Лекція 5

Моделювання бізнес-процесів

Базовий алфавіт Activity Diagram

Activity Diagram складається з невеликої кількості елементів, але кожен має чітке значення. Це фундаментальні блоки для моделювання бізнес-процесів та алгоритмів.

Початковий вузол

Чорне зафарбоване коло ●. Точка входу, звідки починається процес. На діаграмі зазвичай лише один старт.

Дія (Action)

Прямокутник із заокругленими кутами. Атомарний крок процесу. Називається: Дієслово + Іменник (наприклад, "Перевірити баланс").

Потік управління

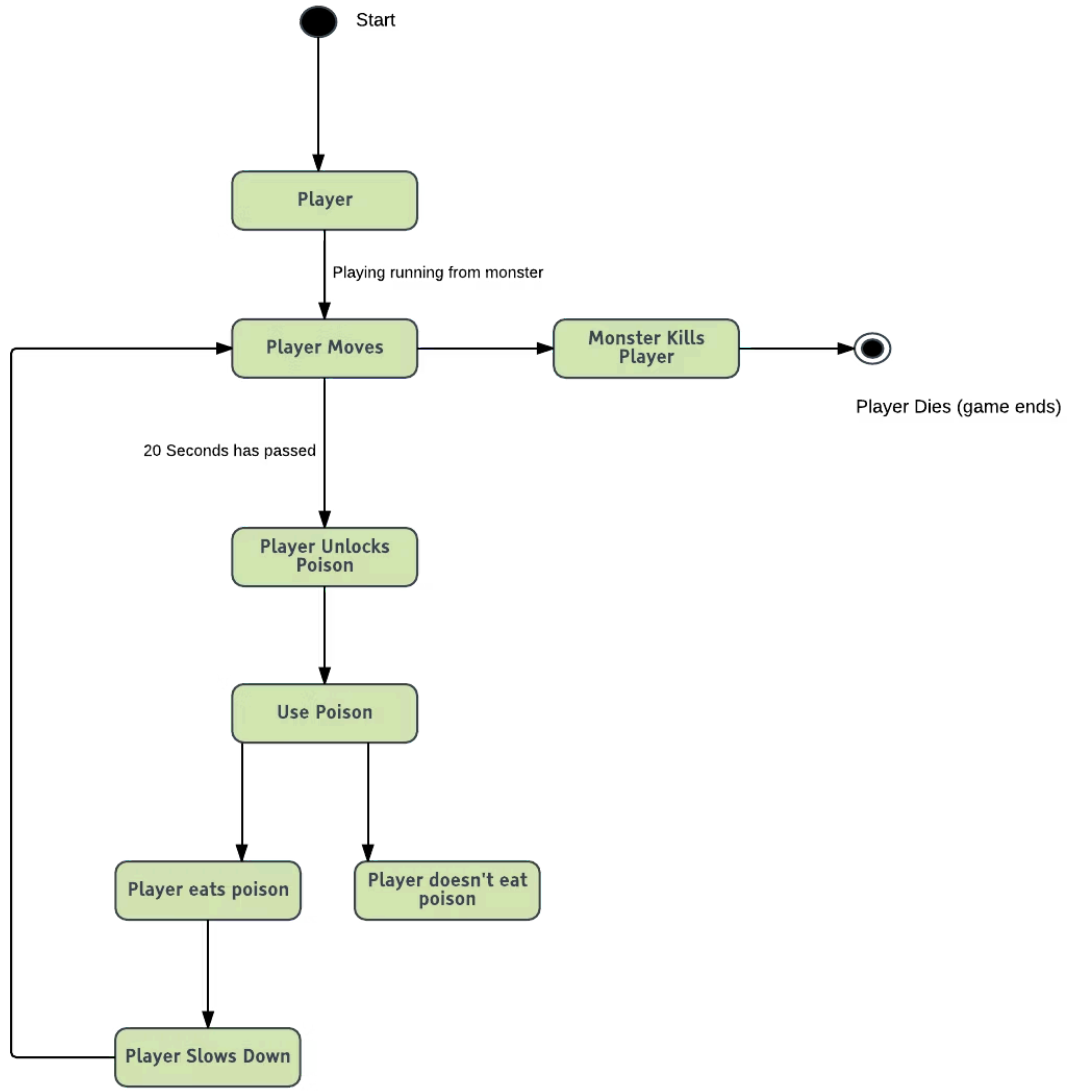
Суцільна стрілка ⊠. Передача естафети між діями. Вказує, яка дія виконується наступною.

Кінцевий вузол

"Bullseye" ◉. Повна зупинка процесу. Кінців може бути декілька для різних сценаріїв завершення.

Завершення потоку

Коло з хрестиком ⊗. Зупиняє лише одну гілку виконання, не вбиваючи весь процес при паралельному виконанні.



Activity Diagram vs Класична Блок-схема

Парадигма

Блок-схема: Процедурна (послідовна)

Activity Diagram: Об'єктно-орієнтована + Подійна

Паралельність

Блок-схема: Неможлива (однопотокова)

Activity Diagram: Підтримується (Multithreading)

Відповідальність

Блок-схема: Не показує виконавця

Activity Diagram: Показує через Swimlanes

Блок-схеми годяться для опису алгоритму сортування масиву. Для опису бізнес-процесу (де Бухгалтер і Склад працюють одночасно) потрібна Activity Diagram.

Правила хорошого тону

Дотримання цих принципів робить діаграми зрозумілими та професійними.

1 Принцип "Зліва-направо" або "Зверху-вниз"

Діаграма має читатися як текст. Не робіть змійку, де стрілки петляють по всьому екрану.

2 Уникайте перетину ліній

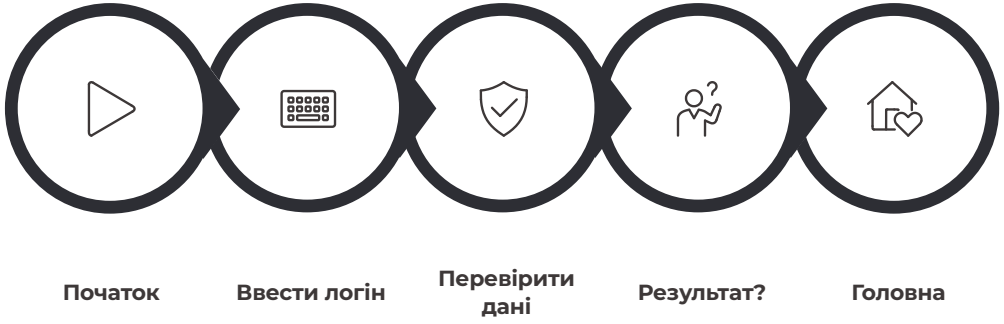
Це ознака поганого лейауту. Переміщуйте блоки так, щоб стрілки не перетиналися.

3 Один рівень абстракції

Не змішуйте дію "Відправити супутник на орбіту" з дією "Заварити каву". Або ви описуєте глобальний процес, або локальний.

Практичний приклад: Процес Login

Розглянемо простий вхід в систему, щоб побачити базові елементи в дії.



Це найпростіша лінійна структура з розгалуженням. Якщо валідація успішна → показуємо головну сторінку. Якщо помилка → показуємо повідомлення і повертаємося до введення даних.

Послідовність дій

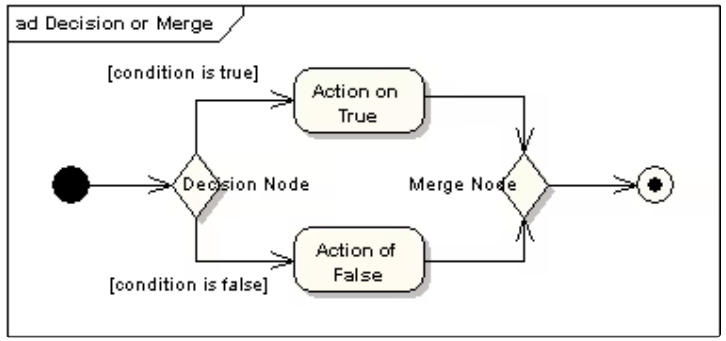
1. Старт процесу ●
2. Ввести логін та пароль
3. Валідувати дані
4. Розгалуження за результатом

Можливі результати

- Успіх: Показати головну сторінку → Кінець ◉
- Помилка: Показати повідомлення → Повернутися до введення

Умовне виконання: Decision & Merge

Для відображення альтернативних шляхів (логіка if/else або switch) використовується ромб — ключовий елемент для моделювання прийняття рішень.



Вузол розгалуження (Decision Node)

Символ: Ромб з одним входом і декількома виходами.

Суть: Точка вибору. Потік піде тільки по одній із гілок.

Вартові умови: Біля кожної вихідної стрілки у квадратних дужках [...] пишеться умова. Наприклад: [Баланс > 0] та [Баланс <= 0].

> Умови мають бути взаємовиключними і покривати всі варіанти. Якщо умова не виконана і немає гілки [else], потік застрягне.

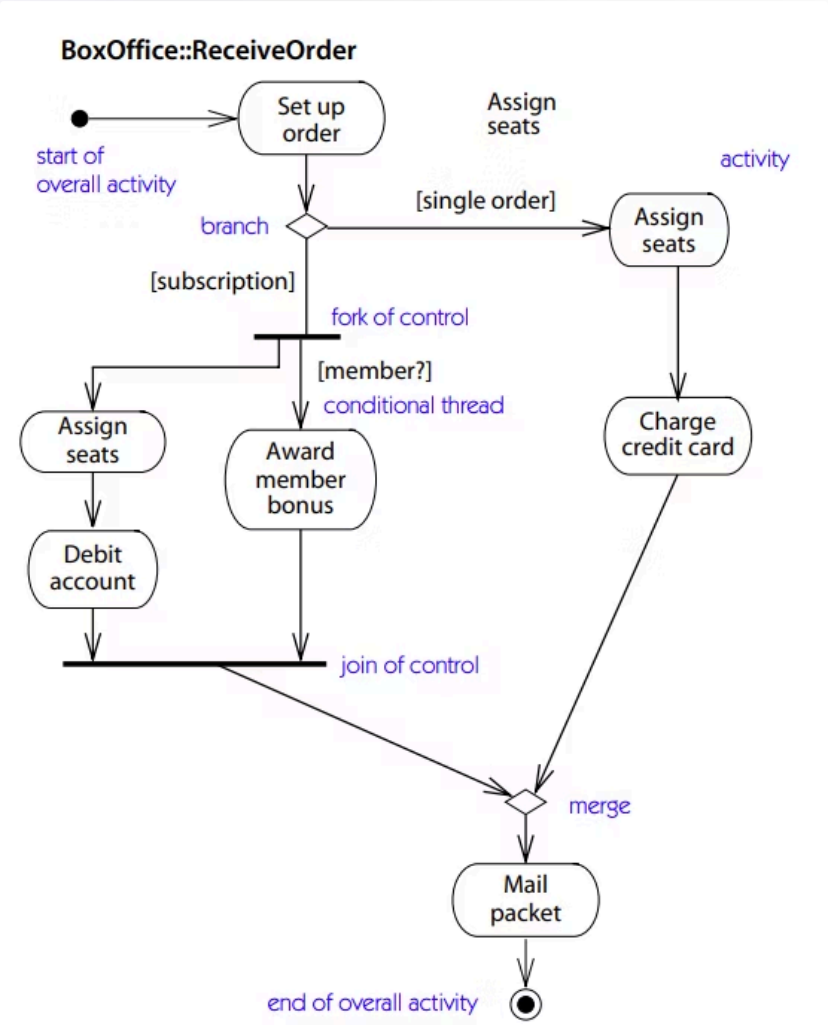
Вузол злиття (Merge Node)

Символ: Той самий ромб, але з декількома входами і одним виходом.

Суть: Об'єднує альтернативні шляхи назад в один потік.

Логіка: Не чекає. Щойно потік приходить по будь-якій вхідній стрілці, він одразу проходить далі. Це логічне "АБО".

Типовий патерн: Ромб розвів потоки (If), потім дії, потім інший ромб звів їх назад (End If).



Паралельне виконання: Fork & Join

Для відображення багатопоточності (Multithreading) або одночасних бізнес-процесів використовується Лінійка синхронізації — товста жирна лінія (горизонтальна або вертикальна).

Fork Node (Розділення)

Символ: Один вхід → Жирна лінія → Багато виходів

Суть: Клонування потоку

Логіка: Коли потік доходить до лінії, він розділяється на N незалежних потоків, які стартують одночасно

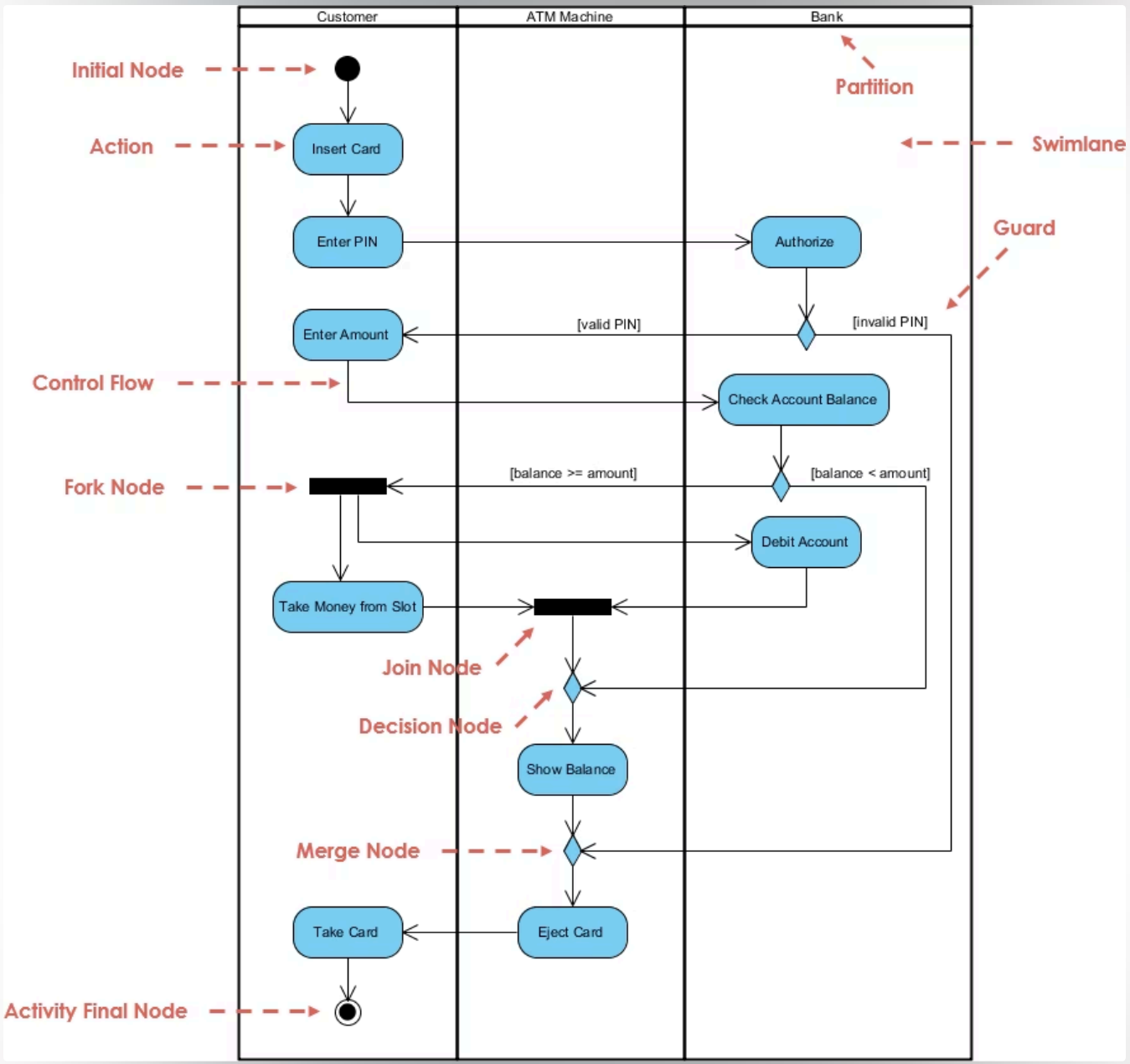
Join Node (Об'єднання)

Символ: Багато входів → Жирна лінія → Один вихід

Суть: Бар'єр синхронізації

Логіка: Чекає всіх. Потік піде далі тільки тоді, коли активність завершиться на всіх вхідних гілках. Це логічне "І"

> Приклад: При оформленні замовлення одночасно Склад пакує товар, Бухгалтерія формує чек, і Клієнту летить email. Ми не можемо відправити посилку, доки не виконаються обидві умови: "Товар упаковано" І "Оплату отримано".



Поняття Доріжки (Swimlane)

Swimlane — це графічна зона на діаграмі, яка обмежує відповідальність певного учасника процесу.

Типи розташування

- **Вертикальні доріжки:** Найпопулярніші. Актори підписані зверху. Час йде зверху вниз.
- **Горизонтальні доріжки:** Актори підписані зліва. Час йде зліва направо.

Кого ставити в заголовок доріжки?

Клас / Об'єкт

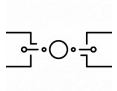
Наприклад, OrderController, Database. Це корисно для проєктування архітектури коду.

Роль / Актор

Наприклад, Менеджер, Клієнт. Корисно для бізнес-аналізу.

Організаційна одиниця

Наприклад, Відділ кадрів, Склад.



Зовнішня система

Наприклад, Google API, Банк.

Якщо ви малюєте діаграму для розробників, використовуйте назви класів або компонентів (Frontend, Backend, DB). Якщо для замовника — назви посад.

Правила малювання Swimlanes

1	**Монолітність дії** Дія (Action) має повністю знаходитися в межах однієї доріжки. Помилка: Малювати прямокутник на лінії розділення. Не можна бути "трошки вагітним". Відповідальний завжди один.
2	**Перетин кордонів** Стрілки (Control Flow) можуть і повинні перетинати лінії доріжок. Це показує передачу управління. Приклад: Дія [Замовити] (у доріжці Клієнта) → стрілка через кордон → Дія [Обробити] (у доріжці Сервера).
3	**Логічні вузли** Ромби та Лінійки теж мають належати комусь. Зазвичай їх малюють у тій доріжці, де приймається рішення.

Практичний приклад: Онлайн замовлення

Розіб'ємо процес на три доріжки: Клієнт, Платіжна система, Склад.

Ця схема чітко показує, що Клієнт не може сам собі [Відправити замовлення]. Кожен учасник має свою зону відповідальності.

Клієнт

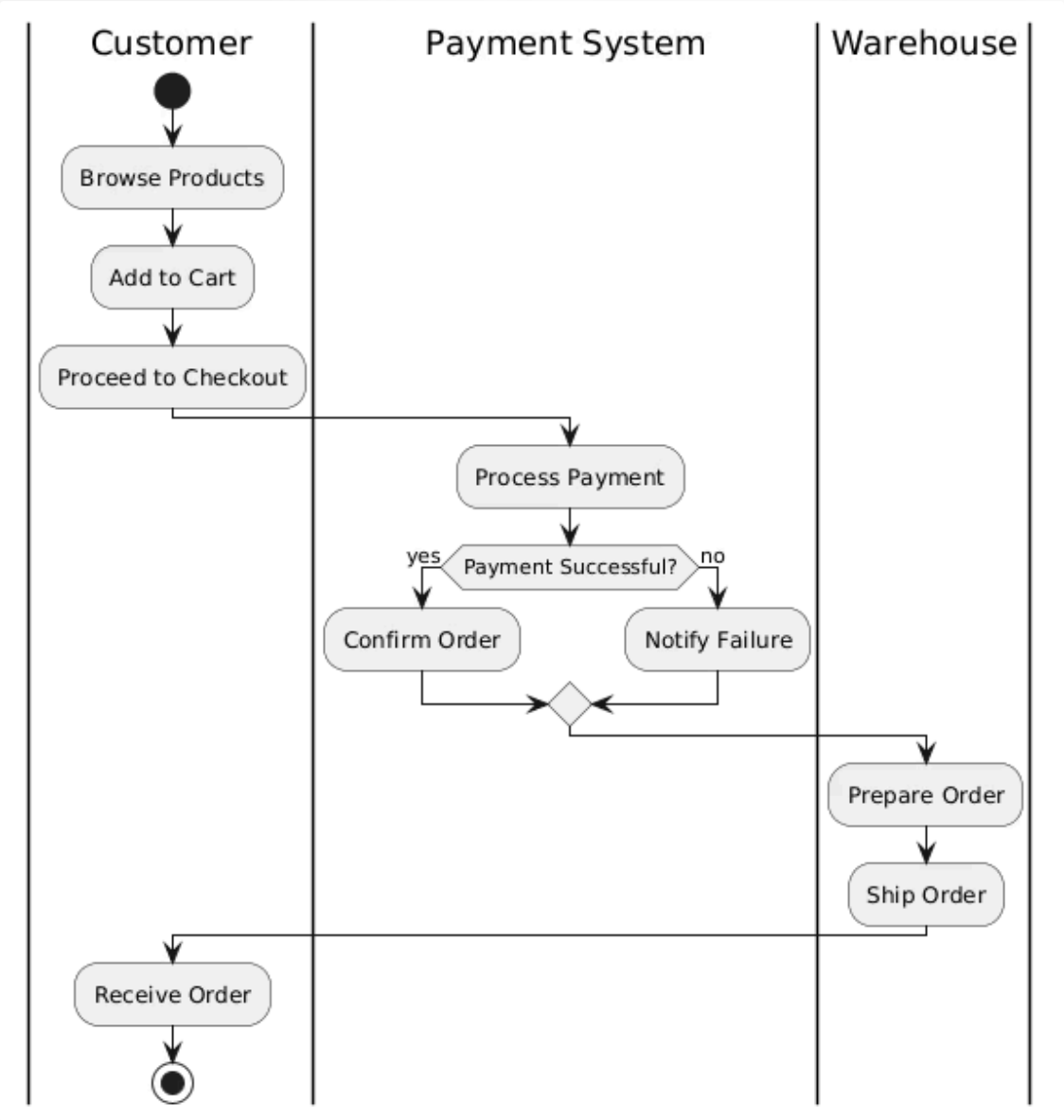
Переглянути товари → Додати в корзину → Оплатити → Отримати замовлення

Платіжна система

Обробити платіж → Підтвердити замовлення або повідомити про помилку

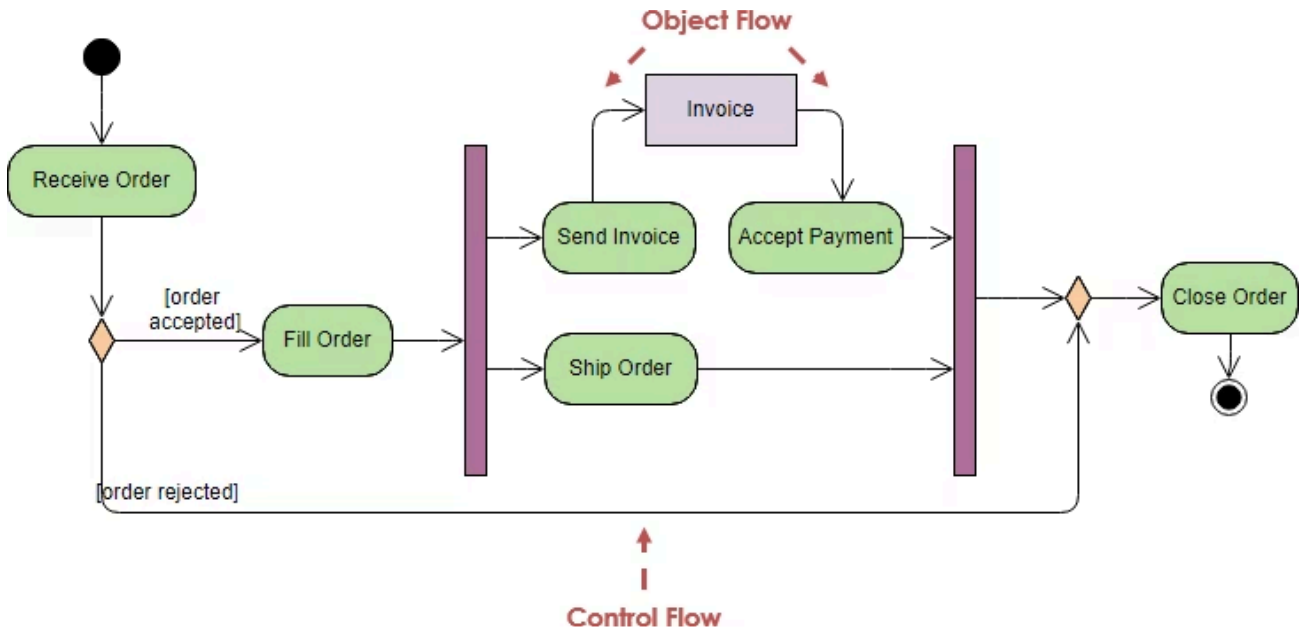
Склад

Підготувати замовлення → Відправити замовлення



Вузол об'єкта (Object Node)

Щоб показати дані на діаграмі, ми використовуємо новий символ, який відрізняється від дії.



Характеристики Object Node

- **Символ:** Звичайний прямокутник (не заокруглений!)
- **Підпис:** Назва класу або документу (наприклад, Замовлення, Рахунок, Товар)
- **Де малювати:** Між двома діями
- **Схема:** [Дія А] ⊠ Прямокутник "Дані" ⊠ [Дія Б]

Сенс використання

Дія А породжує ці дані, а Дія Б не може початися, доки не отримає їх.

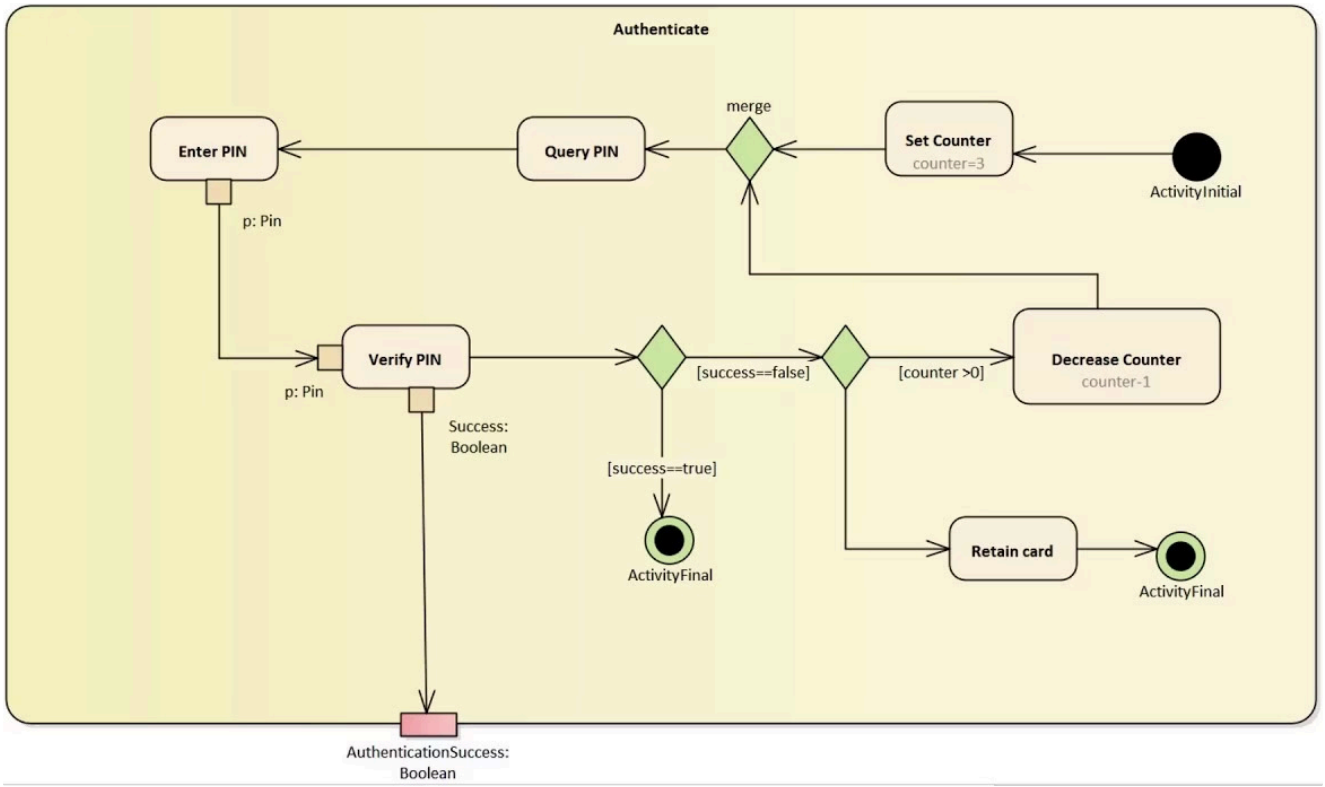
> Приклад: [Сфотографувати] ⊠ Фото (JPG) ⊠ [Обробити фільтром]. Без файлу JPG фільтр не запуститься, навіть якщо управління передано.

Піни (Input/Output Pins)

У сучасних нотаціях UML (починаючи з версії 2.0) замість окремих прямокутників часто малюють Піни. Це ближче до програмування.

Вхідний пін (Input Pin)

Квадратик зліва (куди входить стрілка). Це Аргументи функції



Вихідний пін (Output Pin)

Квадратик справа (звідки виходить стрілка). Це Return Value

Потік управління vs Потік об'єктів

Це важливий нюанс для глибокого розуміння Activity Diagram.

Control Flow

Суцільна стрілка. Несе "Токен управління". Це просто сигнал "Старт".

Object Flow

Теж стрілка, але з'єднує об'єкти. Несе "Токен даних". Це контейнер з інформацією.

1 **Правило виконання**

Дія запускається тільки тоді, коли прийшов сигнал управління (черга дійшла).

2 **Необхідність даних**

І прийшли всі необхідні дані (всі вхідні піни заповнені).

Якщо потік управління прийшов, а даних (об'єкта) ще немає — дія чекає (блокується). Це важливо для моделювання асинхронних процесів (наприклад, чекаємо відповідь від сервера).

Висновки до Лекції 5

Сьогодні ми розглянули інструмент, який покриває 80% потреб у моделюванні бізнес-логіки.

Найкращий спосіб

Activity Diagram — це найкращий спосіб показати алгоритм

Умови та паралельність

Використовуйте Ромби для умов (if) і Лінійки для паралельності (threads)

Відповідальність

Використовуйте Swimlanes, щоб показати відповідальних

Передача даних

Використовуйте Object Nodes, щоб показати, які документи передаються між відділами