

# **PHÂN TÍCH THIẾT KẾ HỆ THỐNG**

## **Nội dung:**

1. Use case diagram
2. Activity diagram
3. Class diagram
4. Sequence diagram
5. Database relationship diagram

## **Phân tích thiết kế hệ thống thông tin sử dụng biểu đồ UML?**

Ngôn ngữ mô hình hóa thống nhất (tiếng Anh: Unified Modeling Language, viết tắt thành UML) là một ngôn ngữ mô hình gồm các ký hiệu đồ họa mà các phương pháp hướng đối tượng sử dụng để thiết kế các hệ thống thông tin một cách nhanh chóng.

- Cách xây dựng các mô hình trong UML phù hợp mô tả các hệ thống thông tin cả về cấu trúc cũng như hoạt động.
- Cách tiếp cận theo mô hình của UML giúp ích rất nhiều cho những người thiết kế và thực hiện hệ thống thông tin cũng như những người sử dụng nó; tạo nên một cái nhìn bao quát và đầy đủ về hệ thống thông tin dự định xây dựng.
- Cách nhìn bao quát này giúp nắm bắt trọn vẹn các yêu cầu của người dùng; phục vụ từ giai đoạn phân tích đến việc thiết kế, thẩm định và kiểm tra sản phẩm ứng dụng công nghệ thông tin.

- Các mô hình hướng đối tượng được lập cũng là cơ sở cho việc ứng dụng các chương trình tự động sinh mã trong các ngôn ngữ lập trình hướng đối tượng, chẳng hạn như ngôn ngữ C++, Java,...
- Phương pháp mô hình này rất hữu dụng trong lập trình hướng đối tượng. Các mô hình được sử dụng bao gồm Mô hình đối tượng (mô hình tĩnh) và Mô hình động.
- UML sử dụng một hệ thống ký hiệu thống nhất biểu diễn các Phần tử mô hình (model elements). Tập hợp các phần tử mô hình tạo thành các Sơ đồ UML (UML diagrams). Có các loại sơ đồ UML chủ yếu sau:

- Sơ đồ lớp (Class Diagram)
- Sơ đồ đối tượng (Object Diagram)
- Sơ đồ tình huống sử dụng (Use Cases Diagram)
- Sơ đồ trình tự (Sequence Diagram)
- Sơ đồ cộng tác (Collaboration Diagram hay là Composite Structure Diagram)
- Sơ đồ trạng thái (State Machine Diagram)
- Sơ đồ thành phần (Component Diagram)
- Sơ đồ hoạt động (Activity Diagram)
- Sơ đồ triển khai (Deployment Diagram)
- Sơ đồ gói (Package Diagram)
- Sơ đồ liên lạc (Communication Diagram)
- Sơ đồ tương tác (Interaction Overview Diagram - UML 2.0)
- Sơ đồ phối hợp thời gian (Timing Diagram - UML 2.0)

# **1. Biểu đồ Use Case (Use Case Diagram)**

Một biểu đồ Use Case chỉ ra một số lượng các tác nhân ngoại cảnh và mối liên kết của chúng đối với Use Case mà hệ thống cung cấp. Một Use Case là một lời miêu tả của một chức năng mà hệ thống cung cấp. Lời miêu tả Use Case thường là một văn bản tài liệu, nhưng kèm theo đó cũng có thể là một biểu đồ hoạt động. Các Use Case được miêu tả duy nhất theo hướng nhìn từ ngoài vào của các tác nhân (hành vi của hệ thống theo như sự mong đợi của người sử dụng), không miêu tả chức năng được cung cấp sẽ hoạt động nội bộ bên trong hệ thống ra sao. Các Use Case định nghĩa các yêu cầu về mặt chức năng đối với hệ thống.

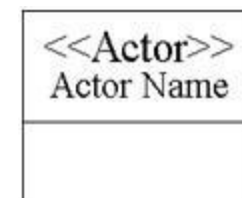
Hệ thống: Với vai trò là thành phần của biểu đồ Use Case, hệ thống biểu diễn ranh giới giữa bên trong và bên ngoài của một chủ thể trong phần mềm chúng ta xây dựng. Một hệ thống ở trong biểu đồ Use Case không nhất thiết là một hệ phần mềm; nó có thể là một chiếc máy, hoặc là một hệ thống thực như một doanh nghiệp, một trường đại học,...

Tác nhân (actor): là người dùng của hệ thống, một tác nhân có thể là một người dùng thực hoặc các hệ thống máy tính khác có vai trò nào đó trong hoạt động của hệ thống. Như vậy, tác nhân thực hiện các Use Case. Một tác nhân có thể thực hiện nhiều Use Case và ngược lại một Use Case cũng có thể được thực hiện bởi nhiều tác nhân

Tác nhân được kí hiệu:

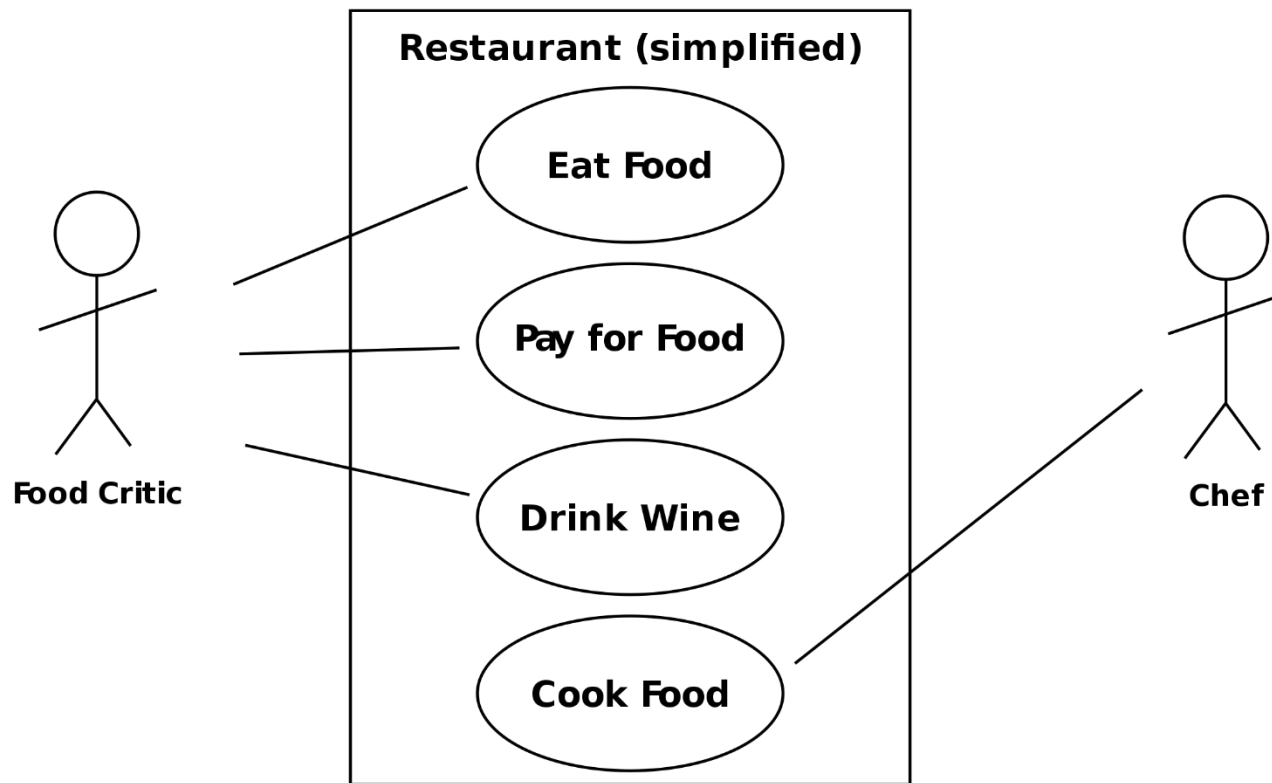


hoặc



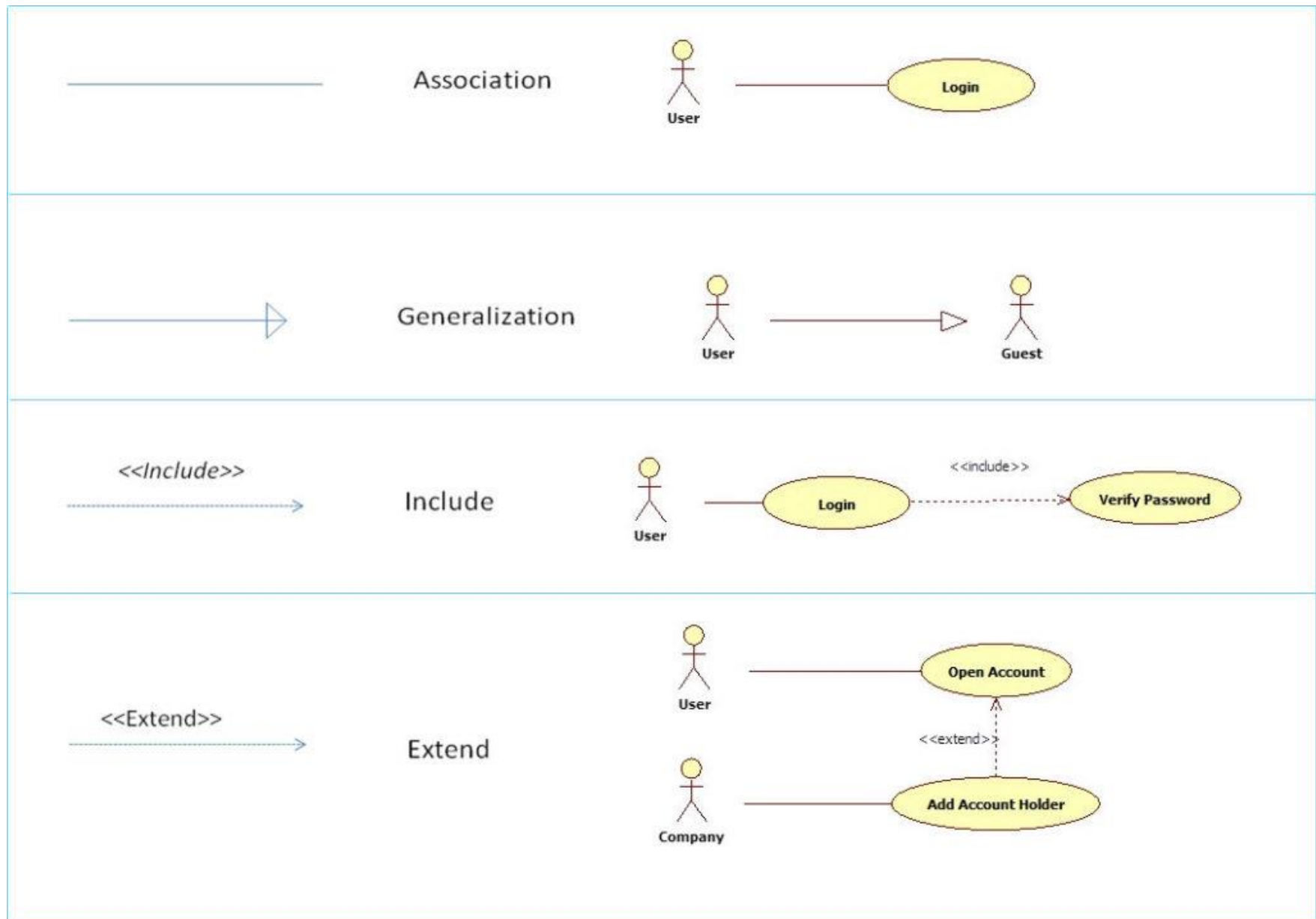
Các Use Case: Đây là thành phần cơ bản của biểu đồ Use Case. Các Use Case được biểu diễn bởi các hình elip. Tên các Use Case thể hiện một chức năng xác định của hệ thống.

Các Use Case được kí hiệu bằng hình elips.





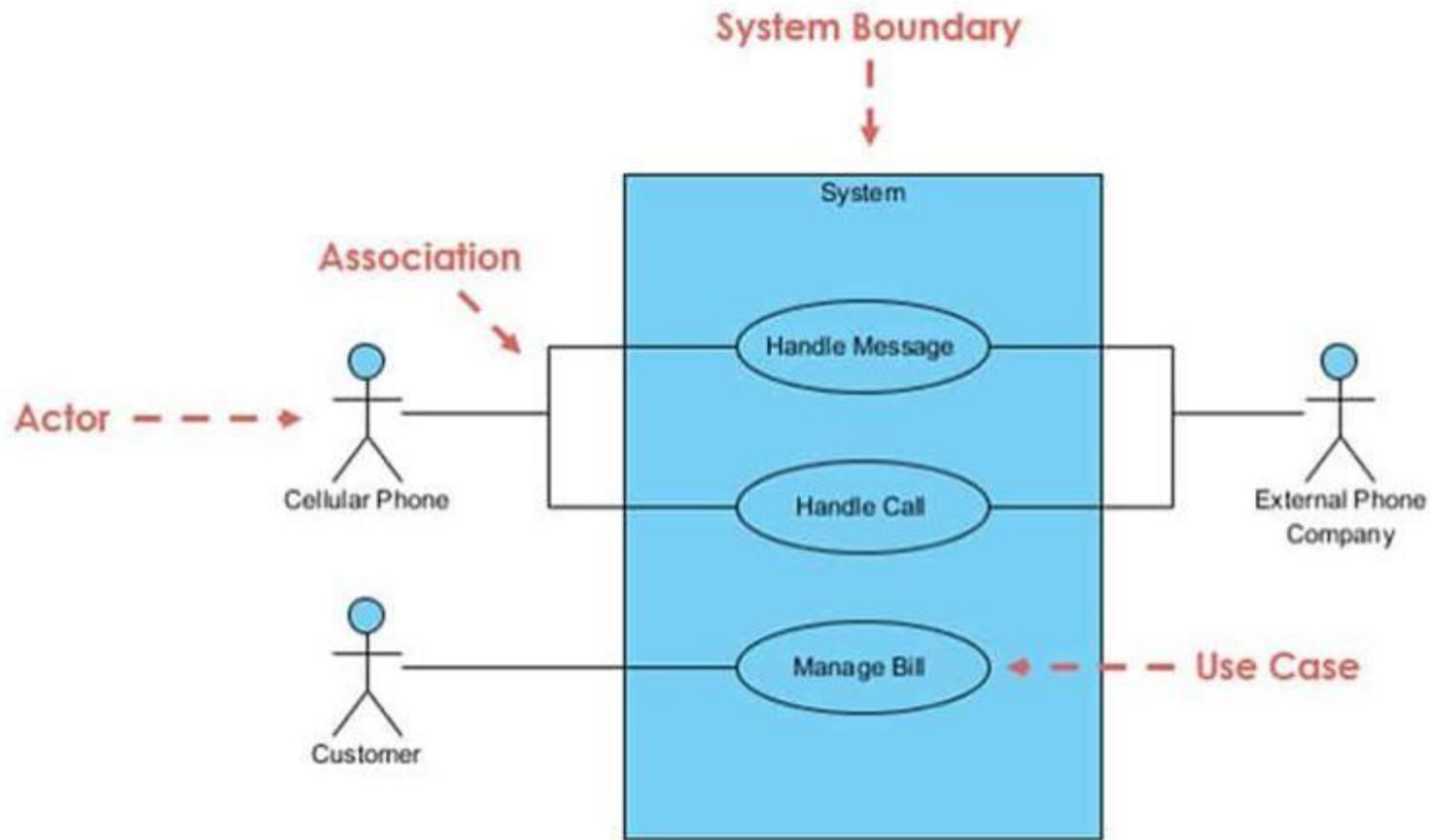
## . Mối quan hệ giữa các Use Case:



- Association: thường được dùng để mô tả mối quan hệ giữa Actor và Use Case và giữa các Use Case với nhau
- Include: là quan hệ giữa các Use Case với nhau, nó mô tả việc một Use Case lớn được chia ra thành các Use Case nhỏ để dễ cài đặt (module hóa) hoặc thể hiện sự dùng lại.
- Extent: Extend dùng để mô tả quan hệ giữa 2 Use Case. Quan hệ Extend được sử dụng khi có một Use Case được tạo ra để bổ sung chức năng cho một Use Case có sẵn và được sử dụng trong một điều kiện nhất định nào đó.
- Generalization: được sử dụng để thể hiện quan hệ thừa kế giữa các Actor hoặc giữa các Use Case với nhau

Để tìm ra được các **Use Case**, ta cần trả lời những câu hỏi sau:

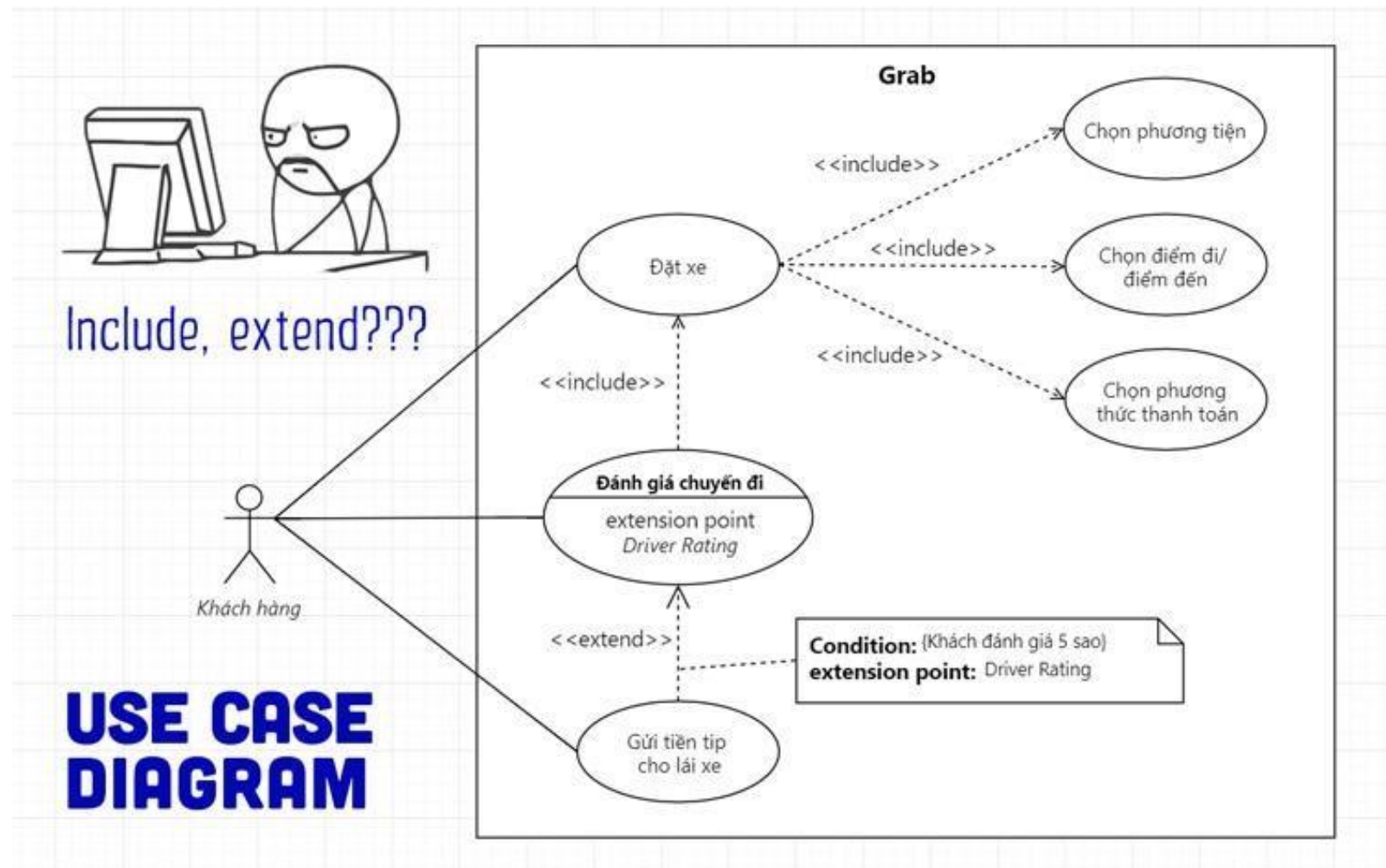
- Actor cần những chức năng nào của hệ thống? Actor có hành động chính là gì?
- Actor có cần đọc, thêm mới, hủy bỏ, chỉnh sửa hay lưu trữ loại thông tin nào trong hệ thống không?
- Hệ thống có cần thông báo những thay đổi bất ngờ trong nội bộ cho Actor không?
- Công việc hàng ngày của Actor có thể được đơn giản hóa hoặc hữu hiệu hóa qua các chức năng của hệ thống?
- Use Case có thể được tạo ra bởi sự kiện nào khác không?
- Hệ thống cần những thông tin đầu vào/đầu ra nào, những thông tin đó sẽ đi từ đâu đến đâu?
- Những khó khăn và thiếu hụt của hệ thống hiện tại nằm ở đâu?



### *Sự tương tác và phạm vi của Use Case*

**Communication Link** thể hiện sự tương tác giữa người dùng (Actor) và hệ thống (System), nó kết nối giữa Actor và Use Case. **Boundary of System** chính là phạm vi mà Use Case xảy ra. Ví dụ với hệ thống CRM, phạm vi có thể là những cụm tính năng lớn như quản lý đơn hàng, quản lý khách hàng hoặc cả một module lớn như quản lý bán hàng.

# Cách xây dựng Use Case Diagram hoàn chỉnh?



*Ví dụ về một Use Case Diagram hoàn hảo*

## **Giai đoạn mô hình hóa:**

- ❖ Bước 1: Thực hiện thiết lập ngữ cảnh của hệ thống.
- ❖ Bước 2: Xác định các Actor.
- ❖ Bước 3: Xác định các Use Case.
- ❖ Bước 4: Định nghĩa các mối quan hệ giữa Actor và Use Case.
- ❖ Bước 5: Đánh giá các mối quan hệ đó để tìm cách chi tiết hóa.

## **Giai đoạn cấu trúc:**

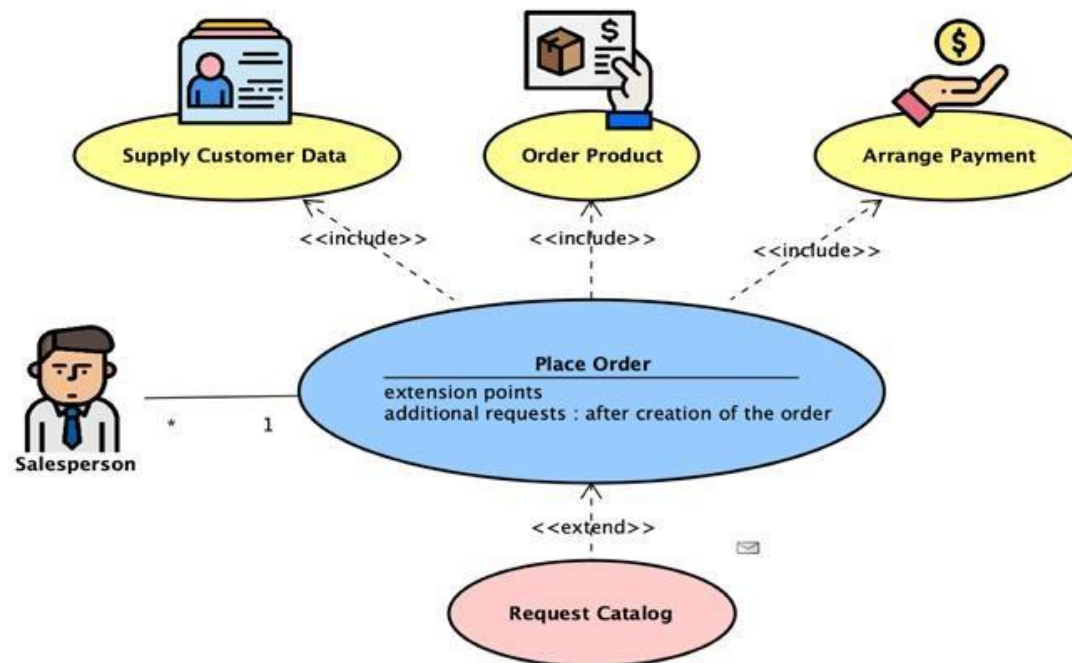
- ❖ Bước 6: Đánh giá các Use Case cho quan hệ Include.
- ❖ Bước 7: Đánh giá các Use Case cho quan hệ Extend.
- ❖ Bước 8: Đánh giá các Use Case cho quan hệ Generalization .

## **Giai đoạn review:**

- ❖ Kiểm tra (verification): đảm bảo hệ thống đúng với tài liệu đặc tả.
- ❖ Thẩm định (validation): đảm bảo hệ thống sẽ được phát triển là thứ mà khách hàng cuối thực sự cần thiết.

**Những sai lầm cần lưu ý khi thiết kế Use Case Diagram**

Sơ đồ Use Case là thứ thể hiện được những yêu cầu từ phía khách hàng. Do vậy, cần thiết kế sao cho đơn giản, chi tiết và dễ hiểu nhất. Để biết những sai lầm thường gặp khi xây dựng Use Case là gì, cách khắc phục như thế nào, các bạn nên lưu tâm:



*Tô màu cho Use Case Diagram để sơ đồ rõ ràng hơn*



- . **Đặt tên không chuẩn:** Vì là mô hình hóa nên cần diễn đạt bằng hình ảnh, cố gắng sử dụng ít chữ nhất có thể. Vì vậy, những gì được ghi trên Use Case Diagram phải thật cô đọng và có giá trị tức thì. Đó là lý do bạn tránh đặt tên quá dài;
- . **Lẫn lộn giữa Use Case và phân rã chức năng:** Sai lầm nhìn thấy ngay là những từ “quản lý” (manage) trên sơ đồ. Use Case cần truyền tải được mục đích sau cùng, chứa đựng góc nhìn của người dùng cuối cùng;
- . **Thiết kế quá nhiều Use Case:** Là một sai lầm nhiều người mắc phải. Bạn nên tận dụng các Relationship để khiến các Use Case liên kết với nhau, sau đó sử dụng Boundary of System để phân nhóm, giới hạn cho các Use Case;
- . **Quá đi sâu vào chi tiết các chức năng CRUD:** Nếu sử dụng mỗi thực thể là 1 lần CRUD thì sẽ rất tốn công. Điều này cũng tạo sự lặp đi lặp lại ở các sơ đồ Use Case nhưng lại không thể

hiện được nhiều thông tin cho người xem. Để giải quyết, bạn có thể thử thêm 1 dòng note trước đoạn mô tả Use Case của tài liệu hoặc tạo hẳn 1 Use Case có tên là manage X với X là 1 đối tượng bất kỳ;

- . **Thiếu thẩm mỹ:** Nhiều sơ đồ Use Case khá thiếu thẩm mỹ, không có thiết kế hợp lý nên không thu hút được người dùng. Vì vậy, bạn cần chú ý thiết kế các Use Case trong Diagram cùng kích cỡ, đánh dấu các Use Case ID, không được chồng chéo các mối quan hệ, có thể tô màu lên Use Case,... để sơ đồ rõ ràng, mạch lạc hơn.

# Thực hành vẽ sơ đồ Use Case

## Bước 1: Thu thập kiến thức liên quan đến hệ thống sẽ xây dựng

Trước hết, để phân tích hệ thống trên bạn phải có kiến thức về hệ thống thương mại điện tử, chúng ta có thể tìm hiểu thông qua các nguồn sau:

- ✚ Xem các trang Web bán hàng qua mạng như amazon, lazada.vn, bkc.vn v.v..
- ✚ Xem các hệ thống mẫu về thương mại điện tử nguồn mở như Magento, OpenCart, Spree Commerce v.v...
- ✚ Đọc sách, báo về eCommerce
- ✚ Hỏi những người chuyên về lĩnh vực này (hỏi chuyên gia)

**Lưu ý:** Bạn không thể thiết kế tốt được nếu bạn không có kiến thức về lĩnh vực của sản phẩm mà bạn sẽ xây dựng.

## Bước 2: Xác định các Actor

*Bạn hãy trả lời cho câu hỏi “Ai sử dụng hệ thống này?”*

- + Những người muốn mua hàng vào website để xem thông tin.  
Những người này là **Khách hàng tiềm năng (Guest)**.
- + Những người đã đặt hàng vào kiểm tra đơn hàng, thanh toán v.v.. gọi là **Khách hàng (Customer)**.
- + Về phía đơn vị bán hàng, có những người sau đây tham gia vào hệ thống:
  - ❖ **Người quản lý bán hàng:** quyết định nhập hàng, giá bán, quản lý tồn kho, doanh thu, chính sách khuyến mãi.
  - ❖ **Người bán hàng:** Tư vấn cho khách hàng, theo dõi đơn hàng, thu tiền, theo dõi chuyển hàng cho khách.
  - ❖ **Quản lý kho:** xuất, nhập hàng, quản lý tồn kho
  - ❖ **Quản trị hệ thống:** Tạo người dùng, Phân quyền, Tạo cửa hàng

*Tiếp theo chúng ta trả lời câu hỏi “Hệ thống nào tương tác với hệ thống này?”*

Giả sử ở đây, chúng ta sử dụng dịch vụ của Ngân Lượng để thanh toán trực tuyến và gọi nó là “**Cổng thanh toán**” thì ta có thêm một Actor tương tác với hệ thống.

Như vậy, chúng ta đã có các Actor của hệ thống gồm: **Khách hàng tiềm năng, khách hàng, Người bán hàng, Quản lý Kho, Quản trị hệ thống, Cổng thanh toán**

Bạn cần khảo sát và phân tích thêm cũng như hỏi trực tiếp khách hàng để xác định đầy đủ các Actor cho hệ thống.




### Bước 3: Xác định Use Case

Bạn cần trả lời câu hỏi “Actor sử dụng chức năng gì trên hệ thống?”.






Trước tiên, xem xét với Actor “**Khách hàng tiềm năng**” trên trang bkc.vn để xem họ sử dụng chức năng nào?

- + Xem trang chủ
- + Xem các sản phẩm theo:
  - + Theo chủng loại
  - + Nhà sản xuất
  - + Tìm kiếm theo văn bản gõ vào
    - Xem chi tiết sản phẩm được chọn
    - Xem khuyến mãi
    - Xem so sánh
    - Mua hàng
    - Quản lý giỏ hàng
    - Chat với người bán hàng
    - Đăng ký tài khoản để trở thành khách hàng

Tiếp theo, xem xét Actor “**Khách hàng**” và nhận thấy họ sử dụng chức năng:

-  Đăng nhập
-  Xem đơn hàng
-  Thanh toán

Tiếp theo, xem xét Actor “**Người bán hàng**” và họ có thể sử dụng các chức năng:

-  Đăng nhập
-  Chat với khách hàng
-  Theo dõi đơn hàng
-  Thu tiền
-  Theo dõi chuyển hàng

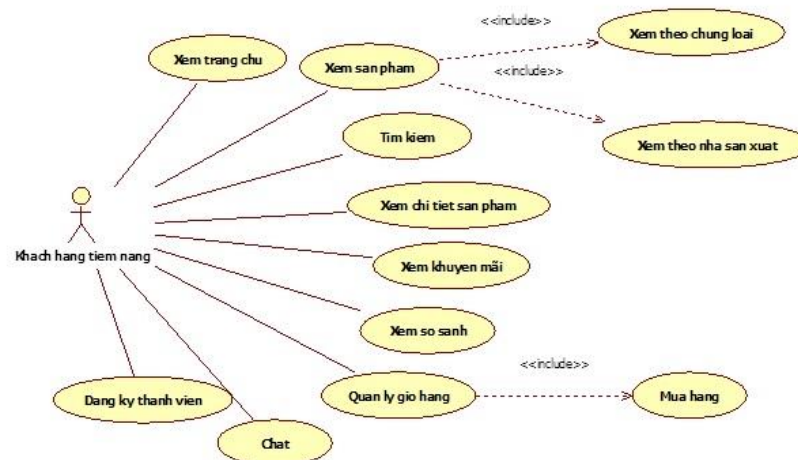
Tương tự như vậy bạn xác định chức năng cho các Actor còn lại.

## Bước 4: Vẽ bản vẽ Use Case

Trước hết chúng ta xem xét và phân tích các chức năng của “Khách hàng tiềm năng” chúng ta nhận thấy.

- + Chức năng xem sản phẩm có 2 cách là chọn loại sản phẩm, nhà sản xuất để xem và gõ vào ô tìm kiếm. Nên chúng ta tách ra làm 2 là *Xem sản phẩm* và *Tìm kiếm*.
- + Chức năng mua hàng, thực chất là thêm vào giỏ hàng nên có thể xem là chức năng con của quản lý giỏ hàng.

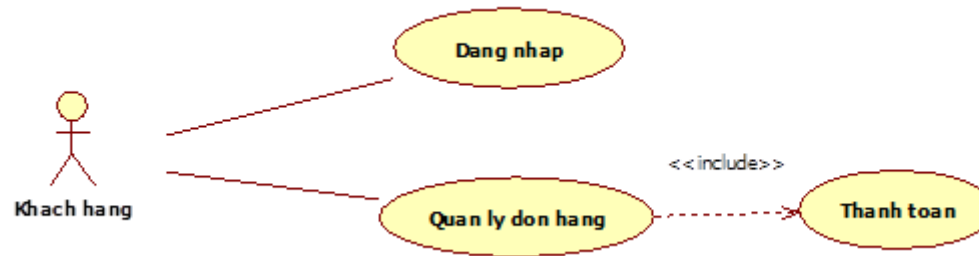
Đặt lại tên và xác định các mối quan hệ, chúng ta được:



Hình 1. Bản vẽ Use Case cho Actor “Khách hàng tiềm năng”



Tiếp theo, chúng xem xét các chức năng cho Actor “Khách hàng” và nhận thấy chức năng “Thanh toán” thường thực hiện cho từng đơn hàng cụ thể nên có thể nó là chức năng con của “Quản lý đơn hàng”. Ngoài ra, các chức năng Actor này sử dụng không giao với Actor “Khách hàng tiềm năng” nên nó được biểu diễn như sau:



**Hình 2. Bản vẽ Use Case cho Actor “Khách hàng”**

Tiếp tục xem xét Actor “**Người bán hàng**” chúng ta nhận thấy:

- Chức năng “**Thu tiền**” thực tế là thanh toán trực tiếp tại quầy cho từng đơn hàng và chức năng “Theo dõi chuyển hàng” được thực hiện trên từng đơn hàng nên nó có thể là chức năng con của “Quản lý đơn hàng”.
- Chức năng “Quản lý đơn hàng” ở đây quản lý cho nhiều khách hàng nên sẽ khác với chức năng “Quản lý đơn hàng” của Actor “Khách hàng” nên để phân biệt chúng ta sửa chức năng “Quản lý đơn hàng ” của Actor “Khách hàng” thành “Quản lý đơn hàng cá nhân”
- Chức năng “Đăng nhập” có thể dùng chung với Actor “Khách hàng”, chức năng Chat dùng chung với Actor “Khách hàng tiềm năng”

Vẽ chúng chung với nhau chúng ta được bản vẽ như sau:



**Hình 2. Bản vẽ Use Case khi bổ sung các chức năng cho “Khách hàng tiềm năng”, “Khách hàng” và “Người bán hàng”**

## 2. Biểu đồ Hoạt động (Activity Diagram)

Biểu đồ hoạt động là biểu đồ mô tả các bước thực hiện, các hành động, các nút quyết định và điều kiện rẽ nhánh để điều khiển luồng thực hiện của hệ thống. Đối với những luồng thực thi có nhiều tiến trình chạy song song thì biểu đồ hoạt động là sự lựa chọn tối ưu cho việc thể hiện. Biểu đồ hoạt động khá giống với biểu đồ trạng thái ở tập các kí hiệu nên rất dễ gây nhầm lẫn. Khi vẽ chúng ta cần phải xác định rõ điểm khác nhau giữa hai dạng biểu đồ này là biểu đồ hoạt động tập trung mô tả các hoạt động và kết quả thu được từ việc thay đổi trạng thái của đối tượng còn biểu đồ trạng thái chỉ mô tả tập tất cả các trạng thái của một đối tượng và những sự kiện dẫn tới sự thay đổi qua lại giữa các trạng thái đó.

### Các thành phần của biểu đồ hoạt động

- ✓ Trạng thái khởi tạo hoặc điểm bắt đầu (Initial State or Start Point)

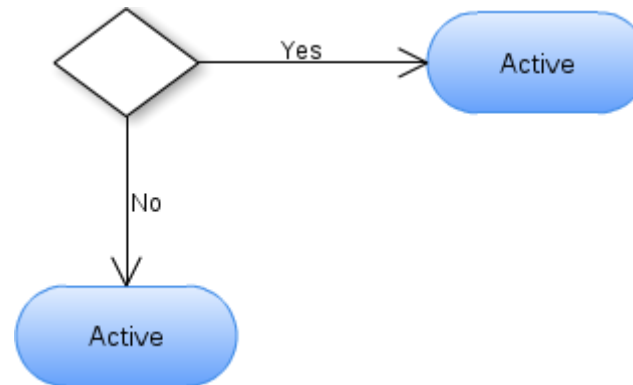


- ✓ Hoạt động hoặc trạng thái hoạt động (Activity or Action State)



Hoạt động và sự chuyển đổi hoạt động được ký hiệu và cách sử dụng hoàn toàn giống như trạng thái trong biểu đồ trạng thái đã nêu ở trên.

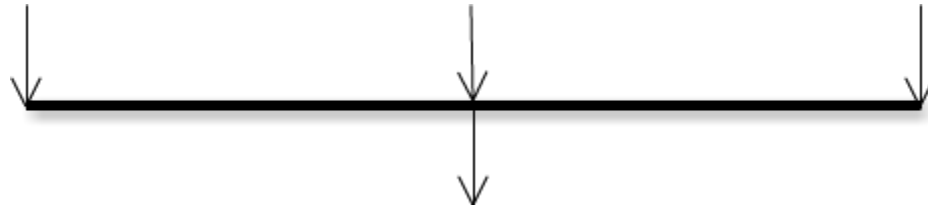
- ✓ Nút quyết định và rẽ nhánh



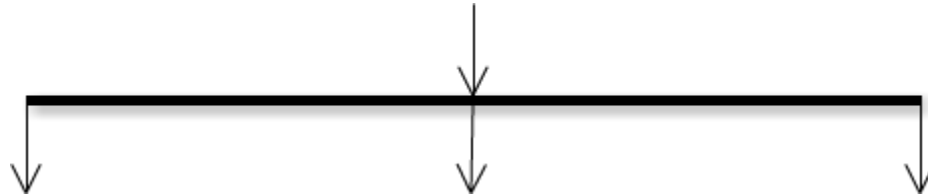
Nút rẽ nhánh trong biểu đồ hoạt động được kí hiệu bằng hình thoi màu trắng.

- ✓ Thanh tương tranh hay thanh đồng bộ  
Có thể có nhiều luồng hành động được bắt đầu thực hiện hay kết thúc đồng thời trong hệ thống.

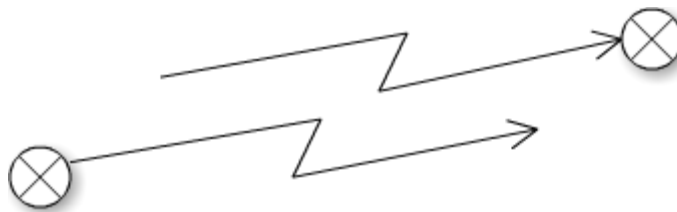
- Thanh đồng bộ kết hợp:



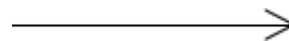
- Thanh đồng bộ chia nhánh:



- ✓ Cạnh gián đoạn (Interrupting Edge)



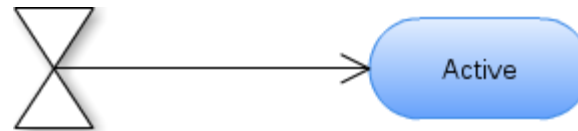
- ✓ Luồng hoạt động (Action Follow)



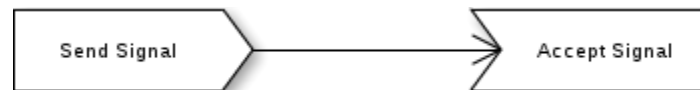
✓ Phân làn (Swimlanes)

Phân làn trong biểu đồ sử dụng là những đường nét đứt thẳng đứng theo các đối tượng. Phần kí hiệu này thường được sử dụng để làm rõ luồng hoạt động của các đối tượng riêng biệt.

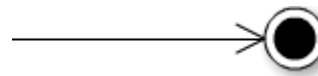
✓ Thời gian sự kiện (Time Event)



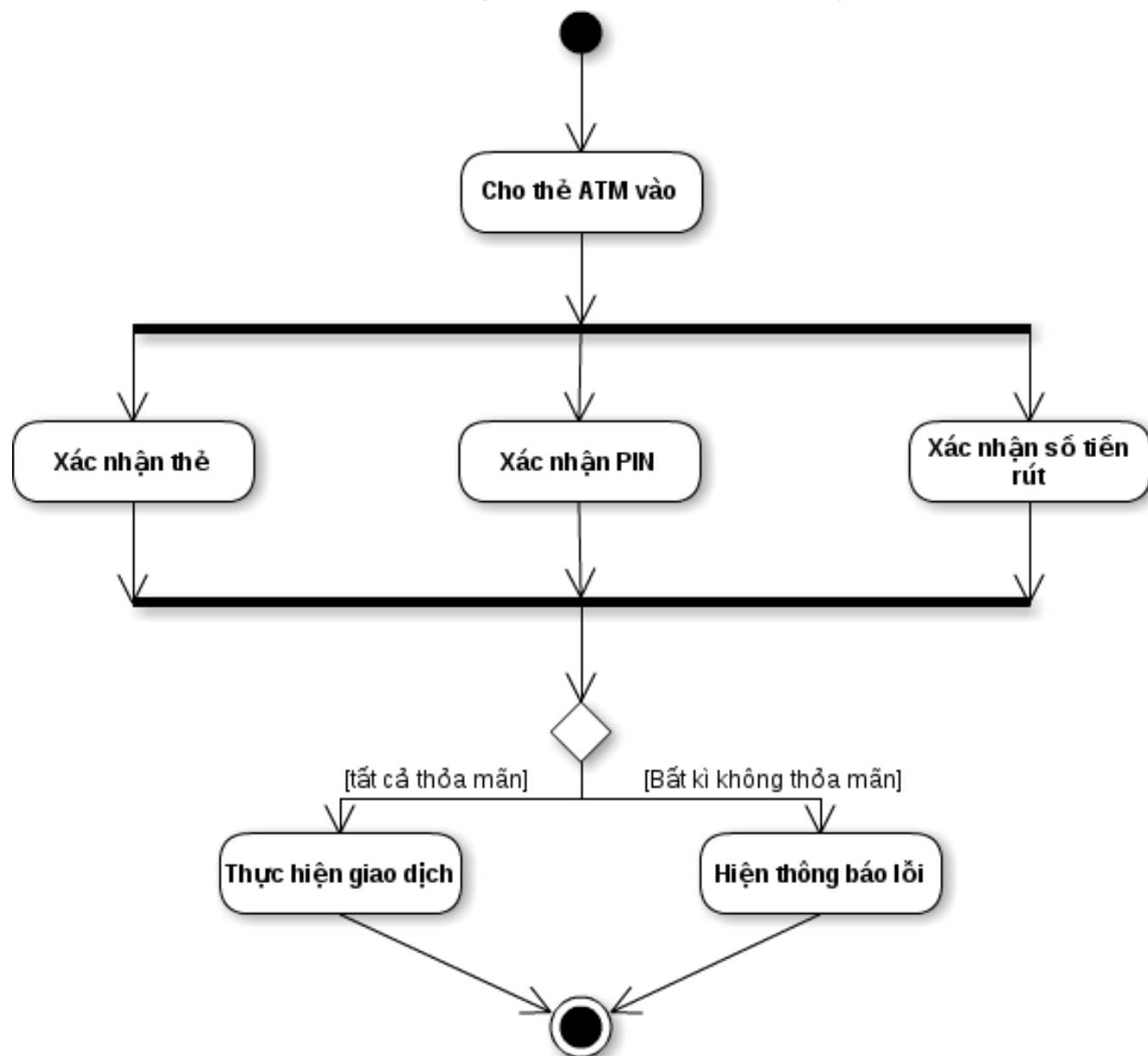
✓ Gửi và nhận tín hiệu (Sent and Received Signals)



✓ Trạng thái kết thúc hoặc điểm cuối (Final State or End Point)

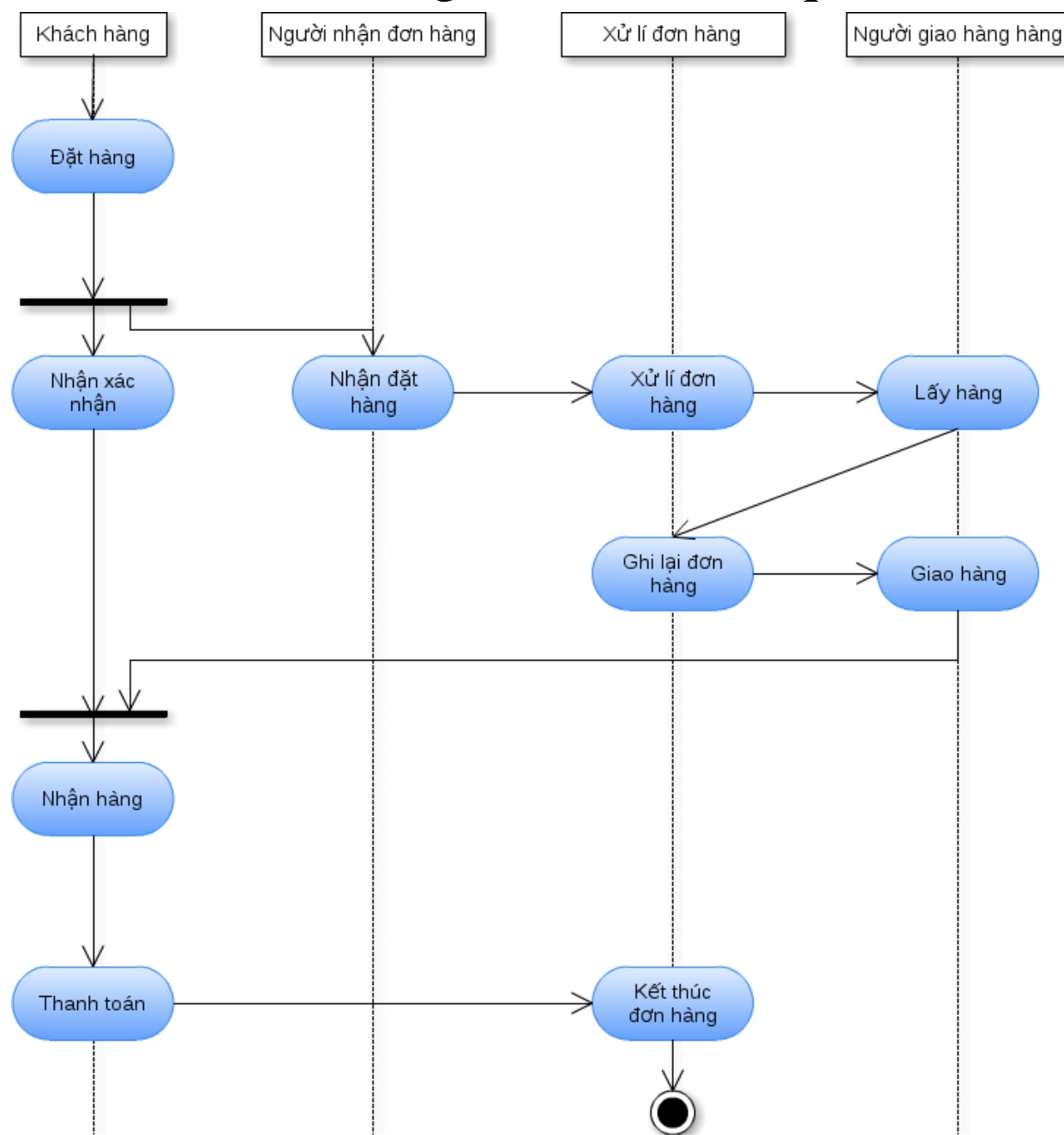


## Ví dụ 1: Biểu đồ hoạt động rút tiền tại cây ATM.





## Ví dụ 2: Biểu đồ hoạt động thể hiện một quá trình đặt hàng.











# Thực hành xây dựng Biểu đồ hoạt động

## *Bước 1: Xác định các nghiệp vụ cần phân tích.*

Trước tiên, chúng ta xem xét các Use Case. Về nguyên tắc bạn phải phân tích và mô tả tất cả các nghiệp vụ của hệ thống để làm rõ hệ thống.

Xem xét bản vẽ Use Case Diagram chúng ta đã vẽ ở bài trước, chúng ta có thể thấy các Use Case sau cần làm rõ:

-  Xem sản phẩm theo chủng loại
-  Thêm sản phẩm theo nhà cung cấp
-  Thêm giỏ hàng
-  Chat
-  Quản lý đơn hàng
-  Thanh toán
-  Theo dõi chuyển hàng
-  Đăng nhập

## ***Bước 2: Xác định các bước thực hiện và đối tượng liên quan***

Để thực hiện chức năng xem sản phẩm theo chuẩn loại hệ thống sẽ thực hiện như sau:

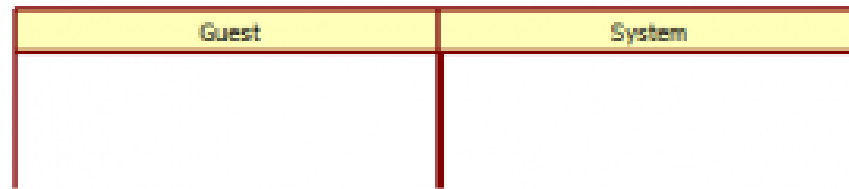
- Điều kiện ban đầu: ở trang chủ
- Điều kiện kết thúc: hiển thị xong trang sản phẩm

### **Các bước như sau:**

- Người dùng chọn loại sản phẩm.
- Hệ thống sẽ lọc lấy loại sản phẩm tương ứng, sau đó lấy giá, lấy khuyến mãi cho tất cả các sản phẩm đã được chọn và hiển thị lên màn hình.
- Người dùng xem sản phẩm.

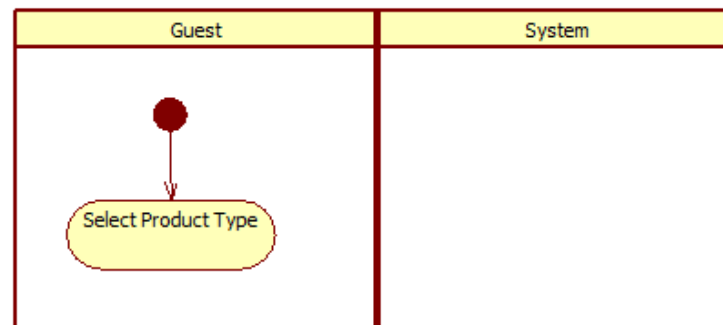
### ***Bước 3: Thực hiện bản vẽ***

– Chúng ta thấy có 2 đối tượng tham gia vào giao dịch này là Người dùng và Hệ thống. Chúng ta nên dùng Swimlane để thể hiện 2 đối tượng trên.



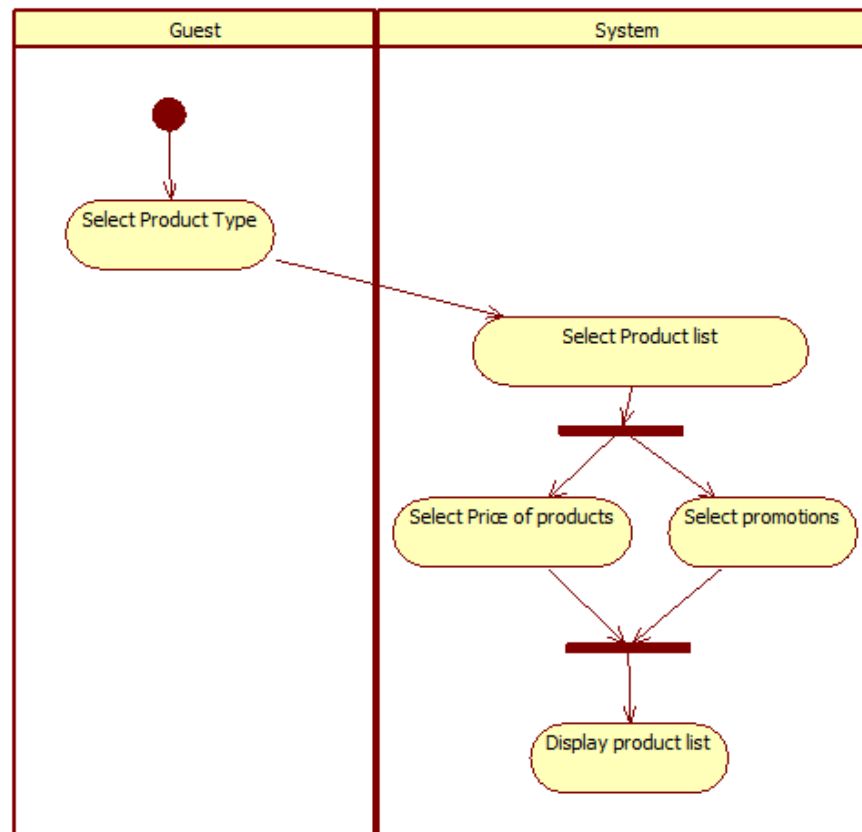
**Hình 1. Sử dụng Swimlane để thể hiện người dùng tham gia vào nghiệp vụ**

- Xác định trạng thái đầu tiên.
- Hành động tiếp theo là Guest chọn loại sản phẩm



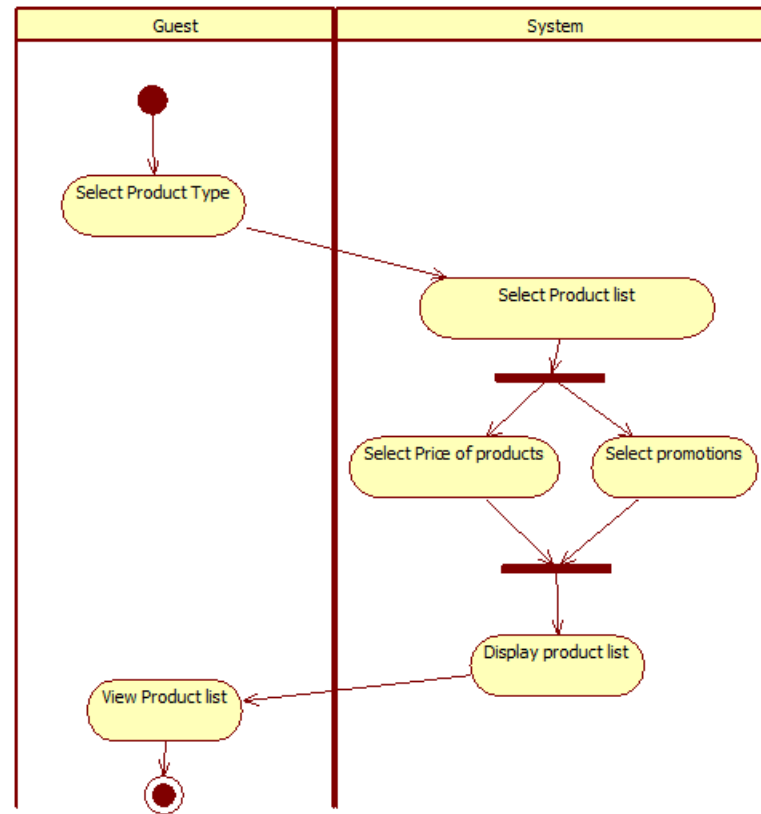
**Hình 2. Người sử dụng chọn loại sản phẩm**

Hệ thống sẽ lấy danh sách sản phẩm tương ứng với loại đó, sau đó lấy giá, lấy khuyến mãi của chúng và hiển thị ra màn hình. Hành động lấy giá và khuyến mãi của sản phẩm có thể làm song song nên chúng ta dùng Fork và Join để thể hiện



**Hình 3. Hệ thống tập hợp danh sách sản phẩm và thông tin liên quan để hiển thị lên Browser**

Người dùng xem danh sách sản phẩm và kết thúc nghiệp vụ của Use Case này.



**Hình 4. Bản vẽ Activity Diagram mô tả cho Use Case xem sản phẩm theo chủng loại**

Tương tự, như vậy bạn hãy hoàn tất Activity Diagram cho các nghiệp vụ còn lại cho hệ thống.

### **3. Biểu đồ Lớp (Class Diagram)**

Trong 1 dự án, việc tổ chức code cũng như clean code là 1 điều rất quan trọng, nếu cách thiết kế các class hợp lý và rõ ràng sẽ giúp ích rất nhiều cho việc mở rộng và bảo trì sau này. Để làm được điều này chúng ta cần phải có 1 bản thiết kế Class Diagram thật sự hợp lý. Vậy Class Diagram là gì, hãy cùng tìm hiểu.

## **Định nghĩa Class Diagram**

Class diagram mô tả kiểu của các đối tượng trong hệ thống và các loại quan hệ khác nhau tồn tại giữa chúng.

- Là một kỹ thuật mô hình hóa tồn tại ở tất cả các phương pháp phát triển hướng đối tượng.
- Biểu đồ hay dùng nhất trong UML và gần gũi nhất với các lập trình viên.
- Giúp các lập trình viên trao đổi với nhau và hiểu rõ ý tưởng của nhau.



# Các tính chất cơ bản của class diagram



Tên class

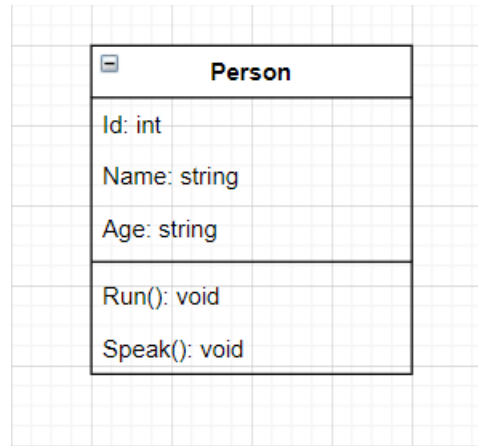


Attribute (field, property)



Operation (method, function)

Ví dụ khai báo tên, attribute, operation kèm theo kiểu trả về của 1 class:

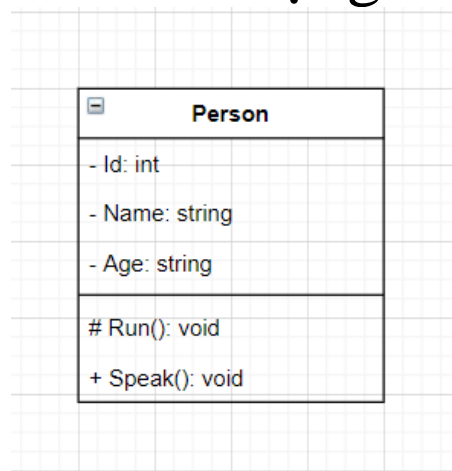


# Access Modifier trong class diagram

Sử dụng để đặc tả phạm vi truy cập cho các Attribute và Operation của 1 class (Cấp quyền cho các class khác sử dụng Attribute và Operation của class này).

❖ Có 4 lựa chọn phạm vi truy cập

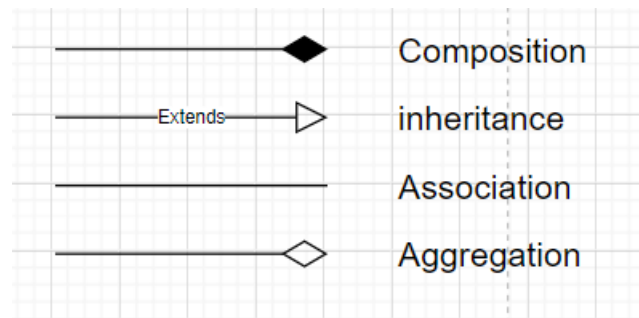
- Private ( - ): Chỉ mình các đối tượng được tạo từ class này có thể sử dụng.
- Public ( + ): Mọi đối tượng đều có thể sử dụng.
- Protected ( # ): Chỉ các đối tượng được tạo từ class này và class kế thừa từ class này có thể sử dụng.
- Package/Default: Các đối tượng được tạo từ class trong lớp cùng gói có thể sử dụng.



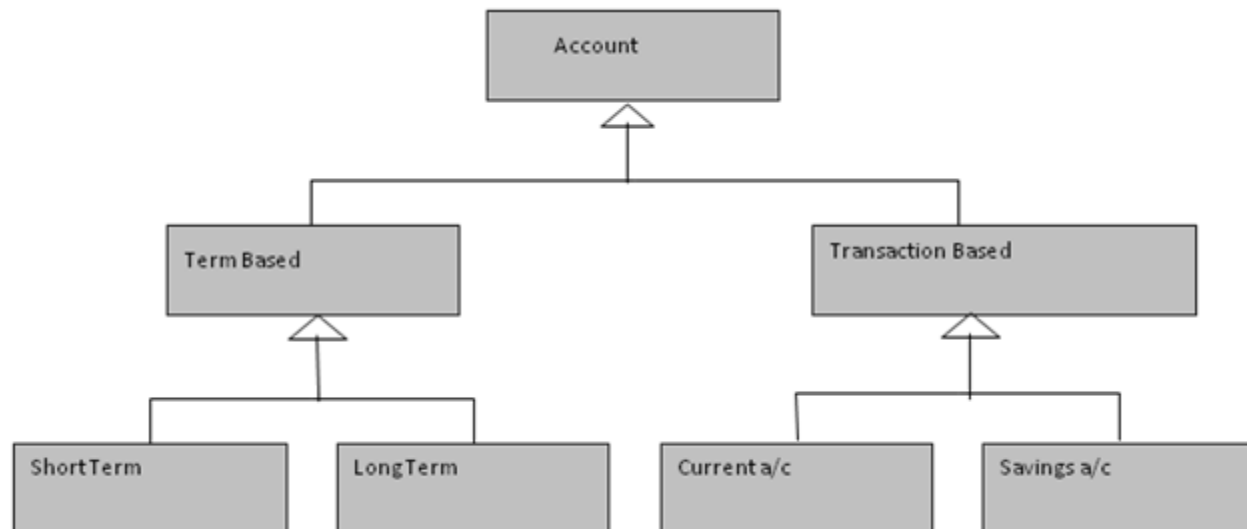
# Relationship trong class diagram

Sử dụng để thể hiện mối quan hệ giữa đối tượng được tạo từ 1 class với các đối tượng được tạo từ class khác trong class diagram.

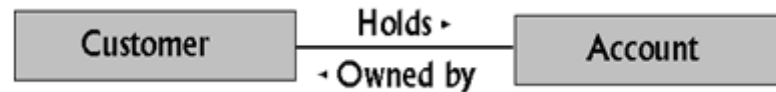
Có 4 loại Relationship:



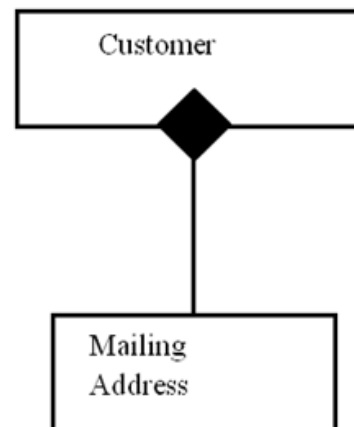
Inheritance: 1 class kế thừa từ 1 class khác.



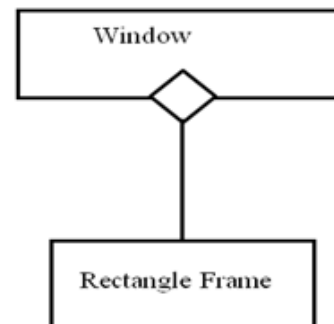
- Association: quan hệ giữa hai lớp với nhau, thể hiện qua các quan hệ như “has: có”, “Own: sở hữu” v.v...

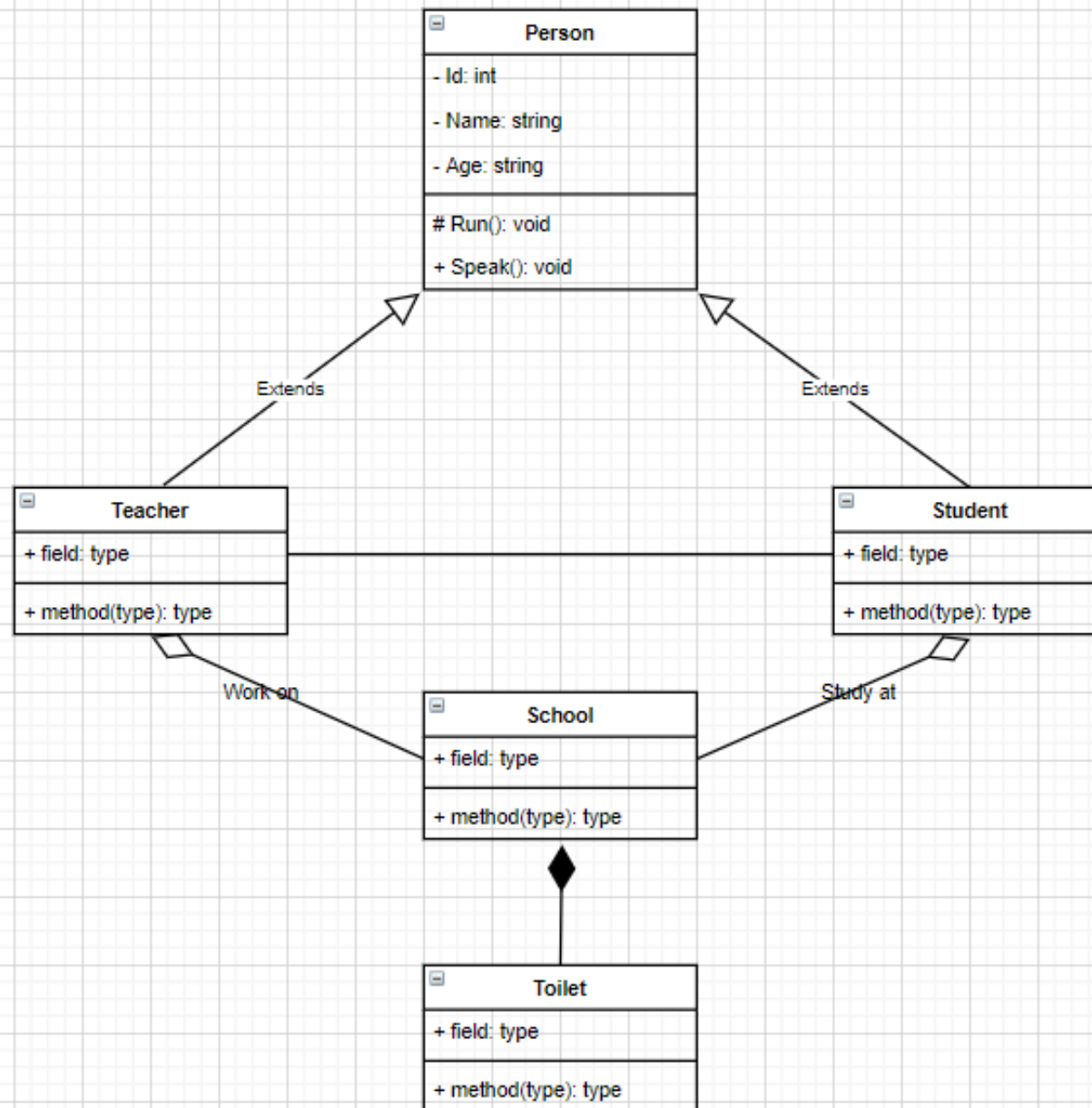


- Composition: Đối tượng tạo từ class A mất thì đối tượng tạo từ class B sẽ mất.



- Aggregation: Đối tượng tạo từ class A mất thì đối tượng tạo từ class B vẫn có thể tồn tại độc lập.

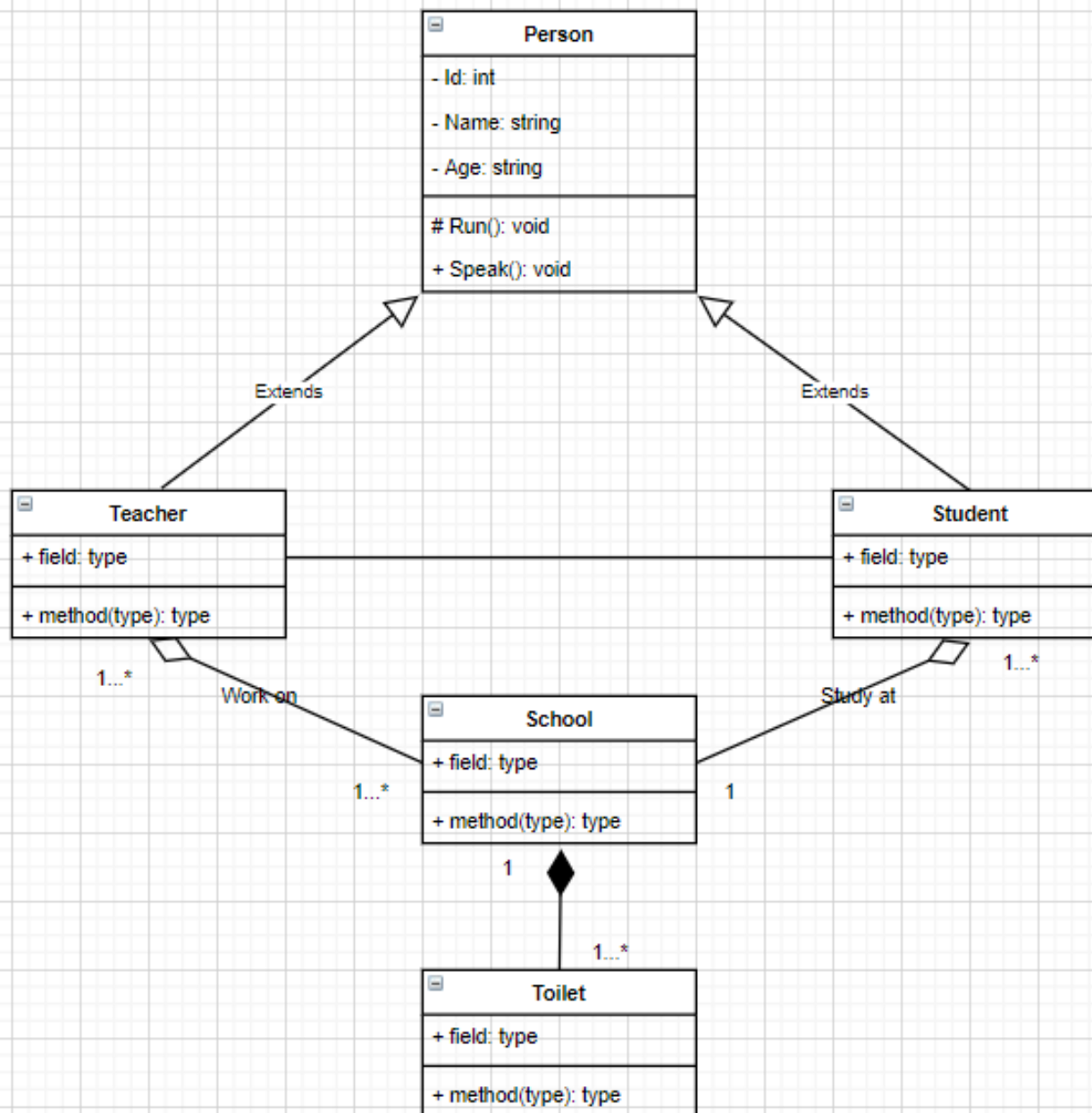




## **Multiplicity trong class diagram**

Sử dụng để thể hiện quan hệ về số lượng giữa các đối tượng được tạo từ các class trong class diagram

- $0...1$ : 0 hoặc 1
- $n$  : Bắt buộc có  $n$
- $0...*$  : 0 hoặc nhiều
- $1...*$  : 1 hoặc nhiều
- $m...n$ : có tối thiểu là  $m$  và tối đa là  $n$



## **Cách xây dựng bản vẽ Class**

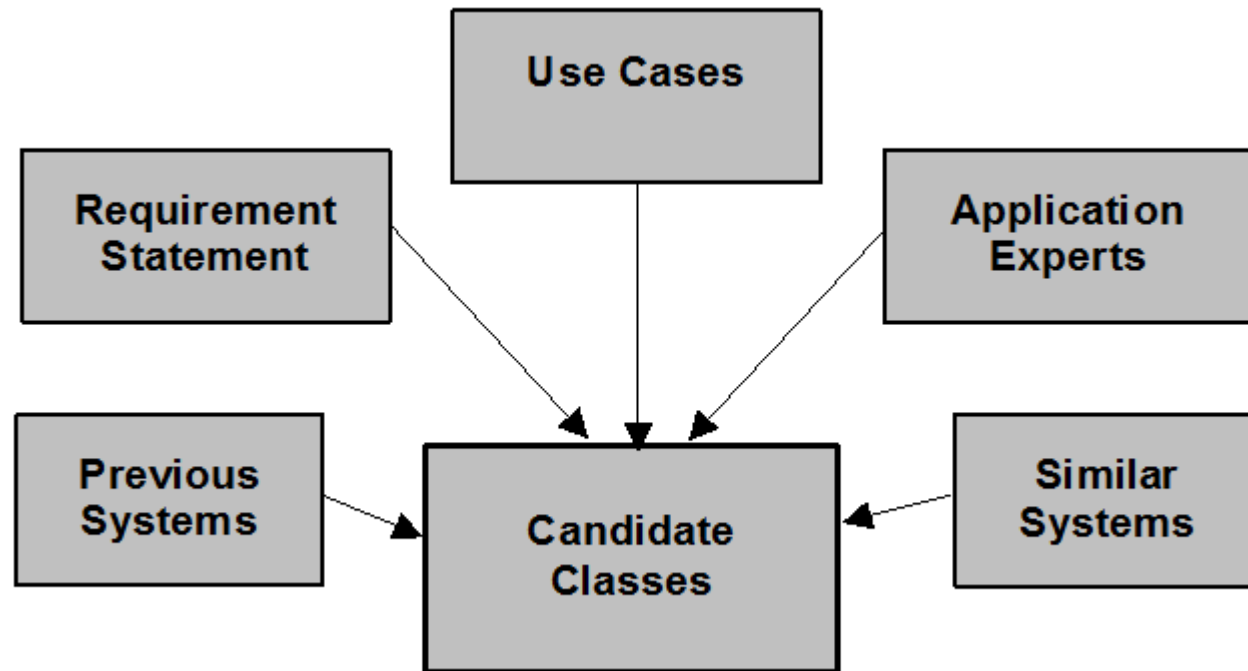
Class Diagram là bản vẽ khó xây dựng nhất so với các bản vẽ khác trong OOAD và UML. Bạn phải hiểu được hệ thống một cách rõ ràng và có kinh nghiệm về lập trình hướng đối tượng mới có thể xây dựng thành công bản vẽ này.

Thực hiện theo các bước sau đây để xây dựng Class Diagram.







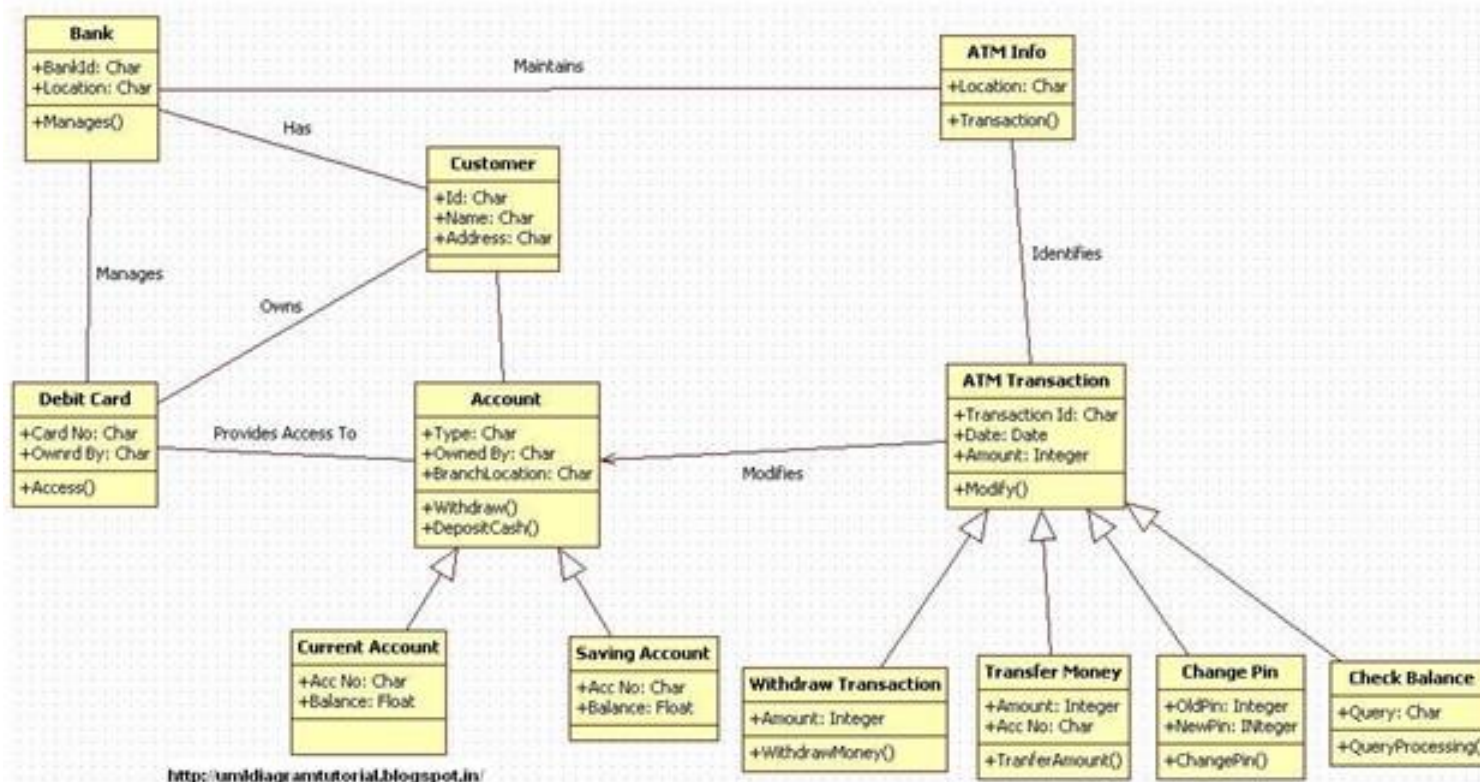
## Bước 1: Tìm các Classes dự kiến

Entity Classes(các lớp thực thể) là các thực thể có thật và hoạt động trong hệ thống, bạn dựa vào các nguồn sau để xác định chúng.



**Hình 7. Các nguồn thông tin có thể tìm Class dự kiến**

-  **Requirement statement:** Các yêu cầu. Chúng ta phân tích các danh từ trong các yêu cầu để tìm ra các thực thể.
-  **Use Cases:** Phân tích các Use Case sẽ cung cấp thêm các Classes dự kiến.
-  **Previous và Similar System:** có thể sẽ cung cấp thêm cho bạn các lớp dự kiến.
-  **Application Experts:** các chuyên gia ứng dụng cũng có thể giúp bạn.





Xem xét, ví dụ ATM ở trên chúng ta có thể thấy các đối tượng là Entity Class như sau:

- ✚ **Customers:** khách hàng giao dịch là một thực thể có thật và quản lý trong hệ thống.
- ✚ **Accounts:** Tài khoản của khách hàng cũng là một đối tượng thực tế.

- ✚ **ATM Cards:** Thẻ dùng để truy cập ATM cũng được quản lý trong hệ thống.
- ✚ **ATM Transactions:** Các giao dịch được lưu giữ lại, nó cũng là một đối tượng có thật.
- ✚ **Banks:** Thông tin ngân hàng bạn đang giao dịch, nếu có nhiều nhà Bank tham gia vào hệ thống bạn phải quản lý nó. Lúc đó Bank trở thành đối tượng bạn phải quản lý.
- ✚ **ATM:** Thông tin ATM bạn sẽ giao dịch. Nó cũng được quản lý tương tự như Banks.

**Lưu ý:** Chỉ các thực thể bên trong hệ thống được xem xét, các thực thể bên ngoài hệ thống không được xem xét. Ví dụ Customers là những người khách hàng được quản lý trong hệ thống chứ không phải người dùng máy ATM bên ngoài. Bạn phải lưu ý điều này để phân biệt Class và Actor.

## **Bước 2: Tìm các thuộc tính và phương thức cho lớp**

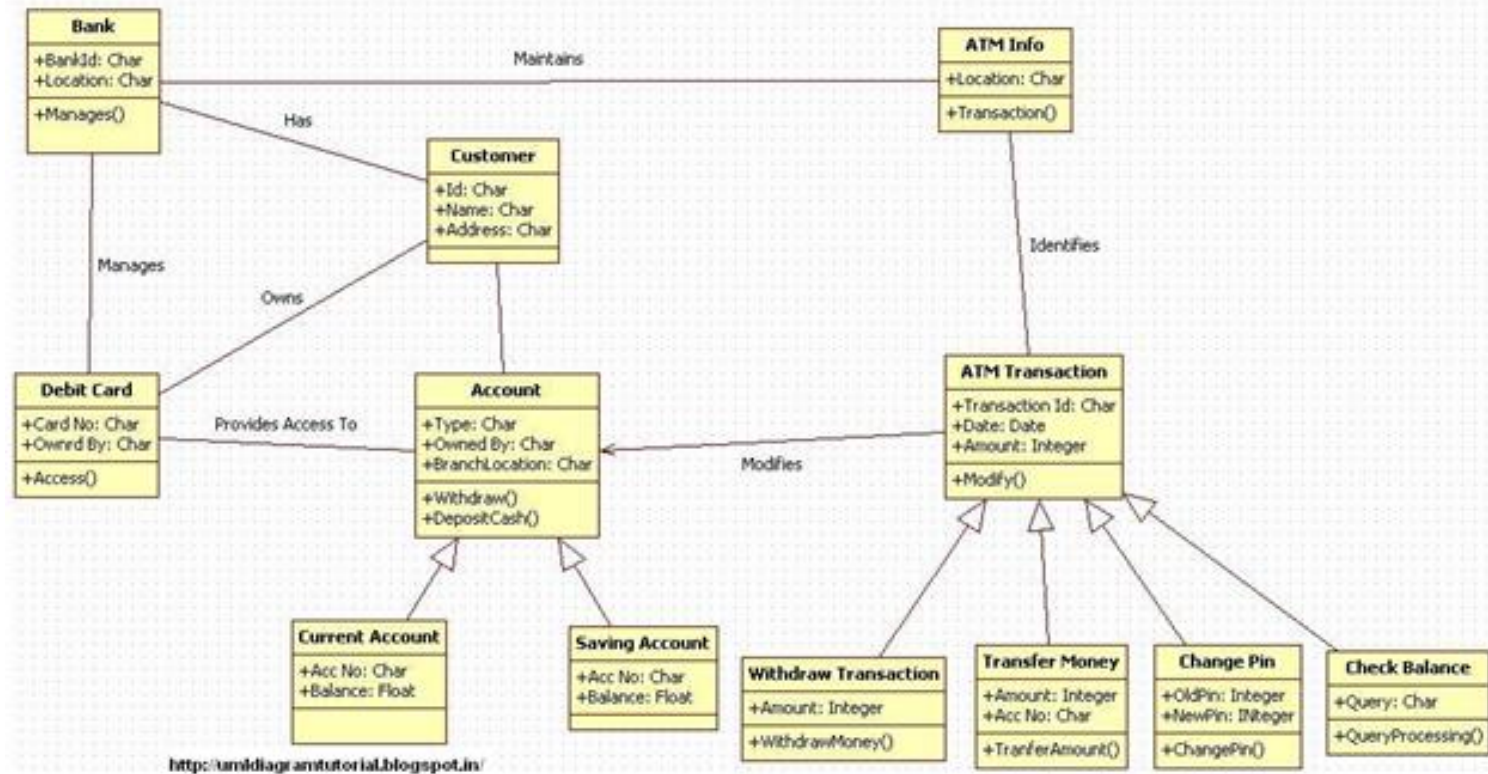
-  **Tìm thuộc tính:** phân tích thông tin từ các form mẫu có sẵn, bạn sẽ tìm ra thuộc tính cho các đối tượng của lớp. Ví dụ các thuộc tính của lớp Customer sẽ thể hiện trên Form đăng ký thông tin khách hàng.
-  **Tìm phương thức:** phương thức là các hoạt động mà các đối tượng của lớp này có thể thực hiện. Chúng ta sẽ bổ sung phương thức đầy đủ cho các lớp khi phân tích Sequence Diagram sau này.

### **Bước 3: Xây dựng các quan hệ giữa các lớp và phát hiện các lớp phát sinh**

Phân tích các quan hệ giữa các lớp và định nghĩa các lớp phát sinh do các quan hệ sinh ra. Chúng ta phân tích các thực thể ở trên và nhận thấy.

- ❖ Lớp *Accounts* có thể chia thành nhiều loại tài khoản như *Current Accounts* và *Saving Accounts* và có quan hệ thừa kế với nhau.
- ❖ Lớp *ATM Transactions* cũng có thể chia thành nhiều loại giao dịch như *Deposit*, *Withdraw*, *Transfer* v.v.. và chúng cũng có quan hệ thừa kế với nhau.

Tách chúng ta và vẽ chúng lên bản vẽ chúng ta sẽ có Class Diagram cho hệ thống ATM như sau:



**Hình 8. Ví dụ về Class Diagram cho hệ thống ATM**

# Thực hành Xây dựng Class Diagram cho hệ thống eCommerce

## Bước 1: Tìm các Classes dự kiến

Nghiên cứu kỹ các yêu cầu, Use Case và nghiên cứu kỹ các hệ thống tương tự để xác định các lớp dự kiến thông qua việc xác định các đối tượng có trong hệ thống.

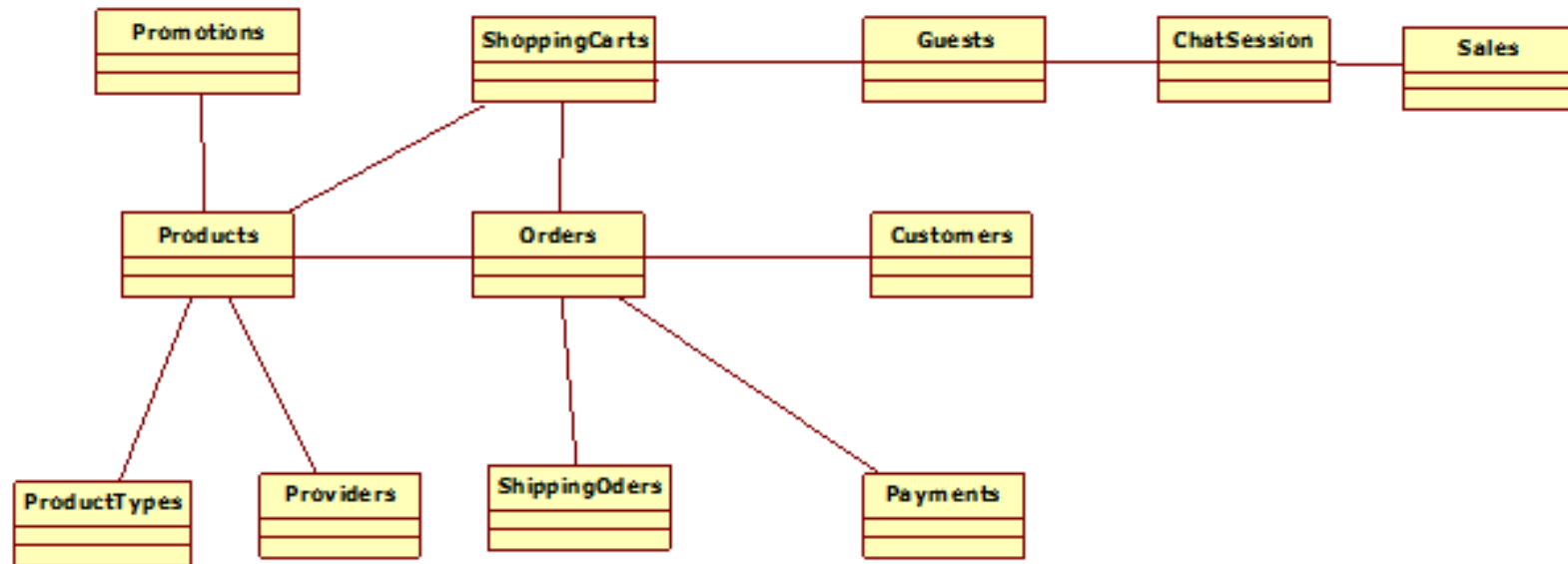
Xem xét Use Case Diagram của hệ thống:

- Phân tích Use Case “**Xem sản phẩm**” chúng ta xác định thực thể sản phẩm (**Products**). Sản phẩm được phân loại theo chủng loại (**Product Types**) và Nhà sản xuất (**Providers**) nên đây có thể là 2 lớp có quan hệ với class **Products**.
- Xem xét Use Case “**Xem khuyến mãi**” xác định Class Chương trình khuyến mãi (**Promotions**)



- Use Case “**Quản lý giỏ hàng**” -> Class giỏ hàng (**Shopping Carts**)
- Use Case Chat -> Class **Chat session**. Những người dùng tham gia Chat là **Sales** và **Guest** có thể là hai class dự kiến.
- Use Case “**Đăng ký thành viên**” -> Khách hàng (**Customers**)
- Use Case “**Quản lý đơn hàng**” -> Class đơn hàng (**Orders**), class thu tiền (**Payments**) và Quản lý chuyển hàng (**Shipping Orders**) có thể là 2 lớp có liên quan với Class **Orders**.

Tạm thời vẽ và xác định quan hệ sơ bộ chúng ta có bản vẽ Class dự kiến như sau:



## Hình 1. Bản vẽ sơ bộ Class Diagram khi phân tích các Use Case

Bản vẽ này giúp chúng ta có cái nhìn cơ bản về cấu trúc hệ thống để tiếp tục phân tích. Tất nhiên, bạn cần phân tích tất cả các Use Case còn lại và tìm hiểu thêm về hệ thống để bổ sung đầy đủ Class dự kiến cho hệ thống.

## **Bước 2: Xác định thuộc tính và quan hệ cho các lớp**

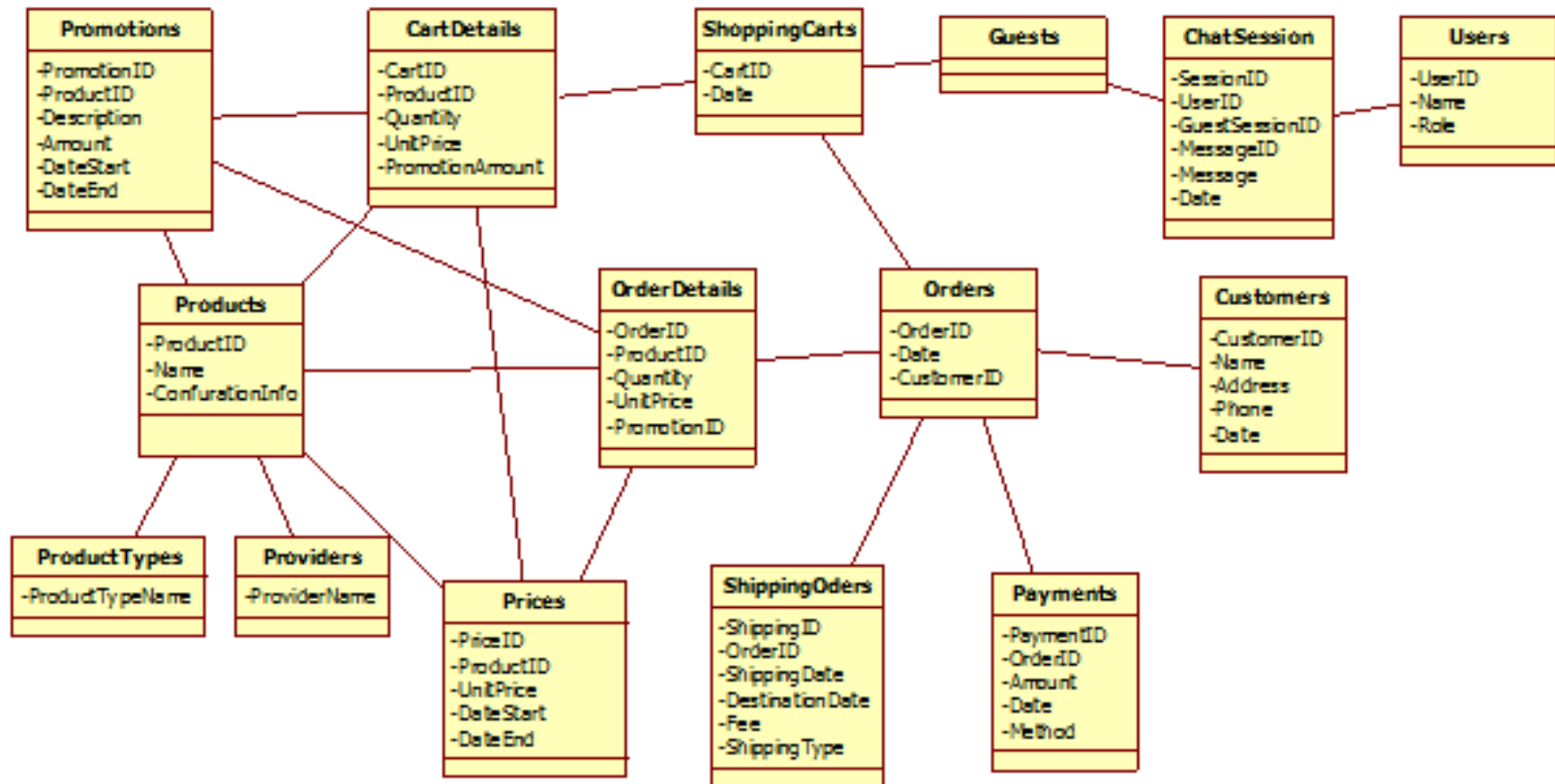
Chúng ta bổ sung các thuộc tính cho các lớp và phân tích quan hệ của chúng.

- **Products:** xem xét tài liệu mô tả sản phẩm của hệ thống chúng ta có thể thấy Class **Products** cần những thuộc tính sau: Tên sản phẩm, mô tả, cấu hình, Giá bán, khuyến mãi, bảo hành (xem mô tả chi tiết sản phẩm trên Website)... Trong đó, thuộc tính giá thay đổi theo thời gian nên chúng ta nên tách ra thành lớp riêng là Giá (**Prices**). Tương tự thuộc tính khuyến mãi cũng được tách ra thành lớp **Promotions**.
- **Prices:** có các thuộc tính là Mã sản phẩm, Giá, ngày bắt đầu, ngày hết hạn.

- **Promotions:** tương tự như giá nó cần có lớp riêng với các thuộc tính là Mã sản phẩm, Mô tả khuyến mãi, Giá trị khuyến mãi, Ngày bắt đầu, Ngày hết hạn.
- **ProductTypes:** chứa loại sản phẩm
- **Providers:** chứa tên nhà sản xuất
- **ShoppingCarts:** chứa các thông tin như: cartID, ngày, mã sản phẩm, số lượng, đơn giá. Chúng ta nhận thấy nếu để nguyên lớp này khi tạo đối tượng chúng sẽ lặp thông tin cartID và ngày mua nên tách chúng ra thành **ShoppingCarts** với các thuộc tính CartID, ngày và **CartDetails** với các thuộc tính ProductID, số lượng, đơn giá.
- Tương tự chúng ta có class **Orders** với OrderID, ngày, customerID và class **Orderdetails** với ProductID, số lượng, đơn giá.

- **Payments:** chứa các thông tin như PaymentID, OrderID, ngày trả, số tiền, hình thức thanh toán.
- **Shippings:** có thể chứa ShippingID, OrderID, Ngày chuyển, ngày đến, số tiền, phương thức vận chuyển.
- **Customers:** CustomerID, Họ và tên, địa chỉ, điện thoại, ngày đăng ký v.v...
- **Guests:** có thể chứa sessionID để xác định thông tin khi chat
- **Sales:** có thể gộp với lớp người dùng (**Users**) chứa UserID, Name
- **ChatSessions:** ChatsessionID, tên người bán hàng, mã khách, mã tin nhắn, nội dung tin nhắn, ngày.

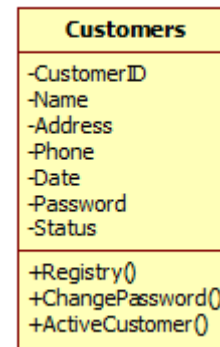
Nhập đầy đủ thuộc tính và vẽ chúng ra, chúng ta có bản vẽ như sau:



**Hình 2. Bản vẽ Class Diagram sau khi thêm thuộc tính và tách các quan hệ**

### Bước 3: Bổ sung phương thức cho các lớp

Phương thức là các hành động mà đối tượng sinh ra từ lớp đó có thể thực hiện trong hệ thống. Ví dụ các đối tượng của lớp **Customers** có thể đăng ký mới, có thể thay đổi mật khẩu (password), kích hoạt người dùng (Active) v.v..



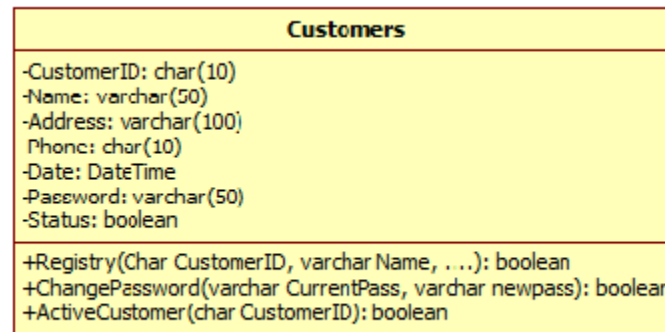
### Hình 3. Ví dụ về các phương thức

Có một vấn đề ở đây là chúng ta rất khó xác định chính xác các phương thức của một lớp. Nếu thiếu bạn sẽ không thể cài đặt đủ yêu cầu chức năng, nếu thừa bạn tốn công cài đặt vô ích mà không dùng đến. Bạn có thể đối chiếu khi phân tích và thiết kế tất cả các Use Case của hệ thống.

## Bước 4: Thiết kế chi tiết các thuộc tính và phương thức cho lớp

Khi đã có được Class Diagram, bạn cần thiết kế chi tiết các lớp bằng cách đặc tả các thuộc tính và phương thức của nó.

- ❖ Đặc tả thuộc tính: chúng ta xác định kiểu dữ liệu và kích thước.
- ❖ Đặc tả phương thức: chúng ta xác định dữ liệu đầu vào, dữ liệu đầu ra.



**Hình 4. Ví dụ về thiết kế lớp Customers**



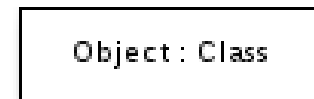
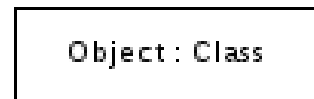
Việc sử dụng các kiểu dữ liệu và mô tả các phương thức của một lớp chúng ta đã học kỹ trong lập trình hướng đối tượng nên chúng ta không bàn ở đây. Hoàn tất các bước trên cho toàn bộ các Use Case chúng ta sẽ có bản vẽ Class hoàn chỉnh.

## 4. Biểu đồ Trình tự (Sequence Diagram)

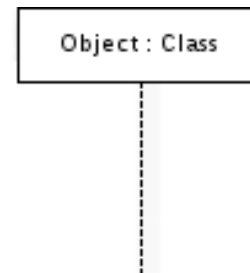
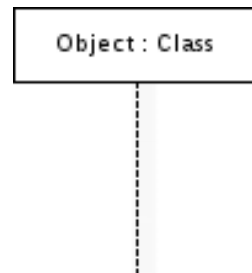
Biểu đồ trình tự (hay tuần tự) là biểu đồ dùng để xác định các trình tự diễn ra sự kiện của một nhóm đối tượng nào đó. Nó miêu tả chi tiết các thông điệp được gửi và nhận giữa các đối tượng đồng thời cũng chú trọng đến việc trình tự về mặt thời gian gửi và nhận các thông điệp đó.

### Các thành phần của biểu đồ tuần tự

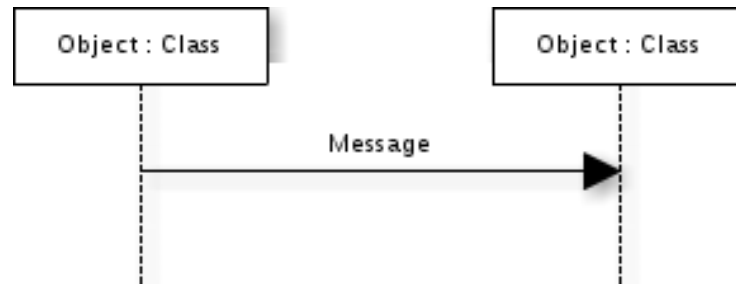
Đối tượng (object or class): biểu diễn bằng các hình chữ nhật



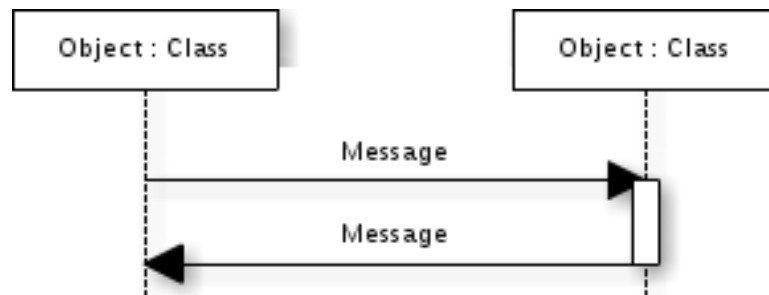
Đường đời đối tượng (Lifelines): biểu diễn bằng các đường gạch rời thẳng đứng bên dưới các đối tượng



Thông điệp (Message): biểu diễn bằng các đường mũi tên



Thông điệp được dùng để giao tiếp giữa các đối tượng và lớp.  
Có nhiều loại thông điệp được định nghĩa ở phần sau.  
Xử lý bên trong đối tượng (biểu diễn bằng các đoạn hình chữ nhật rỗng nối với các đường đời đối tượng)



## Các loại thông điệp trong biểu đồ tuần tự

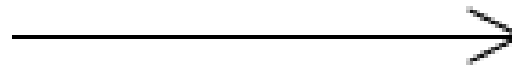
Thông điệp đồng bộ (Synchronous Message)

Thông điệp đồng bộ cần có một request trước hành động tiếp theo.



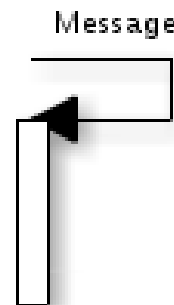
Thông điệp không đồng bộ (Asynchronous Message)

Thông điệp không đồng bộ không cần có một request trước hành động tiếp theo.



Thông điệp chính mình (Self Message)

Là thông điệp mà đối tượng gửi cho chính nó để thực hiện các hàm nội tại.



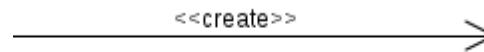
- Thông điệp trả lời hoặc trả về (Reply or Return Message)

Là thông điệp trả lời lại khi có request hoặc sau khi kiểm tra tính đúng đắn của một điều kiện nào đó. Ví dụ thông điệp loại này như tin nhắn trả về là success hoặc fail

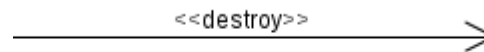


- Thông điệp tạo mới (Create Message)

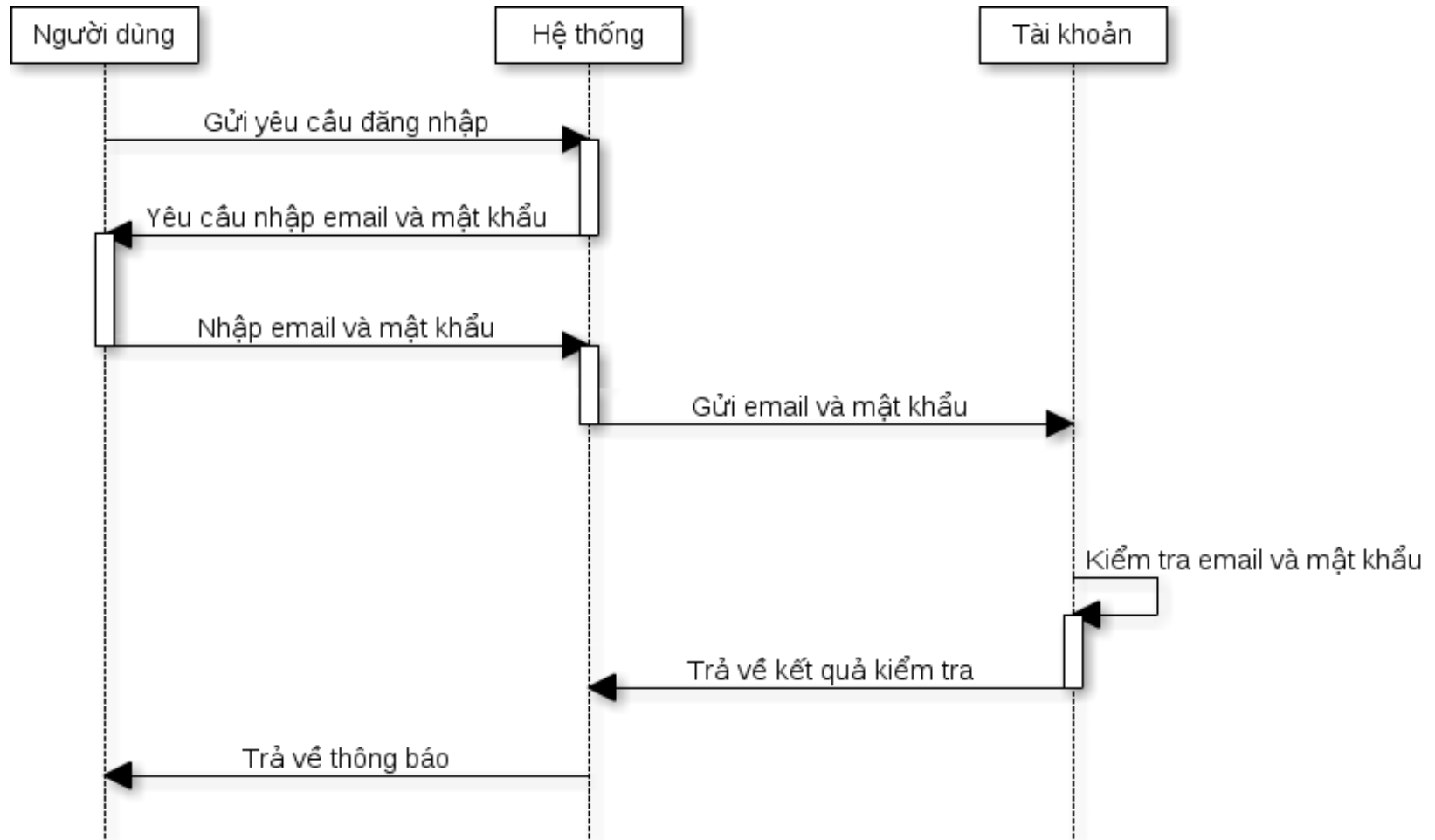
Là thông điệp được trả về khi tạo mới một đối tượng.



- Thông điệp xóa (Delete Message) Là thông điệp được trả về khi xóa một đối tượng.



# Ví dụ



Trong sơ đồ trên có 3 đối tượng là : người dùng, hệ thống và tài khoản. Luồng xử lí của chức năng đăng nhập có thể diễn giải như sau.

1. Người dùng gửi yêu cầu đăng nhập đến hệ thống.
2. Hệ thống yêu cầu người dùng nhập email và mật khẩu.
3. Người dùng nhập email và mật khẩu.
4. Hệ thống gửi email và mật khẩu của người dùng để kiểm tra.
5. Tài khoản kiểm tra thông tin email và password có đúng hay không.
6. Tài khoản trả về kết quả kiểm tra cho hệ thống.
7. Hệ thống trả về thông báo cho người dùng.

# Thực hành xây dựng Sơ đồ trình tự

## *Bước 1: Xác định các Use Case cần thiết kế*

Tương tự như Activity Diagram, chúng ta cũng cần xác định các Use Case mà chúng ta cần sử dụng sequence Diagram để thiết kế chi tiết.

Xem xét bản vẽ Use Case Diagram chúng ta đã vẽ ở bài trước, chúng ta có thể thấy các Use Case sau cần thiết kế:

- Xem sản phẩm theo chủng loại
- Thêm sản phẩm theo nhà cung cấp
- Thêm giỏ hàng
- Chat
- Quản lý đơn hàng
- Thanh toán
- Theo dõi chuyển hàng
- Đăng nhập



Tiếp theo, chúng ta sẽ thiết kế cho chức năng “**Xem sản phẩm theo chủng loại**”.

***Bước 2: Xem Activity Diagram cho Use Case này chúng ta xác định các bước sau:***

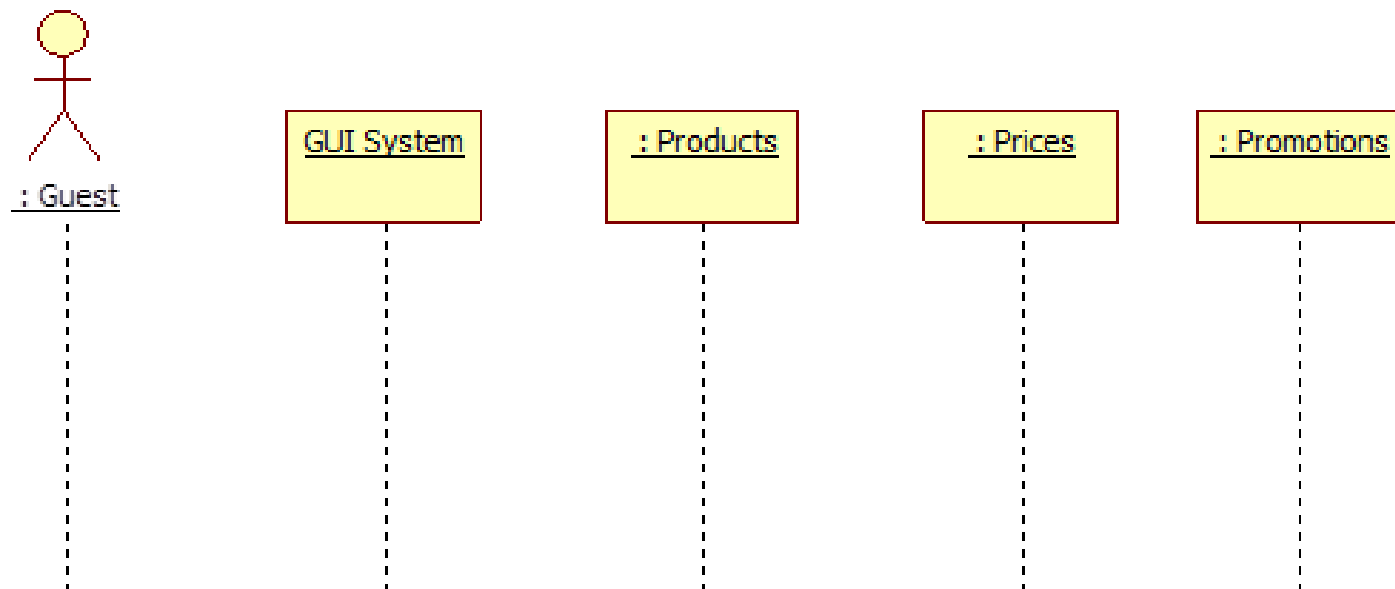
- Người dùng chọn loại sản phẩm
- Hệ thống sẽ lọc lấy loại sản phẩm tương ứng, sau đó lấy giá, lấy khuyến mãi và hiển thị lên màn hình.
- Người dùng xem sản phẩm

***Bước 3: Đối chiếu với Class Diagram chúng ta xác định các đối tượng thực hiện như sau:***

- **Người dùng:** chọn loại sản phẩm qua giao diện
- **Giao diện:** sẽ lấy danh sách sản phẩm tương ứng từ **Products**
- **Giao diện:** lấy giá của từng sản phẩm từ Class **Prices** và Promotion Amount từ lớp **Promotions**
- **Giao diện:** tổng hợp danh sách và hiển thị
- **Người dùng:** Xem sản phẩm

## ***Bước 4: Vẽ sequence Diagram***

- Xác định các lớp tham gia vào hệ thống gồm: người dùng (Guest), Giao diện (GUI System), Sản phẩm (Products), Giá (Prices), Khuyến mãi (Promotions). Trong đó GUI System để sử dụng chung cho giao diện, bạn có thể sử dụng cụ thể trang Web nào nếu bạn đã có Mockup (thiết kế chi tiết của giao diện).

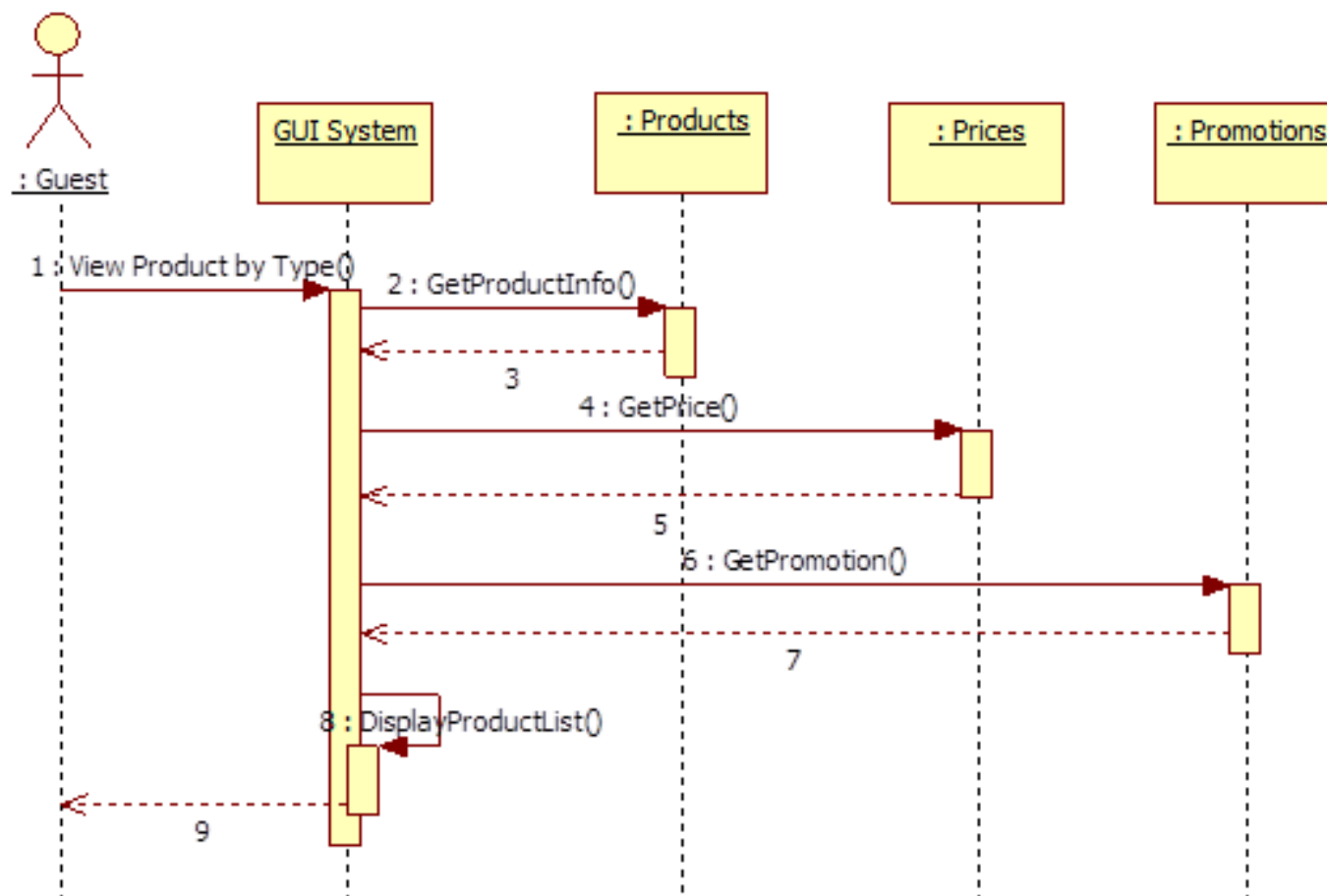


**Hình 1. Xác định các đối tượng tham gia vào bản vẽ**

Các bước thực hiện của Use Case này như sau:

- . – **Guest** gửi yêu cầu xem sản phẩm lên giao diện kèm theo chủng loại
- . – **GUI system**: gửi yêu cầu lấy danh sách các sản phẩm tương ứng với chủng loại cho lớp sản phẩm và nhận lại danh sách.
- . – **GUI system**: gửi yêu cầu lấy Giá cho từng sản phẩm từ Prices
- . – **GUI system**: gửi yêu cầu lấy khuyến mãi cho từng sản phẩm từ Promotions và nhận lại kết quả
- . – **GUI system**: ghép lại danh sách và hiển thị lên browser và trả về cho Guest




Thể hiện lên bản vẽ như sau:



**Hình 2. Bản vẽ Sequence Diagram cho chức năng Xem sản phẩm theo chủng loại.**

## ***Bước 5: Kiểm tra và cập nhật bản vẽ Class Diagram***

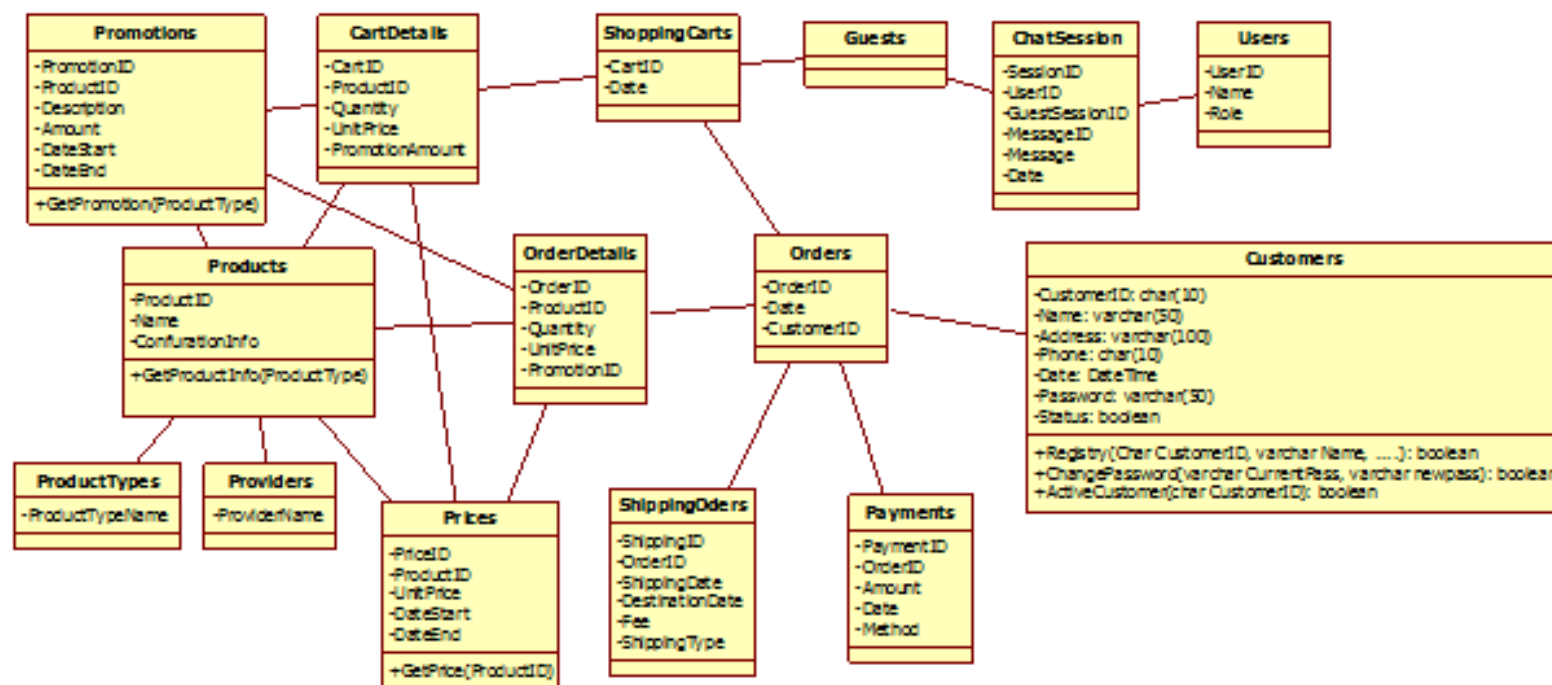
Chúng ta nhận thấy để thực hiện được bản vẽ trên chúng ta cần bổ sung các phương thức cho các lớp như sau:

-  **Products** class: bổ sung phương thức **GetProductInfo(Product Type)**: trả về thông tin sản phẩm có loại được truyền vào. Việc này các đối tượng của lớp Products hoàn toàn làm được vì họ đã có thuộc tính ProductType nên họ có thể trả về được thông tin này.
-  **Prices**: bổ sung phương thức **GetPrice(ProductID)**: UnitPrice. Sau khi lấy được ProductID từ Products, GUI gọi phương thức này để lấy giá của sản phẩm từ lớp giá. Các đối tượng từ lớp Prices hoàn toàn đáp ứng điều này.
-  **Promotions**: tương tự bổ sung phương thức **GetPromotion(ProductID)**.



**GUI System(View Product Page):** bổ sung phương thức **DisplayProductList(List of product)** để hiển thị danh sách lên sản phẩm. Ngoài ra, bạn cần có thêm một phương thức **ViewProductbyType(ProductType)** để mô tả chính hoạt động này khi người dùng kích chọn.

Như vậy, chúng ta thấy các phương thức trên đều thực hiện được trên các đối tượng của các lớp nên thiết kế của trên là khả thi. Bổ sung các phương thức trên vào các Class tương ứng chúng ta có bản vẽ Class Diagram như sau:



**Hình 3. Class Diagram sau khi đã bổ sung các phương thức mới**

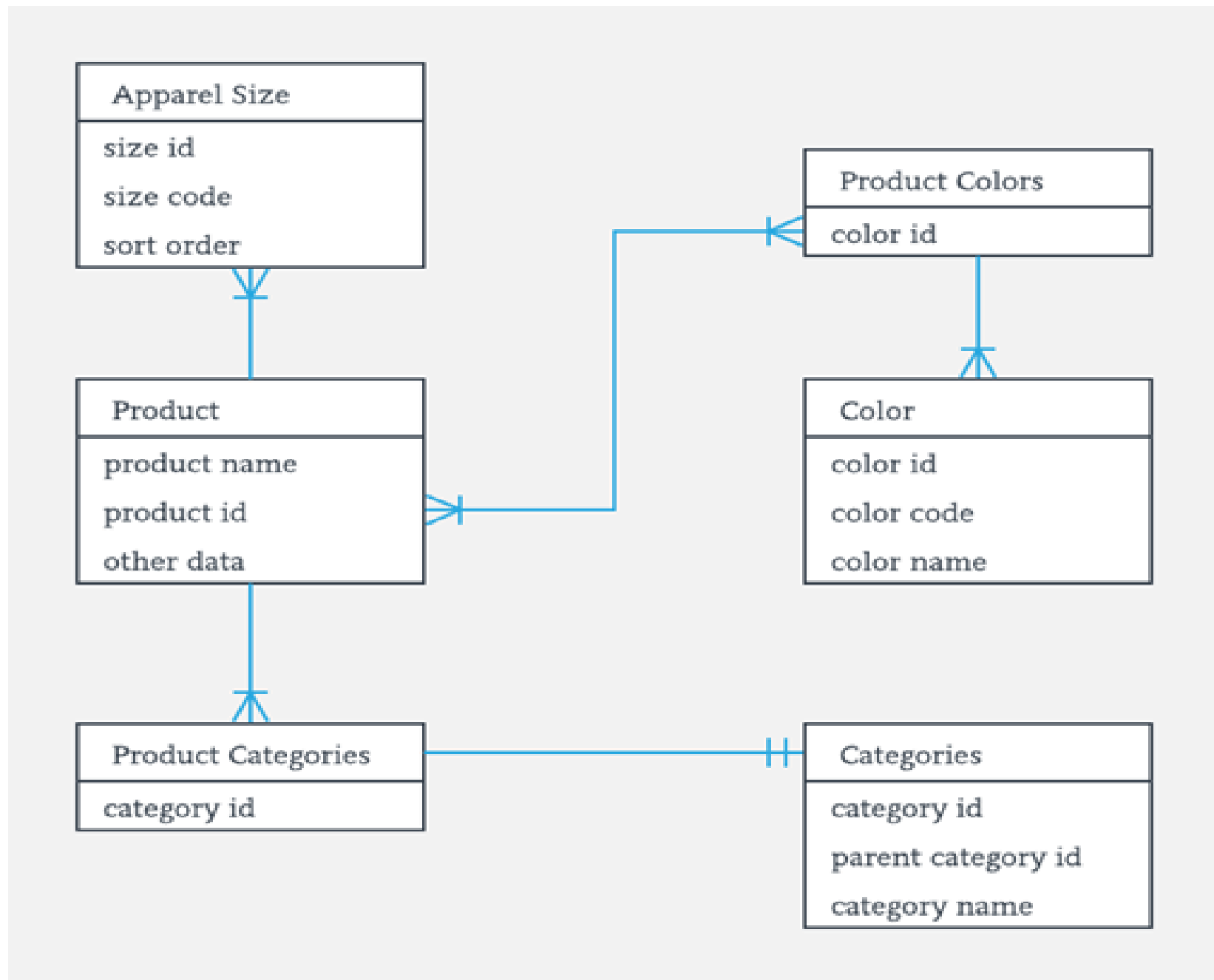


## **5. Sơ đồ Bảng dữ liệu (Database Relationship Diagram)**

## ER Diagram là gì?

**ER Diagram** viết tắt của **Entity Relationship Diagram**, còn được gọi là **ERD** là một sơ đồ hiển thị **mối quan hệ của các tập thực thể được lưu trữ trong cơ sở dữ liệu**. Nói cách khác, sơ đồ ER giúp **giải thích cấu trúc logic** của cơ sở dữ liệu. Biểu đồ ER được tạo ra dựa trên ba khái niệm cơ bản: **thực thể, thuộc tính và mối quan hệ**.

Biểu đồ ER chứa các ký hiệu khác nhau sử dụng **hình chữ nhật** để đại diện cho **các thực thể**, **hình bầu dục** để xác định các **thuộc tính** và **hình thoi** để biểu thị các **mối quan hệ**.



Sơ đồ mối quan hệ thực thể

# Tại sao nên dùng ER Diagram?

- + Giúp bạn xác định các thuật ngữ liên quan đến mô hình mối quan hệ thực thể
- + Cung cấp bản xem trước về cách tất cả các bảng của bạn sẽ kết nối, các trường sẽ có trên mỗi bảng
- + Giúp mô tả các thực thể, thuộc tính, mối quan hệ
- + Biểu đồ ER có thể chuyển thành các bảng quan hệ cho phép bạn xây dựng cơ sở dữ liệu một cách nhanh chóng
- + Biểu đồ ER có thể được sử dụng bởi các nhà thiết kế cơ sở dữ liệu như một bản thiết kế để triển khai dữ liệu trong các ứng dụng phần mềm cụ thể
- + Người thiết kế cơ sở dữ liệu hiểu rõ hơn về thông tin được chứa trong cơ sở dữ liệu với sự trợ giúp của sơ đồ ERP
- + Sơ đồ ERD cho phép bạn giao tiếp với cấu trúc logic của cơ sở dữ liệu với người dùng

## **Các thành phần của Sơ đồ ER**

Mô hình này dựa trên ba khái niệm cơ bản:

- . Thực thể
- . Thuộc tính
- . Các mối quan hệ

## **Ví dụ về Sơ đồ ER**

Ví dụ: trong cơ sở dữ liệu trường Đại học, chúng ta có thể có các thực thể dành cho Sinh viên, Khóa học và Giảng viên. Thực thể sinh viên có thể có các thuộc tính như Rollno, Name và DeptID. Họ có thể có mối quan hệ với các Khóa học và Giảng viên.



Entity Name

### Entity

Person, place, object, event or concept about which data is to be maintained

**Example:** Car, Student

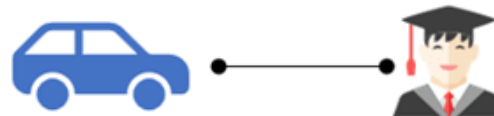


Attribute Name

### Attribute

Property or characteristic of an entity

**Example:** Color of car Entity Name of Student Entity



### Relation

Verb Phrase

Association between the instances of one or more entity types

**Example:** Blue Car Belongs to Student Jack



Các thành phần của Sơ đồ ER

# ENTITY LÀ GÌ?

Một sự vật trong thế giới thực hoặc sống hoặc không sống, có thể dễ dàng nhận ra và không thể nhận ra. Đó là bất kỳ thứ gì trong doanh nghiệp được thể hiện trong cơ sở dữ liệu của chúng tôi. Nó có thể là một điều vật chất hoặc đơn giản là một sự thật về doanh nghiệp hoặc một sự kiện xảy ra trong thế giới thực.

Một thực thể có thể là địa điểm, người, đối tượng, sự kiện hoặc một khái niệm, lưu trữ dữ liệu trong cơ sở dữ liệu. Đặc điểm của các thực thể là phải có một thuộc tính và một khóa duy nhất. Mọi thực thể đều được tạo thành từ một số 'thuộc tính' đại diện cho thực thể đó.

## Ví dụ về các thực thể:

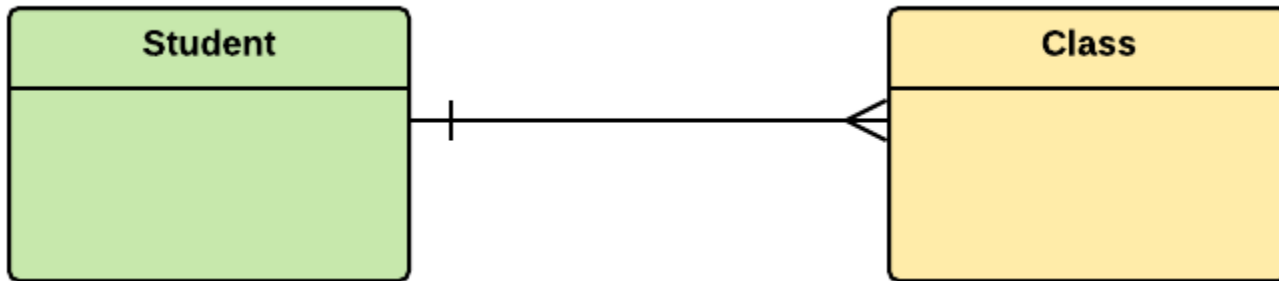
- . **Người:** Nhân viên, Sinh viên, Bệnh nhân
- . **Địa điểm:** Cửa hàng, Tòa nhà
- . **Đối tượng:** Máy móc, sản phẩm và ô tô
- . **Sự kiện:** Bán, Đăng ký, Gia hạn
- . **Khái niệm:** Tài khoản, Khóa học

# Ký hiệu của một thực thể

## Bộ thực thể:

Sinh viên

Tập thực thể là một nhóm các thực thể giống nhau. Nó có thể chứa các thực thể có thuộc tính chia sẻ các giá trị tương tự. Các thực thể được đại diện bởi các thuộc tính của chúng, còn được gọi là các thuộc tính. Tất cả các thuộc tính đều có giá trị riêng biệt. Ví dụ, một thực thể sinh viên có thể có tên, tuổi, lớp, dưới dạng các thuộc tính.





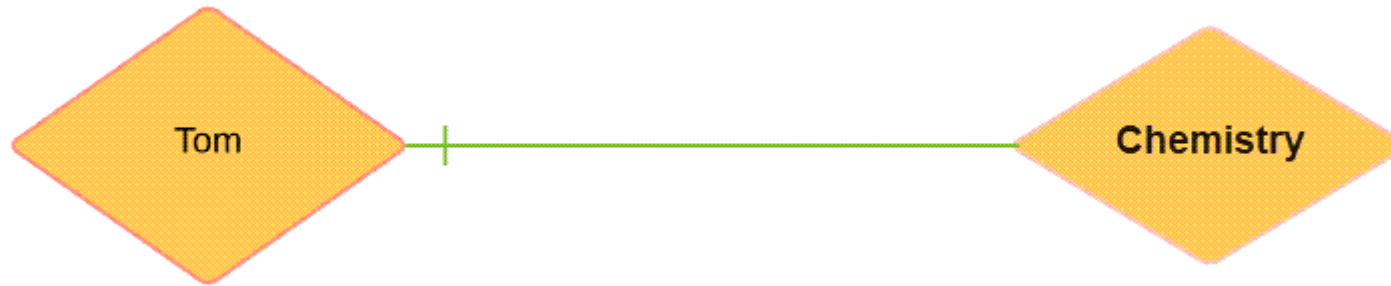
## **Ví dụ về các thực thể:**

Một trường đại học có thể có một số khoa. Tất cả các khoa này tuyển dụng nhiều giảng viên khác nhau và cung cấp một số chương trình.

Một số khóa học tạo nên mỗi chương trình. Học sinh đăng ký trong một chương trình cụ thể và ghi danh vào các khóa học khác nhau. Một giảng viên từ bộ phận cụ thể đảm nhận mỗi khóa học và mỗi giảng viên dạy cho một nhóm sinh viên khác nhau.

# Mối quan hệ

Mối quan hệ không là gì ngoài sự liên kết giữa hai hoặc nhiều thực thể. Ví dụ: Tom làm việc ở khoa Hóa học.



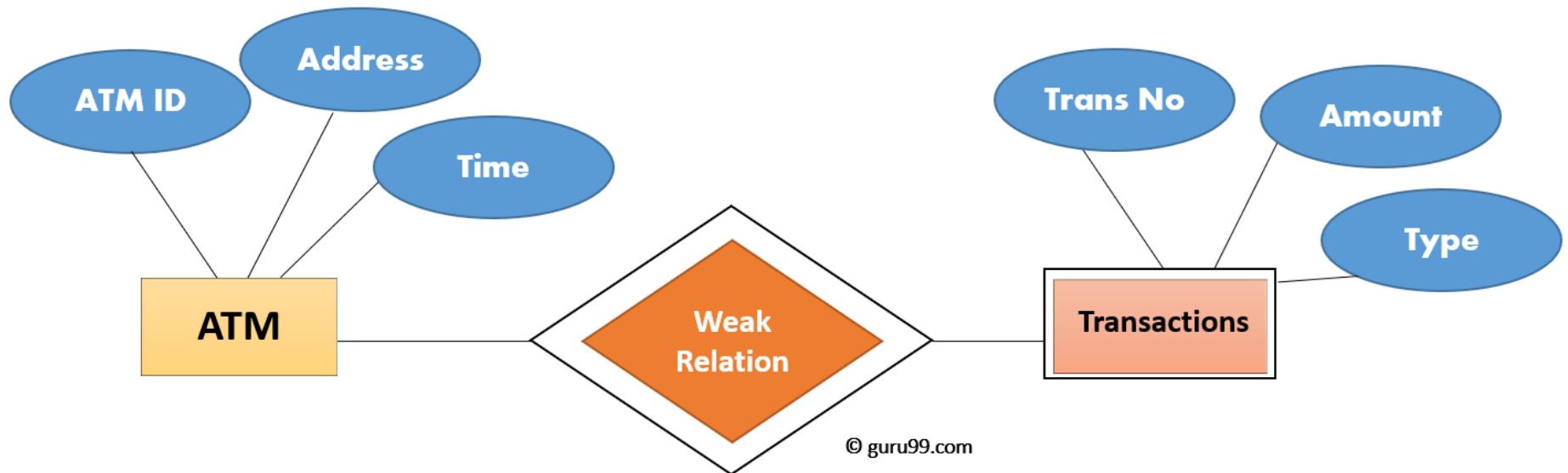
Các thực thể tham gia vào các mối quan hệ. Chúng ta thường có thể xác định mối quan hệ với động từ hoặc cụm động từ.

## Ví dụ:

- . Bạn đang tham dự bài giảng này
- . Chúng ta có thể phân loại các mối quan hệ theo các kiểu mối quan hệ:
- . Một sinh viên tham dự một bài giảng
- . Một giảng viên đang giảng bài.

# Thực thể yếu

Thực thể yếu là một loại thực thể không có thuộc tính khóa của nó. Nó có thể được xác định duy nhất bằng cách xem xét khóa chính của một thực thể khác. Vì vậy, các tập hợp thực thể yếu kém cần phải có sự tham gia.



Trong các ví dụ về Sơ đồ ER ở trên, “Trans No” là một dấu hiệu phân biệt trong một nhóm các giao dịch trong máy ATM.

Hãy cùng tìm hiểu thêm về một thực thể yếu bằng cách so sánh nó với một Thực thể mạnh

### **Bộ thực thể mạnh**

### **Nhóm thực thể yếu**

Tập thực thể mạnh luôn có khóa chính.

Nó không có đủ thuộc tính để tạo khóa chính.

Nó được biểu thị bằng một biểu tượng hình chữ nhật.

Nó được biểu thị bằng một biểu tượng hình chữ nhật đôi.

Nó chứa một khóa chính được biểu thị bằng ký hiệu gạch dưới.

Nó chứa một Khóa một phần được biểu thị bằng một ký hiệu gạch dưới gạch ngang.

Thành viên của một tập thực thể mạnh được gọi là tập hợp thực thể thống trị.

Thành viên của một tập thực thể yếu được gọi là một tập thực thể cấp dưới.

Khóa chính là một trong những thuộc tính giúp xác định thành viên của nó.

Trong tập thực thể yếu, nó là sự kết hợp của khóa chính và khóa một phần của tập thực thể mạnh.

Trong biểu đồ ER, mỗi quan hệ Mỗi quan hệ giữa một tập hợp thực giữa hai tập thực thể mạnh được thể mạnh và yếu được thể hiện thể hiện bằng cách sử dụng biểu bằng cách sử dụng biểu tượng kim tượng hình thoi. cương đôi.

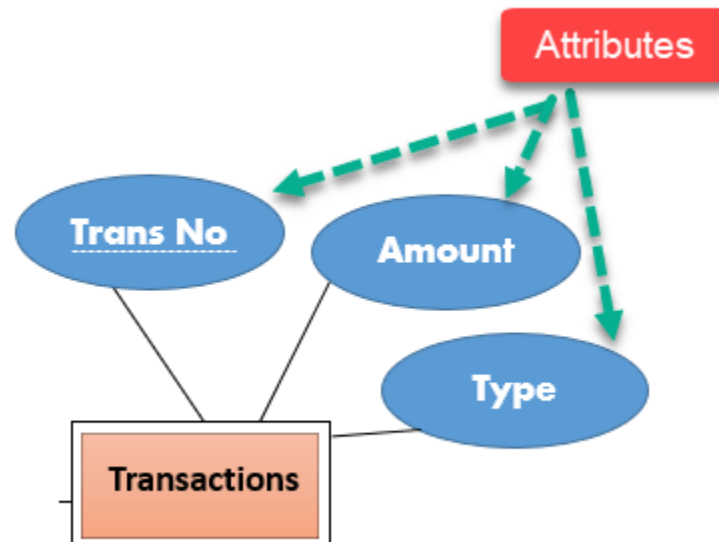
Đường kết nối của tập thực thể Đường kết nối tập thực thể yếu để mạnh với mỗi quan hệ là duy xác định mỗi quan hệ là đường đôi. nhất.

# Thuộc tính

Nó là một thuộc tính có giá trị duy nhất của kiểu thực thể hoặc kiểu quan hệ.

Ví dụ, một bài giảng có thể có các thuộc tính: thời gian, ngày tháng, thời lượng, địa điểm, v.v.

Một thuộc tính trong các ví dụ về Sơ đồ ER, được biểu thị bằng một hình Elip



## Các loại thuộc tính

## Sự miêu tả

### Thuộc tính đơn giản

Các thuộc tính đơn giản không thể được phân chia thêm nữa. Ví dụ, số liên lạc của một sinh viên. Nó còn được gọi là giá trị nguyên tử.

### Thuộc tính tổng hợp

Có thể chia nhỏ thuộc tính tổng hợp. Ví dụ, tên đầy đủ của học sinh có thể được chia thành họ, tên, và họ.

### Thuộc tính có nguồn gốc

Loại thuộc tính này không bao gồm trong cơ sở dữ liệu vật lý. Tuy nhiên, giá trị của chúng được lấy từ các thuộc tính khác có trong cơ sở dữ liệu. Ví dụ, tuổi không nên được lưu trữ trực tiếp. Thay vào đó, nó phải được lấy từ DOB của nhân viên đó.

### Thuộc tính đa giá trị

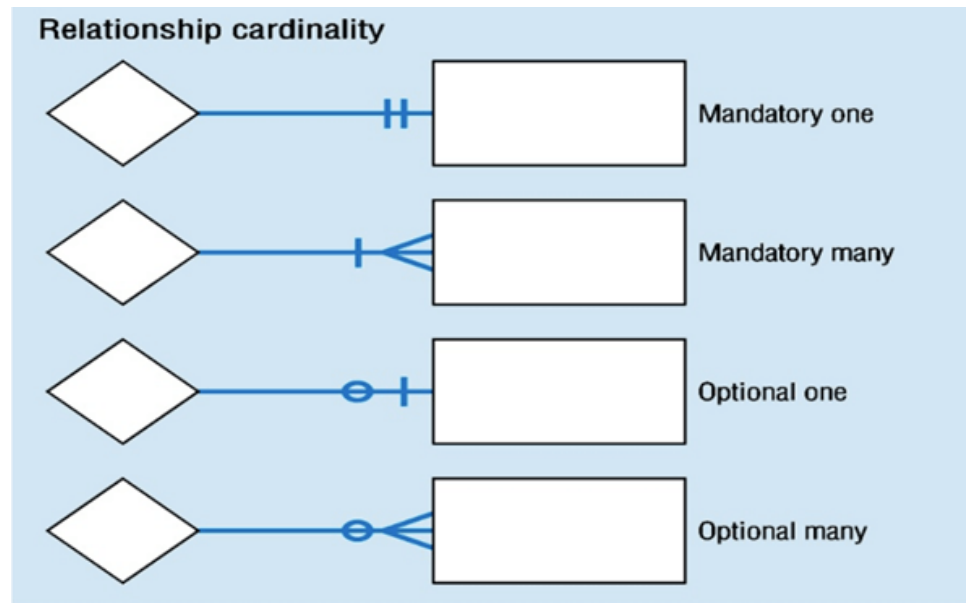
Thuộc tính nhiều giá trị có thể có nhiều giá trị. Ví dụ, một sinh viên có thể có nhiều hơn một số điện thoại di động, địa chỉ email, v.v.

# Cardinality

Xác định các thuộc tính số của mỗi quan hệ giữa hai thực thể hoặc tập thực thể.

Các loại mối quan hệ cơ bản khác nhau là:

- . Mối quan hệ một-một
- . Mối quan hệ một-nhiều
- . Mối quan hệ có thể thành một
- . Mối quan hệ nhiều-nhiều

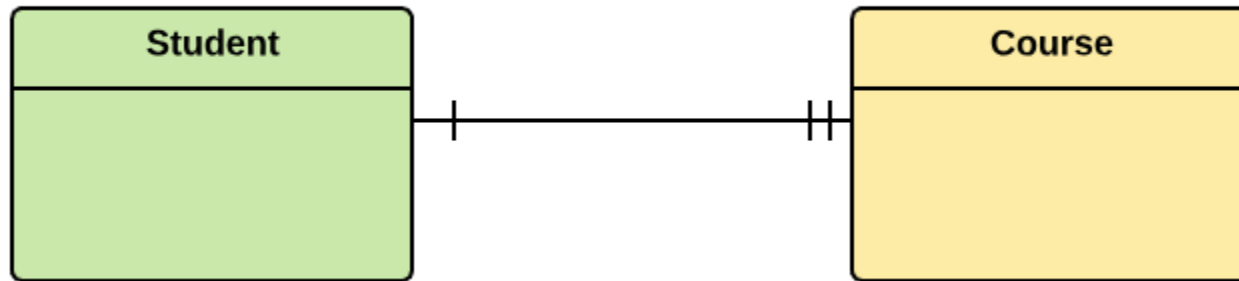




## 1. một-một:

Một thực thể từ tập thực thể X có thể được liên kết với nhiều nhất một thực thể của tập thực thể Y và ngược lại.

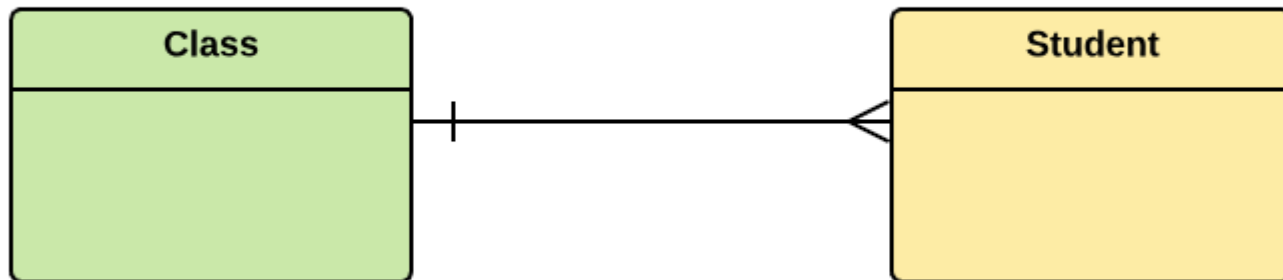
Ví dụ: Một sinh viên có thể đăng ký nhiều khóa học. Tuy nhiên, tất cả các khóa học đó đều có một dòng duy nhất quay lại một sinh viên đó.



## 2. một-nhiều:

Một thực thể từ tập thực thể X có thể được liên kết với nhiều thực thể của tập thực thể Y, nhưng một thực thể từ tập thực thể Y có thể được liên kết với ít nhất một thực thể.

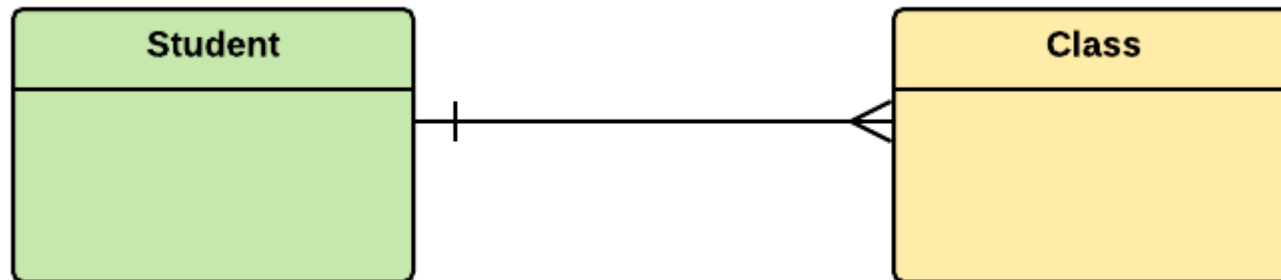
Ví dụ, một lớp học bao gồm nhiều sinh viên.



### 3. Nhiều đến một

Nhiều thực thể từ tập thực thể X có thể được liên kết với nhiều nhất một thực thể của tập thực thể Y. Tuy nhiên, một thực thể từ tập thực thể Y có thể được liên kết hoặc không với nhiều thực thể từ tập thực thể X.

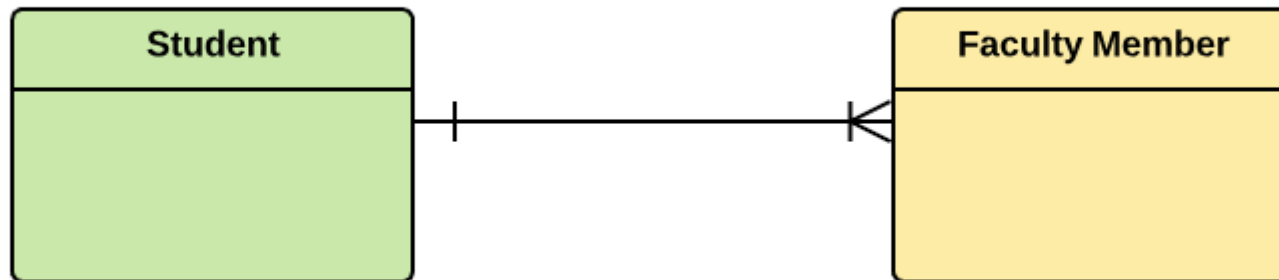
Ví dụ, nhiều học sinh thuộc cùng một lớp.



## 4. Nhiều đến Nhiều:

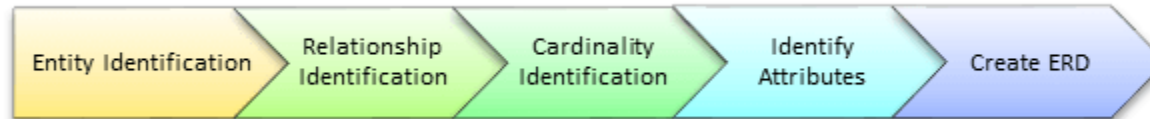
Một thực thể từ X có thể được liên kết với nhiều hơn một thực thể từ Y và ngược lại.

Ví dụ: Sinh viên với tư cách là một nhóm được liên kết với nhiều giảng viên và các thành viên trong giảng viên có thể được liên kết với nhiều sinh viên.



# Cách tạo sơ đồ mối quan hệ thực thể (ERD)

Bây giờ trong Hướng dẫn về Sơ đồ ERD này, chúng ta sẽ học cách tạo Sơ đồ ER. Sau đây là các bước để tạo Sơ đồ ER:



## Các bước tạo sơ đồ ER

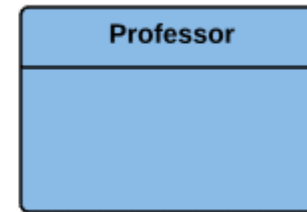
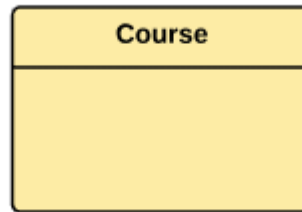
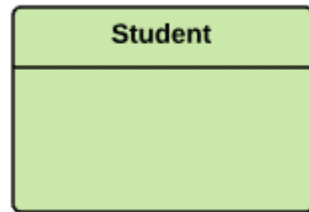
Hãy nghiên cứu chúng với một Ví dụ về Sơ đồ Mối quan hệ Thực thể:

Trong một trường đại học, một Sinh viên đăng ký các Khóa học. Một sinh viên phải được chỉ định cho ít nhất một hoặc nhiều Khóa học. Mỗi khóa học được giảng dạy bởi một Giáo sư duy nhất. Để duy trì chất lượng giảng dạy, một Giáo sư chỉ có thể cung cấp một khóa học

## Bước 1) Nhận dạng thực thể

Chúng tôi có ba thực thể

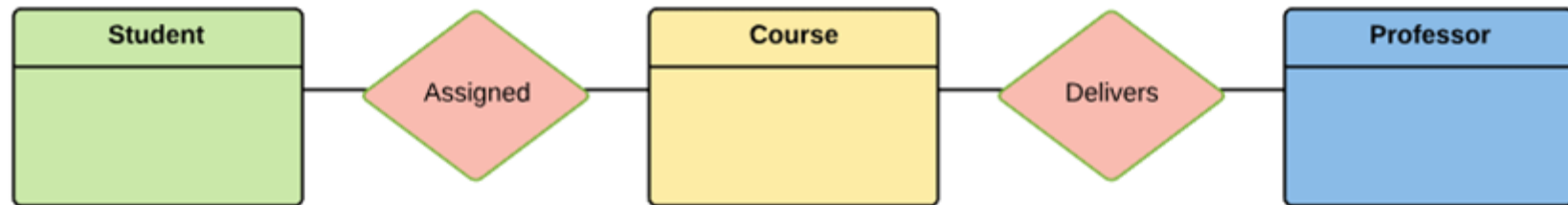
- . Sinh viên
- . Món ăn
- . Giáo sư



## Bước 2) Xác định mối quan hệ

Chúng ta có hai mối quan hệ sau

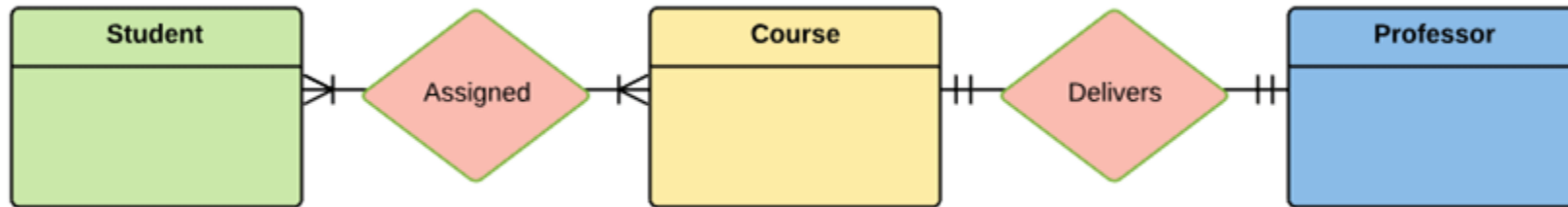
- . Học sinh được **chỉ định** một khóa học
- . Giáo sư **cung cấp** một khóa học



### Bước 3) Nhận dạng Cardinality

Đối với họ báo cáo vấn đề, chúng tôi biết rằng,

- . Một sinh viên có thể được chỉ định **nhiều** khóa học
- . Một giáo sư chỉ có thể cung cấp **một** khóa học





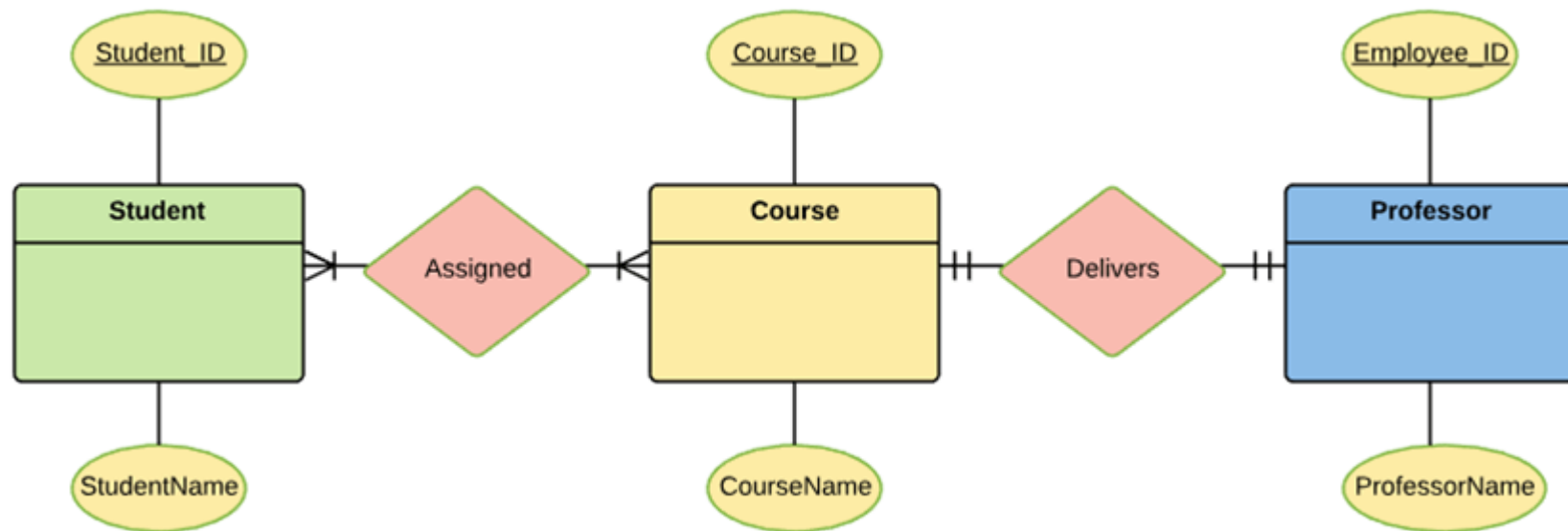
## Bước 4) Xác định các thuộc tính

Bạn cần nghiên cứu các tệp, biểu mẫu, báo cáo, dữ liệu hiện đang được tổ chức duy trì để xác định các thuộc tính. Ban đầu, điều quan trọng là xác định các thuộc tính mà không ánh xạ chúng với một thực thể cụ thể.

Khi bạn đã có danh sách Thuộc tính, bạn cần ánh xạ chúng tới các thực thể đã xác định. Đảm bảo một thuộc tính được ghép nối với chính xác một thực thể. Nếu bạn cho rằng một thuộc tính nên thuộc về nhiều thực thể, hãy sử dụng công cụ sửa đổi để làm cho nó trở thành duy nhất.

Sau khi ánh xạ xong, hãy xác định các Khóa chính. Nếu không có sẵn một khóa duy nhất, hãy tạo một khóa.

Thực thể	Khóa chính	Thuộc tính
Sinh viên	Thẻ học sinh	Tên học sinh
Giáo sư	Mã hiệu công nhân	Giáo sư
Món ăn	Mã khóa học	Tên khóa học



Đối với Thực thể khóa học, các thuộc tính có thể là Thời lượng, Tín chỉ, Bài tập, v.v. Để dễ hiểu, chúng tôi chỉ xem xét một thuộc tính.

## Bước 5) Tạo sơ đồ ERD

Một biểu diễn hiện đại hơn của Ví dụ về Sơ đồ Mối quan hệ Thực thể

