**UML diagram** – спосіб візуалізації аплікухи використовуючи набір діаграм різного типу. Ми можемо зрозуміти де які компоненти знаходяться, як виглядають і як взаємодіють між собою. UML має десь 9 типів діаграм, а в UML 2.0 їх вже порядка 13-14. Є дві гупи діаграм:

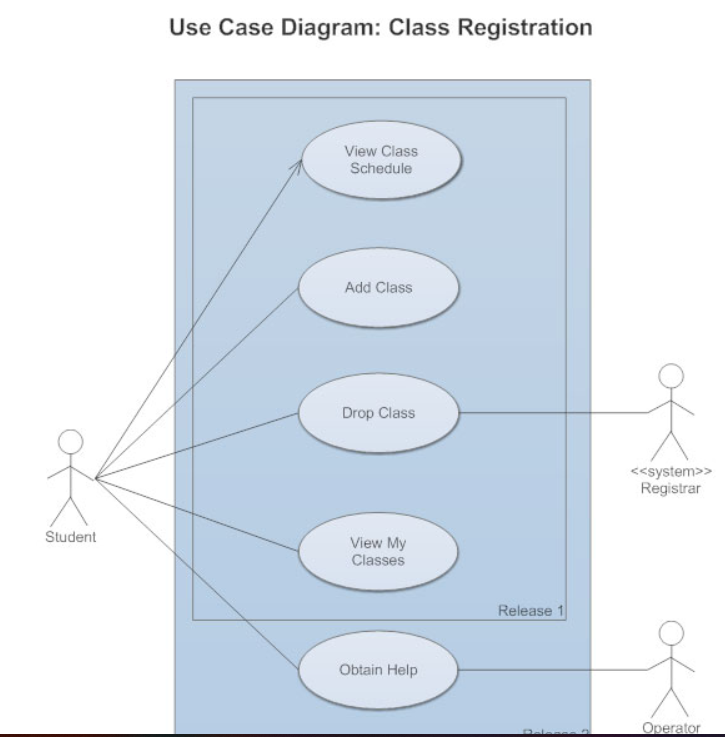
**Структурні**:

* Class diagram
* Package diagram (організує елементи по підгрупам шоб мінімізувати депенденсі)
* Object diagram (статитчна структура обєктів системи в певний час)
* Component diagram (описує організацію компонентів включно до коду і бінарників)
* Composite structure diagram (показує внутрішні частини класу)
* Deployment diagram (описує ресурси системи включно ноди, конекшени, компоненти)

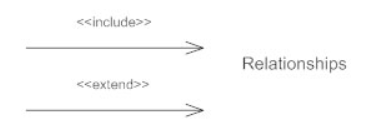
**Поведінкові**:

* Use case diagram
* Activiti diagram (показує флоу системи від одного екшена до іншого)
* Communication diagram (описує почергову взаємодію між обєктами)
* Sequence diagram (описує взаємодію обєктів при обміні меседжами)
* State diagram (описує стан системи при різних ситуаціях)
* Interaction diagram (комбінація activity i sequence diagram)

**Use Case diagram –** описує як веде себе система відносно функціональних реквайрментів використовуючи акторів які взаємодіють з системою і можливі для них кейси. Наприклад у нас є реєстрація на курси. Схематично система позначається в квадратику, актори малюються як человечкі, а юзкейси в кружечку.

  
На схемі видно, що може робити кожен з акторів і що Оператор допомагає поза системою.

**Include** означає, що юзкей потрібен для іншого юзкейса що завершити таск.

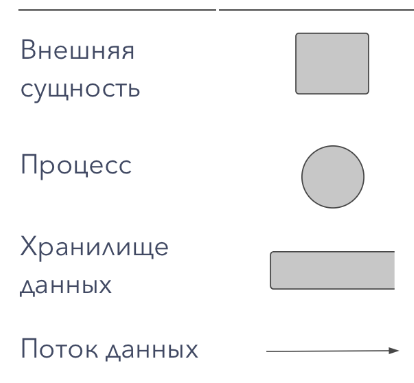


**Extend –** каже, що може бути альтернативний юзкейс.

**Data Flow Diagram** – графічно зображає як дані проходять через систему, де які інпути\оутпути і як дані зберігаються.

Нотації:

Любі обєкти які не входять в нашу систему але які є джерелом даних для системи. Клієнт чи інша бд наприклад.

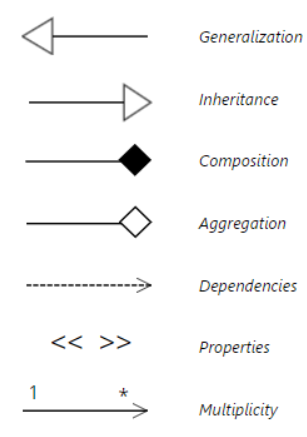


Функція чи послідовність дій які треба зробити шоб обробити дані. Має мати хоча б один in i out даних або з іншим процесом, або з храніліщем даних.

Місце де зберігаються дані.

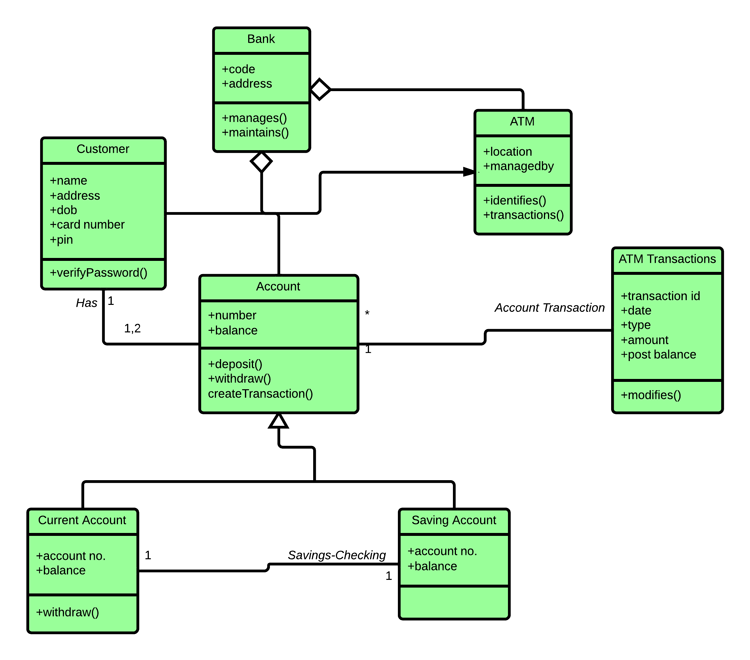
Які дані звідки виходять і куди входять. Стрілки не можуть напряму з’єднувати храніліща даних бо дані мають проходити через процеси.

**Class diagram** – описує статичну структуру системи. Містить класи, їх атрибути і методи. Взаємодія між класами позначається лініями. Модифікатори доступу для членіві класу позначаються:



Взаємодія між класами 🡪

Наприклад:



**Асоціація** – коли є два обєкта які якось взаємодіють між собою, але якшо один з них забрати, то інший зможе спокійно продовжити працювати. Популярний приклад це викладч і студент. Вони взаємодіють між собою, але якшо студента відчислять, то викладач продовжить спокійно роботу.

**Агрегація** – це ніби форма асоціації, тобто обєкти мають свій lifecycle але тут вже один з обєктів містить інший. Наприклад, університет має викладачів. То в випадку коли викладач піде, то університет всеодно продовжить працювати.

**Композиція** – це така форма агрегації, але коли є вже завязаність обєктів одне на одне. І якщо забрати один з них, то інший скаже «я немогу». Наприклад якщо в якомусь будинку знести приміщення з несучою стіною, то воно розвалиться.

