**Phần 1: giới thiệu**

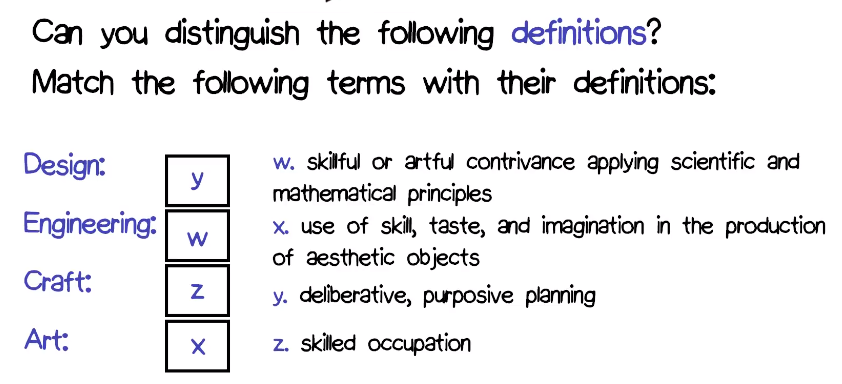
**Bài 1: giới thiệu**

Kiến trúc phần mềm của một chương trình máy tính là cấu trúc của các thành phần trong hệ thống đó. Kiến trúc phần mềm bao gồm các phần tử phần mềm, các thuộc tính và muốn quan hệ giữa chúng.

Kiến trúc phần mềm giúp việc đưa ra các quyết định ở mức **high-level** trở nên dễ dàng hơn và cho phép tái sử dụng các thành phần và các design pattern của các dự án phần mềm

**Bài 2: khái niệm thiết kế**

Design: là planning một cách có chủ ý, có chủ đích



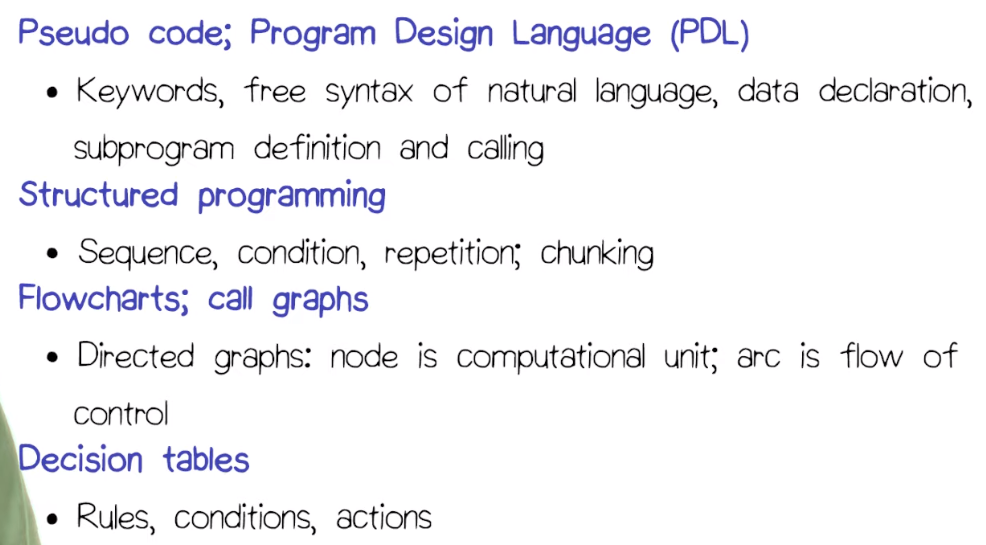
Điểm khác biệt giữa software design và programming:

- Scale

- Nhấn mạnh vào các Non-functional Requirements

Như vậy, **Software Design** là gì? Là process building một program trong khi vẫn thỏa mãn các vấn đề của functional requirements mà không làm vi phạm đến các non-functional constraints. Software Design được thường chia làm 2 giai đoạn:

1. Architectural Design: process của việc xác định và định hình trách nhiệm của các modules hay các components khác nhau của phần mềm trong việc đưa ra các hàng vi trong quá trình chạy của ứng dụng
2. Detail Design: process của việc dealing với các components riêng biệt mà chúng ta đã định hình trong architectural design



**Approaches to Software Design:**

- Method: là một chuỗi các bước nối tiếp nhau 1 cách có hệ thống trình bày một cách để giải quyết 1 vấn đề.

**Traditional Design Documentation:**

* Subcomponents:
  + Processes / activities
  + Data / Data flows
* Control Flow:
  + Control regime
* Performance
* Resources

**Design Rationale** (cơ sở lý luận của design)**:** lý do vì sao chúng ta lại coming up với design solution hiện tại. Design decisions là các lựa chọn rõ ràng chúng ta trade off giữa 2 non-functional aspects của một design, ví dụ giữa tốc độ và size

1 A / B

2 D / A

3 B

4 D / B

5 D

**--------------- --------------- --------------- --------------- --------------- --------------- --------------- --------------- --------------- --------------- --------------- ---------------**

**Phần 2: ngôn ngữ mô hình hóa thống nhất và ngôn ngữ ràng buộc đối tượng**

**Bài 3: đánh giá kiến thức về UML**

**Ngôn ngữ UML (ngôn ngữ mô hình hợp nhất UML)** là một ngôn ngữ để xác định, hình dung, xây dựng và ghi lại các khía cạnh của hệ thống phần mềm, cũng như mô hình nghiệp vụ và các hệ thống non-software khác. UML đại diện cho một tập hợp các minh họa thực tế dưới dạng kỹ thuật ở mức tốt nhất để chứng minh thành công trong việc mô hình hóa các hệ thống lớn và phức tạp.

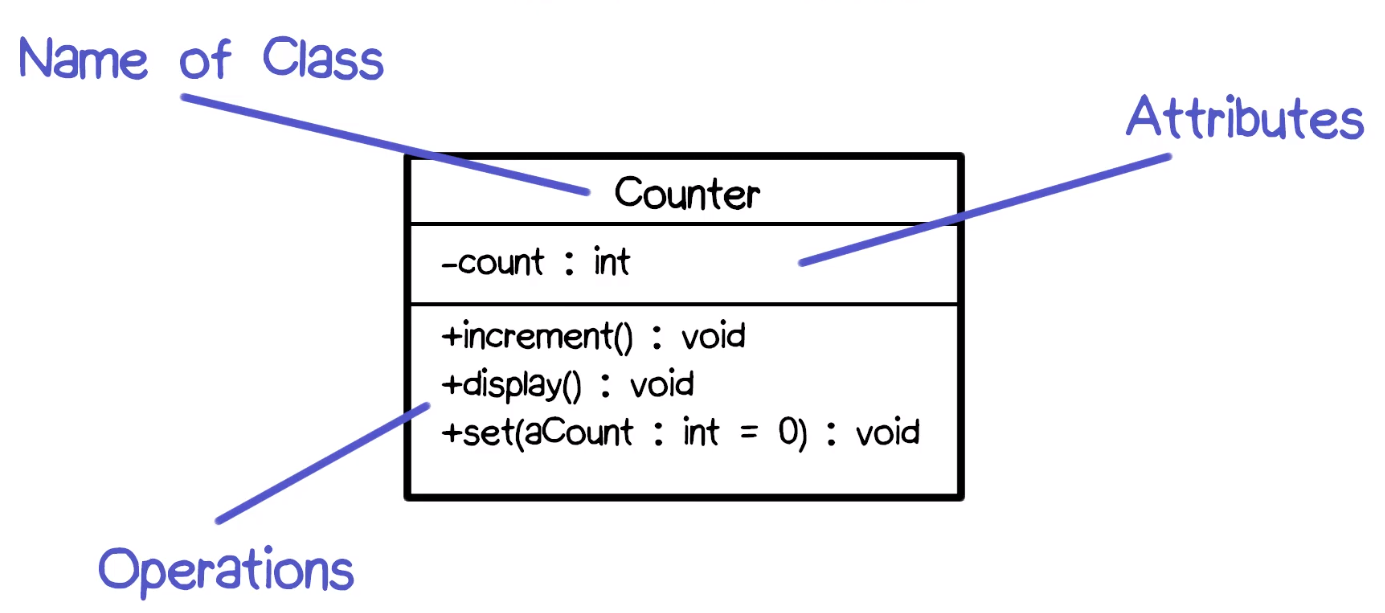
Các loại UML sẽ có một tính chất riêng biệt khác nhau, có loại giúp biểu diễn được thuật toán, có loại giúp hiểu được ngữ cảnh của hệ thống, có loại giúp nắm được tuần tự của hoạt động nghiệp vụ để biểu đạt bằng UML.

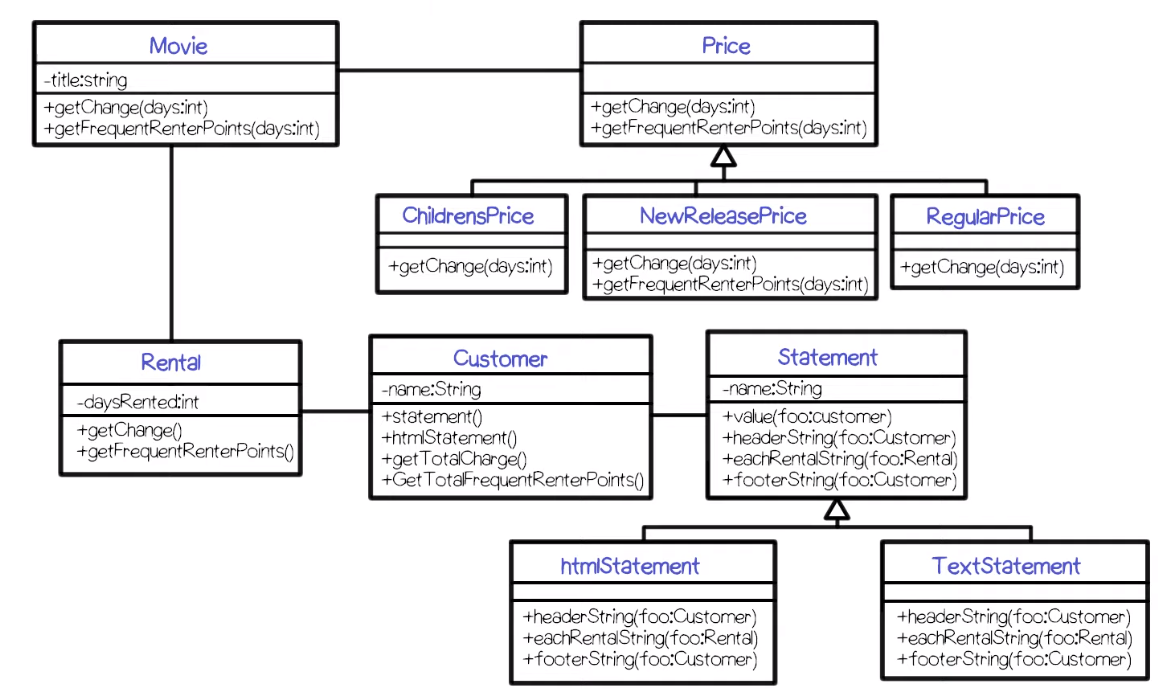
Các Diagrams có thể được chia làm 2 loại chính:

* Structural Diagrams: cho chúng ta thấy 1 phần hoặc toàn bộ hệ thống, luôn luôn static, và mối liên hệ giữa các thành phần này
* Behavioral Diagrams: tập trung vào executions của hệ thống

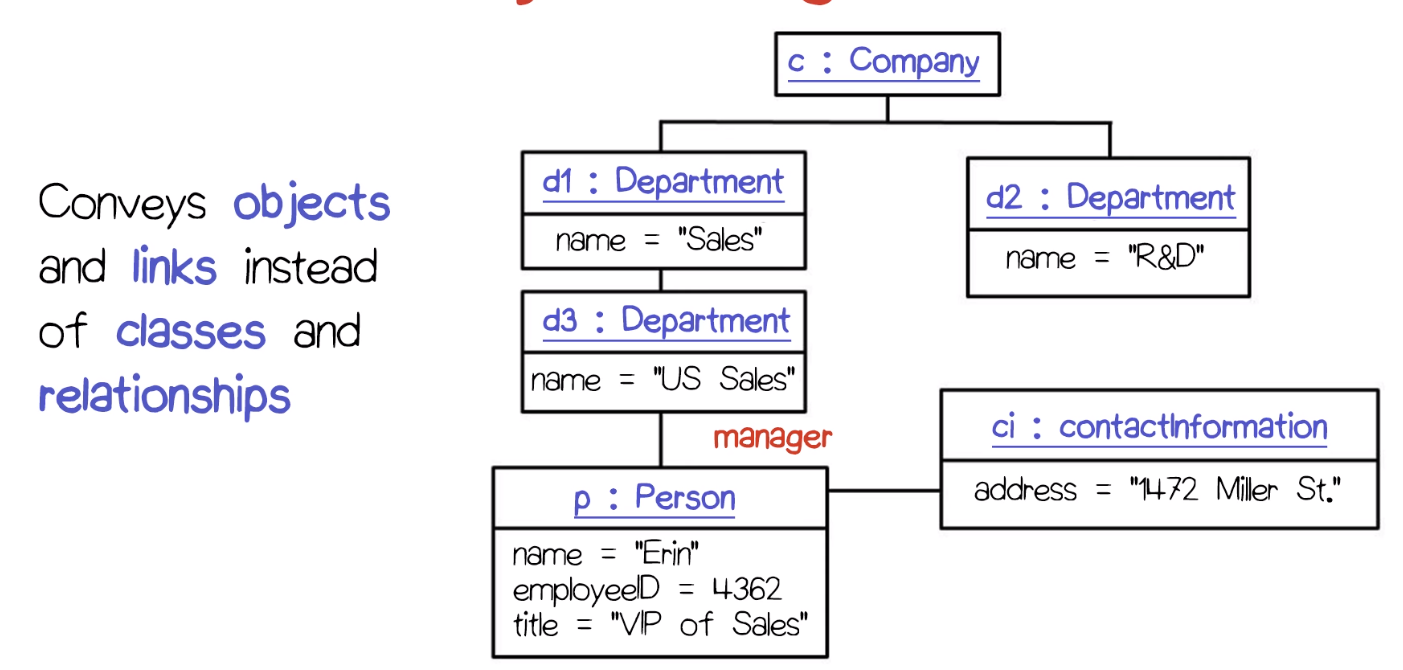
**ĐI QUAN TỪNG LOẠI UML DIAGRAM:**

1. UML Classes:
   1. Relationships:
      1. Dependency: X uses Y, sử dụng *dashed* hoặc *directed line*
      2. Asociations: X affects Y, sử dụng các line vô hướng
      3. Generalization: X is a kind of Y, sử dụng solid line với mũi tên





1. Object Diagram: related visually đối với class model diagram là object digram, thay vì mention đến classes, nó đề cập đến các instances được tạo nên từ Class.



**1 D**

**2 E**

**3 B**

**4 A + C**

**5 A + B + E / A + E /**

**6 D**