

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**

**КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

**Тетяна Марусенкова,**

**Євгенія Левус,**

**Дмитро Петров**

*МЕТОДИЧНІ ВКАЗІВКИ*

*ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ №6*

*З ДИСЦИПЛІНИ «ВСТУП ДО ІНЖЕНЕРІЇ ПЗ»*

*ДЛЯ СТУДЕНТІВ першого (бакалаврського) рівня вищої освіти*

*СПЕЦІАЛЬНОСТІ 121*

*«ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»*

**2023**

## ЛАБОРАТОРНА РОБОТА №5.

**Тема.** Формування діаграми послідовностей.

**Мета.** Навчитися використовувати діаграму послідовностей як інструмент моделювання поведінки програмної системи.

### ТЕОРЕТИЧНІ ВІДОМОСТІ

#### 1.1. Діаграми взаємодії

Будь-яка програмна чи програмно-апаратна система призначена для виконання певних функцій. Наприклад, в платіжного терміналу з-поміж основних функцій – переказ коштів з одного рахунку на інші, в телефона – здійснення дзвінків і прийом/передача СМС, в графічного редактора – перетворення зображень, у системи «ВНС» – завантаження та перегляд навчальних матеріалів, здача та оцінка робіт тощо.

В процесі виконання цих функцій, між об'єктами системи виникають *взаємодії*, спрямовані на виконання заданих функцій. Наприклад, у випадку платіжного терміналу залучені щонайменше людина-користувач, сам термінал і сервер з базою даних, в яких фіксуються проведені операції. Співвідношення між взаємодіями та функціями можуть бути довільними. В одних випадках одна функція потребує виконання серії взаємодій, в інших – одна взаємодія забезпечує виконання кількох функцій.

В загальному випадку, взаємодією називається поведінка, яка полягає в упорядкованому обміні повідомленнями між множиною об'єктів в деякому контексті, в результаті чого досягається поставлена мета. Під повідомленням розуміється своєрідна специфікація обміну даними між об'єктами, яка у відповідь на передачу деякої інформації передбачає виконання деякої дії або передачу повідомлення далі. Успішна передача повідомлень завжди повинна завершуватися виконанням деякої дії. Такою дією може бути перевірка деякої умови, виконання арифметичної операції, команда периферійному пристрою тощо. Взаємодія завжди виконується в деякому контексті, який визначається структурною сутністю, на якій фокусується аналіз динамічних процесів.

Моделювання взаємодій в програмних систем найчастіше виконується в контексті всієї системи або її окремої підсистеми. В таких випадках взаємодії розглядаються з точки зору функцій, властивих всій системі або лише одній її підсистемі, відповідно. Моделювання взаємодій може виконуватися і на більш низьких рівнях, зокрема, в контексті класу або окремої операції. В першому випадку моделюються поведінка визначених в класі атрибутів та операцій, яка реалізує функціональність класу. Контекст операції передбачає моделювання окремих алгоритмів, реалізованих в операціях. Особливість контексту взаємодій полягає в тому, що він безпосередньо не визначає та не обмежує коло сутностей, які можуть брати участь у взаємодіях. Наприклад, моделювання взаємодій в контексті операції може супроводжуватися включенням в модель інших зовнішніх об'єктів, які беруть участь в процесі виконання операції. В цьому полягає принципова відмінність моделювання взаємодій від моделювання інших аспектів системи, таких як класи та прецеденти.

Моделювання взаємодій можна виконувати одним з двох підходів. Перший підхід зосереджується на часовій впорядкованості повідомлень, а другий – на структурній організації об'єктів, що беруть участь у взаємодії. Відповідно до цих двох підходів розрізняють два типи діаграм взаємодій (Interaction diagrams): діаграми послідовностей (Sequence diagrams) та діаграми кооперацій (Collaboration diagrams).

## **1.2. Призначення діаграми взаємодії**

Діаграма послідовностей має ряд застосувань. Одне з них – деталізація прецедента (випадку використання). Наприклад, в діаграмі прецедентів для банкомату буде «Видача готівки». Під цією стислою назвою криється ряд дій між клієнтом і банкоматом. Користувач вставляє картку. Банкомат розпізнає картку і просить ввести PIN-код. Користувач вводить PIN-код. Банкомат пропонує на вибір ряд функцій, які він може виконати для клієнта. Користувач вибирає, що саме (в даному випадку – зняття готівки). Банкомат запитує, скільки саме потрібно. Користувач зазначає суму. Банкомат видає готівку і запитує, чи обслуговування завершено чи потрібно ще щось. Ці дії виконуються

послідовно в часі, саме в такому порядку. Описаний пінг-понг дій між клієнтом і банкоматом зручно зобразити на діаграмі послідовностей і розкрити тим самим зміст сценарію використання, детально змодельовати його. Діаграма послідовностей, що уточнює сценарій використання, зменшує ризик щось упустити при реалізації програмної системи. Загалом же діаграма послідовностей може бути застосована всюди, де потрібно змодельовати хитру поведінку якоїсь функції чи складний обмін повідомленнями між учасниками взаємодії.

### 1.3. Основні елементи діаграми послідовностей

Основними елементами діаграм послідовностей є *об'єкти* та *повідомлення*.

Об'єкти відображають структурні сутності. Якщо на діаграмі присутній лише один екземпляр деякого класу, то він може не мати власного імені і позначатися лише іменем класу. Якщо діаграма містить кілька об'єктів одного класу, то всі вони крім імені класу повинні мати свої власні унікальні імена. Атрибути та операції об'єктів, як правило, на діаграмах взаємодій не вказуються. Об'єкт зображується у вигляді прямокутника, в якому написано ім'я екземпляра класу (опційно) та через двокрапку ім'я класу. Наприклад:

<u>:User</u>	Об'єкт класу User. У платіжного терміналу в один момент часу клієнт лише один, тому для нас він просто безіменний користувач; немає потреби в імені об'єкта
<u>:Terminal</u>	Платіжний термінал теж лише один, тому давати йому ім'я не потрібно, плутанини і так не виникне
<u>student:Person</u>  <u>teacher:Person</u>	В системі «ВНС» різні ролі. Є студент, є викладач, є адміністратор. Викладач в одному курсі може бути слухачем на іншому курсі. Мабуть, логічно спільні атрибути та методи студента, викладача та адміністратора винести в окремий клас Person або User. Щоб описати взаємодію між цими об'єктами

	слід тим не менш розрізняти ролі, тому в цьому випадку іменування об'єктів є необхідним
--	---

В окремих випадках, в якості імен класів об'єктів вказуються імена абстрактних класів або інтерфейсів. Реально, ці об'єкти представляють собою екземпляри конкретних класів, які наслідують вказані абстрактні класи або реалізують вказані інтерфейси. Подібне позначення використовують для відображення поліморфних або інтерфейсних об'єктів.

В нашому прикладі люди (студент, викладач) представлені об'єктами. Однак, можна зустріти діаграми послідовностей, в яких в якості об'єктів принципово показані тільки складові самої системи (інтерфейс застосунку, сервер, база даних, а також екземпляри різноманітних класів, наприклад, Рахунок, Замовлення тощо), а людина показана сутністю «Актор», щоб підкреслити, що вона поза системою. Актором може бути і інша система.

Іноколи підкреслюють лише імена екземплярів класів, тобто, якщо об'єкт представляє цілий клас, ім'я класу не підкреслюють.

На діаграмі послідовностей об'єкти розташовуються вгорі діаграми горизонтально в порядку їх залучення в процес взаємодії. Тобто зліва вгорі буде об'єкт, який ініціював обмін повідомленнями. Від кожного об'єкта вертикально вниз тягнеться штрихова лінія – це його *лінія життя* (Рис. 1):

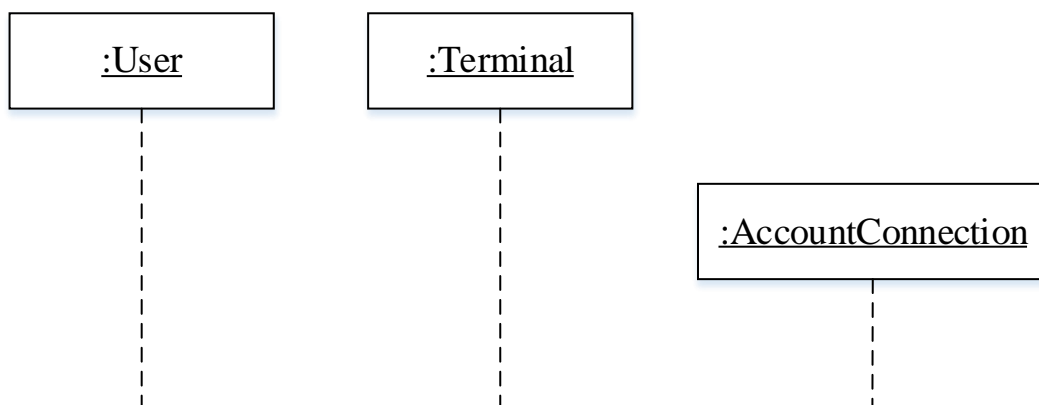


Рис. 1. Об'єкти та лінії життя на діаграмі послідовностей

Лінія життя символізує час існування об'єкта. Не у фізичному змісті, звісно, а лише в контексті реалізації якогось сценарію використання. Наприклад, лінія життя користувача платіжного терміналу слугуватиме для


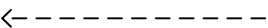


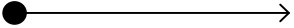
відображення послідовності обміну повідомлень між моментом часу, коли користувач підійшов до терміналу і ініціював перше повідомлення (вибір дії, яку йому необхідно виконати) і до моменту, коли користувач отримав все що йому було потрібно і завершив взаємодію з терміналом, повернувшись до головного меню. В частини об'єктів лінія життя тягнеться з самого верху діаграми до самого низу. Зокрема, користувач і термінал залучені у взаємодію від початку до кінця. Можливо, однак, що якийсь об'єкт існуватиме лише частину часу. Наприклад, поки користувач не ідентифікував себе, не буде доступу до його рахунку. Об'єкт :AccountConnection з'явиться протягом взаємодії користувача з терміналом в якийсь момент пізніше, тому його лінія життя почнеться на діаграмі нижче і правіше. Не всі об'єкти «доживають» до низу діаграми послідовностей. Вони можуть бути знищені, поки інші об'єкти на цій же діаграмі ще живуть. Пам'ятаймо, що в діаграмі послідовностей увага фокусується саме на послідовності обміну повідомленнями в часі. Таким чином, по горизонталі отримуємо «зріз» з життя об'єктів діаграми. Діаграма послідовностей унаочнює розвиток подій в часі.

Як вже відзначалося, основними елементами діаграм послідовностей є об'єкти та повідомлення. Повідомлення зображується у вигляді лінії зі стрілкою. Стрілка вказує напрям повідомлення. Лінія може бути суцільною або пунктирною, а стрілка – простою або замальованою. Пунктирна лінія символізує повідомлення-відповідь. Тобто, якщо повідомлення не є самостійним, а є реакцією об'єкта на інше повідомлення, відправлене раніше іншим об'єктом, його слід відобразити пунктирною лінією. Замальована стрілка символізує синхронне повідомлення, а проста – асинхронне. Синхронне повідомлення означає, що для подальшого розвитку подій необхідно дочекатися відповіді. Асинхронне повідомлення припускає, що його відправник може спокійно продовжити свої справи і відправляти наступні повідомлення, абсолютно не чекаючи, поки його попереднє повідомлення буде оброблене. Наприклад, при використанні електронної пошти людина може відправити електронний лист і продовжити інші операції з іншими листами (прочитання

інших листів, написання нових і видалення старих) не чекаючи відповіді адресата. Натомість, поговорити по телефону не вдасться, поки адресат не підніме слухавку. У випадку платіжного терміналу прикладом синхронного повідомлення буде вибір пункту меню (вибрали пункт і чекаємо, поки перемалюється екран), ввід особового рахунку при оплаті комунальних платежів (маємо почекати, поки рахунок знайдеться і дані про нього відобразяться на екрані).

По відношенню до ВНС прикладом асинхронного повідомлення є завантаження роботи на перевірку. Після цього без очікування на оцінку можна взятися за щось інше – прочитання наступної лекції, наприклад.

Стрілка є лише на одному кінці лінії. З боку відправника стрілки немає. Однак, не завжди повідомлення має визначеного відправника або адресата. Якщо повідомлення прийшло звідкись поза системою і відправник невідомий, з його кінця лінії ставлять круг (замальоване коло). Аналогічно, якщо система відправляє повідомлення невідомому адресату (кудись, немає значення куди), таке повідомлення теж позначається кругом, але тепер вже з боку адресата (поруч зі стрілкою). В специфікації UML повідомлення без адресата називаються «загубленими» (lost messages), а повідомлення без відправника – «знайденими» (found messages).

	Асинхронне повідомлення (відправник розташований лівіше одержувача)
	Повідомлення-відповідь
	Синхронне повідомлення (подальший обмін повідомленнями між цими ж об'єктами в часі слідує після отримання відповіді)
	«Загублене» повідомлення
	«Знайдене» повідомлення

Повідомлення передаються між парами об'єктів, але в окремих випадках об'єкт може надсилати повідомлення самому собі.

Передача повідомлень супроводжується однією з наступних чотирьох дій: викликом операції (можливо, з передачею параметрів) об'єкта-адресата, поверненням управління (і, можливо, значення) відправнику, створенням об'єкта та знищенням об'єкта. При виклику операції поруч з лінією, що символізує повідомлення, вказується ім'я операції і фактичні параметри, якщо такі є. Повернення управління позначається пунктирною лінією, поруч з якою може вказуватися значення, що повертається. Повідомлення, призначені для створення та знищення об'єктів, позначаються стереотипами «create» та «destroy», відповідно.

На лініях життя об'єктів вказують *фокуси управління*, що відображають проміжки часу, протягом яких об'єкти тримають потік управління в собі. Тримаючи фокус управління, об'єкт може відсилати повідомлення іншим об'єктам. Тому фокуси управління не визначають місце знаходження потоку управління в кожний момент часу, а відображають вкладеність викликів та їх послідовність в часі. Графічно фокуси управління позначаються прямокутниками, розміщеними вздовж лінії життя об'єктів. Початок такого прямокутника відповідає моменту передачі управління об'єкту, а кінець – моменту повернення управління. Для рекурсивних викликів, при яких об'єкт передає повідомлення самому собі, створюється новий фокус управління, який накладається на існуючий з невеликим зміщенням вправо.

Вся діаграма може бути взята в рамку (Рис. 2), в якій зліва зазначається тип діаграми (sd – скорочення від sequence diagram) та її назва (Name).

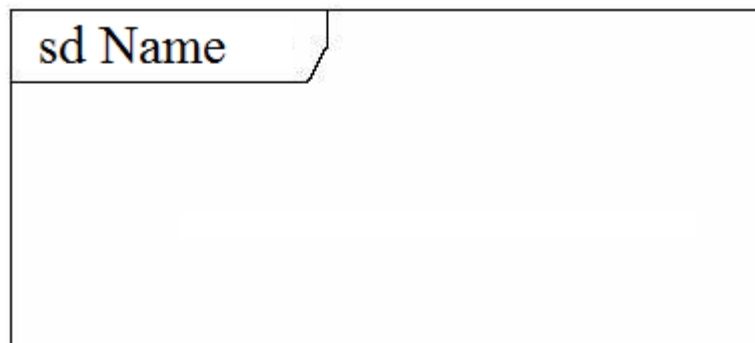


Рис. 2. Опційна рамка



Рамка відділяє проектовану систему від зовнішнього світу. З її країв можуть входити повідомлення (ось звідки беруться «знайдені» повідомлення) і навпаки – система відправляє повідомлення зовнішньому світу поза рамку без конкретизації, хто ловитиме повідомлення.

## 1.4. Приклад діаграми послідовностей

Для прикладу розглянемо діаграму послідовностей (Рис. 3), яка відображає сеанс інформаційного обміну між деякими передавачем та приймачем.

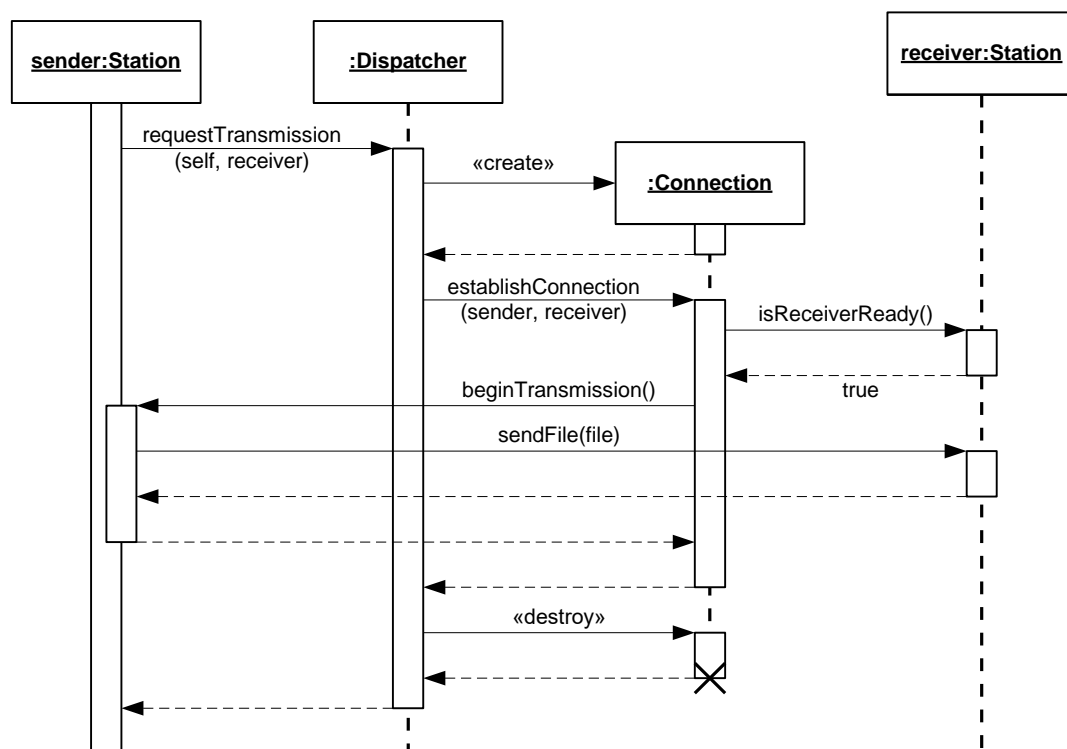


Рис. 3. Приклад діаграми послідовностей

Взаємодія починається із звернення передавача `sender:Station` до об'єкта `:Dispatcher`, який реалізує функції комунікаційного диспетчера, і виклику його операції `requestTransmission()`. Для ідентифікації обох сторін в з'єднанні, `sender:Station` передає в операцію `requestTransmission()` два параметри: вказівник на себе (`self`) та на приймач (`receiver:Station`). Диспетчер `:Dispatcher` створює об'єкт `:Connection`, який забезпечує з'єднання між передавачем та приймачем, і викликає його метод `establishConnection()` з двома параметрами – об'єктами `sender` та `receiver`. Далі об'єкт `:Connection` перевіряє готовність приймача до одержання даних шляхом виклику операції `isReceiverReady()`.

Якщо `receiver:Station` підтверджує свою готовність і повертає логічне значення `true`, то об'єкт `:Connection` дозволяє передавачу розпочати передачу даних, про що свідчить виклик методу `beginTransaction()`. Після цього передавач виконує передачу даних, представлених на діаграмі параметром `file`, що передається в метод `sendFile()` приймача. Після завершення передачі даних, диспетчер `:Dispatcher` знищує об'єкт `:Connection` і сеанс інформаційного обміну вважається завершеним. Вельми символічно, після знищення об'єкта на його лінії життя «ставиться хрест».

Часто на діаграмах взаємодій не відображають повідомлення, які символізують повернення управління викликаючому об'єкту. Таке спрощення можливе, коли воно не призводить до втрати інформації. Наприклад, в наведеній вище діаграмі можна видалити всі повертаючі управління повідомлення, крім повернення з операції `isReceiverReady()`, оскільки останнє крім управління повертає логічне значення.

## 1.5. Моделювання розгалужень

Розглянута діаграма представляє собою лінійну послідовність дій, без будь-яких розгалужень. Теж саме можна сказати про наведений вище приклад зі зняттям готівки. Але в реальності все може відбуватися не так вже й гладко. Наприклад. Користувач вставляє картку. Банкомат розпізнає картку і просить ввести PIN-код. Користувач вводить PIN-код. Маємо два варіанти: PIN-код правильний або неправильний. Ясно, що всі наступні дії, розглянуті раніше, відповідають лише умові, що PIN введено правильно....Банкомат запитує, скільки саме потрібно. Користувач зазначає суму. Що буде, якщо на рахунку користувача недостатньо коштів? Якщо в самого банкомата недостатньо коштів? Якщо запитану суму не можна укласти з наявних купюр? Якщо перевищено ліміт коштів, які можна списати за одну транзакцію? Маємо продумати і описати, як розгортаються події в часі за всіх цих умов.

На діаграмах послідовностей умови записуються в квадратних дужках, поруч з позначенням повідомлення. Наприклад, ми могли б викликати метод

establishConnection(server, receiver) лише в робочий час. Тоді над стрілкою повідомлення мало б бути записано щось типу  
[hour >= 9 & hour <=17 & day != Sunday] establishConnection(server, receiver)

UML не накладає обмежень на формат запису умови розгалуження. Тобто, в квадратних дужках умова записується у довільному форматі.

Для моделювання логіки «якщо ... то ... інакше» (if then else) використовуються альтернативи. Умови для всіх альтернативних повідомлень повинні бути взаємовиключними та взаємодоповнюючими, тобто в кожній ситуації одна і лише одна умова повинна приймати значення “істина”. Якщо дві умови істинні одночасно, все одно виконається лише одна гілка. В UML 2.0 для розгалужень на лінії життя накладається рамка з назвою alt. Коли лінії життя перебувають в межах цієї рамки, читач діаграми розуміє, що на цій ділянці розгортаються альтернативні сценарії. Скористаємося дуже наочним прикладом, запозиченим з ресурсів компанії IBM (Рис. 4), який якраз відповідає випадку, коли клієнт запитує суму, що може бути, а може і не бути в нього на рахунку.

Тут перевіряється стан рахунку (повідомлення з викликом методу getBalance()) і у відповідь повертається баланс. Якщо баланс більший за запитану суму, реалізується щасливий сценарій для користувача, а інакше – альтернативний. На лінії життя першого об’єкта зображено повідомлення самому собі (тобто, виклик власного ж методу, що зображається новим фокусом управління, зміщеним вправо по відношенню до основного).

В даному прикладі показано лише дві гілки. Якщо потрібно більше, просто додається більше умов в межах одного блока alt.

Якщо потрібно змоделювати опцію, для цього є блок opt. Він схожий на вже розглянутий alt, але в ньому немає галуження. Якщо виконується умова, буде обмін повідомленнями, якого за інших умов би не трапилося.

В UML 2.0 передбачений також блок loop для моделювання повторюваних дій.

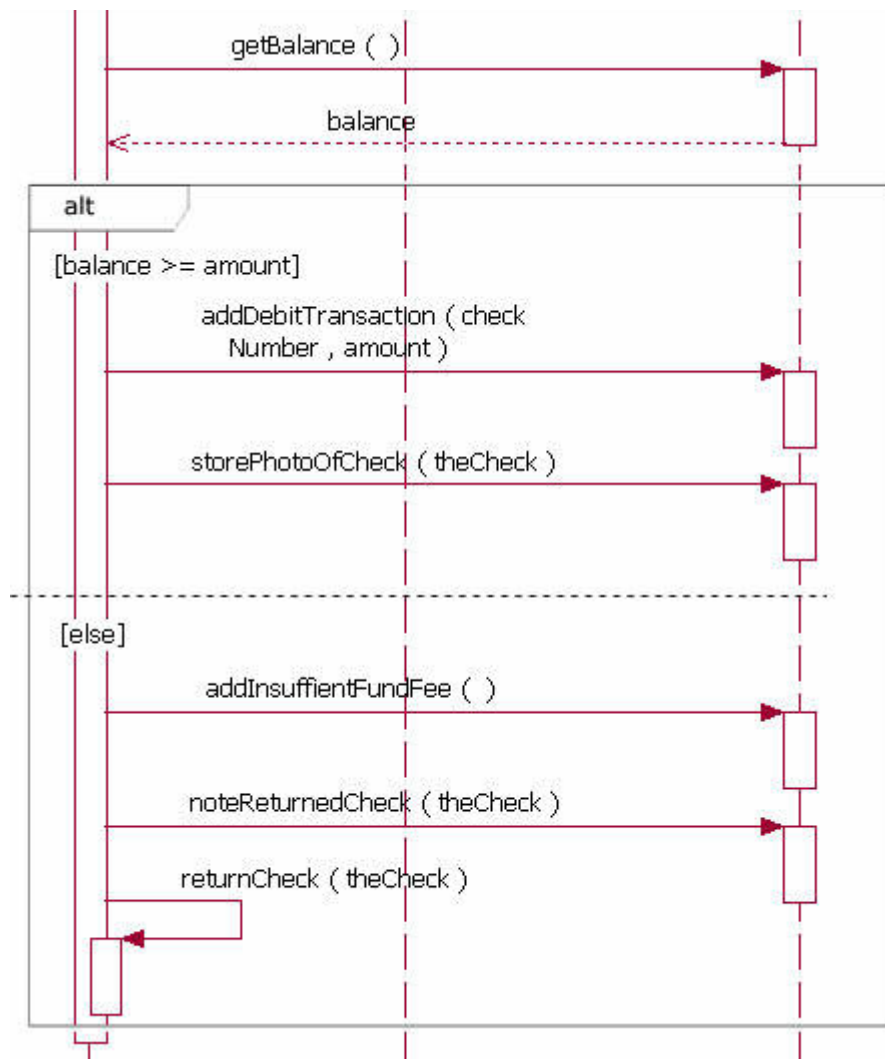


Рис. 4. Приклад альтернативи (фрагмент діаграми)

Джерело: <https://developer.ibm.com/articles/the-sequence-diagram/>

### Висновки

Діаграма послідовностей є різновидом діаграми взаємодії. Основне її призначення – детальне моделювання випадків використання. При цьому основна увага приділяється послідовності взаємодії між об'єктами в часі, звідки і назва діаграми. В UML 2.0 визначено засоби моделювання розгалужень та циклів, що дозволяє, на противагу простій лінійній послідовності обміну повідомленнями, описувати складні сценарії.

### Контрольні питання для самоперевірки

1. Для чого призначені діаграми взаємодії?
2. Яким чином діаграма послідовностей зв'язана з діаграмою прецедентів?
3. Назвіть основні складові діаграми послідовностей.

4. Чи можна стверджувати, що діаграми послідовностей та діаграми кооперації описують одне і те саме?
5. Що означає розташування об'єктів на діаграмі послідовностей?
6. Чи може об'єкт розміщатися не вгорі діаграми? Відповідь поясніть.
7. Як на діаграмі послідовностей зображують створення об'єкта?
8. Як на діаграмі послідовностей зображують видалення об'єкта?
9. Запишіть позначення об'єкта для випадку, коли він уособлює цілий клас.
10. Запишіть позначення об'єкта Pluto класу Dog.
11. Як дізнатися, який об'єкт є ініціатором взаємодії?
12. Що таке лінія життя?
13. Що таке фокус управління?
14. Що означають повідомлення на діаграмі послідовностей?
15. Що означає суцільна лінія повідомлення?
16. Що означає штрихова лінія повідомлення?
17. Що означає відкрита стрілка на повідомленні?
18. Що означає текст, записаний над лінією повідомлення?
19. До яких різновидів сутностей UML відносяться об'єкт і повідомлення?
20. Поясніть різницю між синхронними та асинхронними повідомленнями.
21. Що таке «загублені» повідомлення і як вони позначаються?
22. Що таке «знайдені» повідомлення і як вони позначаються?
23. Чи може об'єкт надсилати повідомлення собі?
24. В яких випадках можна не відображати повідомлення-відповіді?
25. Що роблять у випадках, коли повідомлення має бути надіслано лише за певної умови?
26. Як записуються умови?
27. Чи може умова бути складеною?
28. Наведіть приклад запису умови на діаграмі послідовностей.
29. Як моделюються розгалуження?
30. Для чого призначений блок opt?
31. Як Ви розумієте поняття «взаємодія»?

32. Як співвідноситься взаємодія та функції системи?
33. Придумайте приклад розгалуження по відношенню до системи ВНС.
34. Придумайте приклад розгалуження по відношенню до процесу захисту лабораторної роботи.
35. Придумайте приклад розгалуження для отримання допуску до іспиту.
36. Чи можна однією діаграмою послідовностей описати декілька випадків використання? Відповідь поясніть.
37. Придумайте приклад застосування блока орт по відношенню до системи ВНС.
38. Придумайте приклад застосування блока орт по відношенню до захисту лабораторної роботи.
39. На Вашу думку, чи варто використовувати вкладені розгалуження на діаграмі послідовностей? Відповідь аргументуйте.
40. Чи може об'єкт на діаграмі послідовностей бути базою даних? Людиною? Мобільним телефоном? Відповіді аргументуйте.

### ***Необхідні програмні засоби для виконання завдання***

- 1) Текстовий редактор/процесор,
- 2) MS Visio або онлайн-редактор для побудови діаграми послідовностей.

### **Рекомендований порядок виконання лабораторної роботи**

1. Ознайомитися з темою й метою роботи.
2. Вивчити необхідний теоретичний матеріал та розглянути приклади, які реалізують аналогічні задачі до тематики лабораторної роботи.
3. Отримати індивідуальний варіант завдання. Проаналізувати його й скласти план виконання для дотримання термінів виконання та захисту роботи.
4. Ознайомитися з необхідним для виконання завдання програмним забезпеченням.
5. Створити необхідні файли згідно умови завдання з допомогою необхідних програмних засобів.

6. Перевірити свої знання за списком контрольних питань та з допомогою відповідних тестів.

7. Захистити роботу за комп'ютером.

8. Проаналізувати отриманий результат й оформити письмовий звіт про роботу, дотримуючись наступного змісту:

- *тема та мета лабораторної роботи;*
- *теоретичні відомості;*
- *постановка завдання;*
- *отримані результати;*
- *висновки, в яких необхідно зазначити: які знання отримано в процесі виконання даної роботи; для чого призначена розроблена документація (вказати область її застосування).*

### **Завдання**

Побудувати діаграму послідовностей для предметної області, заданої індивідуальним варіантом. Діаграма послідовностей має відповідати таким вимогам:

- 1) Вона має деталізувати один з прецедентів, описаних в попередній лабораторній роботі на побудову діаграми прецедентів.
- 2) Розроблена діаграма послідовностей має бути реалістичною і добре узгоджуватися з життєвим досвідом пересічної людини.
- 3) На діаграмі послідовностей має бути щонайменше три об'єкти.
- 4) На діаграмі послідовностей має бути застосовано альтернативні сценарії.
- 5) На діаграмі послідовностей має бути використано щонайменше два типи повідомлень

### **Індивідуальні варіанти**

1. Інформаційна система «Бібліотека».
2. Інформаційна система «Деканат».

3. Інформаційна система «Магазин електротехніки».
4. Інформаційна система «Кінотеатр».
5. Інформаційна система «Фітнес-центр».
6. Інформаційна система «Ресторан ».
7. Інформаційна система «Комунальні платежі».
8. Інформаційна система «Ательє одягу».
9. Інформаційна система «Віртуальне начальне середовище».
10. Інформаційна система «Театр».
11. Інформаційна система «Книжковий магазин».
12. Інформаційна система «Стоматологічний кабінет».
13. Інформаційна система «Ательє з ремонту побутової техніки».
14. Інформаційна система «Щоденник школяра».
15. Інформаційна система «Студентський квиток».
16. Інформаційна система «Журнал відвідування та успішності студентів».
17. Інформаційна система туристичної фірми.
18. Система «Конкурс» (вступ у ЗВО).
19. Сервіс відео-, телетрансляції (Megogo).
20. Інтернет-банкінг (Приват24).
21. Інформаційна система «Відділ кадрів» підприємства.
22. Сервіс бронювання житла (Booking).
23. Інформаційна система мобільного оператора.
24. Інформаційна система авіакомпанії ( Ryanair).
25. Інформаційна система салону краси.
26. Сервіс віддаленого конференц-зв'язку (Zoom).
27. Інформаційна система залізничних пасажирських перевезень (УкрЗалізниця).
28. Мобільний застосунок «Таксі» (Uber).



### Варіанти завдань до теоретичних відомостей у звіті

Номер варіанту	Перелік контрольних питань
1	4, 10, 29
2	3, 15, 24
3	2, 20, 19
4	1, 25, 17
5	5, 30, 36
6	6, 26, 34
7	7, 22, 32
8	8, 27, 40
9	11, 33, 39
10	12, 18, 38
11	13, 29, 37
12	16, 20, 1
13	17, 28, 39
14	18, 32, 2
15	24, 34, 3
16	25, 35, 4
17	38, 21, 1
18	39, 22, 5
19	36, 19, 6
20	37, 20, 7
21	35, 18, 8
22	40, 17, 9
23	39, 16, 10
24	35, 15, 11
25	14, 33, 29
26	15, 34, 37

27	24, 12, 2
28	28, 11, 3
29	27, 10, 4
30	26, 9, 5

### Вимоги до звіту

Захист лабораторних робіт включає також виконання звіту у текстовому редакторі, виконаного згідно вимог: розмір шрифту основного тексту – 14 пт; міжрядковий інтервал – 1,25; абзацний відступ зліва – 1 см; перед- й після-абзацні міжрядкові інтервали – 0; вирівнювання основного тексту – двостороннє; заголовки відцентровані з накресленням – **Bold**, усі поля сторінки – 2 см, орієнтація аркушів книжкова.

Аркуші звіту формату А4 нумеруються без вказання номера на першій сторінці. При потребі звіт друкується, паперові аркуші скріплюються.

Складовими звіту є **титульний аркуш, тема, мета, теоретичні відомості, постановка завдання, отримані результати, висновки**. Не допускається переписування чи копіювання матеріалів теоретичних відомостей до лабораторних робіт у теоретичних відомостях власного звіту.

У теоретичних відомостях згідно індивідуального завдання необхідно дати вичерпну відповідь на три контрольні питання з наведеного вище списку.

У розділі «Отримані результати» розмістити сформоване технічне завдання.

У висновках зазначити й охарактеризувати отримані результати в ході виконання лабораторної роботи.

### ***Список рекомендованої літератури***

1. Левус Є. В. Вступ до інженерії програмного забезпечення: навч.посібник/ Є.В. Левус, Н.Б. Мельник. – Львів: Видавництво Львівської політехніки, 2018. – 248 с.
2. Левус Є. В. Життєвий цикл програмного забезпечення: навчальний посібник / Є. В. Левус, Т. А. Марусенкова, О. О. Нитребич. – Львів: Видавництво "Львівська політехніка", 2017. – 208 с.
3. Левус Є. В. Вступ до інженерії програмного забезпечення: 1024 завдання для підготовки до контрольних заходів: навчальний посібник / Є. В. Левус, Т. А. Марусенкова, – Львів: Видавництво "Львівська політехніка", 2021. – 188 с.
4. Матеріали навчально-методичного комплексу у ВНС НУ «Львівська політехніка» (сертифікат №02431, Номер E41-163-146/2018 від 03.09.2018 р.).
5. Explore the UML sequence diagram [Electronic source]: – Available at <https://developer.ibm.com/articles/the-sequence-diagram/>