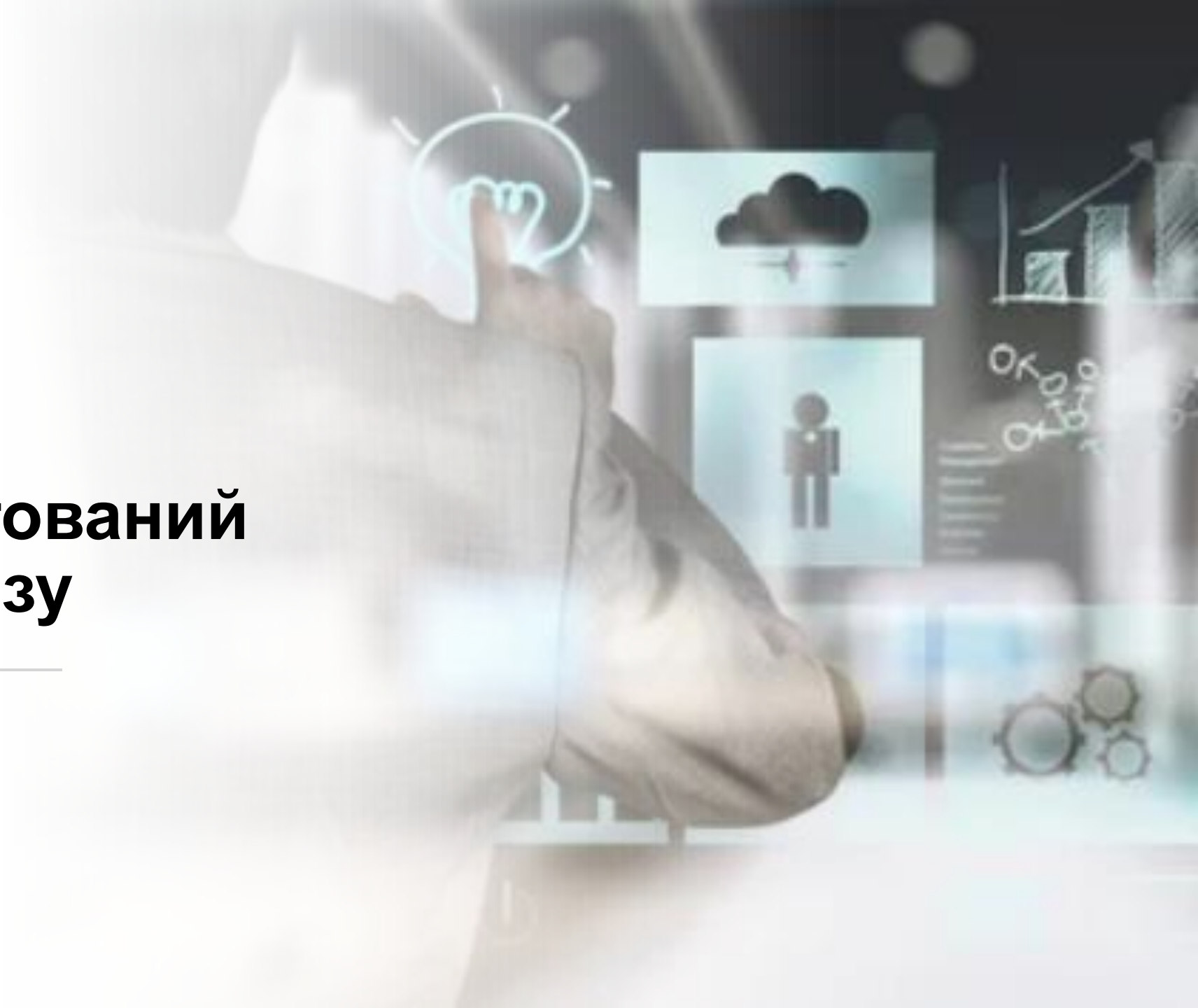
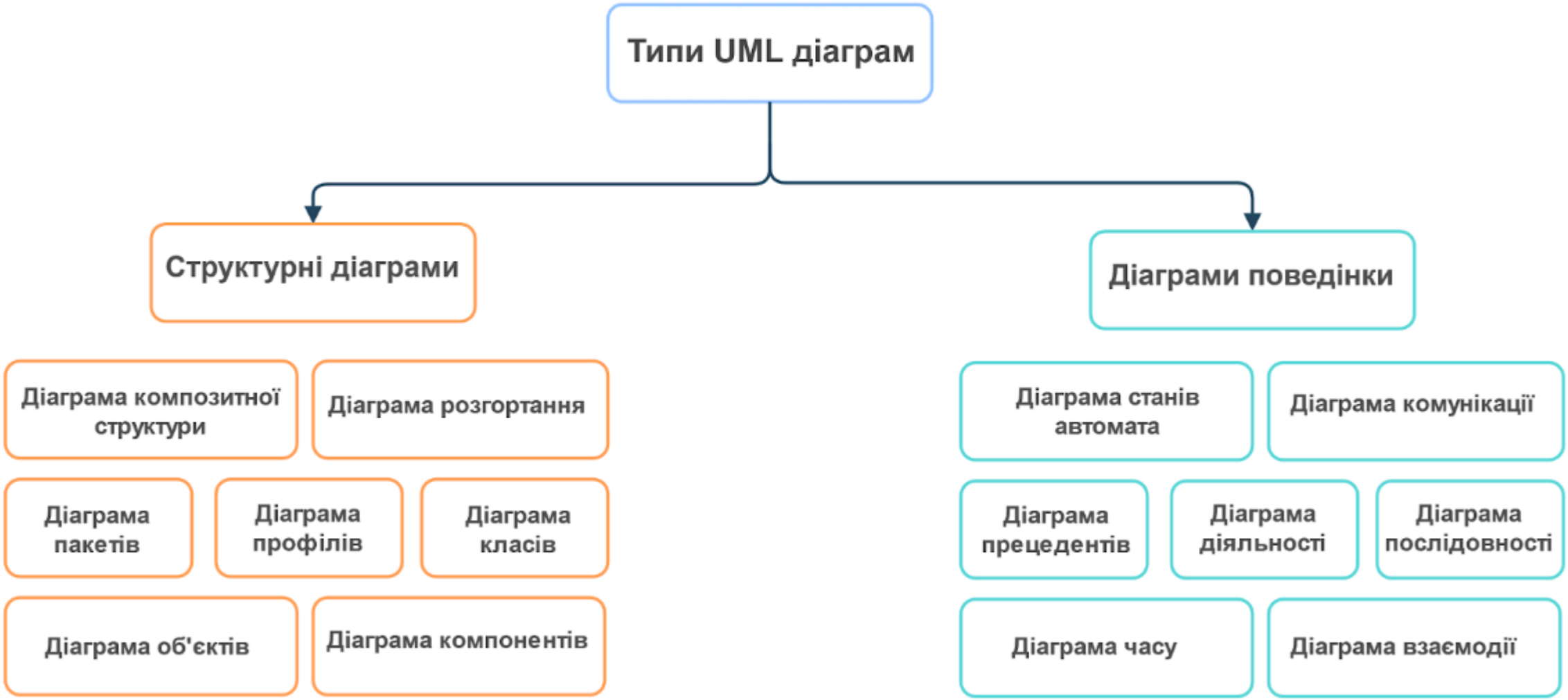
Об’єктно-орієнтований підхід до аналізу

UML: діаграми діяльності та послідовності



UML діаграми



Діаграма послідовності (Sequence Diagram)

показує часові особливості передачі і прийому повідомлень об’єктами. Впорядкованість за часом слід розуміти як послідовність дій

1

Об'єкти

Діаграма послідовності зосереджується на взаємодії між окремими об'єктами системи та показує їх участь у часовій послідовності.

2

Повідомлення

Вона демонструє обмін повідомленнями між об'єктами, вказуючи на тип та порядок цих взаємодій.

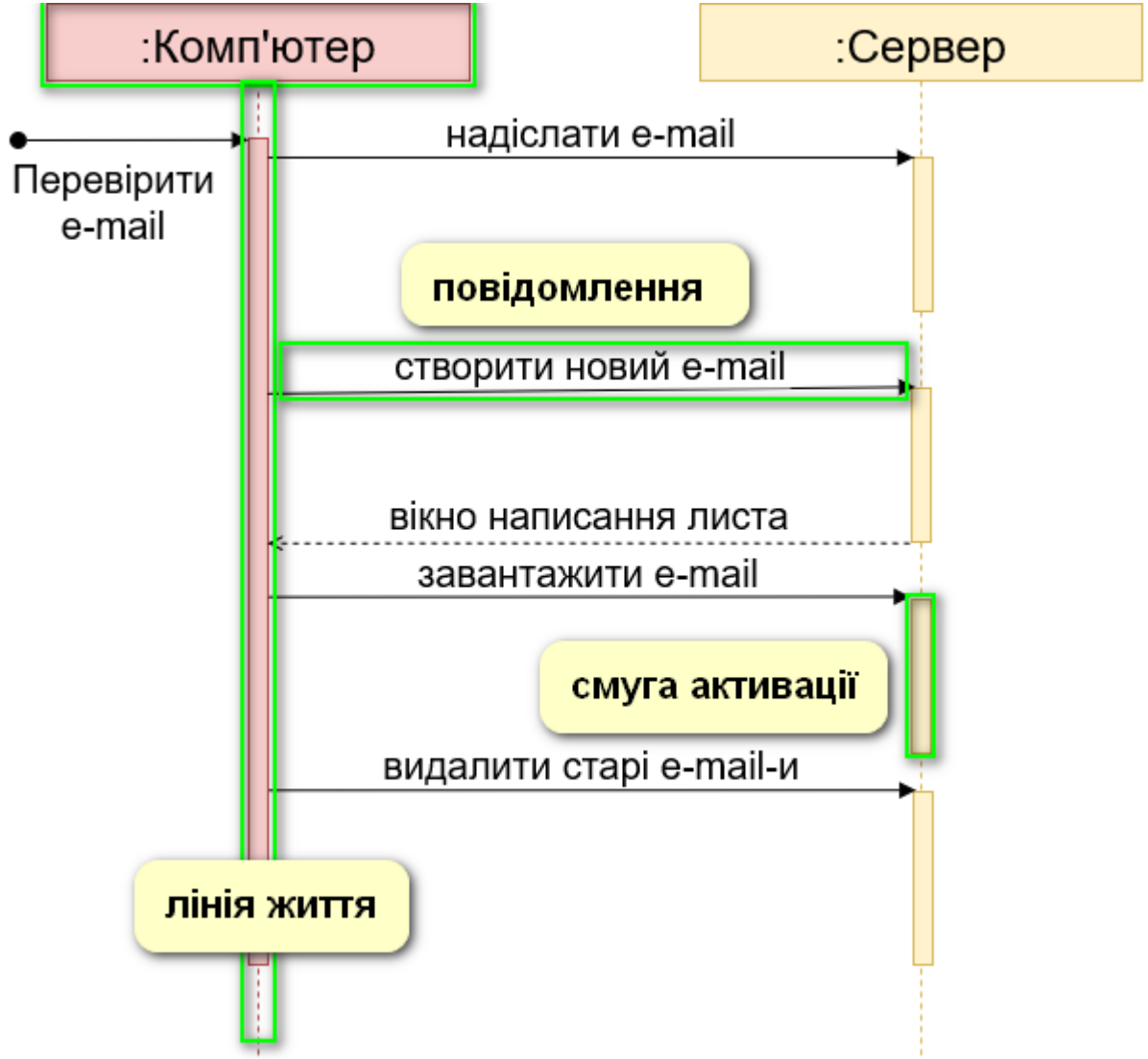
3

Час

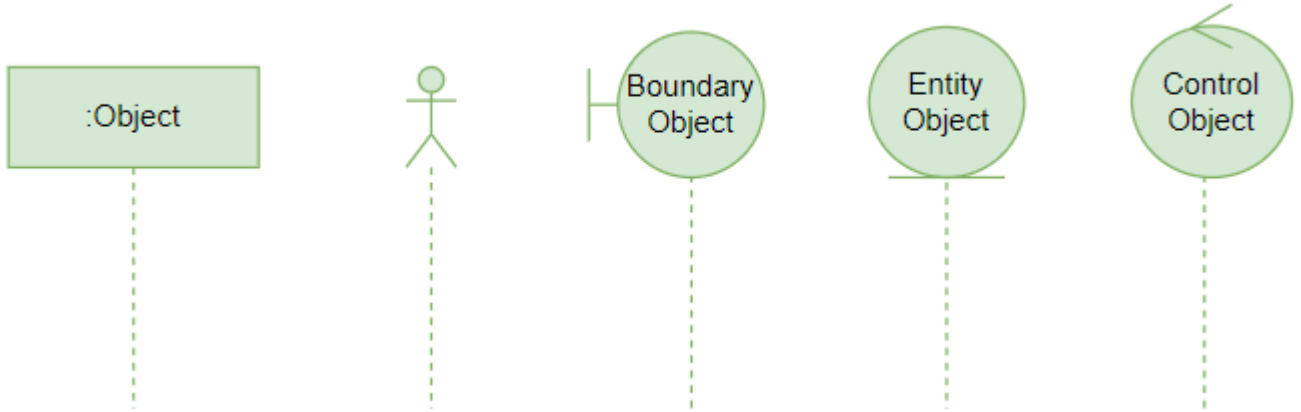
Діаграма розгортається в часі, показуючи, коли і в якій послідовності виконуються різні операції.

Діаграма послідовності (Sequence Diagram)

- Об’єкт
- Лінія життя
- Повідомлення
- Смуга активації (фокус керування)



Виявлення об’єктів



- **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; (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

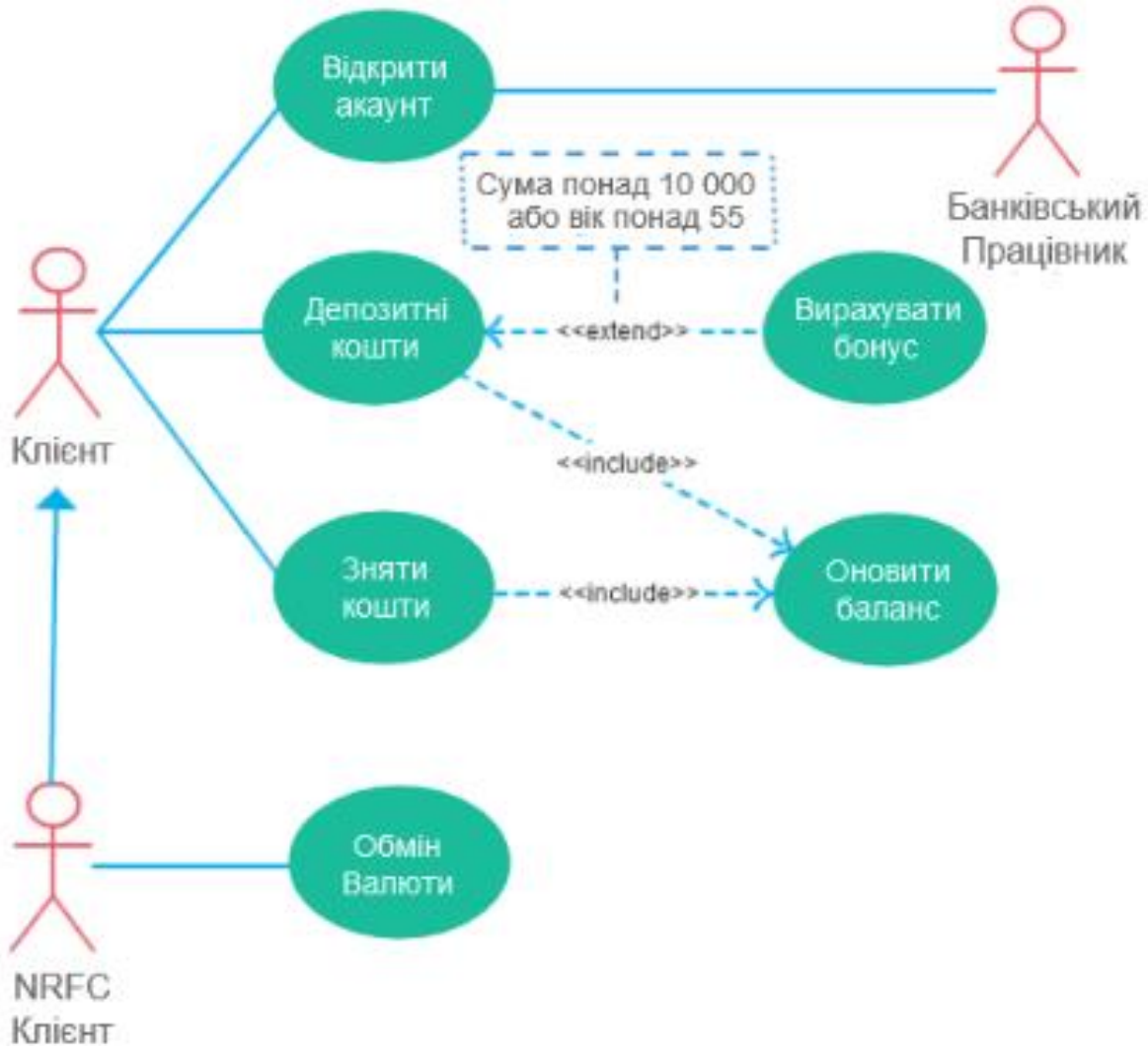
Виявлення об’єктів

- **Виявлення об’єктів** – перший важливий крок на шляху до розроблення діаграм класів
- **Як виявляти об'єкти?** Один з варіантів – шляхом дослідження іменників у сценаріях. (Частина з них може бути атрибутами або ж окремими класами)

Типи об’єктів:

- Об’єкти, що взаємодіють з системою (актори)
- Об’єкти в середині системи

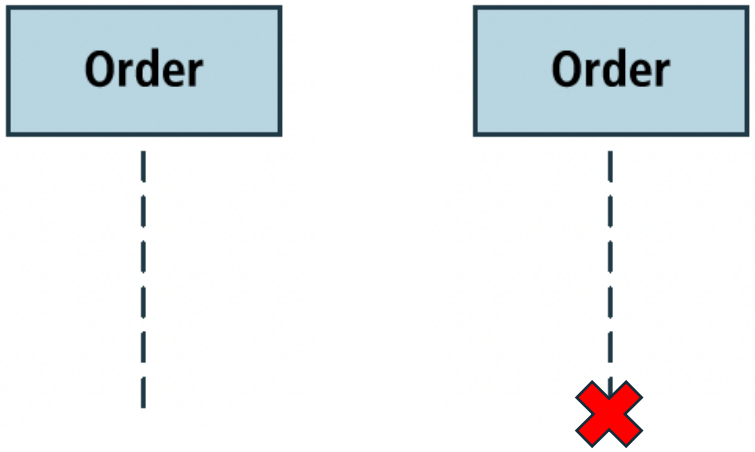
Виявлення об’єктів



- Акаунт (рахунок)
- Баланс
- Бонус
- Внесені кошти
- Виведені кошти
- Клієнт

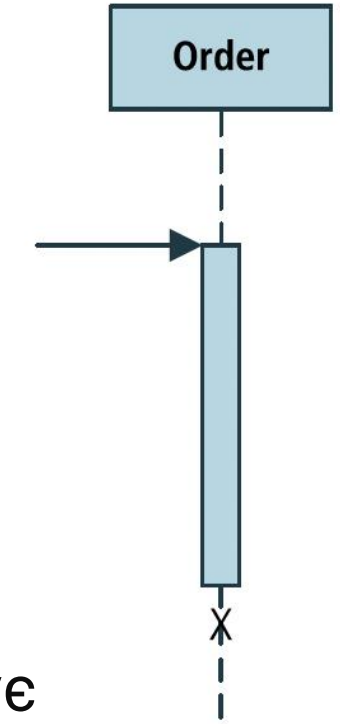
Лінія життя

- 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



Фокус керування

- тонкий прямокутний на лінії життя, протягом якого елемент виконує операцію (отримує а відправляє повідомлення)
- довжина прямокутника вказує на тривалість перебування об’єктів в активному режимі



Повідомлення (Message)

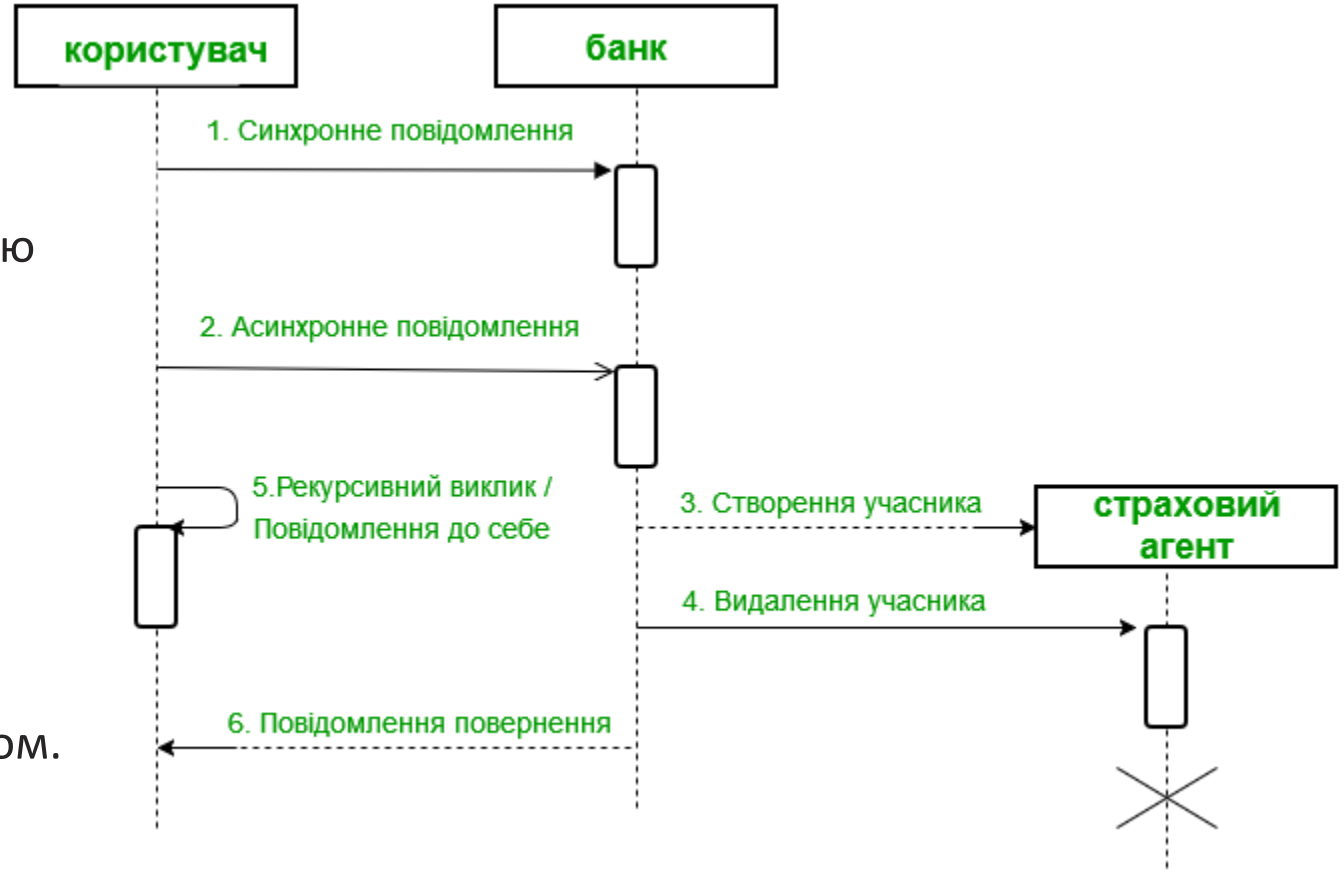
Повідомлення - це взаємодія між двома об'єктами. Повідомлення відображається у вигляді стрілки, що йде від поля активації об'єкта, який надсилає повідомлення, до поля активації об'єкта, який отримує повідомлення.

Синхронні

Очікують відповіді, зображуються суцільною лінією із зафарбованим вказівником.

Асинхронні

Не очікують відповіді, зображуються пунктирною лінією зі звичайним вказівником.



Повідомлення (Message)

Рекурсивне

направлене до себе (починається та закінчується на одній лінії життя)
отримання доступу до камери смартфоном

Повернення

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

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

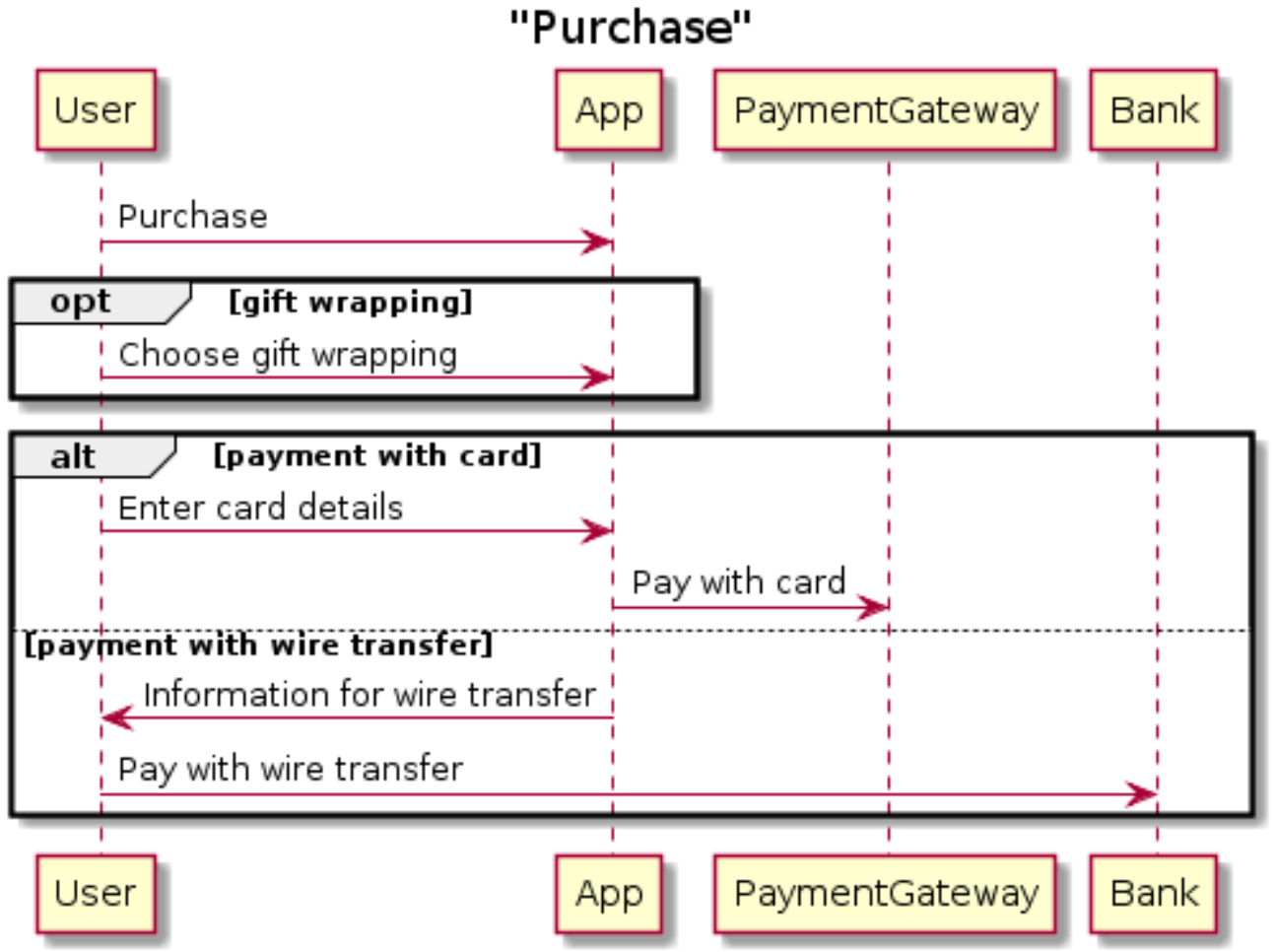
Фрагменти діаграми послідовності

alt

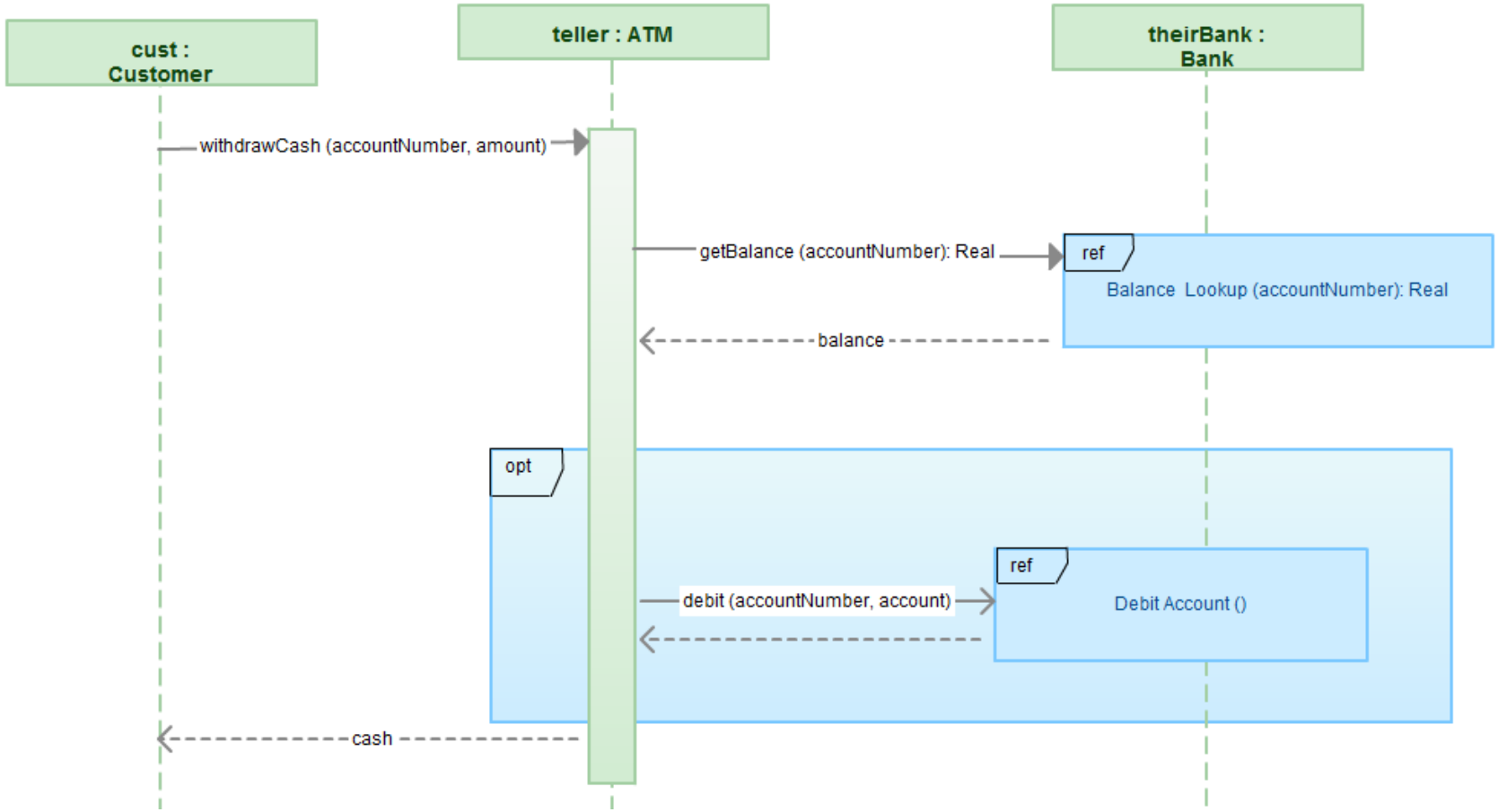
Альтернативний фрагмент для опису декількох сценаріїв.

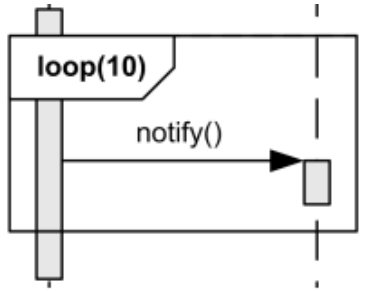
opt

Опціональний фрагмент для необов'язкових кроків.

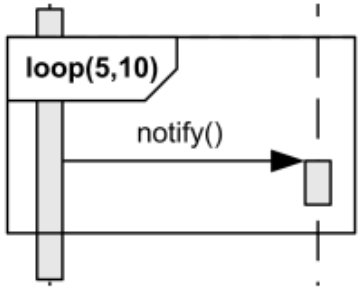


Фрагменти діаграми послідовності





Виконати цикл 10 разів



Виконати цикл мінімум 5 але не більше 10 разів

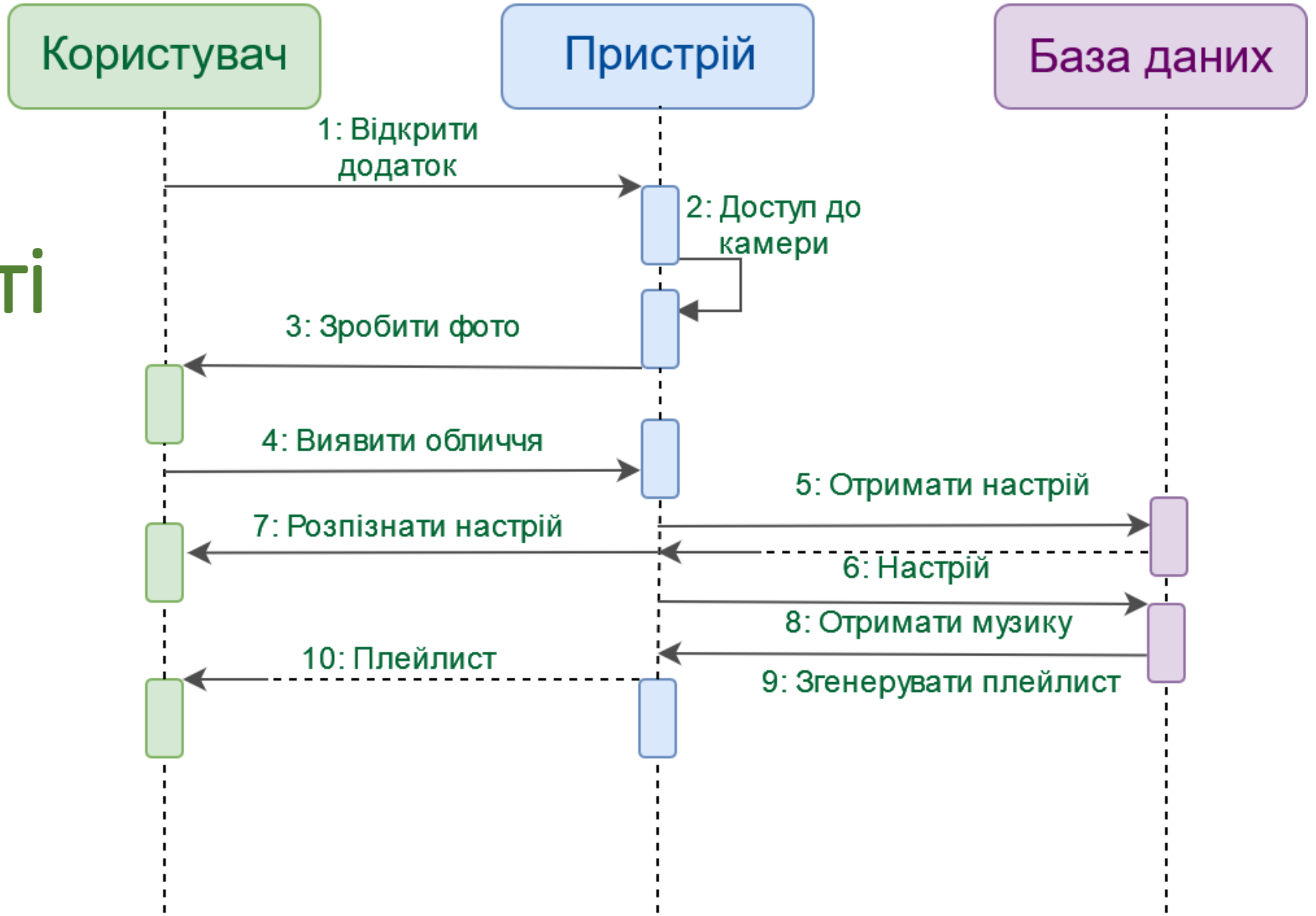
ref

Фрагмент посилання для повторного використання частини послідовності.

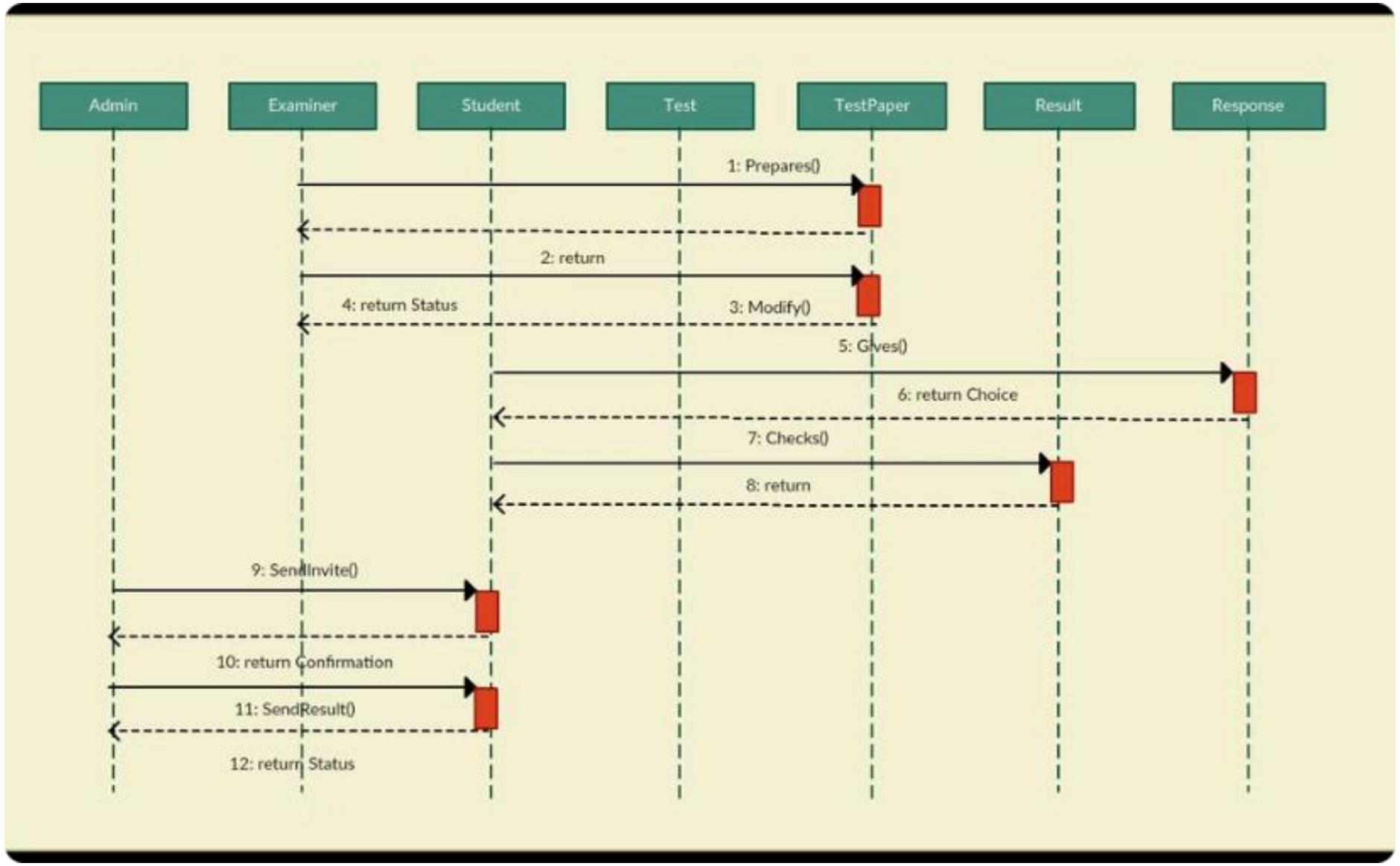
loop

Фрагмент циклу для повторюваних послідовностей.

Приклад діаграми послідовності

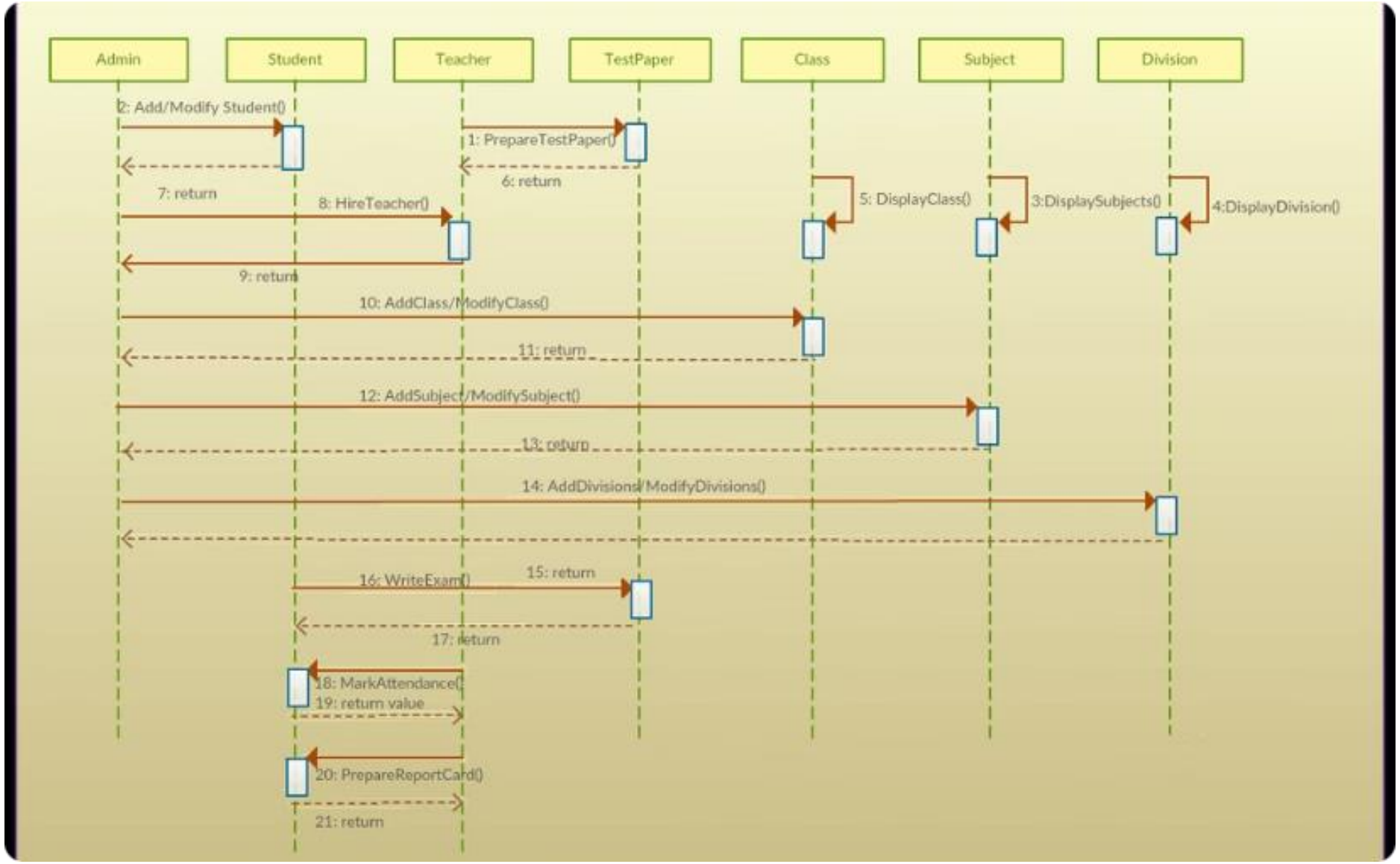


Приклад діаграми послідовності



Онлайн-система проведення іспитів

Приклад діаграми послідовності



Система управління школою

Переваги діаграми послідовності

Візуалізація логіки

Діаграма послідовності наглядно демонструє логіку взаємодії об'єктів у часі.

Розуміння поведінки

Вона допомагає глибоко зрозуміти динамічну поведінку системи.

Документування

Діаграма послідовності є корисним інструментом для документування архітектури ПЗ.

Аналіз продуктивності

Вона може використовуватися для виявлення вузьких місць у продуктивності.

Рекомендації зі створення діаграми послідовності

- Створіть UseCase
- Визначіть прецедент, для якого будуватиметься діаграма
- Визначте об’єкти або акторів, які будуть залучені до створення діаграми
- Створіть короткий список взаємодій об’єктів, що відображатимуться на діаг
- Визначте, якими типами повідомлень будуть обмінюватись об’єкти

Типові помилки

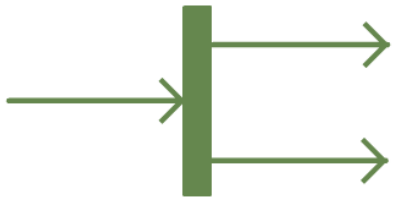
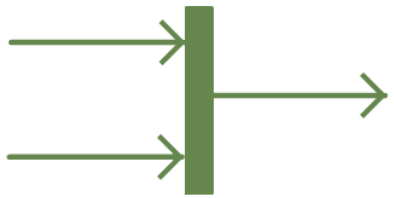
- Не додавайте багато деталей. Це захаращує діаграму та ускладнює її читання
- Початок стрілки повідомлення має завжди торкатись лінії життя відправника, а вказівник — лінії життя об’єкта-одержувача
- Повідомлення мають будуватись зліва направо
- Не використовуйте діаграму послідовності, якщо необхідна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

Діаграма діяльності (Activity Diagram)

візуалізує процес використання та ілюструє потік повідомлень від однієї дії до іншої

Символ	**Ім’я**	**Використання**
	Початковий вузол	Відправна точка, або початковий стан
	Дія	Представлення діяльності, завдання для виконання
	Потік керування	Спрямований потік, контрольний потік діяльності
	Кінцевий вузол активності	Кінцевий стан, завершення усіх потоків процесу

Діаграма діяльності (Activity Diagram)

Символ	Ім’я	Використання
	Вузол прийняття рішення	Розгалуження з умовою та кількома варіантами дій. Має один вхід декілька виходів
	Вузол злиття	Об’єднання потоків створених вузлом прийняття рішень. Має кілька входів і один вихід
	Вилка	Розподілення потоку на кілька паралельних без прийняття рішення
	Злиття	Об’єднання декількох паралельних потоків

Приклад використання основних елементів

Потік керування
з’єднує два вузли на діаграмі активності

Вузол розгалуження (вилка)
вузол керування, який розділяє потік на кілька одночасних. Потрібен для створення декількох одночасних завдань

Вузол злиття
об’єднує ці потоки

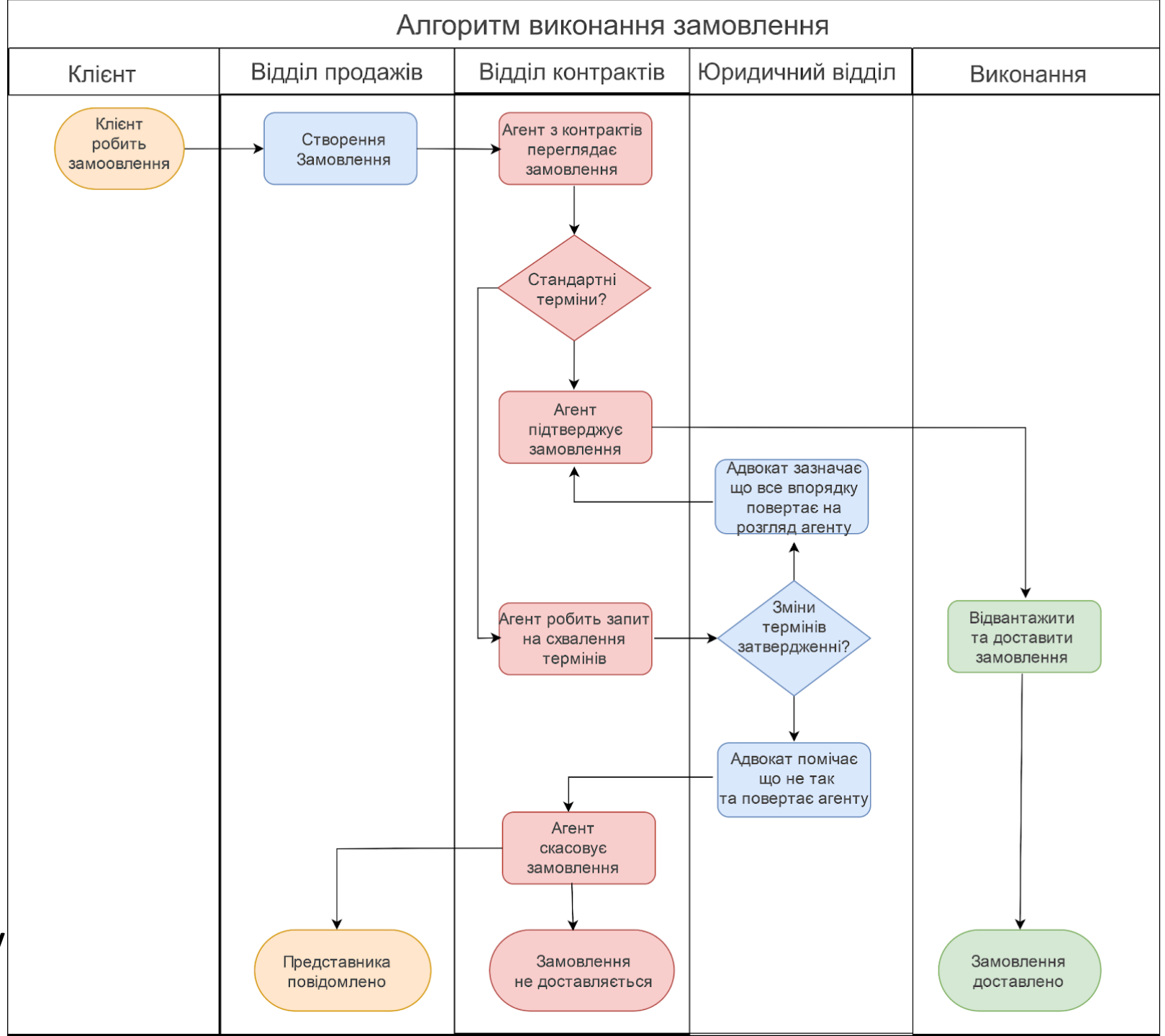
Вузол прийняття рішень
вузол керування, який вибирає між декількома потоками один істинний. Сожий на оператор if. Умови записуються в квадратних дужках поруч з потоком

Приклад діаграми з доріжками

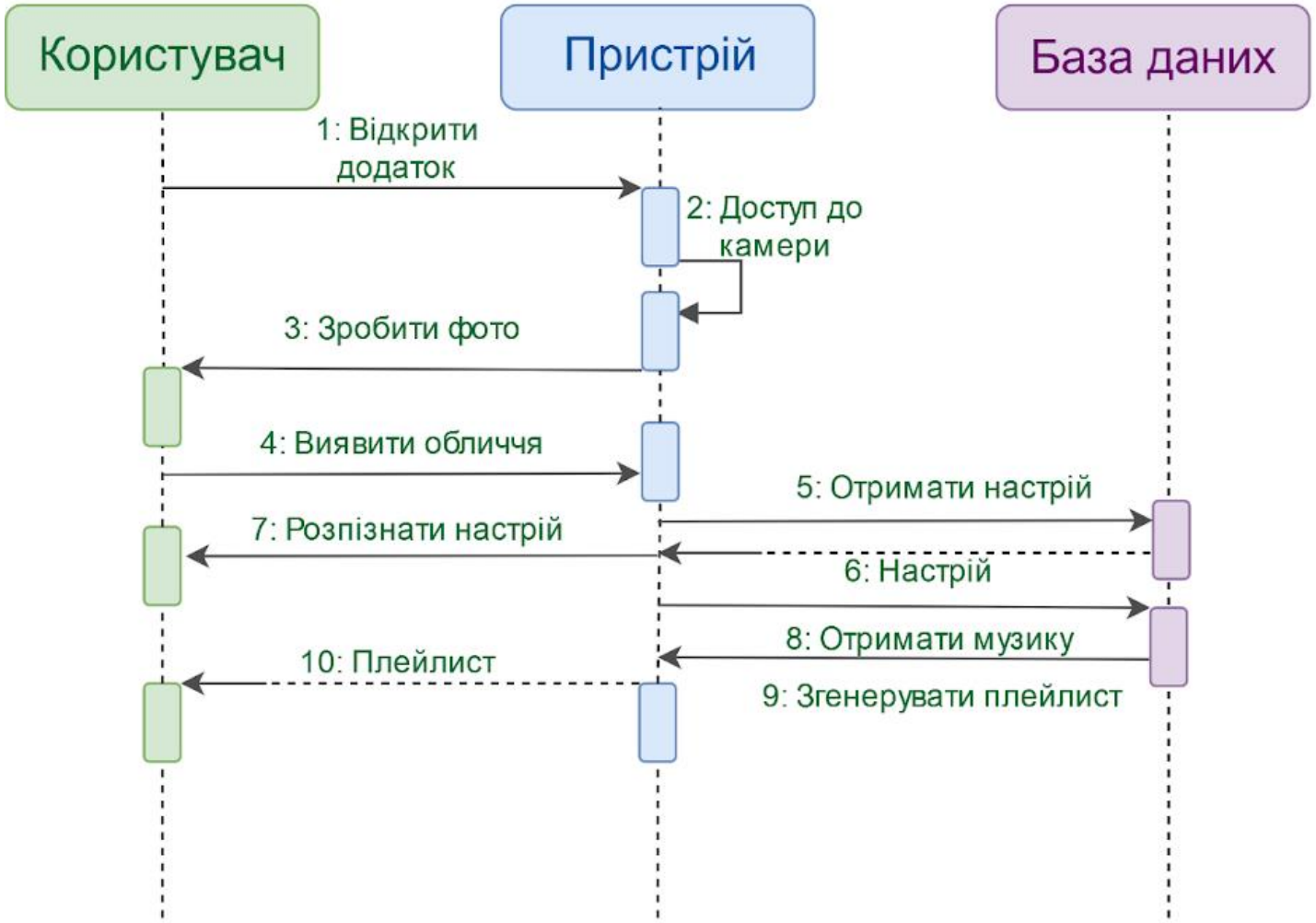
Доріжки

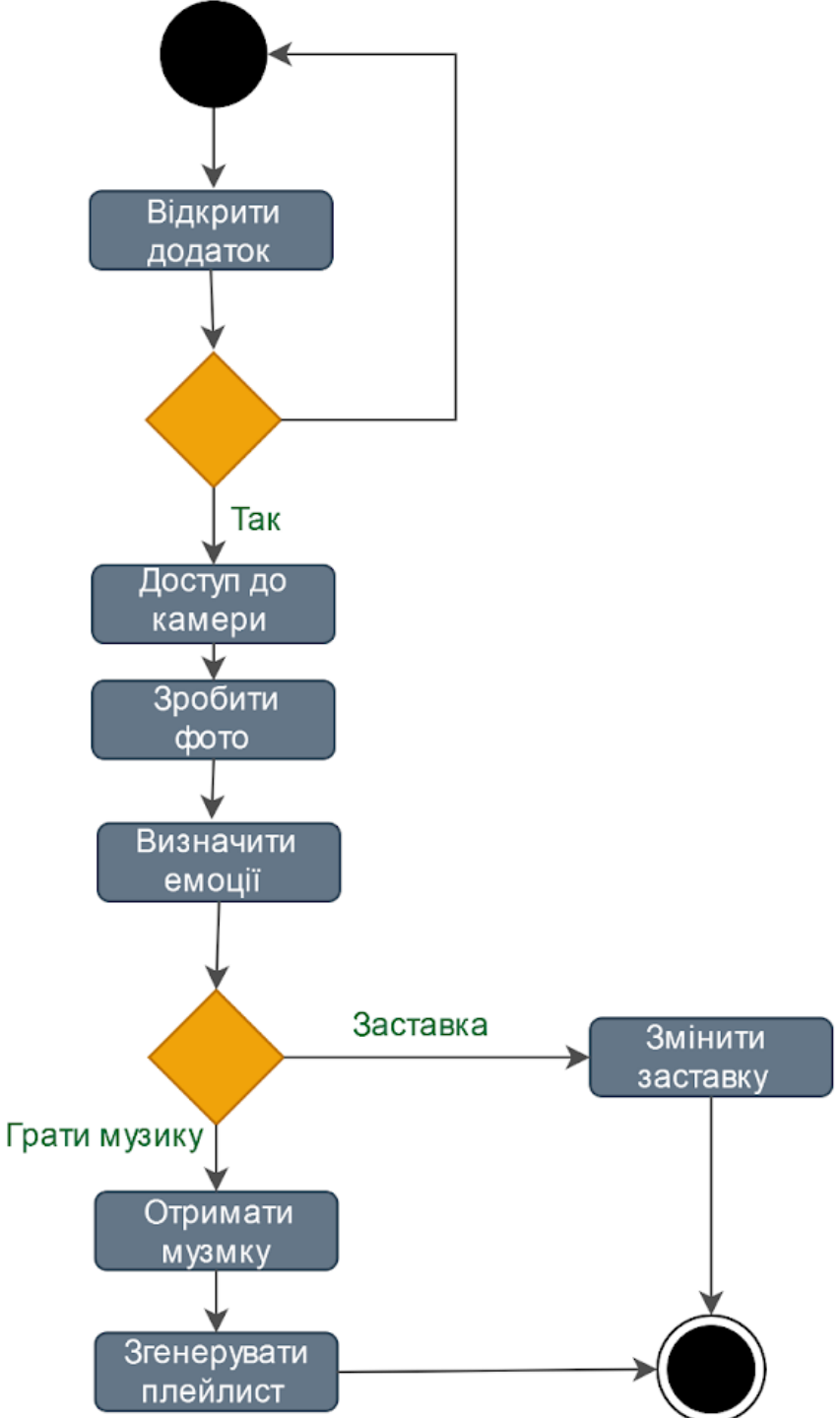
спосіб групування дій, які виконуються одним актором, або декількома акторами в одному потоці

- не більше 5 доріжок одночасно
- розміщення у логічному порядку



Відмінність діаграм





Поняття класу

Клас – абстракція, що описує групу об'єктів із загальними:

- властивостями (атрибутами);
- поведінкою (операціями);
- відношеннями з об'єктами інших класів;
- семантикою, зокрема відповідальностями.

Клас включає:

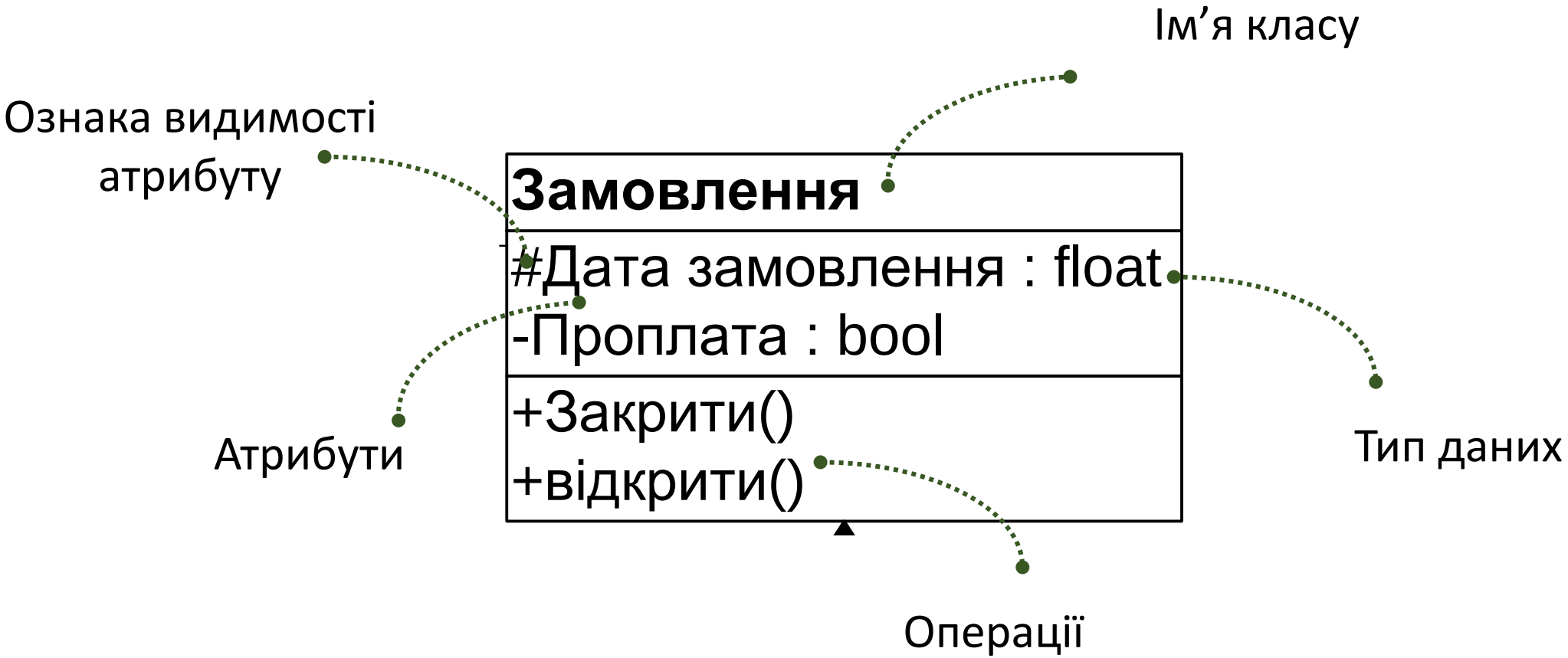
- Унікальне імя
- Атрибут – властивість класу
- Домен – множина допустимих значень атрибута
- Операція

Операції класу

- Реалізація функцій яку може виконати об’єкта або яка може бути виконана над об’єктом

- Виконання операції пов’язане зі зміною значень атрибутів

Графічне представлення класу



Видимість атрибутів

Public - загальний – клас або атрибут видимий іншими об’єктами і може бути ними використаний

Protected – захищений – лише нащадок заданого класу може використовувати його атрибути та інші властивості

Private – закритий, не доступний для інших класів та об`єктів

Стереотипи класів

Клас керування (control class) відповідає за координацію поведінки об'єктів системи. У кожного прецеденту, зазвичай, є хоча б один клас керування, який контролює послідовність виконання дій цього прецеденту. Зазвичай, цей клас є активним та ініціює розсилання множини повідомлень іншим класам моделі. Клас керування може бути відсутнім у прецедентах, які здійснюють прості маніпуляції зі збереженими даними.

Клас-сутність (entity) містить інформацію, яка повинна зберігатися постійно і не знищуватися зі знищенням об'єктів певного класу чи припиненням роботи системи (вимиканням системи чи завершенням програми). Цей клас може відповідати окремій таблиці бази даних; у цьому випадку його атрибути є полями таблиці, а операції - збереженими процедурами. Зазвичай, цей клас є пасивним і лише приймає повідомлення від інших класів моделі. Класи-сутності є ключовими абстракціями (поняттями) ПО системи.

Граничний клас (boundary) розташовується на межі системи з зовнішнім середовищем, однак є складовою частиною системи. Цей клас слугує посередником під час взаємодії зовнішніх об'єктів із системою. Зазвичай, для кожної пари актор - прецедент визначається один граничний клас.

Клас сутності

Клас сутності у мові UML служить для позначення конкретної сутності, яка має однакову структуру та поведінку

Граничний клас

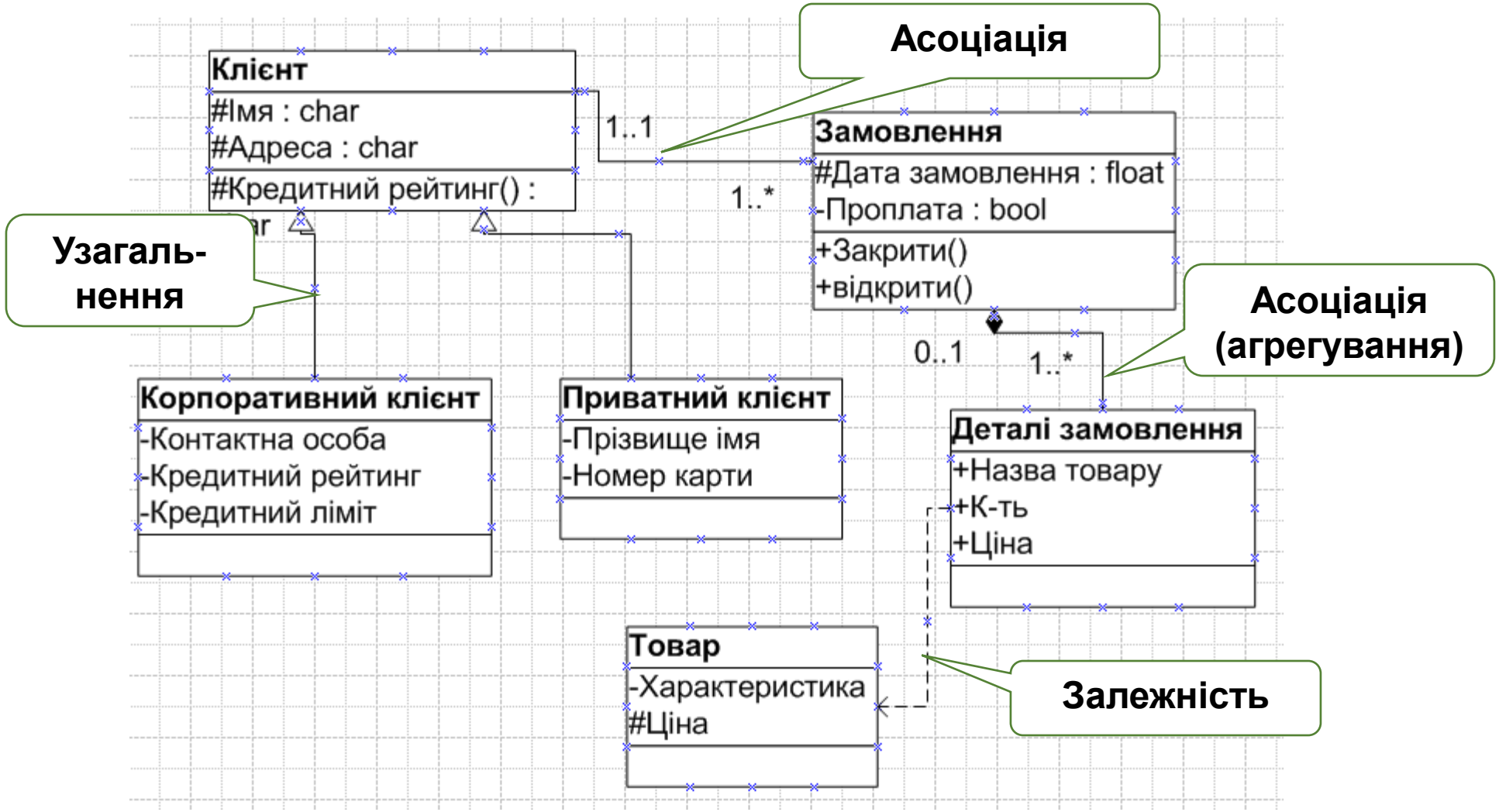
Інтерфейс (interface) - іменована множина операцій, які характеризують поведінку окремого елемента моделі

Інтерфейс у контексті мови UML є спеціальним випадком класу, у якого є операції, але відсутні атрибути

Для позначення інтерфейсу використовується стандартний спосіб – прямокутник класу зі **стереотипом <<interface>>**

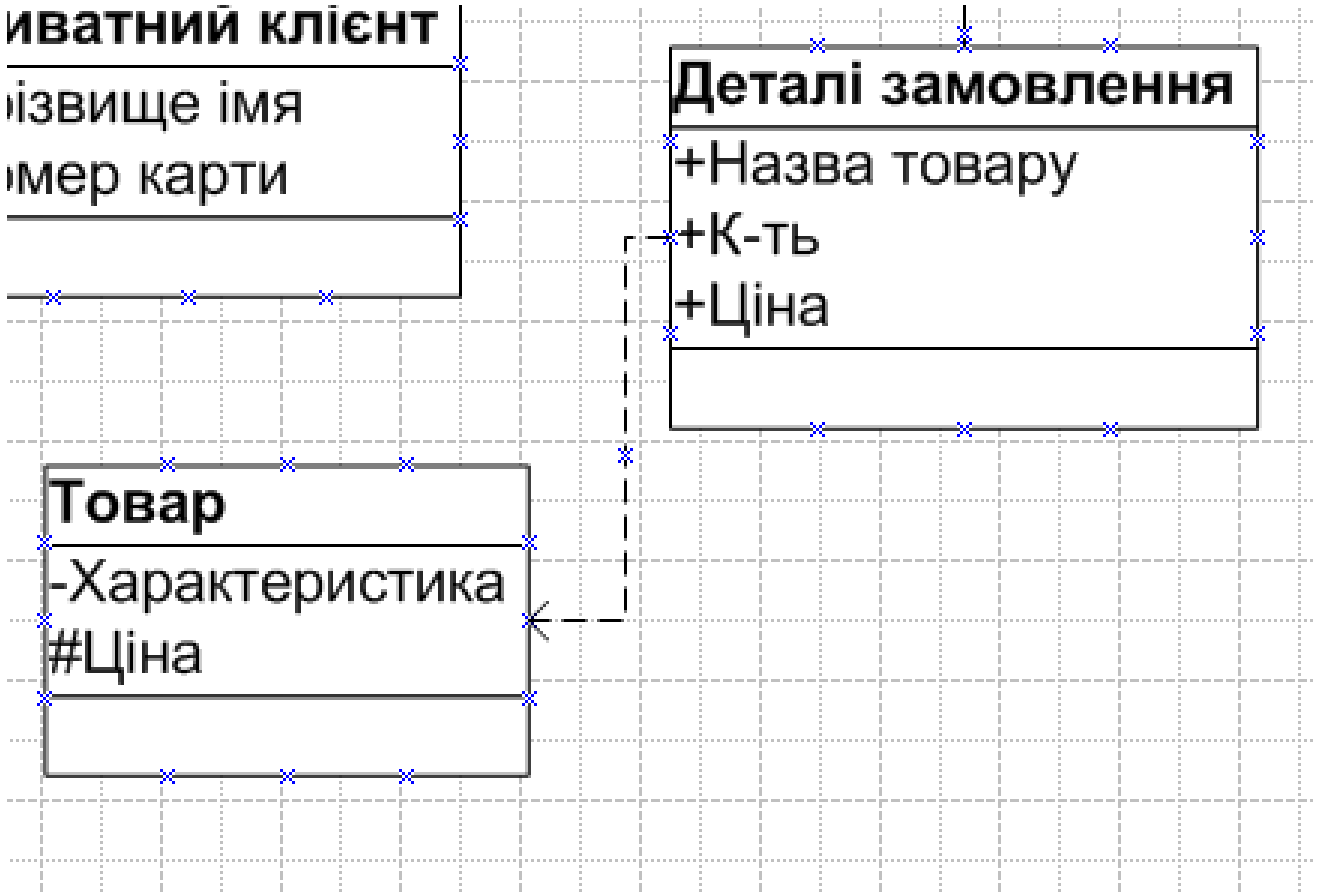
- Датчик температури
- Відеокамера
- Форма вводу
- Кнопка

Відношення між класами сутності



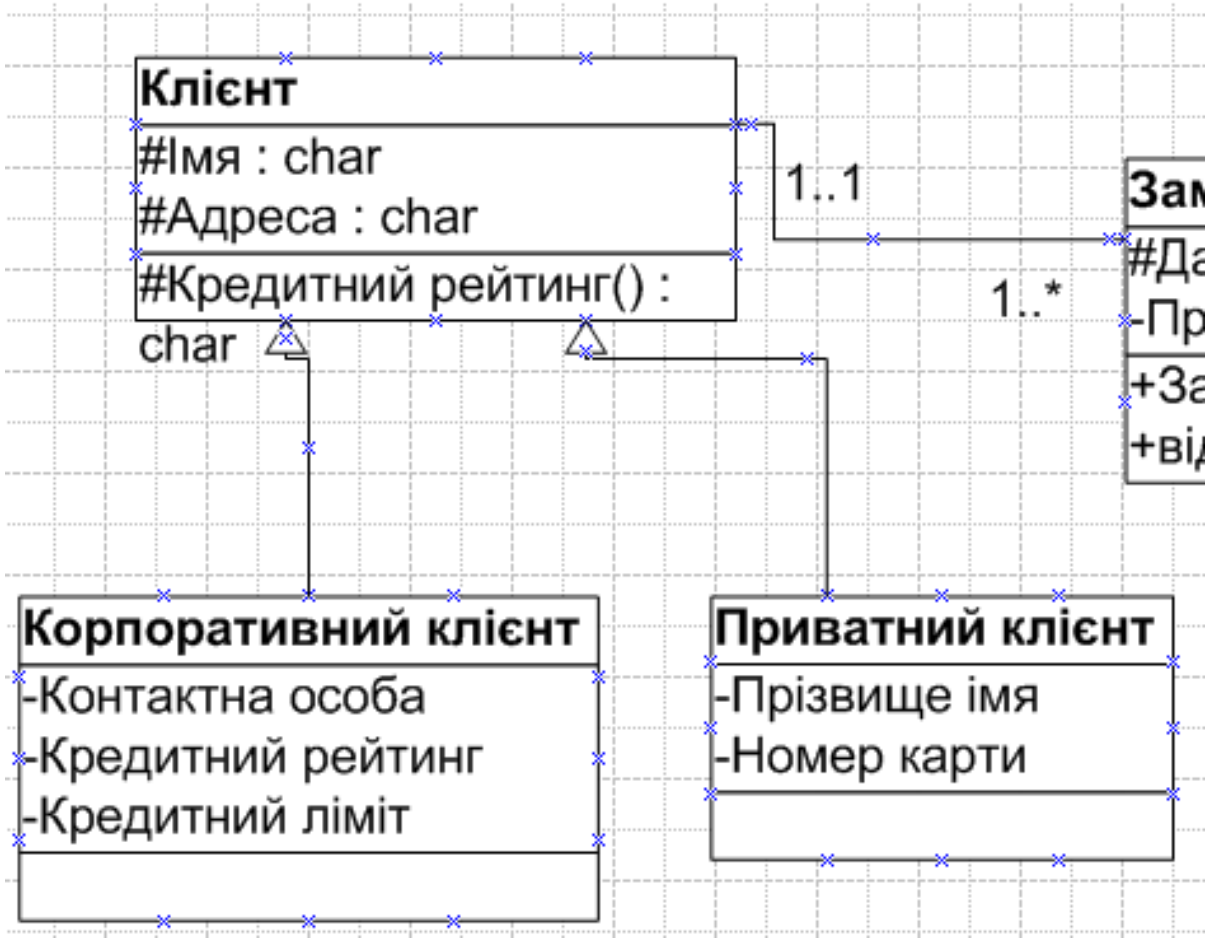
Залежність

Зміни у специфікації одного елемент впливають на елемент, що його використовує



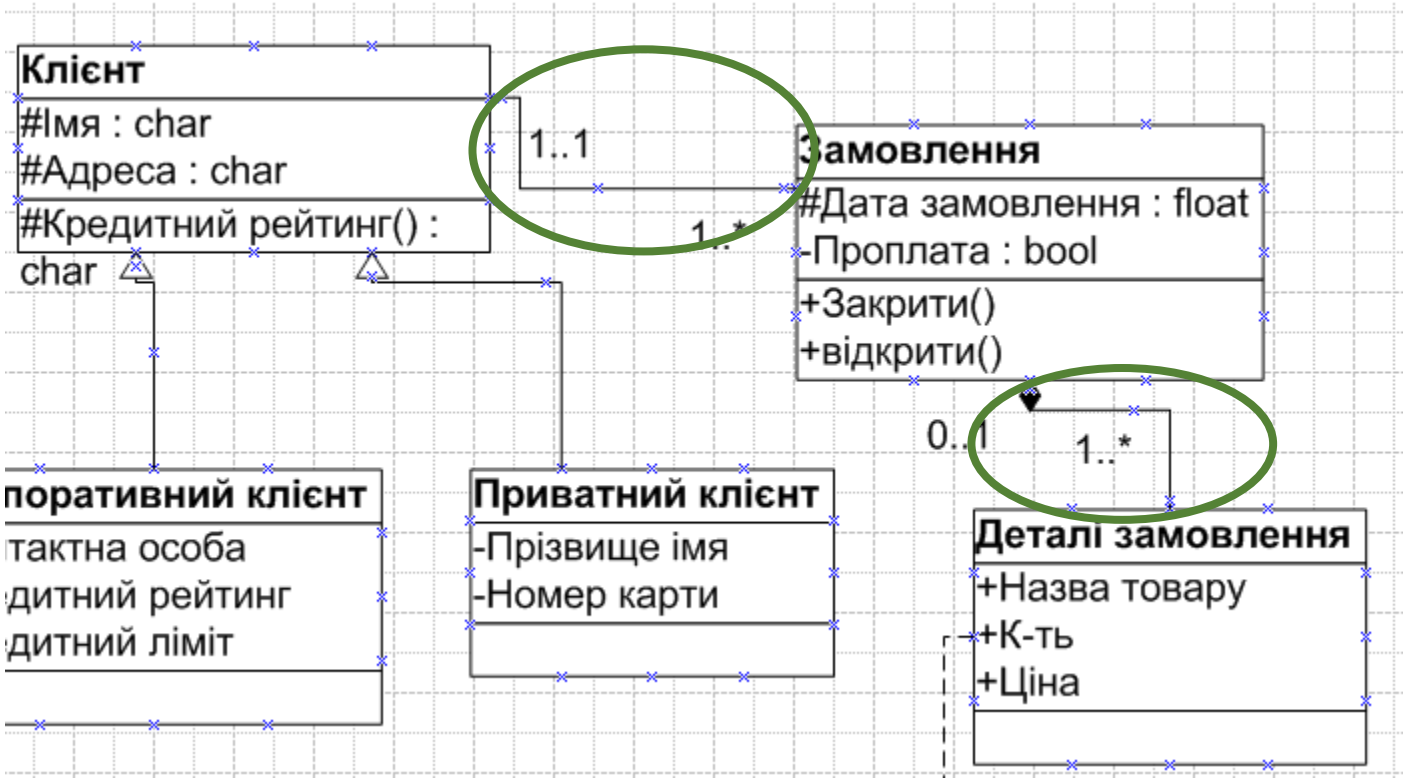
Узагальнення / наслідування

- Відношення між загальною сутністю та його частиною
- Між батьківським класом та класами-нащадками



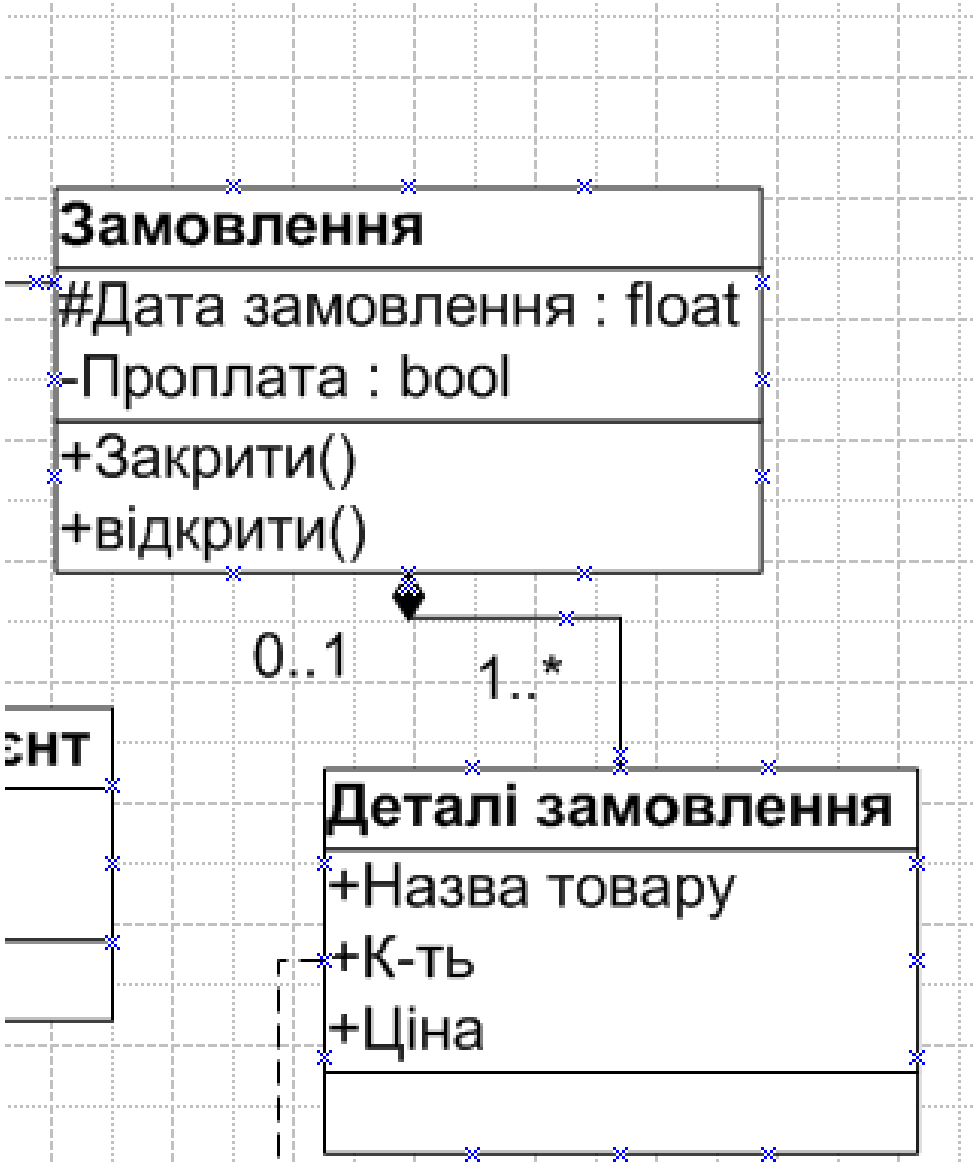
Асоціація

- Показує як об’єкти одного типу пов’язані з об’єктами іншого
- Напрям навігації показує прив’язку до певного об’єкта
- Множинність показує скільки об’єктів може приймати участь у даній асоціації

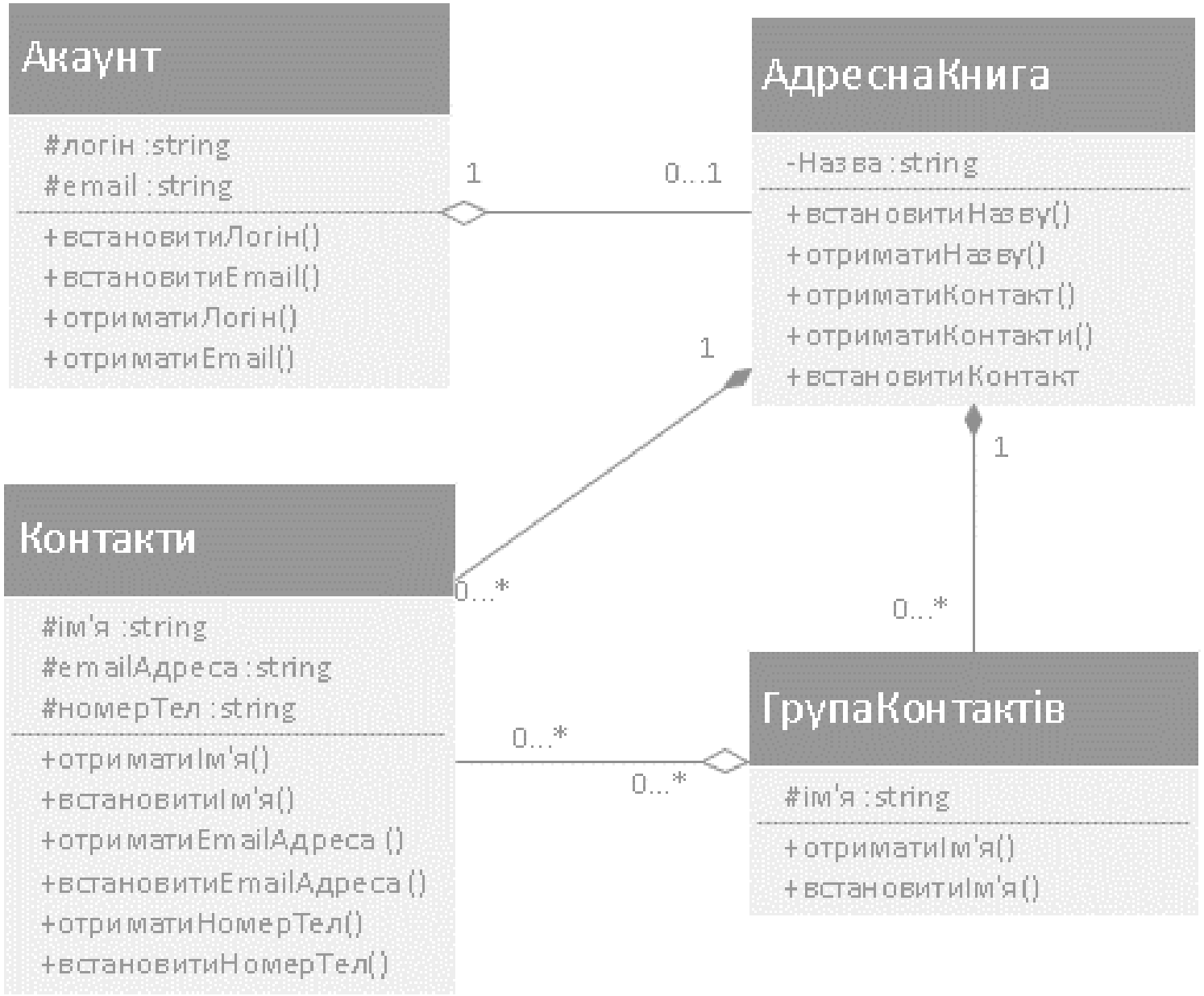


Агрегація

Агрегування - тип асоціації між нерівноправними класами



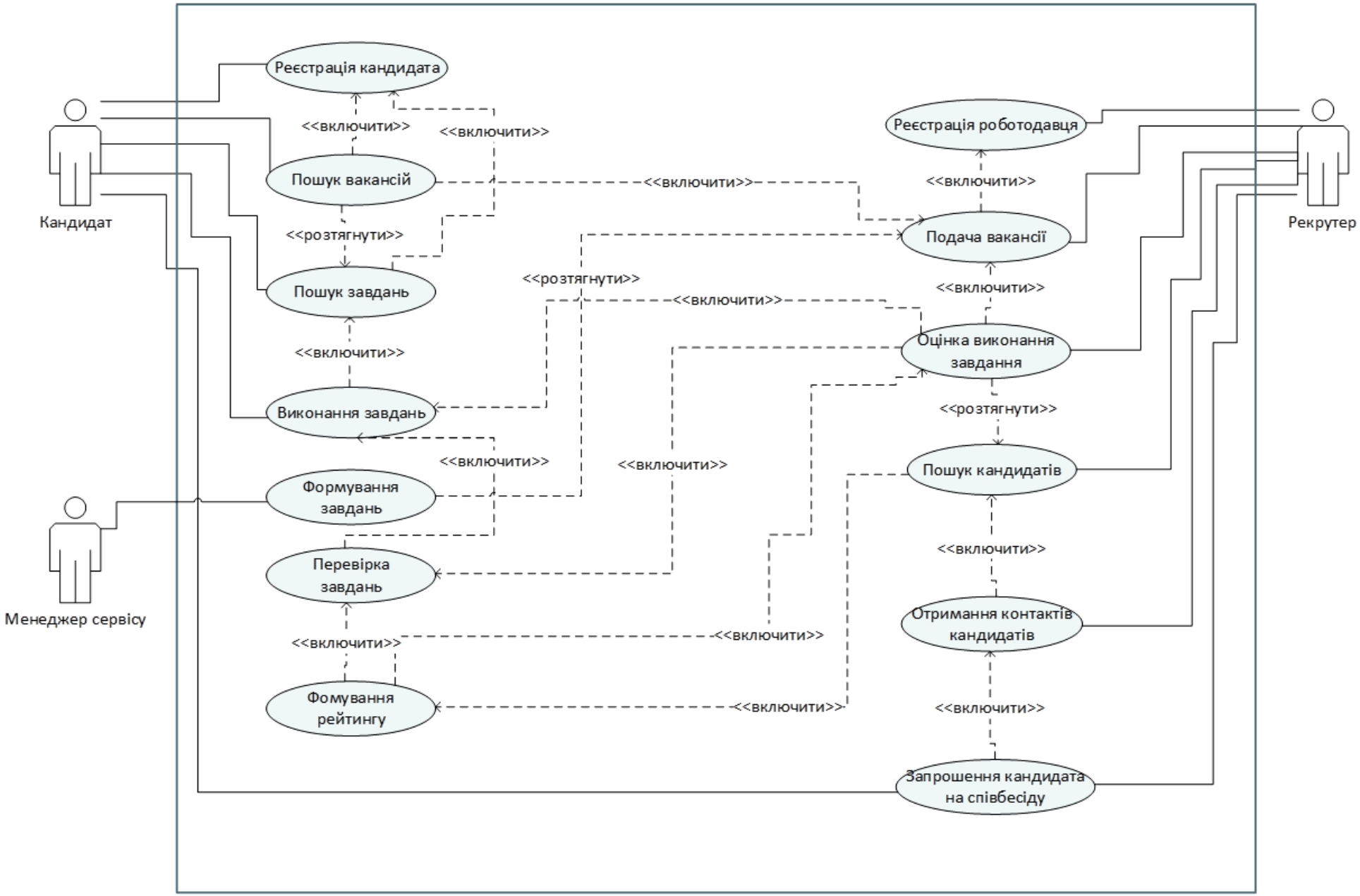
Приклад моделювання класів сутності



Методика моделювання

- Обрати прецедент
- Виявити об’єкти
- Змоделювати класи сутності
- Встановити зв'язки між класами
- Змоделювати класи граничні та керування
- Спроектувати стани об'єктів
- Спроектувати поведінку керуючих класів

UseCase модель



Приклад діаграми класів