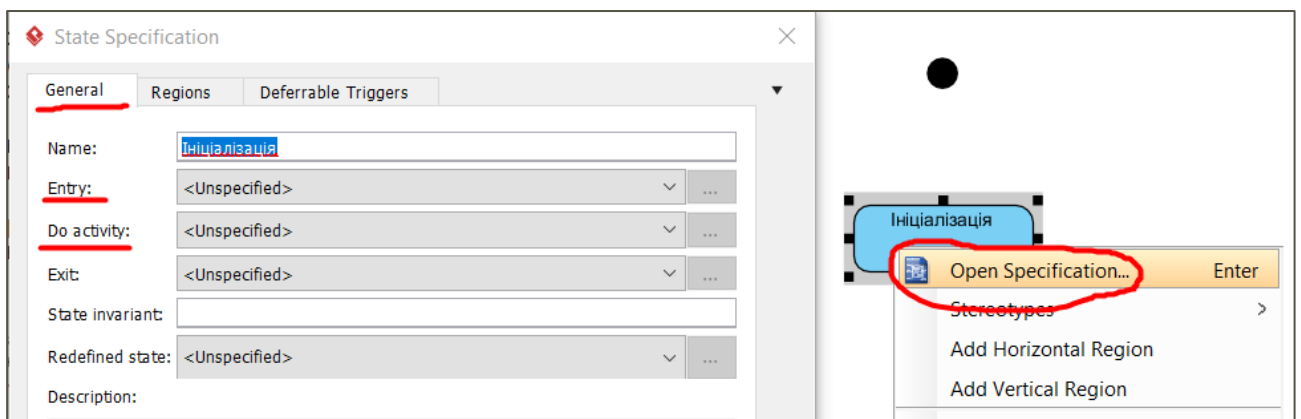
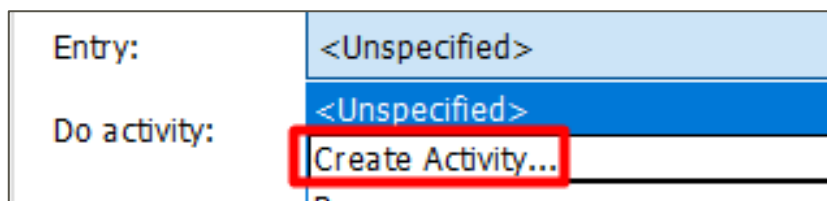


Рис. 3. Відкриття вікна специфікації стану

Для стану Ініціалізація створюється дві активності. Одна типу Entry , а інша Do. Створюєте їх та вводите відповідні назви (рис. 4).



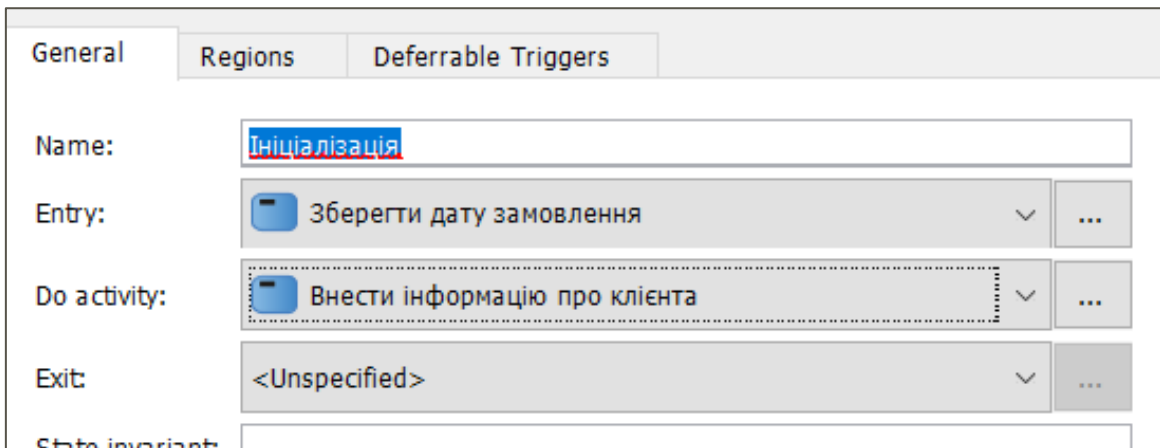
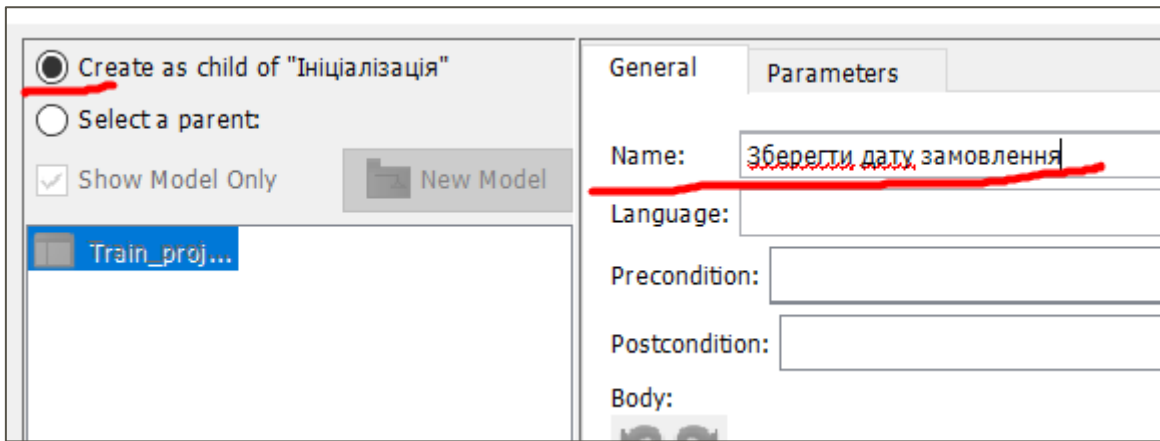


Рис. 4. Створення деталізованого опису стану

Після цього, початок процесу потрібно зв'язати зі станом Ініціалізація. Тип зв'язку – Transition (рис. 5). Для одного із станів (Відкрите), потрібно буде використати зв'язок типу Self-Transition, який доступний через Resource Catalog (рис. 6).

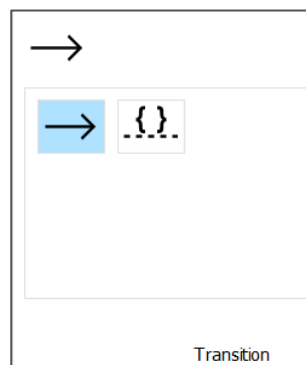
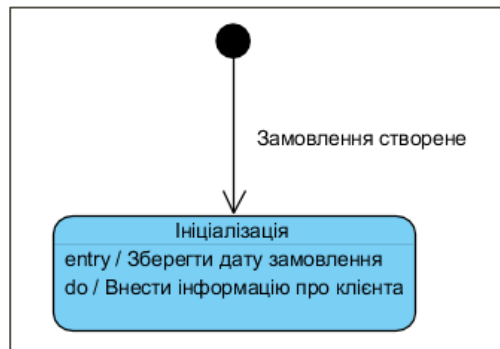


Рис. 5. Встановлення зв'язків між станами

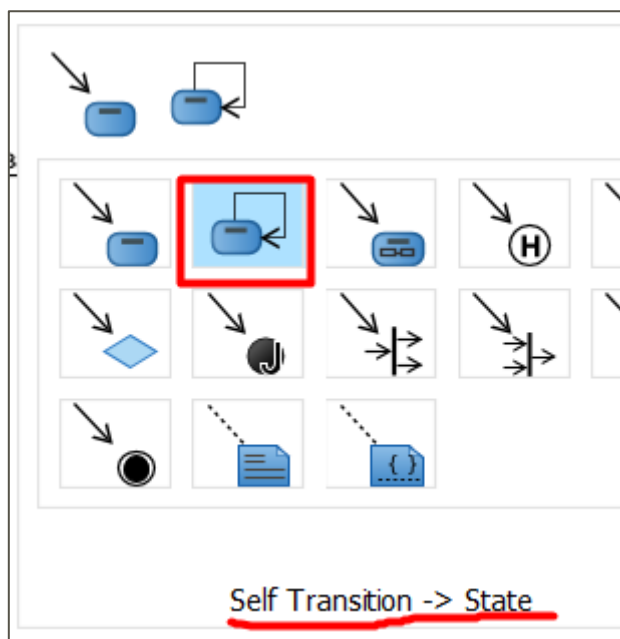


Рис. 6. Зв'язок типу Self-Transition

Діаграма станів для класу *Order* представлена на рис. 7.

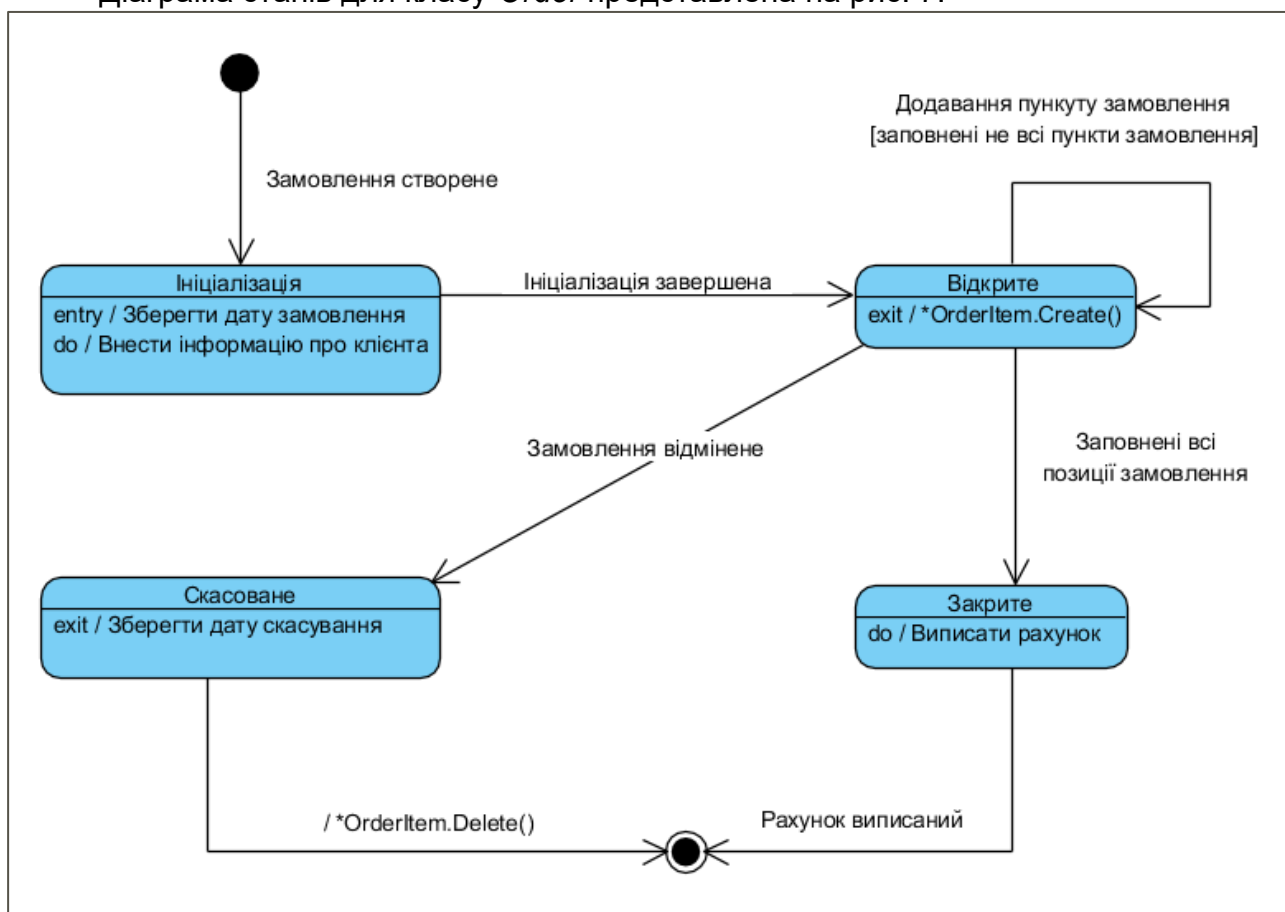


Рис. 7. Діаграма станів для класу *Order*

Першим станом на діаграмі станів є початковий стан. При виконанні події "Замовлення створене" замовлення переходить у стан *Ініціалізація*. При вході в цей стан виконується вхідна дія "Зберегти дату замовлення". Основна дія, що буде виконуватися протягом всього часу, поки замовлення буде перебуває в цьому стані,

це *"Внести інформацію про клієнта"*. Перехід із цього стану в стан *Відкрите* відбудеться тільки при виконанні умови *"Ініціалізація завершена"*.

Для стану *Відкрите* є вихідна дія й перехід у себе. Перехід у себе означає, що подія ініціює перехід, відбувається вихід з поточного стану, виконується деяка дія, після чого відбувається повернення у вихідний стан. Оскільки при переході в себе відбувається вихід зі стану й повторний вхід у нього ж, то виконується дія, асоційована з переходом, і, крім того, дія при вході в стан.

У стані *Відкритий* до замовлення додаються нові пункти, причому їх можна додати тільки в тому випадку, якщо є незаповнені пункти. Для показу цього ми використали перехід у себе *"Додавання пункту замовлення"* з умовою *"заповнені не всі пункти замовлення"*. Вихід із цього стану відбудеться у двох випадках - або коли виконається умова *"заповнені всі позиції замовлення"* (при цьому замовлення перейде в стан *Закрите*), або коли наступить подія *"замовлення скасоване"* (при цьому замовлення перейде в стан *Скасоване*). При виході зі стану виконається дія виходу *"* OrderItem.Create()"* (створення пункту замовлення). Символ "*" указує на те, що ця дія виконається багато разів (по числу доданих пунктів у замовлення).

У стані *Закрите* наявна тільки внутрішня дія - *"Виписати рахунок"*. У цей стан замовлення переходить зі стану *Відкрите* тільки при виконанні умови *"заповнені всі позиції замовлення"*. Вихід із цього стану й перехід у кінцевий відбудеться при настанні події *"рахунок виписаний"*.

У стан *Скасоване* замовлення переходить зі стану *Відкрите* при настанні події *"замовлення скасоване"*. При виході з нього виконується дія виходу *"Зберегти дату скасування"*. При переході із цього стану до кінцевого виконується дія *"*OrderItem.Delete()"* (видалення пункту замовлення). Тут також використовується знак "*", оскільки ця дія буде виконуватися багато разів.

Контрольні запитання до лабораторної роботи 5

1. Що відображає діаграма станів?
2. Що таке стан об'єкта? Як його зображають?
3. Що таке анонімний стан об'єкта?
4. Що таке початковий стан об'єкта? Як його зображають?
5. Що таке кінцевий стан об'єкта? Як його зображають?
6. Які події використовують для характеристики поведінки об'єкта у деякому стані?
7. Як активізують просту внутрішню дію для стану об'єкта?
8. Як активізують долучення події зовнішнього об'єкта для даного стану об'єкта?

Список використаних джерел

1. Кватрани Т. Rational Rose 2000 и UML. Визуальное моделирование: Пер. с англ. – М.: ДМК Пресс, 2001. – 176 с.
2. Леоненков А. В. Самоучитель UML. – Санкт-Петербург: БХВ-Петербург, 2006. – 432 с.
3. Дудзяний І. М. Об'єктно-орієнтоване моделювання програмних систем: навчальний посібник. – Львів: Видавничий центр ЛНУ імені Івана Франка, 2007. – 108 с.
4. Принципи проектування відкритих розподілених систем: лабораторний практикум. / Укладачі І.Е. Райчев, О.Г. Харченко. – Київ: Національний авіаційний університет, 2007. – 65 с.
5. Табунчик Г.В., Кудерметов Р.К., Брагіна Т.І. Інженерія якості програмного забезпечення: навчальний посібник. – Запоріжжя: ЗНТУ, 2013. – 180 с.
6. Бевз О.М., Папінов В.М., Скидан Ю.А. Проектування програмних засобів систем управління. Частина 1. Основи об'єктно-орієнтованого проектування: навчальний посібник. – Вінниця: ВНТУ, 2010. – 125 с.
7. Лаврищева К.М. Програмна інженерія. – Київ: Академперіодика, 2008. – 319 с.
8. Методичні вказівки до виконання лабораторних робіт з дисципліни «Планування та управління ГІМ-проектами». / Упорядники: Лагоднюк О.А., Прокопчук А.В., Ревуцький В.Р. – Рівне: НУВГП, 2012. – 30 с.
9. Visual Paradigm Official Site, [Електронний ресурс]. Режим доступу: <https://www.visual-paradigm.com/>.
10. Visual Paradigm User's Guide. Part II. UML modeling. https://www.visual-paradigm.com/support/documents/vpuserguide/94_umlmodeling.html

Підписано до друку 12.12.2019 р.
Формат 60×84 1/16. Папір офсетний.
Умов друк. арк. 2,6. Зам. № 1358.
Тираж 100 прим.

Видавець: Чабаненко Ю.А.
Свідоцтво про внесення
до Державного реєстру видавців
серія ДК № 1898 від 11.08.2004 р.
Україна, м. Черкаси, вул. О. Дашковича, 39
Тел: 0472/56-46-66; (093) 788 99 99
E-mail: office@2post.com

Друк ФОП Чабаненко Ю.А.
Україна, м. Черкаси, вул. О. Дашковича, 39
Тел: 0472/56-46-66; (093) 788 99 99
E-mail: office@2post.com