ПРАКТИЧНЕ ЗАНЯТТЯ № 04

ТЕХНОЛОГІЯ СТВОРЕННЯ ПРОГРАМНИХ ПРОДУКТІВ

Тема: **ОБ'ЄКТНО-ОРІЄНТОВАНИЙ АНАЛІЗ ТА ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ: ДІАГРАМА ПОСЛІДОВНОСТІ, ДІАГРАМА КООПЕРАЦІЇ**

Мета: дослідження предметної області, вивчення процесу побудови діаграми послідовності та кооперації за допомогою інструмента об'єктного моделювання StarUML.

Програмне забезпечення: ОС Windows/Linux, StarUML (https://staruml.io/).

Теоретичний базис:

1. Діаграма послідовності

Діаграма послідовності (Sequence Diagram) — показує часові особливості передачі і прийому повідомлень об'єктами. Впорядкованість за часом слід розуміти як послідовність дій і не плутати з часовими діаграмами.

Позначення діаграми послідовності

1. Лінія життя починається з об'єкта-прямокутника (голова) та зображується вертикальною пунктирною лінією (стеблом). Вона служить для позначення періоду часу, протягом якого об'єкт існує в системі. Якщо об'єкт існує в системі постійно, то його лінія життя повинна продовжуватися по всій площині діаграми зверху донизу. Зазвичай об'єкти перераховуються зліва направо.

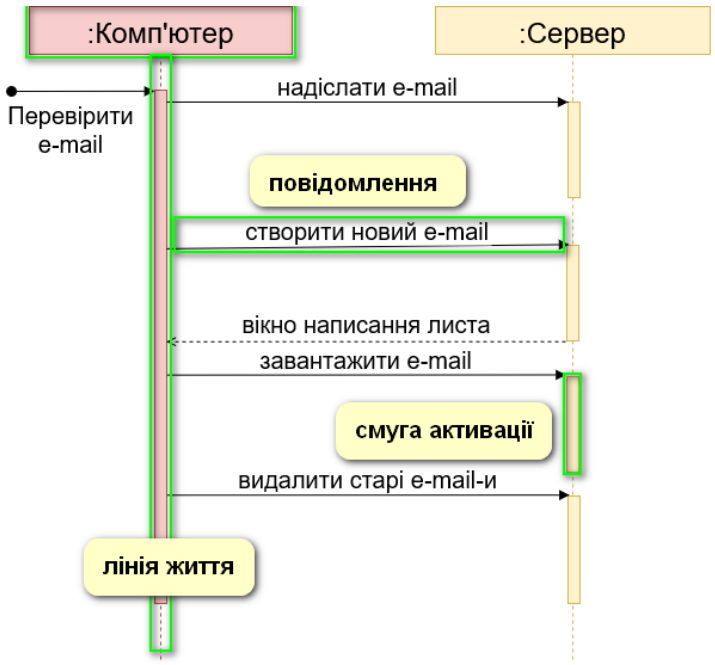

Рис. Загальні елементи діаграми послідовності

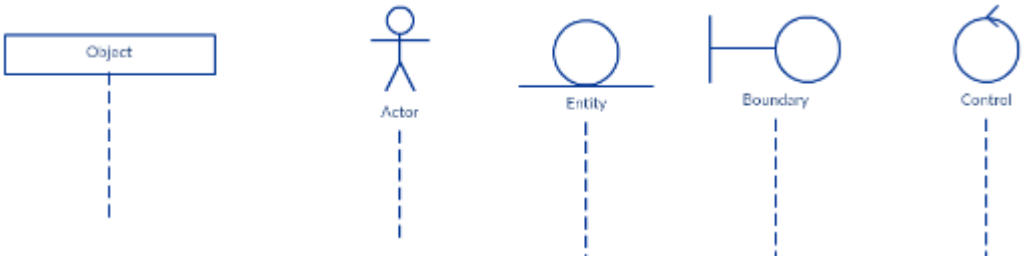

Рис. Позначення на лінії життя

Об’єкт (учасник) — позначення лінії життя та екземпляр класу у горизонтального прямокутника.

Актор — використовується, коли конкретна діаграма послідовності належить варіанту використання.

Сутність (entity) — представляє системні дані. Наприклад, у програмі обслуговування клієнтів суб’єкт — клієнт керує даними (сутність), пов’язаними з клієнтом.

Межа/кордон (boundary) — вказує на межу системи/ граничний елемент у системі; наприклад, екрани інтерфейсу користувача, шлюзи баз даних або меню, з якими взаємодіють користувачі

Управління (control) вказує на керівну сутність або менеджера. Він організовує та планує взаємодії між кордонами та сутностями та служить посередником між ними.

2. Смуга активації (фокус управління) — тонкий прямокутник на лінії життя, протягом якого елемент виконує операцію. Довжина прямокутника вказує на тривалість перебування об’єктів в активному режимі.

3. Повідомлення (виклики) з’являються в послідовному порядку на лінії життя. Повідомлення зображується за допомогою стрілок. Початок стрілки завжди торкається лінії життя відправника та лінії життя об’єкта, що приймає повідомлення. Підпис може знаходитись над або всередині стрілки повідомлення. Для зручності перед повідомленням можна проставляти нумерацію дій. Повідомлення можна розділити на такі категорії:

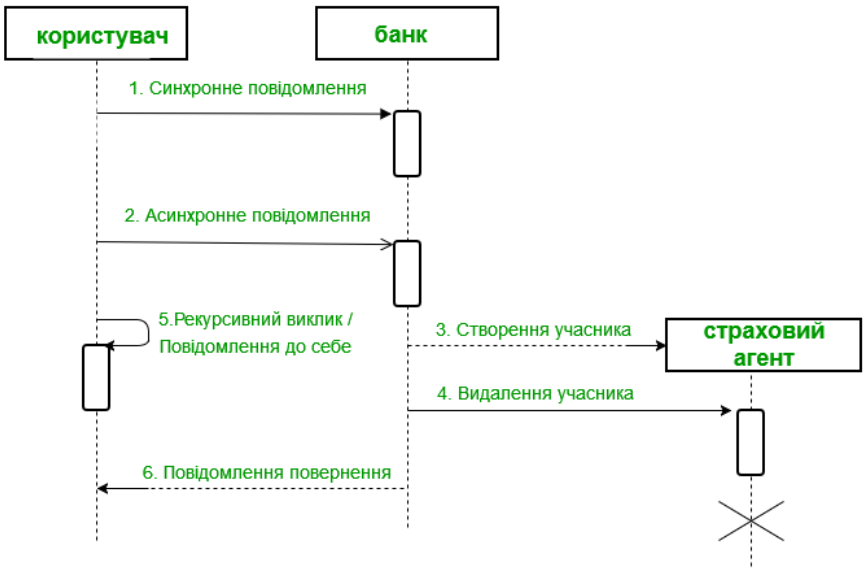

Рис. Позначення типів повідомлень

Повідомлення поділяються на *синхронні*, що очікують відповіді (зафарбований вказівник), та *асинхронні* — не очікують відповіді (пунктирна лінія, звичайний вказівник).

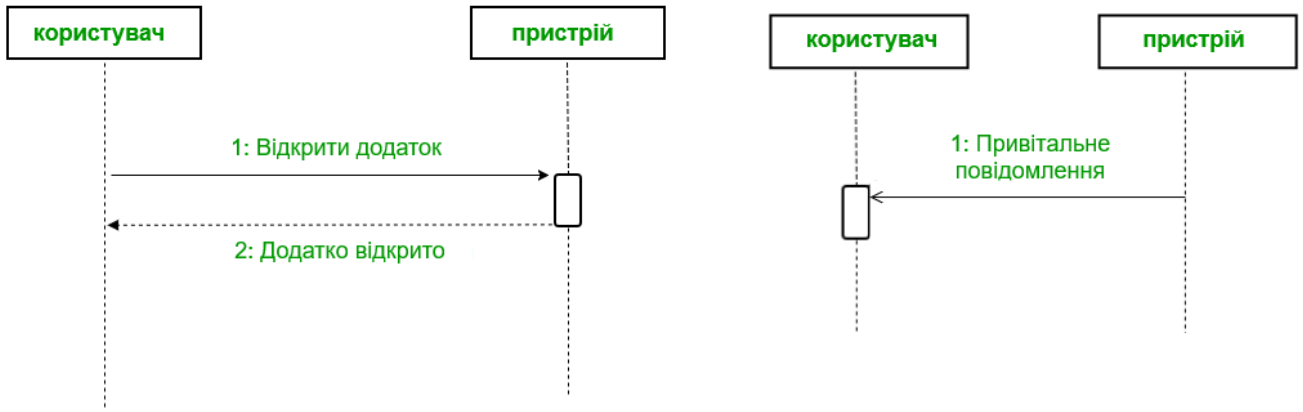

Рис. Приклад синхронного та асинхронного повідомлення

Рекурсивне повідомлення — направлене до себе (починається та закінчується на одній лінії життя). Прикладом може служити отримання доступу до камери смартфоном.

Повідомлення *повернення* (пунктирна лінія із зафарбованим вказівником) — зворотне повідомлення викликаючій стороні. Можна уникнути захаращення діаграм, вказуючи значення в самій стрілці початкового повідомлення.

Створення учасника — повідомлення, що створює нову лінію життя. Позначається пунктирною лінією, направленою до прямокутника учасника. Якщо створений учасник робить щось після його створення, слід додати вікно активації під його полем.

Видалення учасника позначається хрестом в кінці лінії життя учасника. При цьому, якщо стрілка повідомлення направлена від одного учасника до іншого, значить, один учасник видаляє іншого. Якщо стрілка відсутня, учасник самознищується.

Знайдене повідомлення — повідомлення від невідомого джерела.

4. Інші позначення:

- Коментар — прямокутник із загнутим кутом. Може бути з'єднаний з об'єктом пунктирною лінією.
- Фрагмент використовується, якщо процеси утворюють цикл або вимагають виконання умов для його закінчення. Він складається з вікна (рамки) та оператора фрагмента (напису) в п'ятикутнику зверху зліва.

Фрагмент діаграми послідовності (sd) використовується для оточення усієї діаграми. Після напису sd вказується назва діаграми.

Альтернативний фрагмент (alt) — показує один або декілька альтернативних сценаріїв, де виконується лише один, той, чия умова істинна.

Для опису двох або більше альтернативних сценаріїв використовуються пунктирні лінії — операнд взаємодії (interaction operands). Кожен операнд має умову захисту в квадратних дужках (guard condition).

Опціональний (не обов’язковий) фрагмент (opt) — виконується, лише якщо вказана умова, істинна. Він має лише одну умову і не поділяється на операнди. Використовується для опису не обов’язкового кроку робочого процесу.

Прикладом може слугувати послідовність покупок в інтернет-магазині: opt, щоб описати, як користувач може додати подарункове пакування. Та alt для опису двох варіантів оплати: за допомогою кредитної картки або грошового переказу.

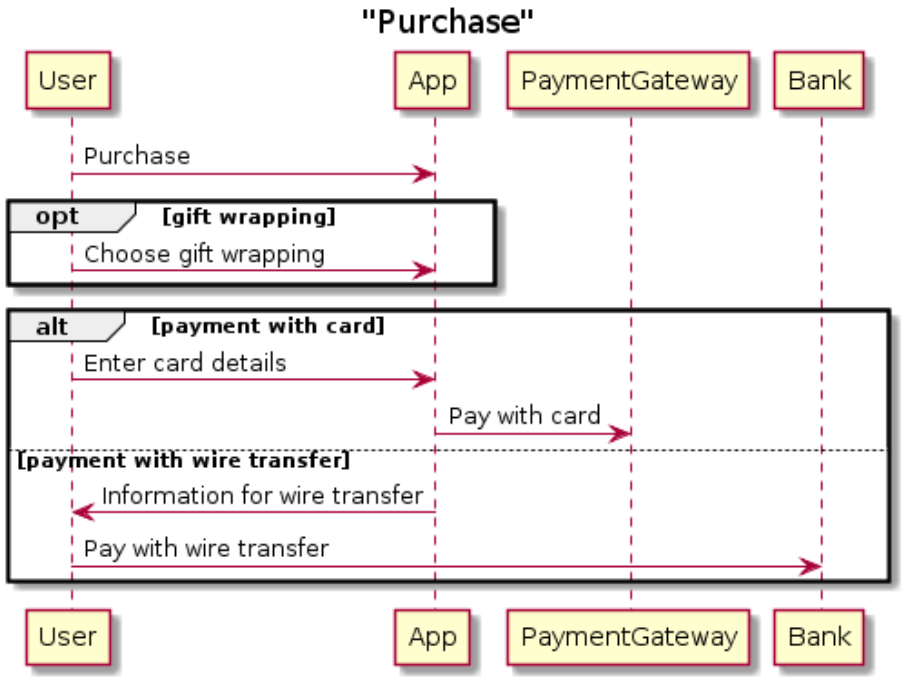

Рис. Приклад фрагментів alt та opt

Фрагмент циклу (loop) — використовується для послідовності, що повторюється. Може мати межі ітерації, що пишуться біля назви loop

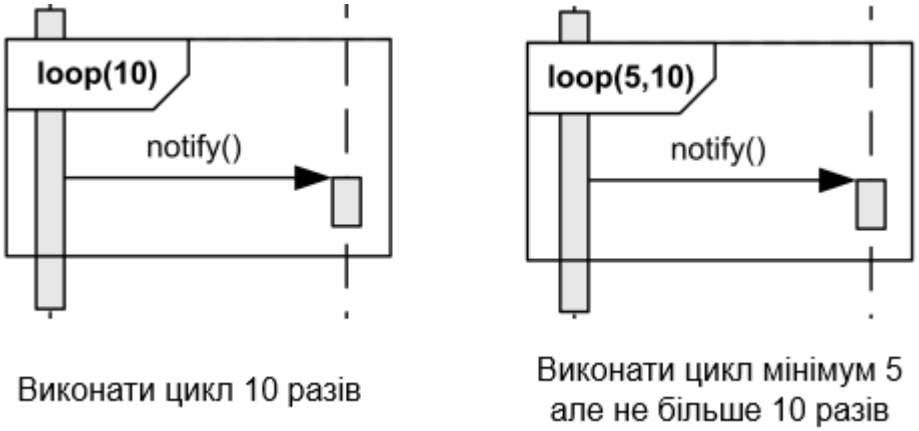

Рис. Приклад фрагмента циклу

Фрагмент посилання (ref) — для повторного використання частини послідовності в іншому місці діаграми.

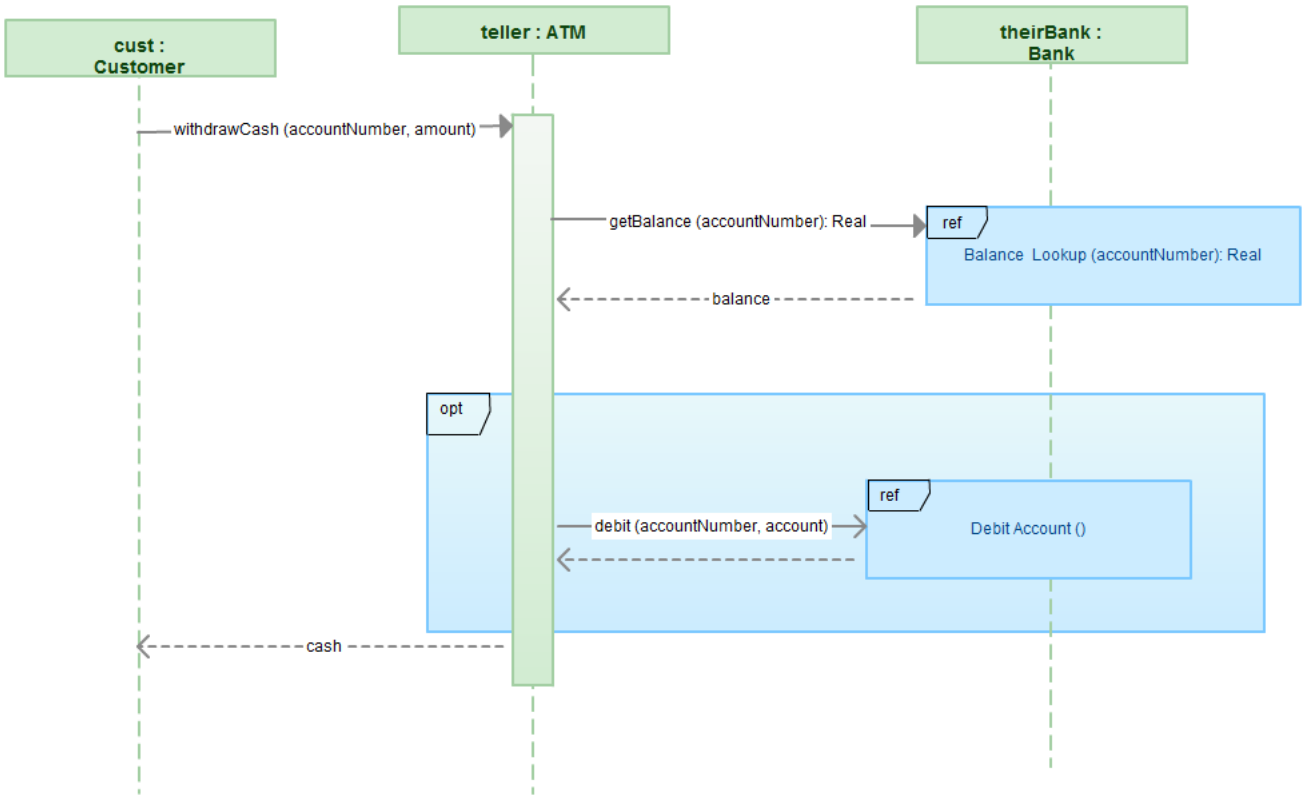

Рис. Приклад фрагмента посилання

Рекомендації зі створення діаграми послідовності

- Оберіть тему діаграми послідовності.
- Визначте об’єкти або акторів, які будуть залучені до створення діаграми.
- Створіть короткий список взаємодій об’єктів, що відображатимуться на діаграмі послідовності.
- Визначте, якими типами повідомлень будуть обмінюватись об’єкти.
- Ви можете розфарбувати діаграму.
- Поширені помилки
- Не додавайте багато деталей. Це захаращує діаграму та ускладнює її читання.
- Початок стрілки повідомлення має завжди торкатись лінії життя відправника, а вказівник — лінії життя об’єкта-одержувача.
- Повідомлення мають будуватись зліва направо.
- Не використовуйте діаграму послідовності, якщо необхідна реалізація простої логіки.

Поширені помилки

- Не додавайте багато деталей. Це захаращує діаграму та ускладнює її читання.
- Початок стрілки повідомлення має завжди торкатись лінії життя відправника, а вказівник — лінії життя об’єкта-одержувача.
- Повідомлення мають будуватись зліва направо.
- Не використовуйте діаграму послідовності, якщо необхідна реалізація простої логіки

Приклад діаграм послідовності

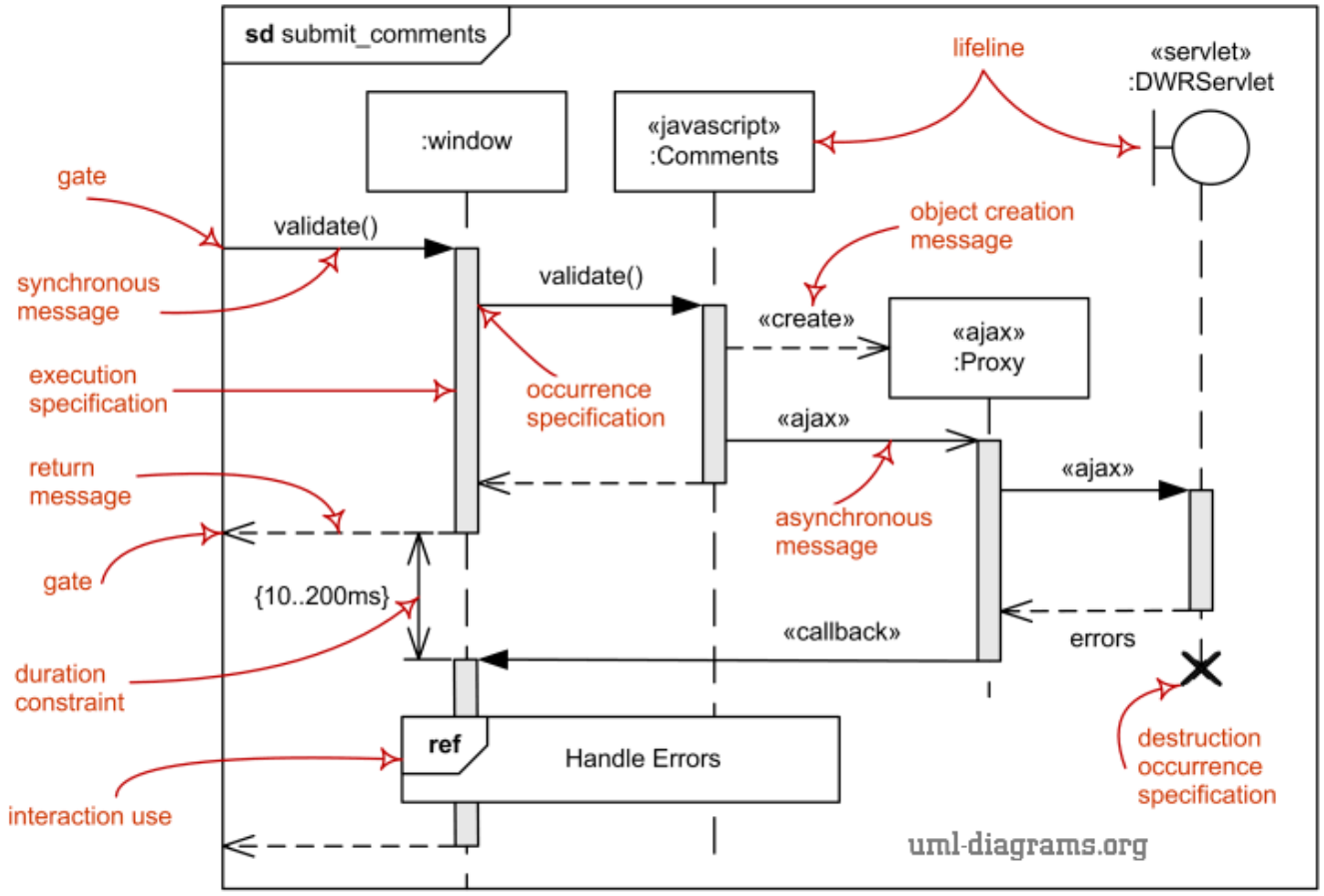

Рис. Приклад 01

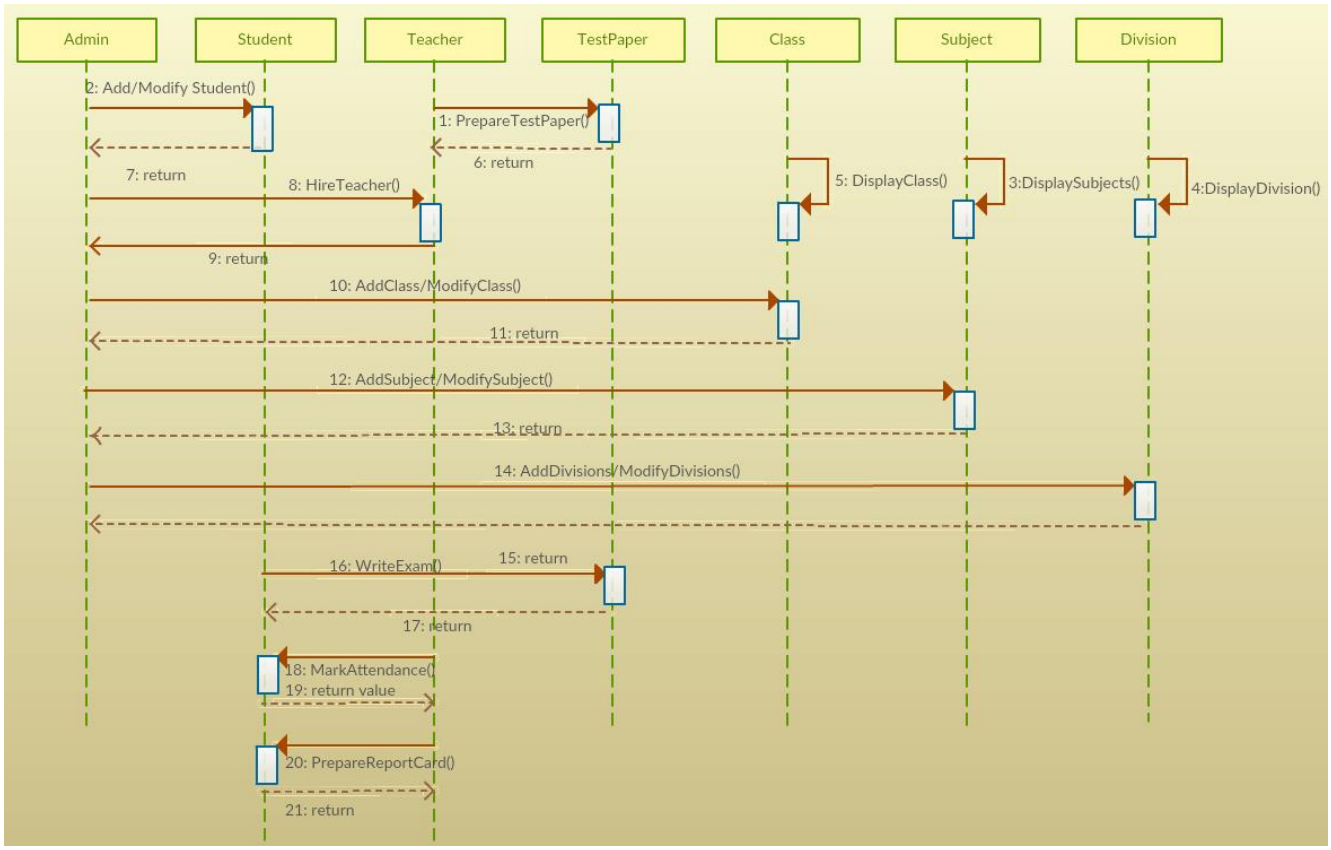

Рис. Діаграма послідовності системи управління школою

2. Діаграма кооперації

Символи та компоненти діаграми коллаборації

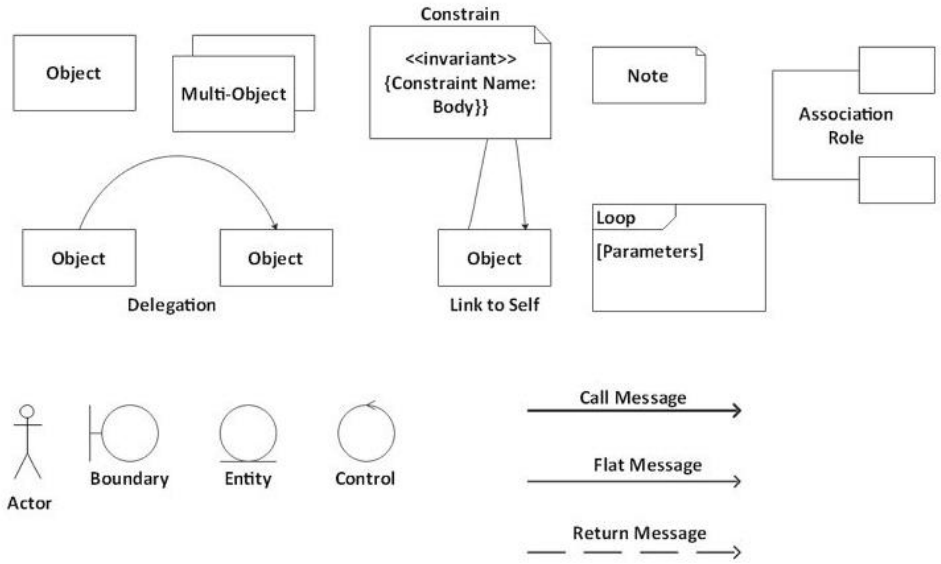

Рис. Основні нотації діаграми коллаборації

- Зв'язки: Зв'язки з'єднують об'єкти, а також акторів. Це випадки асоціацій, і кожен зв'язок на діаграмі класів відноситься до зв'язку.
- Актор (Actor): Зазвичай екземпляр актора існує як початок взаємодії у зв'язку. Якщо на одній діаграмі присутні кілька екземплярів акторів, зосередьтеся на тому, щоб тримати їх на зовнішній стороні діаграми.
- Об'єкт: Об'єкт зображується символом об'єкта, який відображає назву об'єкта та підкреслює його клас, відокремлений двокрапкою.
- Повідомлення Повідомлення - це взаємодія між об'єктами, які передають інформацію з припущенням, що за нею послідує дія. Повідомлення відображається на діаграмах взаємодії у вигляді стрілки з міткою, розташованої біля посилання.

Кроки для створення діаграм взаємодії

1. Визначте поведінку, реалізація та виконання якої задається
2. Визначте структурні елементи (ролі класів, об'єкти, підсистеми), необхідні для реалізації функціональності взаємодії
   - Визначте контекст взаємодії: система, підсистема, варіант використання та операція
3. Змоделюйте структурні взаємозв'язки між цими елементами, щоб створити діаграму, яка відображає контекст взаємодії
4. Розглянути альтернативні сценарії, які можуть знадобитися
   - Намалюйте діаграми взаємодії на рівні екземплярів, якщо це необхідно.

- За бажанням намалюйте діаграму взаємодії на рівні специфікацій, щоб узагальнити альтернативні сценарії на діаграмах послідовності на рівні екземплярів

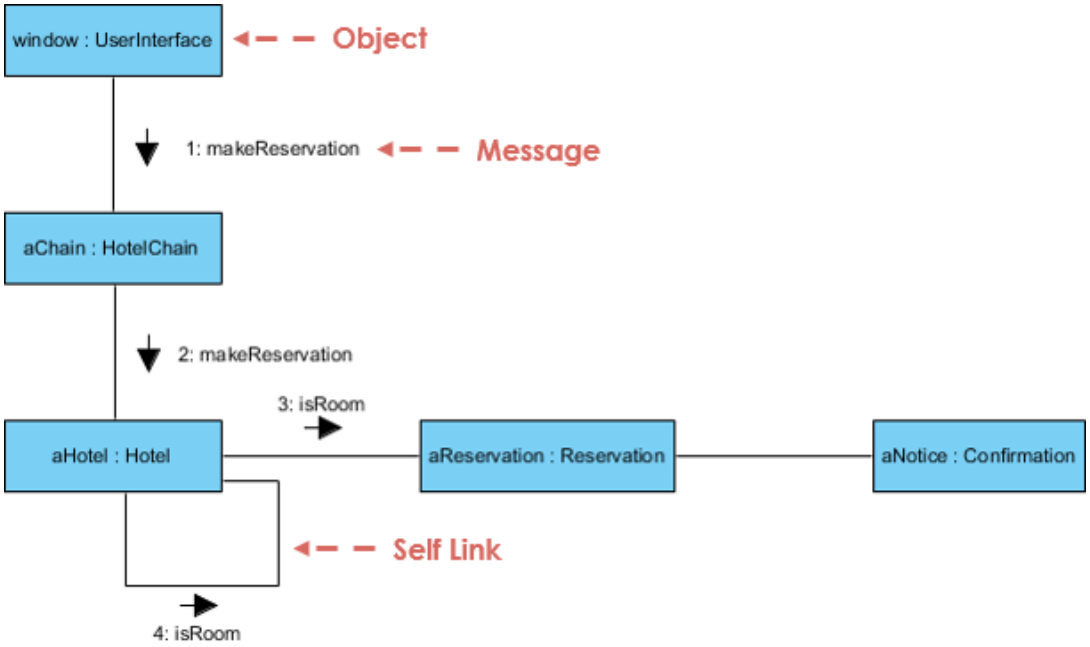

Рис. Основні елементи діаграми кооперації

Діаграма взаємодії у форматі діаграми надійності

На діаграмах взаємодії ви можете мати об'єкти та екземпляри акторів разом з посиланнями та повідомленнями, що описують, як вони пов'язані та як вони взаємодіють. Наведена нижче діаграма "Об'єкт отримання депозиту в системі переробних машин" описує те, що відбувається в об'єктах-учасниках, з точки зору того, як об'єкти взаємодіють, надсилаючи один одному повідомлення. Ви можете створити діаграму взаємодії для кожного варіанту потоку подій варіанту використання.

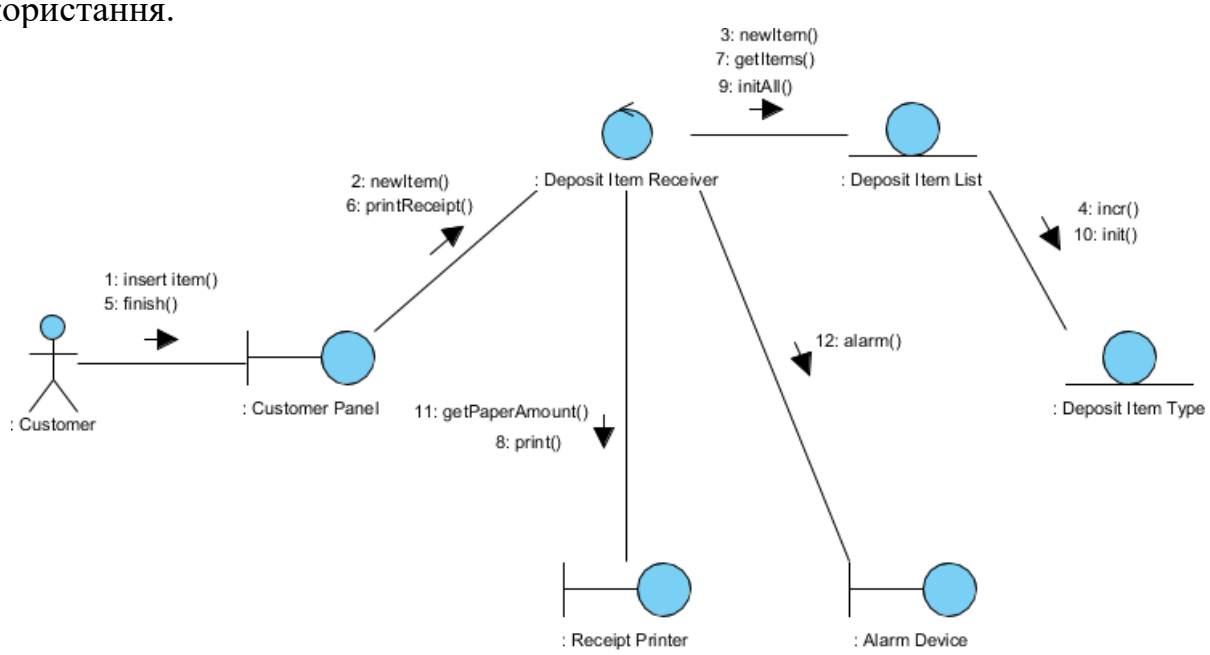

Рис. Діаграма в нотаціях діаграми робастності

Приклад діаграм послідовності

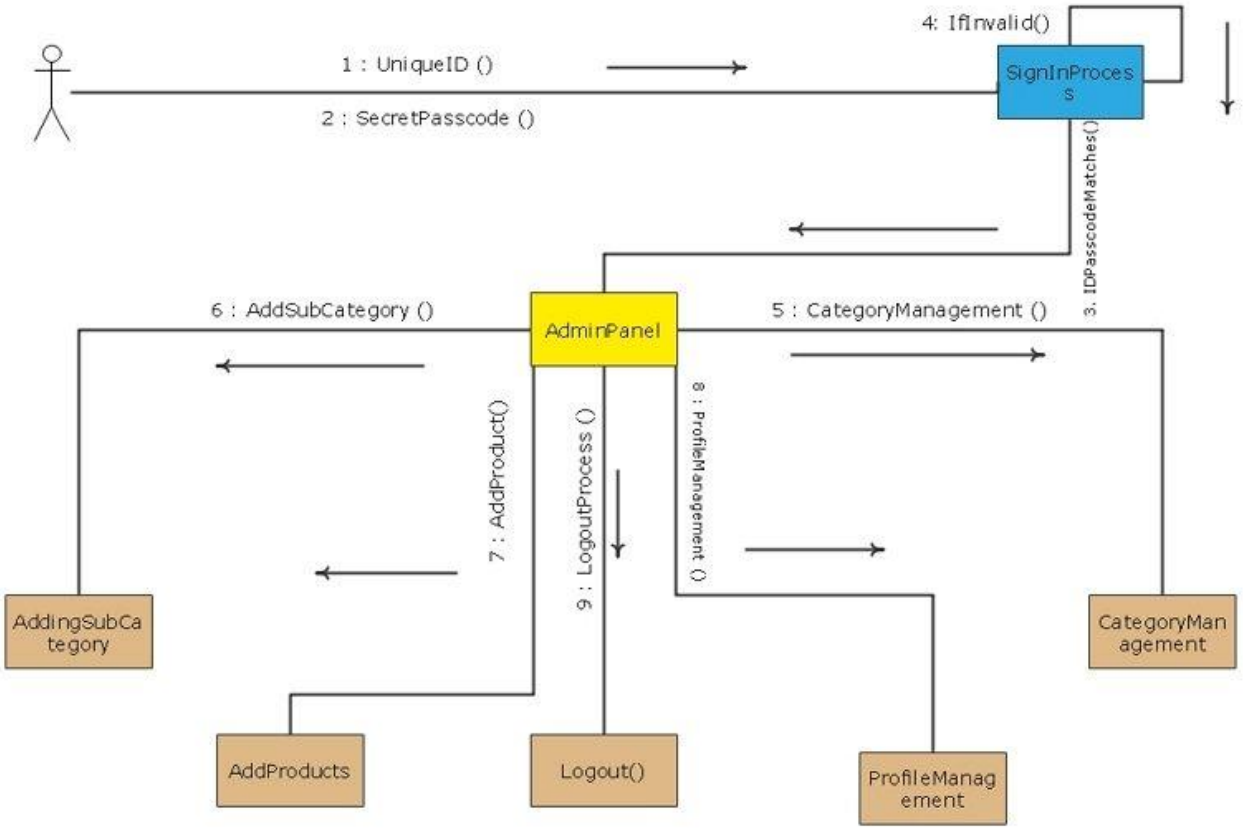

Рис. Приклад для адмін-панелі

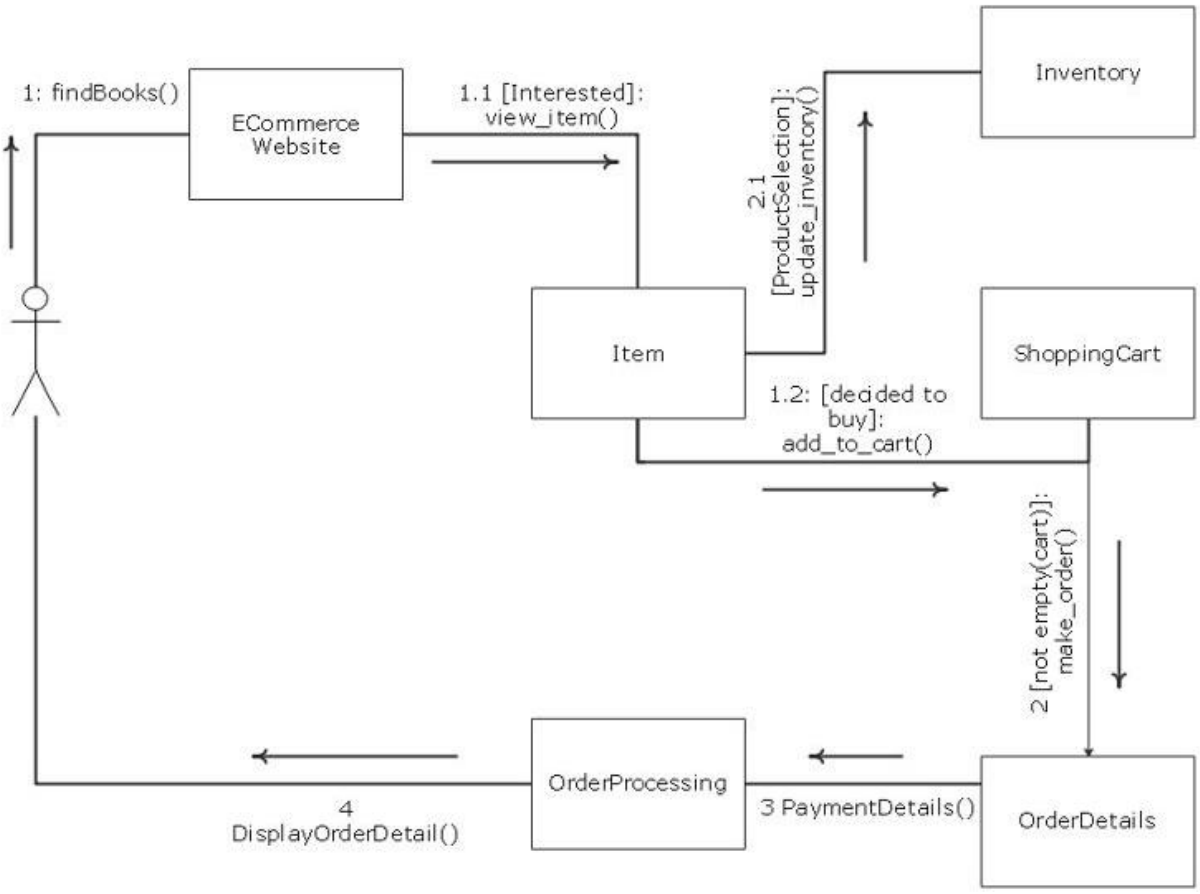

Рис. Приклад для Ecommerse WebSite

Порядок виконання роботи:

1. У відповідності до власного варіанту індивідуального завдання здійснити побудову діаграми послідовності у середовищі StarUML.
2. У відповідності до власного варіанту індивідуального завдання здійснити побудову діаграми кооперації у середовищі StarUML.

Контрольні питання:

1. Чи може діаграма послідовностей містити об'єкт з лінією життя, але без фокусу управління?
2. Чим відрізняються уявлення кооперації на рівні специфікації і на рівні прикладів?
3. У чому різниця між активними і пасивними об'єктами?
4. Чим асинхронне повідомлення відрізняється від синхронного?
5. Що таке мультиоб’єкт?
6. Що таке композитний об'єкт і як він пов'язаний з поняттям кооперації?
7. Як можна уникнути ускладнення діаграми взаємодії з розгалуженим потоком управління?

Література:

1. Grady Booch, James Rumbaugh, Ivar Jacobson. Unified Modeling Language User Guide. Addison-Wesley, 1999. 496 p. ISBN: 0-201-57168-4.
2. Martin Fowler. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley, 2004. 208 p. ISBN: 0-321-19368-7.
3. Craig Larman. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Prentice Hall, 2004. 736 p. ISBN: 0-13-148906-2.
4. Grady Booch, James Rumbaugh, Ivar Jacobson. The Unified Modeling Language Reference Manual. Addison-Wesley, 1999. 720 p. ISBN: 0-201-30998-X.
5. Doug Rosenberg, Kendall Scott. Applying Use Case Driven Object Modeling with UML: An Annotated e-Commerce Example. Addison-Wesley, 2001. 512 p. ISBN: 0-201-60489-7.
6. Jean-Marc Nerson. Object-Oriented Software Construction. Prentice Hall, 2002. 704 p. ISBN: 0-13-629155-4.
7. Jim Arlow, Ila Neustadt. UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design. Addison-Wesley, 2005. 624 p. ISBN: 0-321-26779-8.
8. Michael Blaha, James Rumbaugh. Object-Oriented Modeling and Design with UML. Prentice Hall, 2004. 416 p. ISBN: 0-13-196845-0.
9. Alan Dennis, Barbara Haley Wixom, David Tegarden. Systems Analysis and Design: An Object-Oriented Approach with UML. Wiley, 2014. 544 p. ISBN: 978-1-118-56497-6.
10. Craig Larman. UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design. Prentice Hall, 2005. 736 p. ISBN: 0-13-148906-2.
11. Richard F. Schmidt. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Pearson, 2019. 480 p. ISBN: 978-0131489066.
12. Dan Pilone, Neil Pitman. UML 2.0 in a Nutshell. O'Reilly Media, 2005. 258 p. ISBN: 0-596-00795-7.
13. Doug Rosenberg, Matt Stephens. Use Case Driven Object Modeling with UML: A Practical Approach. Apress, 2007. 648 p. ISBN: 978-1-59059-774-3.