Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інформаційні системи

Викладач: к.т.н., доц. Саяпіна Інна Олександрівна

План заняття:

- **Ділова гра «Відчуй себе бізнес-аналітиком»**
- **Нотація C4**
- **UML Class Diagram**

Ділова гра. Завдання 1.

- **Ситуація:**
- Ви група бізнес-аналітиків, якій треба провести інтерв'ю з клієнтом, який хоче, щоб Ваша компанія побудувала йому кавомашину.
- Зберіть початковий набір вимог. Пам'ятайте про усі типи вимог та властивості гарної вимоги.
- Напишіть питання, які Ви вважаєте треба задати, за посиланням:

1 Go to wooclap.com

2 Enter the event code in the top banner

Event code
XHDNBO

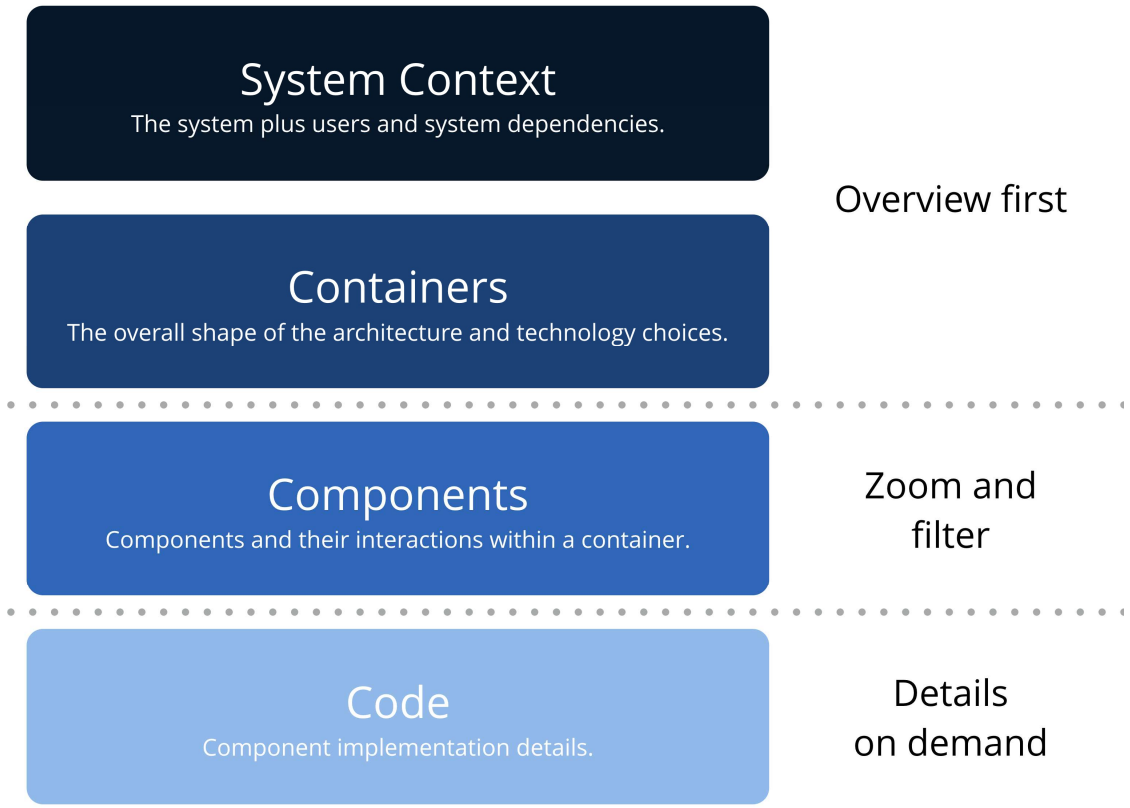
Ділова гра. Що ж клієнт насправді хотів?

1. Унікальна кавоварка, яка стане додатковим плюсом роботи в моїй компанії та допоможе найняти +20% співробітників.
2. Заварювання кави, чаю та какао з додаванням коров'ячого та рослинного молока.
3. Розмір звичайного банкомату.
4. Підключений до водопроводу з додатковою фільтрацією.
5. Замовлення через мобільний.
6. Розпізнавання клієнта, коли він/вона підходить до машини.
7. Оплата кредитними картками, готівкою або корпоративним акаунтом.
8. Автоматичне замовлення сервісу очистки, зміни фільтрів, додавання молока, кави.
9. Безконтактний інтерфейс з голосовим асистентом для замовлення напоїв
10. Персональний гороскоп для клієнта з інтернету та побажання гарного дня.

Ділова гра. Що ж клієнт насправді хотів?

1. Унікальна кавоварка, яка стане додатковим плюсом роботи в моїй компанії та допоможе найняти +20% співробітників.
2. Заварювання кави, чаю та какао з додаванням коров'ячого та рослинного молока.
3. Розмір звичайного банкомату.
4. Підключений до водопроводу з додатковою фільтрацією.
5. Замовлення через мобільний.
6. Розпізнавання клієнта, коли він/вона підходить до машини.
7. Оплата кредитними картками, готівкою або корпоративним акаунтом.
8. Автоматичне замовлення сервісу очистки, зміни фільтрів, додавання молока, кави.
9. Безконтактний інтерфейс з голосовим асистентом для замовлення напоїв
10. Персональний гороскоп для клієнта з інтернету та побажання гарного дня.

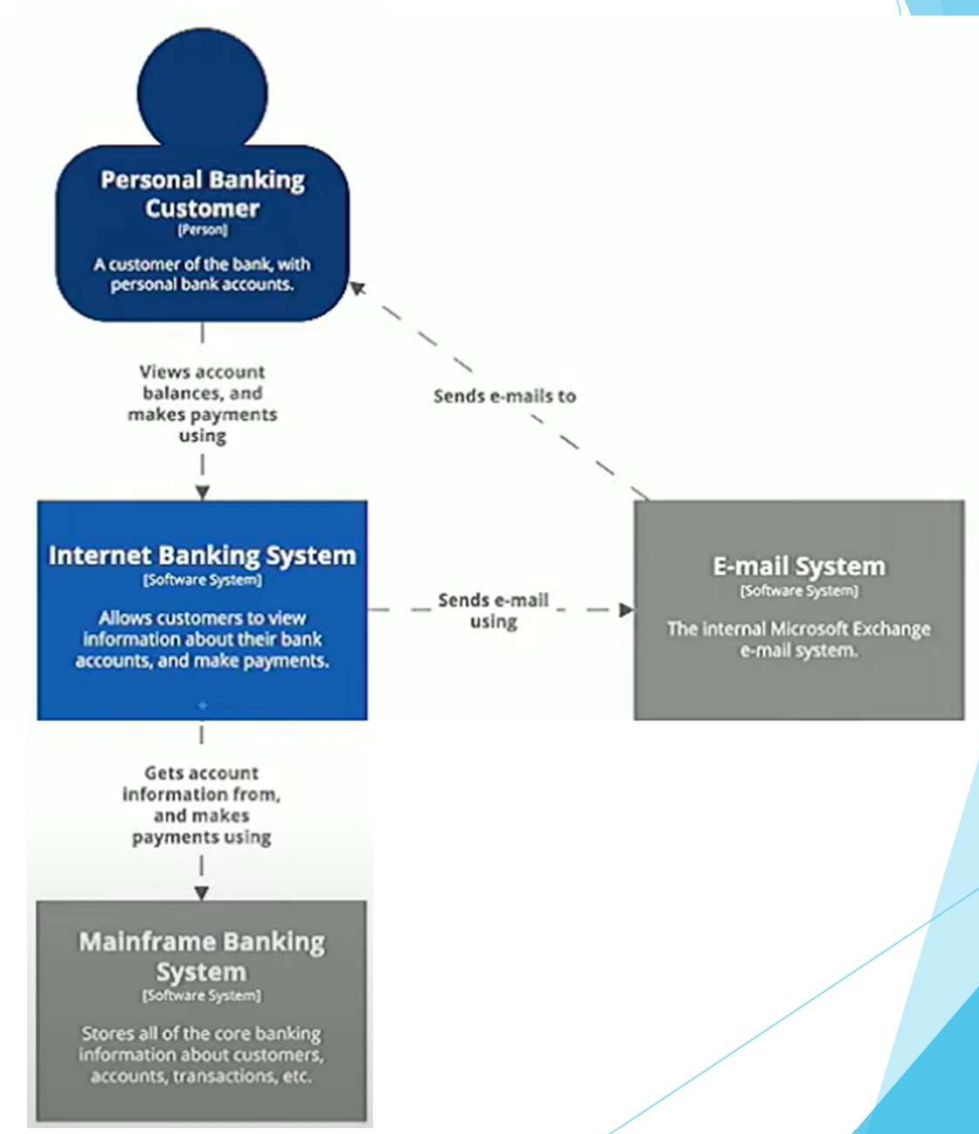
C4 model


System Context
The system plus users and system dependencies.
Containers
The overall shape of the architecture and technology choices.
Components
Components and their interactions within a container.
Code
Component implementation details.
Overview first
Zoom and filter
Details on demand

C4 model Example

- Level 1 Контекстна діаграма - це діаграма потоку даних вищого рівня, яка відображає потік даних між системою та зовнішніми компонентами. Вона відображає оточення системи, над якою ви працюєте, з точки зору систем, з якими вона взаємодіє, і людей, які її використовують.
- Створення:
  1. Визначте користувачів.
  2. Визначте зовнішні системи.
  3. Створіть єдиний прямокутник, який зображує вашу систему.
  4. Додайте зв'язок між системою, користувачами та зовнішніми системами.
  5. Напишіть змістовні коментарі щодо кожного компоненту.
- Приклад: Internet banking system

C4 model Example. Context Diagram

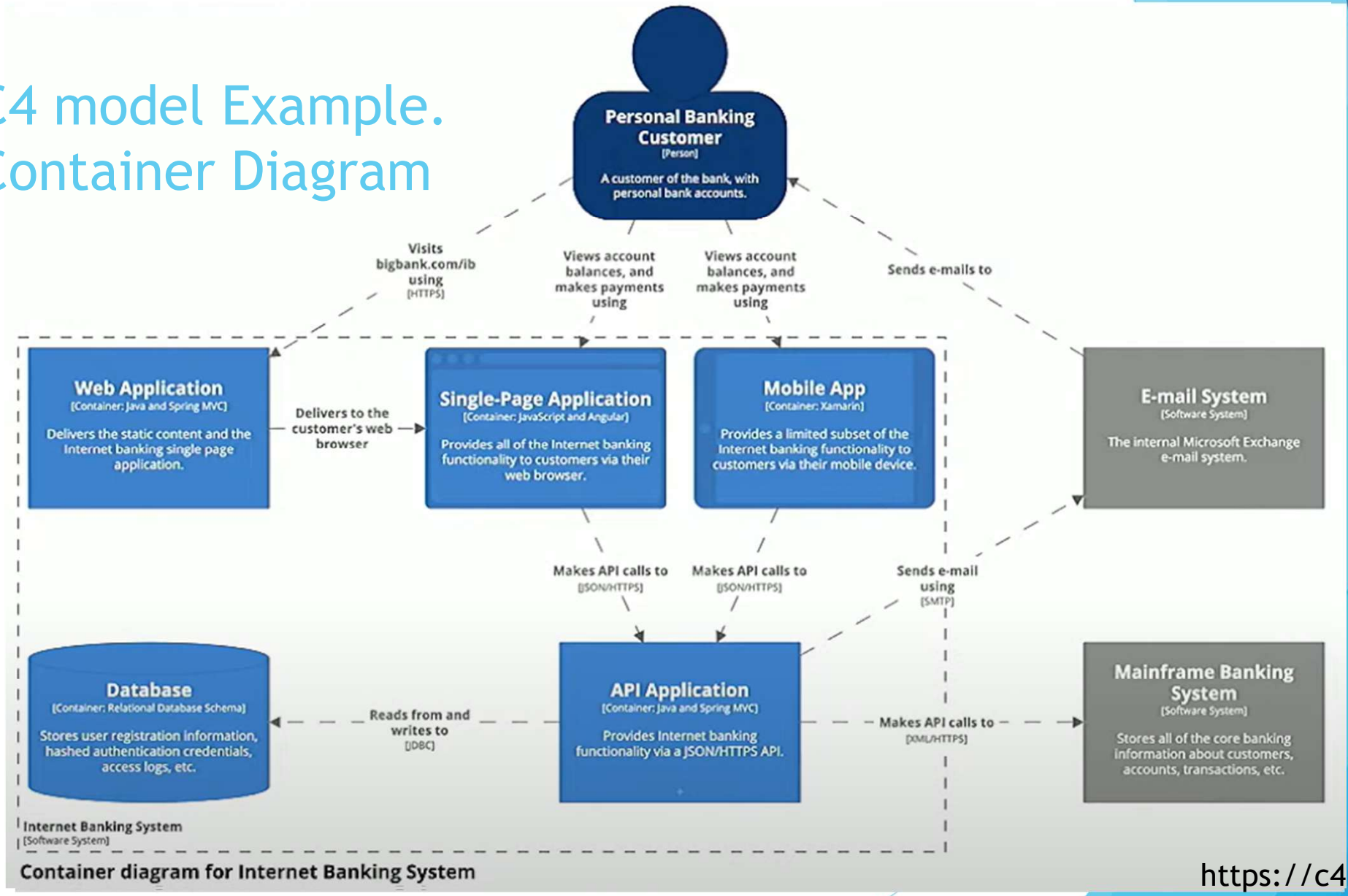


https://c4model.com/

C4 model
Container diagram

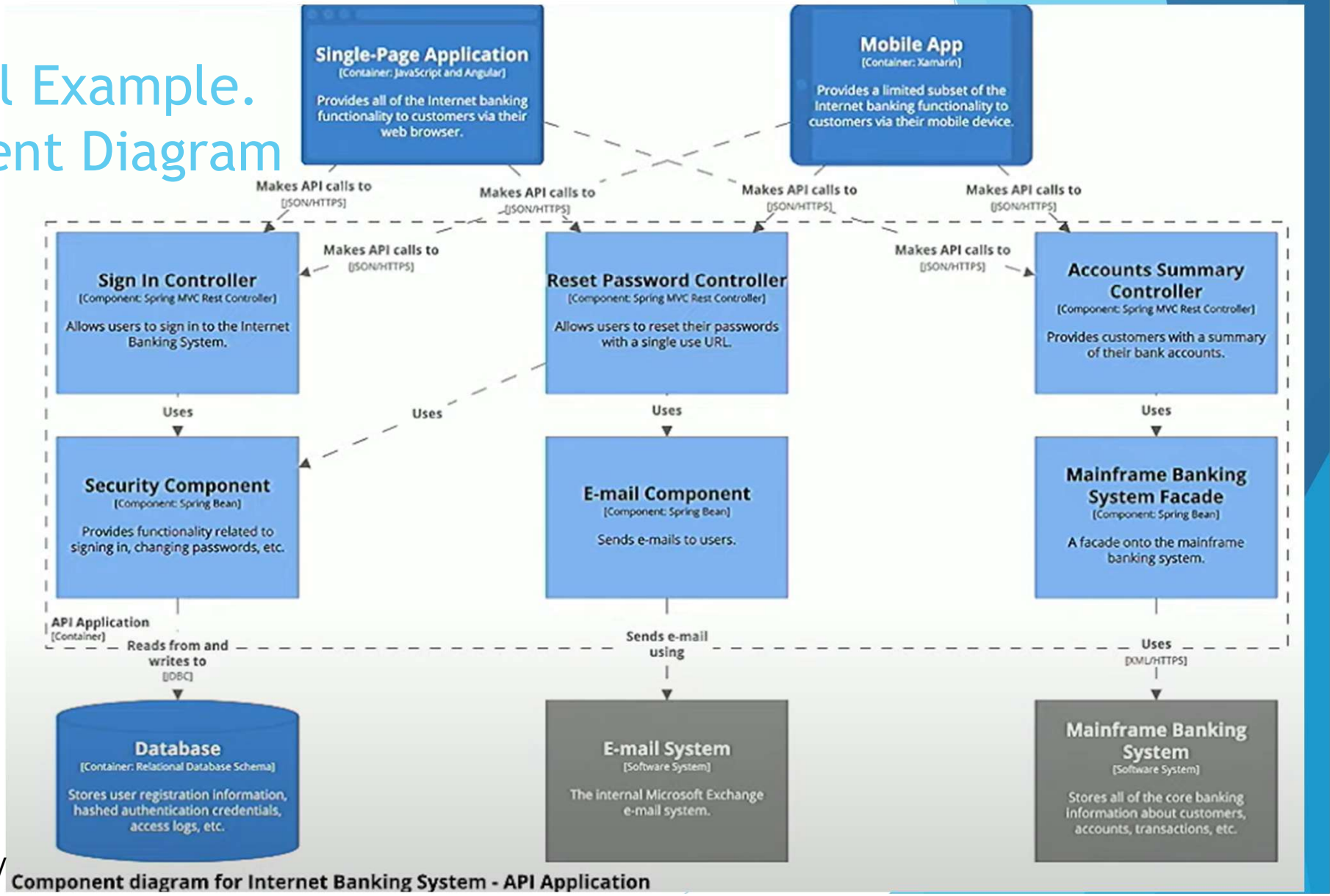
- Level 2 Діаграма контейнерів - дає уявлення про те, з яких елементів, що розгортаються, складається серверна частина, і як ці компоненти взаємодіють один з одним.
- Контейнери тут не обов'язково означають докер-контейнери. Контейнер — це будь-який об'єкт, що розгортається, або сховище даних з точки зору C4. Це може бути мобільна програма, веб-сайт, віртуальна машина, докер-контейнер, база даних або сховище об'єктів; все, що ви можете розгорнути.
- Створення:
  1. Визначте перелік сутностей: мікросервіси, сховища, зовнішні послуги.
  2. Помістіть їх у діаграму.
  3. Додайте коментарі про призначення кожного компонента та технології, яку він реалізує.
  4. Додайте з'єднання зі стрілками.
  5. Додайте значні позначки до кожної стрілки.
  6. Виберіть колір схеми.
  7. Створіть легенду.

C4 model Example. Container Diagram



https://c4model.com/

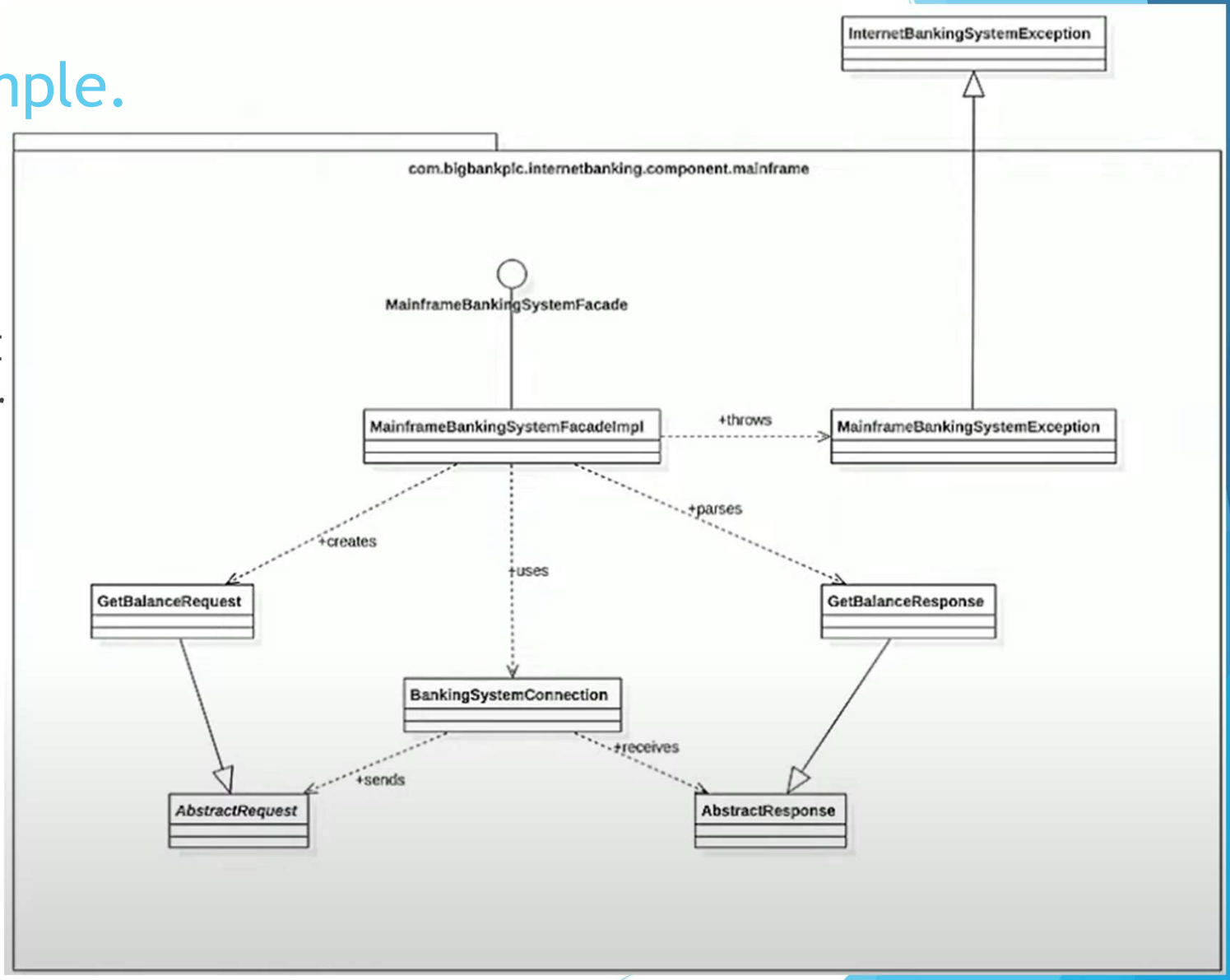
C4 model Example. Component Diagram



https://c4model.com/

C4 model Example. Code Diagram

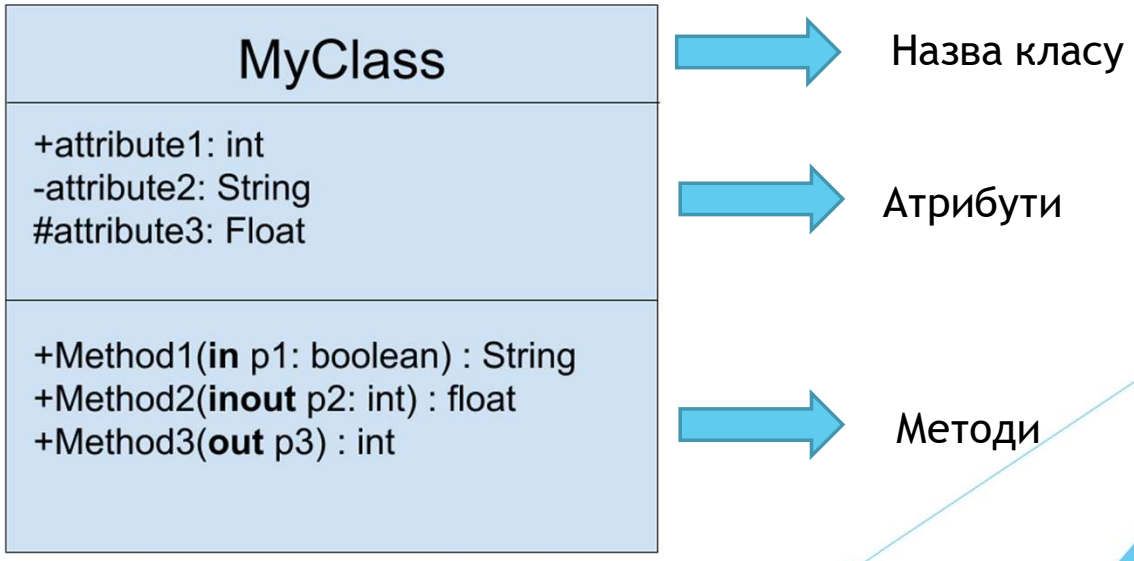
- Зазвичай ця діаграма генерується вже з IDE після написання коду.



https://c4model.com/

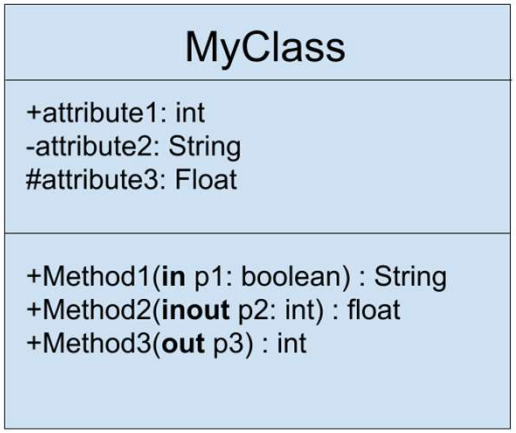
UML Class Diagram

- *Що таке клас?*
- Клас є шаблоном для створення об'єктів, що забезпечує початкові значення станів: ініціалізацію полів-змінних і реалізацію поведінки полів і методів.
- У діаграмі класів ім'я класу є обов'язковою інформацією до заповнення



UML Class Diagram

- *Що значить* **in, out, inout** *у поясненні метода?*
- Кожен параметр у методі може мати опис спрямованості методу: **in, out, inout.**
- Method1 використовує p1 як **вхідний** параметр і значення p1 якимось чином використовується методом, метод **не змінює** p1.
- Method2 приймає p2, як **параметр введення/виведення**, значення p2 якимось чином використовується методом і приймає вихідне значення методу, але сам метод також **може змінювати** p2.
- Method3 використовує p3 як **вихідний** параметр, іншими словами, параметр служить сховищем для вихідного значення методу.

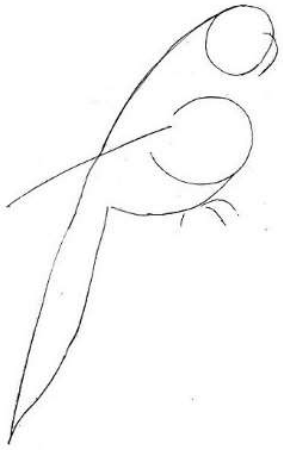


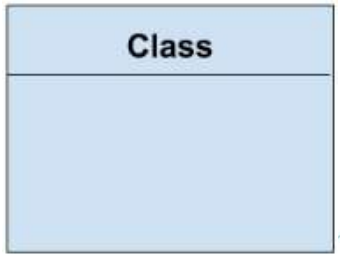
Використання UML Class Diagram у життєвому циклі розробки програмного забезпечення

- Можна використовувати діаграми класів на різних етапах життєвого циклу розробки програмного забезпечення і, як правило, поступово моделюючи діаграми класів із трьох різних точок зору в міру нашого просування за рівнями деталізації:
- Концептуальна перспектива
- Специфікаційна перспектива
- Імплементаційна перспектива

Використання UML Class Diagram у життєвому циклі розробки програмного забезпечення

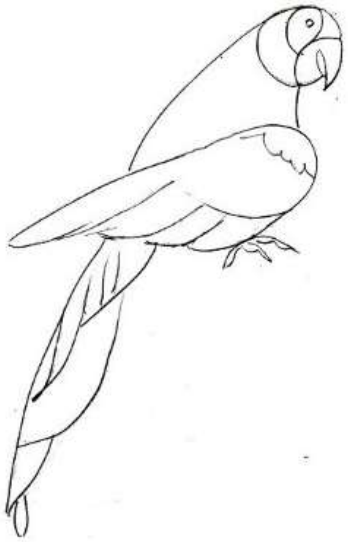
- **Концептуальна перспектива** — коли діаграми інтерпретуються як опис речей у світі. Таким чином, якщо ми беремо концептуальну перспективу, ми малюємо діаграму, яка представляє концепції в області, що вивчається. Ці концепції відносяться до класів, які їх реалізують. Концептуальна перспектива вважається незалежною від мови.
- Приклад: намалювати папугу: 1 етап

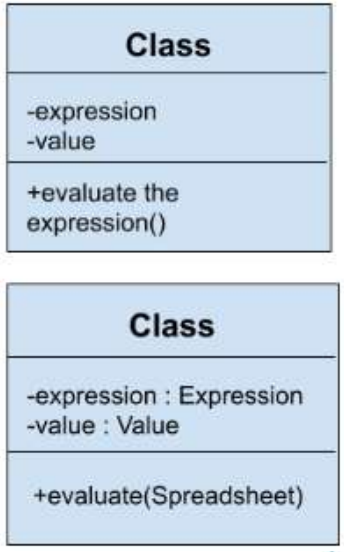




Використання UML Class Diagram у життєвому циклі розробки програмного забезпечення

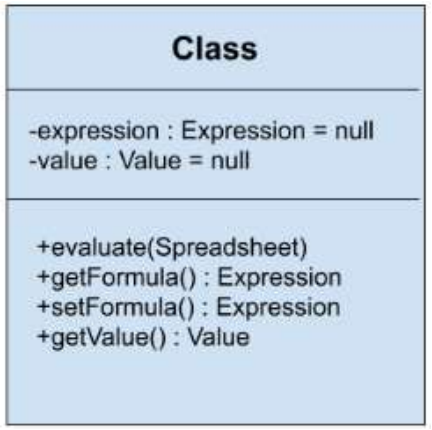
- **Специфікаційна перспектива** — це коли діаграми інтерпретуються як опис абстракцій програмного забезпечення або компонентів зі специфікаціями та інтерфейсами, але без прив'язки до конкретної реалізації.





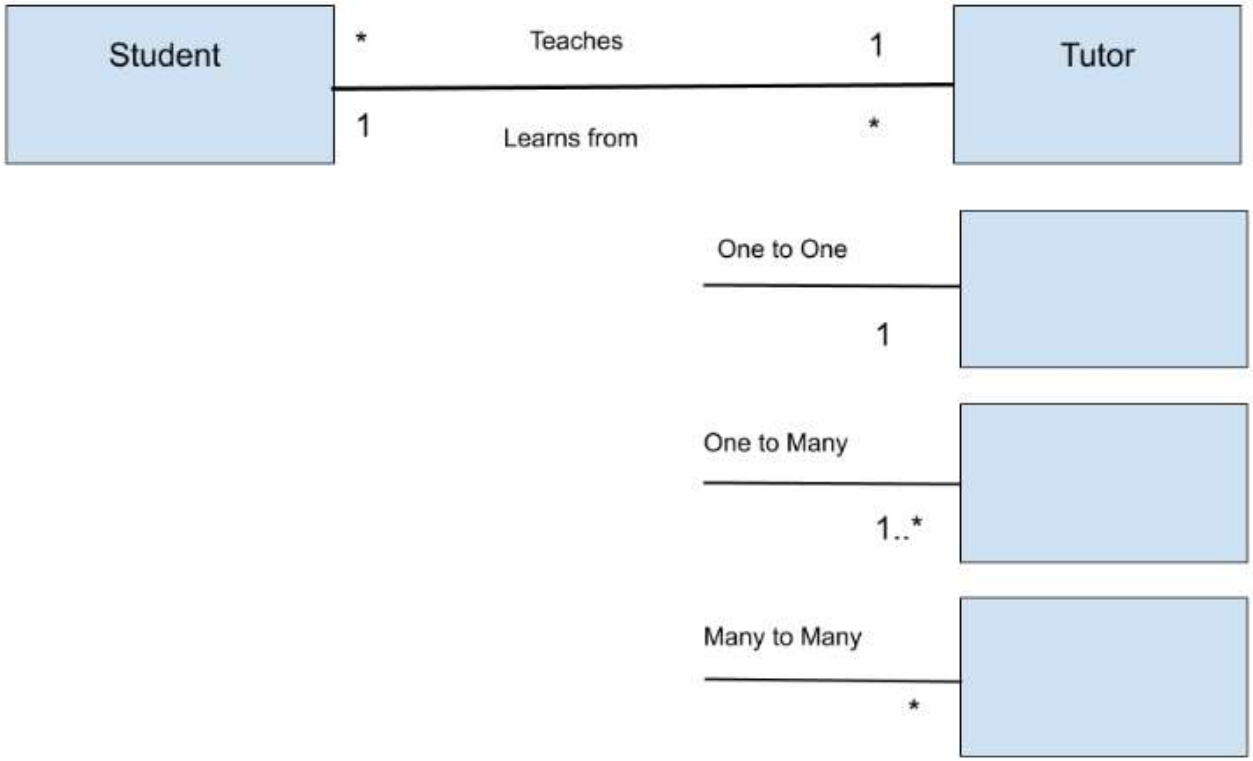
Використання UML Class Diagram у життєвому циклі розробки програмного забезпечення

- **Імплементаційна перспектива** — це коли діаграми інтерпретуються як опис реалізацій програмного забезпечення певною технологією та мовою.Таким чином, якщо ти береш імплементаційну перспективу,ти дивишся на реалізацію програмного забезпечення.



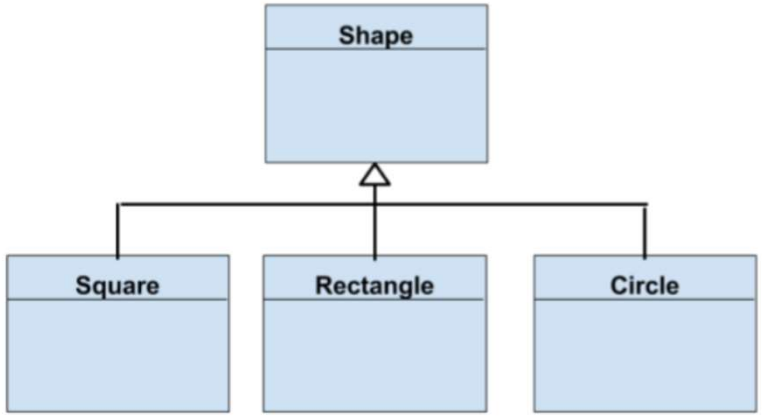
Типи відношень у UML Class Diagram

- **Асоціація**



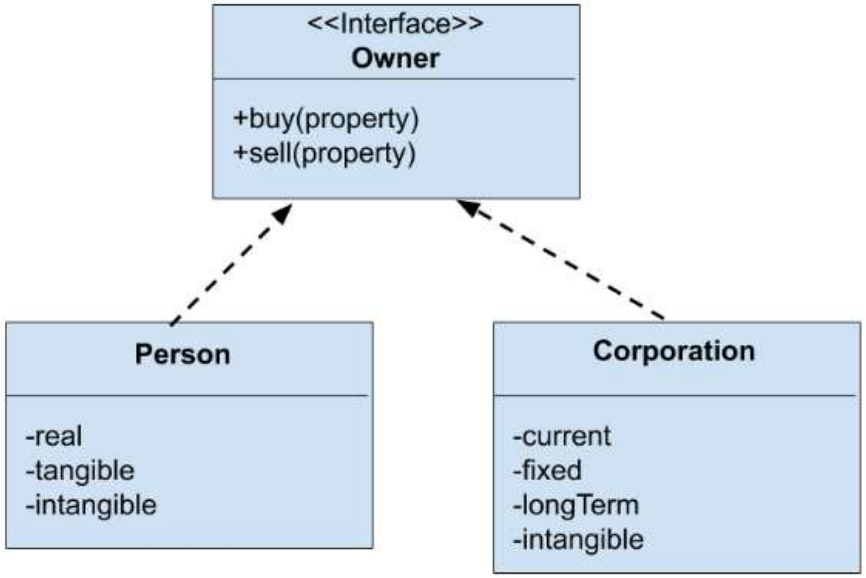
Типи відношень у UML Class Diagram

- **Узагальнення (наслідування)** це схематичне зображення відносин між батьківським класом та його спадкоємцями. Порожня стрілка завжди спрямована до батьківського класу.
- Можна зображувати успадкування як окремо кожного класу, і об'єднувати їх.
- Якщо успадкування походить від абстрактного класу, ім'я такого батьківського класу записується курсивом.



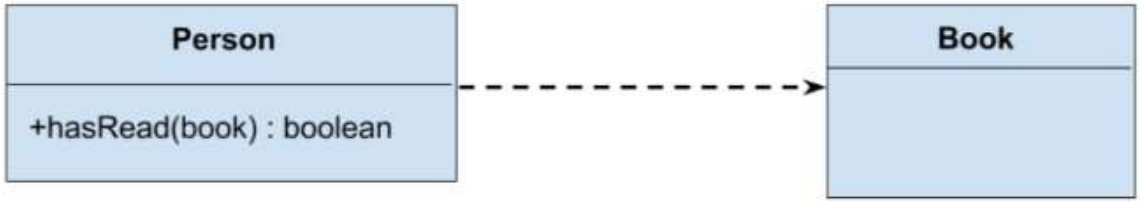
Типи відношень у UML Class Diagram

- Під **реалізацією** мається на увазі відношення інтерфейсу та об'єктів, що реалізують цей інтерфейс. Наприклад, інтерфейс Owner має методи для купівлі та продажу приватної власності, а відносини класів Person і Corporation, що реалізують цей інтерфейс, на діаграмі позначатимуться у вигляді пунктирної лінії зі стрілкою у напрямку до інтерфейсу.



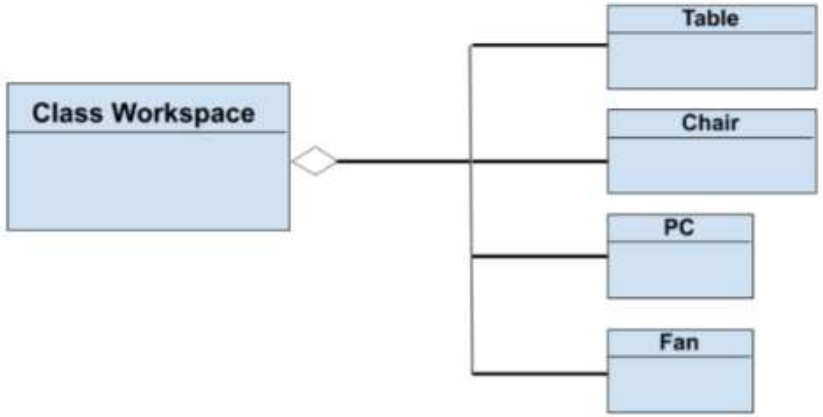
Типи відношень у UML Class Diagram

- Об'єкт одного класу може використовувати об'єкт іншого класу у своєму методі. Якщо об'єкт не зберігається у полі класу, такий вид міжкласових відносин моделюється як **залежність**.



Типи відношень у UML Class Diagram

- Агрегація - особливий тип відносин між класами, коли один клас є частиною іншого.



Типи відношень у UML Class Diagram

- **Композиція** - по суті, різновид агрегації, тільки в цьому випадку, класи, що є частиною іншого класу, знищують, коли знищується клас-агрегатор.