

## Giới thiệu về Rational Rose.

Rational rose là phần mềm công cụ mạnh hỗ trợ phân tích, thiết kế hệ thống phần mềm theo đối tượng. Nó giúp ta mô hình hóa hệ thống trước khi viết mã chương trình. Rational rose hỗ trợ cho việc làm mô hình doanh nghiệp, giúp bạn hiểu được hệ thống của mô hình doanh nghiệp. Giúp chúng ta phân tích hệ thống và làm cho chúng ta có thể thiết kế được mô hình.

Mô hình Rose là bức tranh của hệ thống từ những phối cảnh khác nhau nó bao gồm tất cả các mô hình UML, actors, use cases, objects, component và deployment nodes v.v., trong hệ thống. Nó mô tả chi tiết mà hệ thống bao gồm và nó sẽ làm việc như thế nào vì thế người lập trình có thể dùng mô hình như một bản thiết kế cho công việc xây dựng hệ thống.

Theo phong cách lập trình truyền thống thì sau khi đã xác định yêu cầu hệ thống, người phát triển sẽ lấy một vài yêu cầu, quyết định thiết kế và viết mã chương trình.

Ưu điểm: Cung cấp nhiều tính năng

- + Mô hình hướng đối tượng.
- + Cung cấp cho UML, COM, OMT, và Booch '93.
- + Kiểm tra ngữ nghĩa.
- + Hỗ trợ phát sinh mã cho một số ngôn ngữ.

v.v..

Nhược điểm:

- + Phải cân chỉnh nhiều cho mô hình được đẹp.
- + Trong bản free không hỗ trợ phát sinh mã cho một số ngôn ngữ.
- + Không lùi về những bước trước đã làm.
- + Dung lượng khá nặng.

v.v..

## Hướng dẫn các Diagram.

### Use case diagram

**Use case** là một kỹ thuật được dùng trong kỹ thuật phần mềm và hệ thống để nắm bắt yêu cầu chức năng của hệ thống. Use case mô tả sự tương tác đặc trưng giữa người dùng bên ngoài (actor) và hệ thống. Nó mô tả các yêu cầu đối với hệ thống, có nghĩa là những gì hệ thống phải làm chứ không phải mô tả hệ thống làm như thế nào. Tập hợp tất cả Use case của hệ thống sẽ mô tả tất cả các trường hợp mà hệ thống có thể được sử dụng.(theo wikipedia)

#### I. Use Case Model (mô hình trường hợp sử dụng)

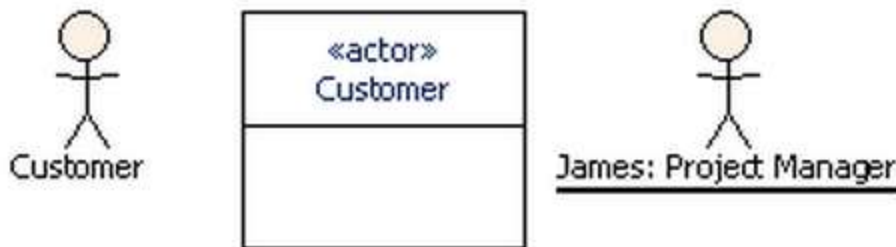
Mô hình Use Case là một kỹ thuật được sử dụng để miêu tả những yêu cầu mang tính chức năng của một hệ thống. Tài nguyên của một mô hình Use Case bao gồm:

- Name (tên) và Description (mô tả): Một Use Case thường có tên như một ngữ-động từ (verb-phrase) giúp người đọc có một thông tin mô tả nhanh về một chức năng của hệ thống.
- Use Case Diagram (lược đồ Use Case).
- Use Case Narrative (tường thuật Use Case): mô tả chi tiết Requirements (Yêu cầu), Constraints (Ràng buộc).
- Use Case Scenarios (kịch bản Use Case).
- Additional information (Thông tin thêm) .

## II. Các thành phần trong bản vẽ Use Case:

### 1) Actors (tác nhân):

Các thực thể bên ngoài này được tham chiếu đến như các Actor. Actor giữ vai trò người dùng hệ thống, phần cứng hoặc các hệ thống khác bên ngoài. Một Actor thường được vẽ như hình một người (hình trái), hoặc cách khác, giống như hình chữ nhật mô tả lớp với stereotype «actor» (hình giữa). Một thực thể Actor (Actor Instance) chỉ một Actor cụ thể của lớp Actor đó (hình phải).



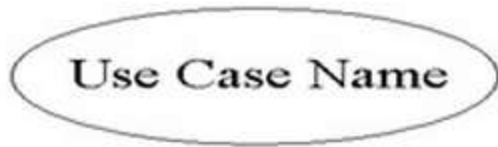
Actor này có thể sinh ra Actor khác (Actor Generalization) như trong diagram sau:



Trong rational rose để vẽ 1 actor ta chọn biểu tượng “actor” bên thanh công cụ .

### 2) Use case

Một Use Case là một đơn vị công việc riêng lẻ. Nó cung cấp một cái nhìn cấp cao của người ngoài hệ thống về một hành vi có thể nhận thấy được của hệ thống. Hay hiểu cách khác Use Case là chức năng mà các Actor sẽ sử dụng. Ký hiệu mô tả một Use Case là một hình ellipse.



Với việc xác định các chức năng mà Actor sử dụng bạn sẽ xác định được các Use Case cần có trong hệ thống.

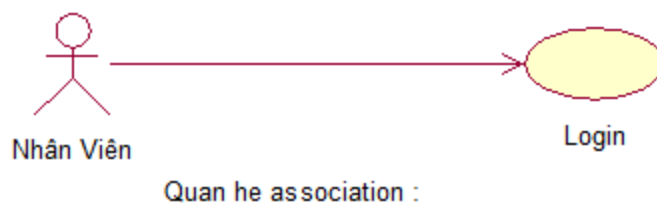
Trong rational rose để vẽ 1 use case ta chọn biểu tượng hình ellipse bên thanh công cụ

### 3) Relationship(Quan hệ)

Relationship hay còn gọi là connector được sử dụng để kết nối giữa các đối tượng với nhau tạo nên bản vẽ Use Case. Có các kiểu quan hệ cơ bản sau:

- Association
- Generalization
- Include
- Extend

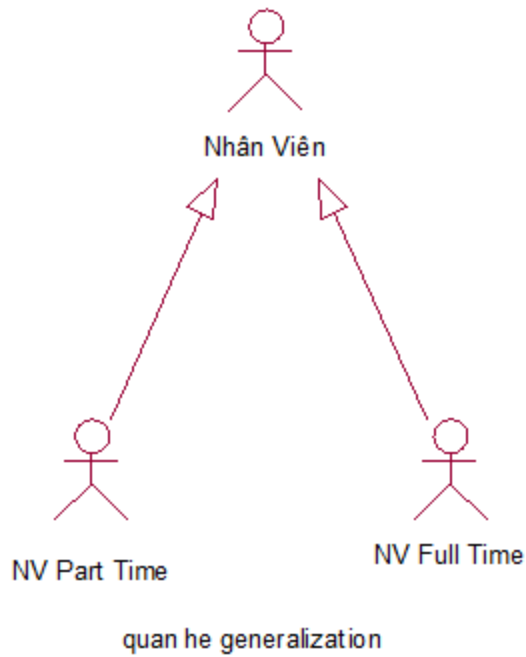
#### Quan hệ association :



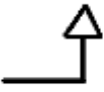
Association thường được dùng để mô tả mối quan hệ giữa **Actor** và **Use Case** và giữa các Use Case với nhau. Trong rational rose quan hệ association được thể hiện qua kí

hiệu  ( Unidirectional association)

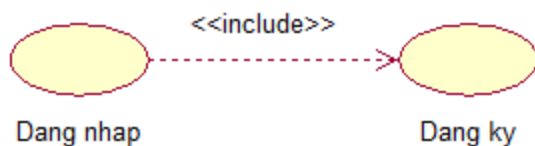
#### Quan hệ Generalization (Tổng quát hóa):



Generalization được sử dụng để thể hiện quan hệ thừa kế giữa các Actor hoặc giữa các Use Case với nhau. Trong rational rose quan hệ generalization được thể hiện qua kí

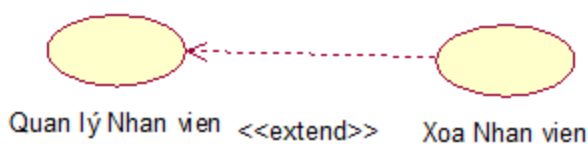
hiệu  ( Generalization).


### Quan hệ Include:



Include là quan hệ giữa các Use Case với nhau. A include B thì B là pre-condition của A. Bắt buộc phải thực hiện use case B trước sau đó use case A mới được thực hiện.

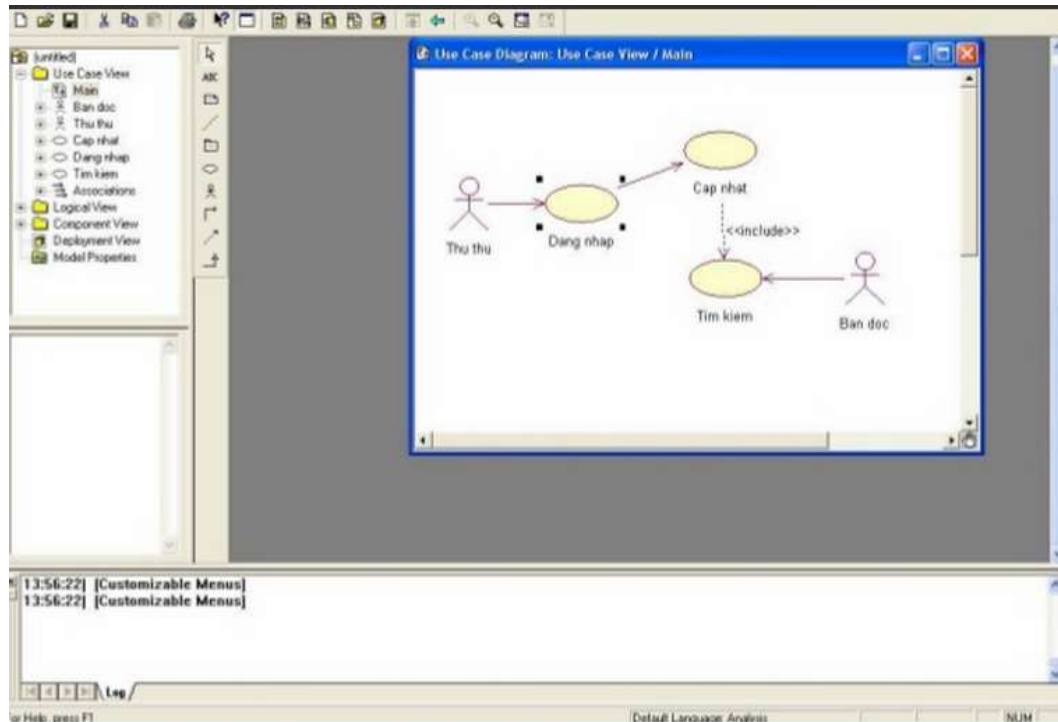
### Quan hệ Extend:



Extend là quan hệ giữa các Use Case với nhau. B extend A thì B là mở rộng của A. Trong rational rose quan hệ generalization được thể hiện qua kí hiệu( Dependency or instantiates) 

Ngoài ra trong thanh công cụ còn có các công cụ phục vụ cho việc vẽ use case diagram

- Selection tool
- Textbox
- Note
- Anchor Note to item
- Package



## Giao diện của biểu đồ use case

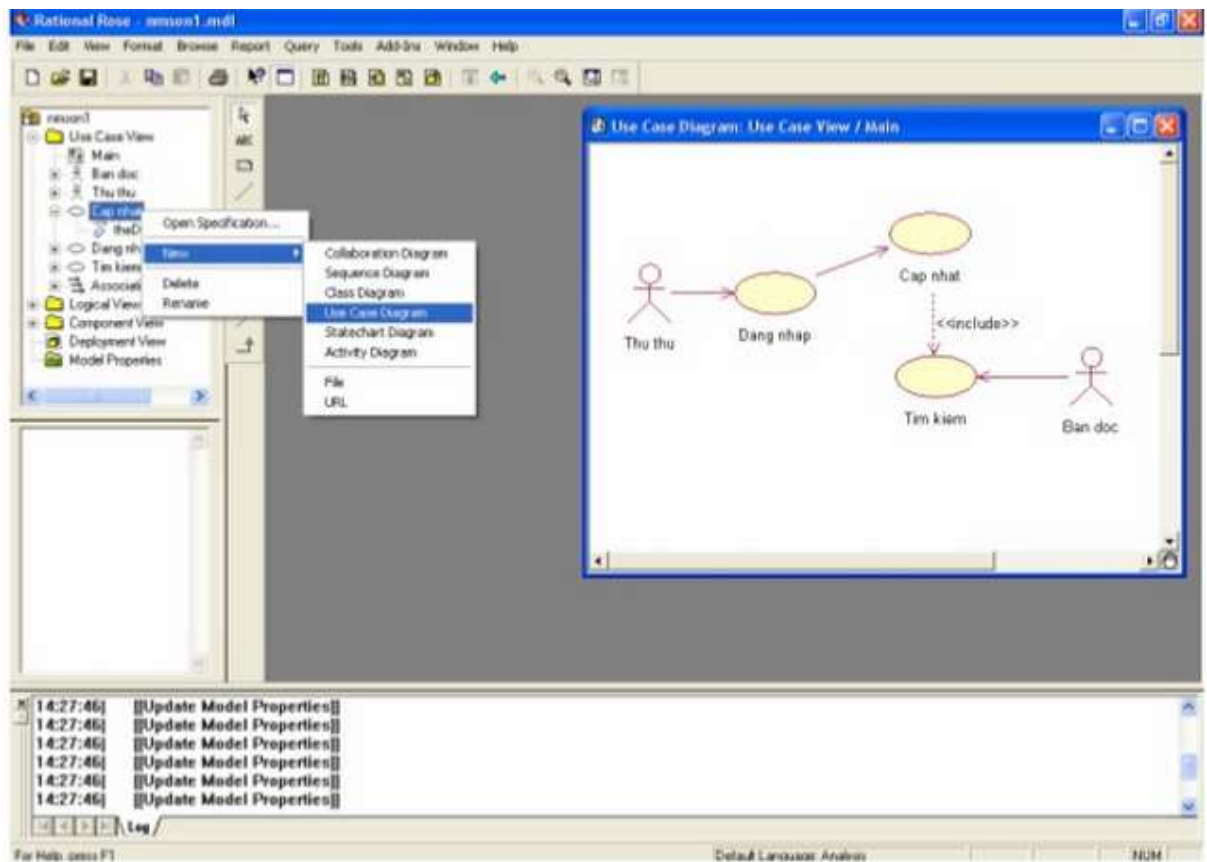


- **Tab General:** thông tin chung về use case như tên, kiểu...
- **Tab Diagram:** cho biết các biểu đồ đi kèm của use case.
- **Tab Relations:** liệt kê các mối quan hệ của use case với các use case và actor khác.
- **Tab Files:** là các file kèm theo use case.

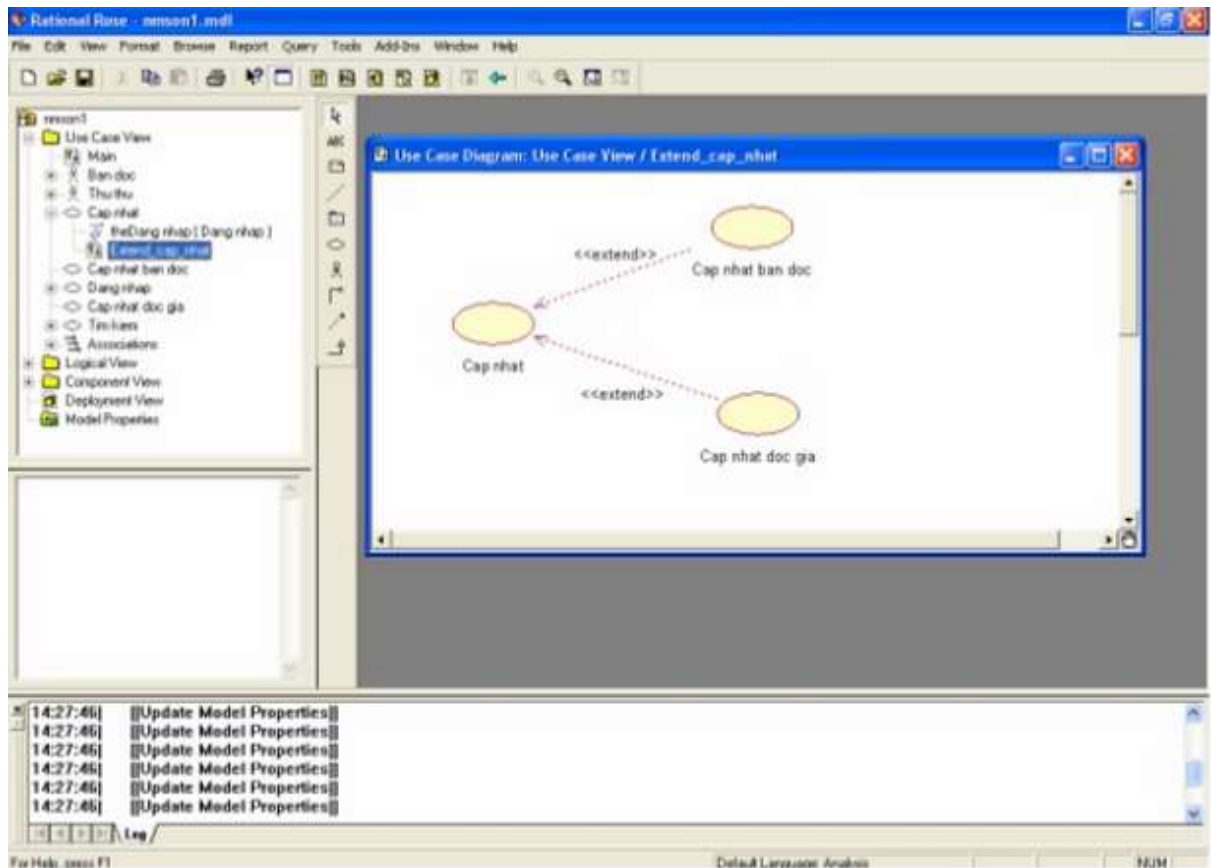
Đặc tả quan hệ  
Dependency

Đặc tả quan hệ association

Đặc tả quan hệ association



Phân rã use case



sơ đồ use case mức 2

### III. Các giai đoạn xây dựng một Use Case Diagram

Giai đoạn mô hình hóa:

- **Bước 1:** Thiết lập ngữ cảnh của hệ thống đích.
- **Bước 2:** Chỉ định các Actor.
- **Bước 3:** Chỉ định các Use Case.
- **Bước 4:** Định nghĩa các quan hệ giữa các Actor và các Use Case.
- **Bước 5:** Đánh giá các Actor và các Use Case để tìm cách chi tiết hóa.

Giai đoạn cấu trúc:

- **Bước 6:** Đánh giá các Use Case cho quan hệ phụ thuộc «include».
- **Bước 7:** Đánh giá các Use Case cho quan hệ phụ thuộc «extend».
- **Bước 8:** Đánh giá các Use Case cho quan hệ generalizations.

Giai đoạn review:

Sau khi định nghĩa Use Case, cần tiến hành thử nghiệm Use Case

## Activity Diagram

### I. Biểu đồ hoạt động (Activity diagram) do Odell đề xuất cho UML để

- mô tả luồng công việc trong tiến trình nghiệp vụ trong mô hình hóa nghiệp vụ
- mô tả luồng sự kiện trong mô hình hóa hệ thống

- Sử dụng text như trước đây sẽ khó đọc khi logic phức tạp, có nhiều rẽ nhánh
- Biểu đồ hoạt động sử dụng để mô hình hóa
  - khía cạnh động của hệ thống
  - các bước trình tự hay tương tranh trong quá trình tính toán

Activity Diagram là bản vẽ tập trung vào mô tả các hoạt động, luồng xử lý bên trong hệ thống. Nó có thể được sử dụng để mô tả các qui trình nghiệp vụ trong hệ thống, các luồng của một chức năng hoặc các hoạt động của một đối tượng.

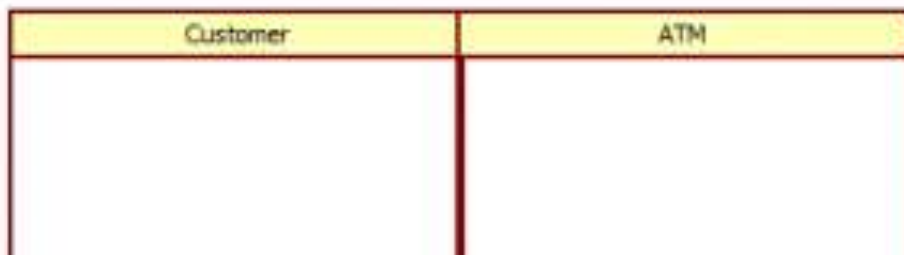
Các thành phần của mô hình động :

- **Event:** là sự kiện, mô tả một hoạt động bên ngoài tác động vào đối tượng và được đối tượng nhận biết và có phản ứng lại.
- **Activity:** mô tả một hoạt động trong hệ thống. Hoạt động có thể do một hoặc nhiều đối tượng thực hiện.
- **State:** là trạng thái của một đối tượng trong hệ thống, được mô tả bằng giá trị của một hoặc nhiều thuộc tính.
- **Action:** chỉ hành động của đối tượng.
- **Condition:** mô tả một điều kiện.

## II. Các thành phần trong bản vẽ Activity Diagram:

### 1) Swimlane

Swimlane được dùng để xác định đối tượng nào tham gia hoạt động nào trong một qui trình



Trong rational rose Swimlane được thể hiện qua kí hiệu

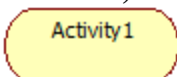


### 2) Nút End ,Start

Start thể hiện điểm bắt đầu qui trình, End thể hiện điểm kết thúc qui trình.

- Kí hiệu nút Start: ●
- Kí hiệu nút End: ⊙

### 3) Activity



Activity mô tả một hoạt động trong hệ thống. Các hoạt động này do các đối tượng thực hiện.

Trong rational rose Activity được thể hiện qua kí hiệu (Activity)





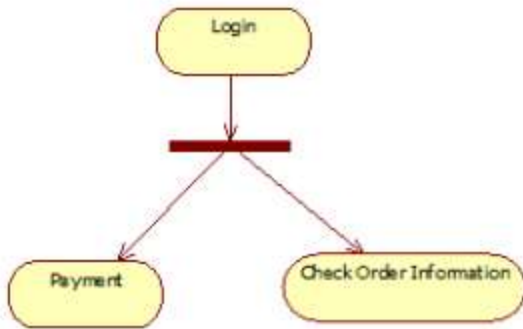
#### 4) Branch

Branch thể hiện rẽ nhánh trong mệnh đề điều kiện.



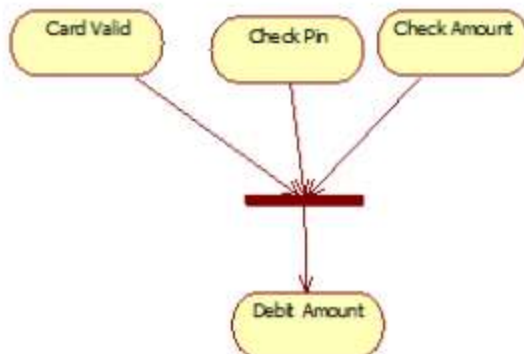
#### 5) Fork

Fork thể hiện cho trường hợp thực hiện xong một hoạt động rồi sẽ rẽ nhánh thực hiện nhiều hoạt động tiếp theo.



#### 6) Join

Cùng ký hiệu với Fork nhưng thể hiện trường hợp phải thực hiện hai hay nhiều hành động trước rồi mới thực hiện hành động tiếp theo.



### III) Cách xây dựng Activity Diagram

Thực hiện các bước sau đây để xây dựng bản vẽ Activity Diagram.

#### Bước 1: Xác định các nghiệp vụ cần mô tả

Xem xét bản vẽ Use Case để xác định nghiệp vụ nào bạn cần mô tả.

#### Bước 2: Xác định trạng thái đầu tiên và trạng thái kết thúc

#### Bước 3: Xác định các hoạt động tiếp theo

Xuất phát từ điểm bắt đầu, phân tích để xác định các hoạt động tiếp theo cho đến khi gặp điểm kết thúc để hoàn tất bản vẽ này.

Bạn có thể hỏi chuyên gia, học hệ thống tương tự, hỏi khách hàng để nắm rõ về qui trình của hệ thống.

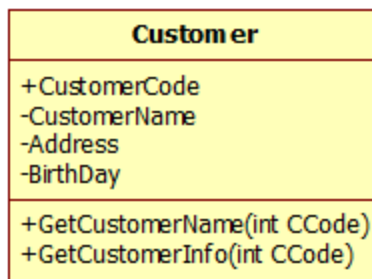
## Class Diagram

Class là thành phần chính của bản vẽ Class Diagram. Class mô tả về một nhóm đối tượng có cùng tính chất, hành động trong hệ thống. Ví dụ mô tả về khách hàng chúng ta dùng lớp “Customer”. Class được mô tả gồm tên Class, thuộc tính và phương thức

<b>Class Name</b>
<b>Attributes</b>
<b>Methods</b>

Trong đó,

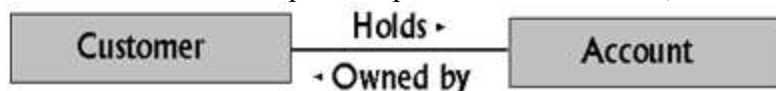
- Class Name: là tên của lớp.
- Attributes (thuộc tính): mô tả tính chất của các đối tượng. Ví dụ như khách hàng có Mã khách hàng, Tên khách hàng, Địa chỉ, Ngày sinh v.v...
- Method (Phương thức): chỉ các hành động mà đối tượng này có thể thực hiện trong hệ thống. Nó thể hiện hành vi của các đối tượng do lớp này tạo ra.



## Relationship

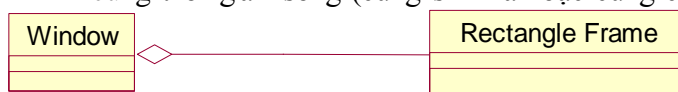
Relationship thể hiện mối quan hệ giữa các Class với nhau.

- Association
  - Aggregation
  - Composition
  - Generalization
  - Dependency
  - Multiplicity
- Association là quan hệ giữa hai lớp với nhau, thể hiện chúng có liên quan với nhau. Association thể hiện qua các quan hệ như “has: có”, “Own: sở hữu” v.v...



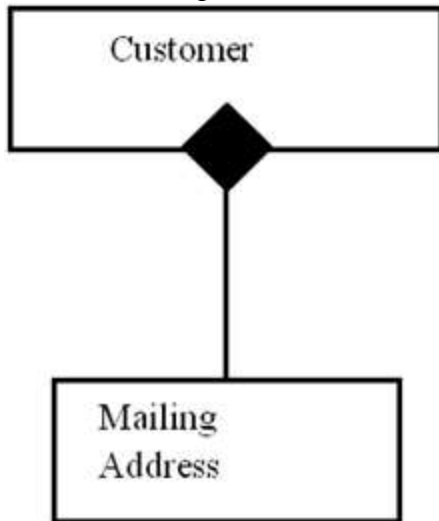
Ví dụ quan hệ trên thể hiện Khách hàng nắm giữ Tài khoản và Tài khoản được sở hữu bởi Khách hàng.

- Aggregation là một loại của quan hệ Association nhưng mạnh hơn. Nó có thể cùng thời gian sống (cùng sinh ra hoặc cùng chết đi)



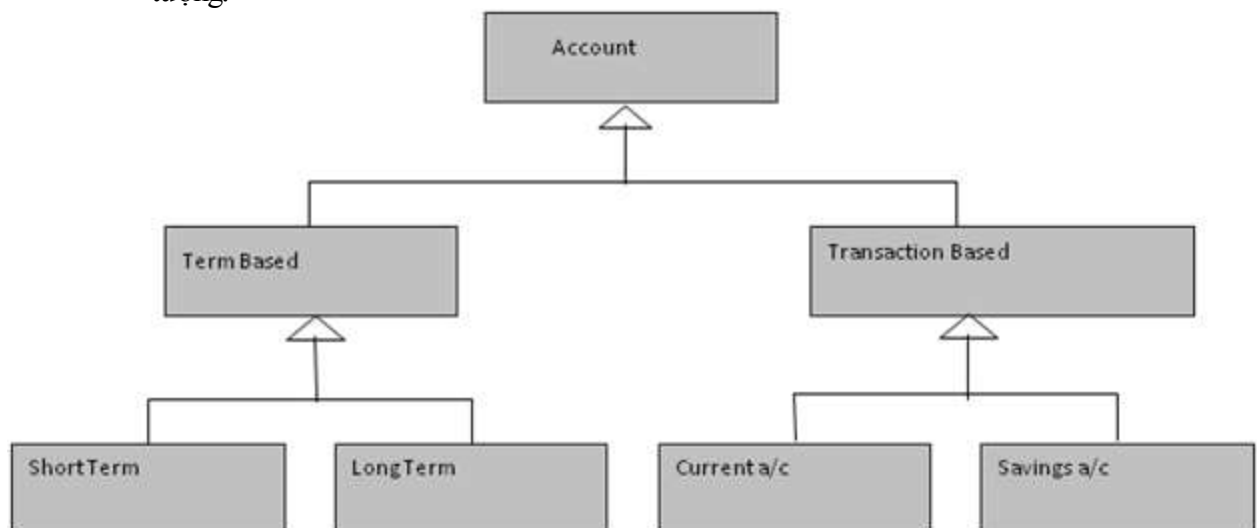
Ví dụ quan hệ trên thể hiện lớp Window(cửa sổ) được lắp trên Khung cửa hình chữ nhật. Nó có thể cùng sinh ra cùng lúc.

- Composition là một loại mạnh hơn của Aggregation thể hiện quan hệ class này là một phần của class kia nên dẫn đến cùng tạo ra hoặc cùng chết đi.



Ví dụ trên class Mailing Address là một phần của class Customer nên chỉ khi nào có đối tượng Customer thì mới phát sinh đối tượng Mailing Address.

- Generalization là quan hệ thừa kế được sử dụng rộng rãi trong lập trình hướng đối tượng.



Các lớp ở cuối cùng như Short Term, Long Term, Curent a/c, Savings a/c gọi là các lớp cụ thể (concrete Class). Chúng có thể tạo ra đối tượng và các đối tượng này thừa kế toàn bộ các thuộc tính, phương thức của các lớp trên.

Các lớp trên như Account, Term Based, Transaction Based là những lớp trừu tượng (Abstract Class), những lớp này không tạo ra đối tượng.

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.

Bước 1: Tìm các Classes 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.

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

- Dependency là nhấn mạnh rằng sự tồn tại của lớp đối tượng này phụ thuộc hoàn toàn vào lớp đối tượng kia. Giữa hai lớp đối tượng phải có sự tham chiếu.



Ví dụ trên: Class A “phụ thuộc” Class B

Client → Supplier (phần tử phục thuộc → phần tử độc lập)

- Multiplicity xác định số lượng đối tượng của một lớp có quan hệ với một đối tượng của lớp khác. Thể hiện chặn trên và chặn dưới số lượng đối tượng, cho biết mỗi kết hợp có bắt buộc hay không.

Bảng ý nghĩa:

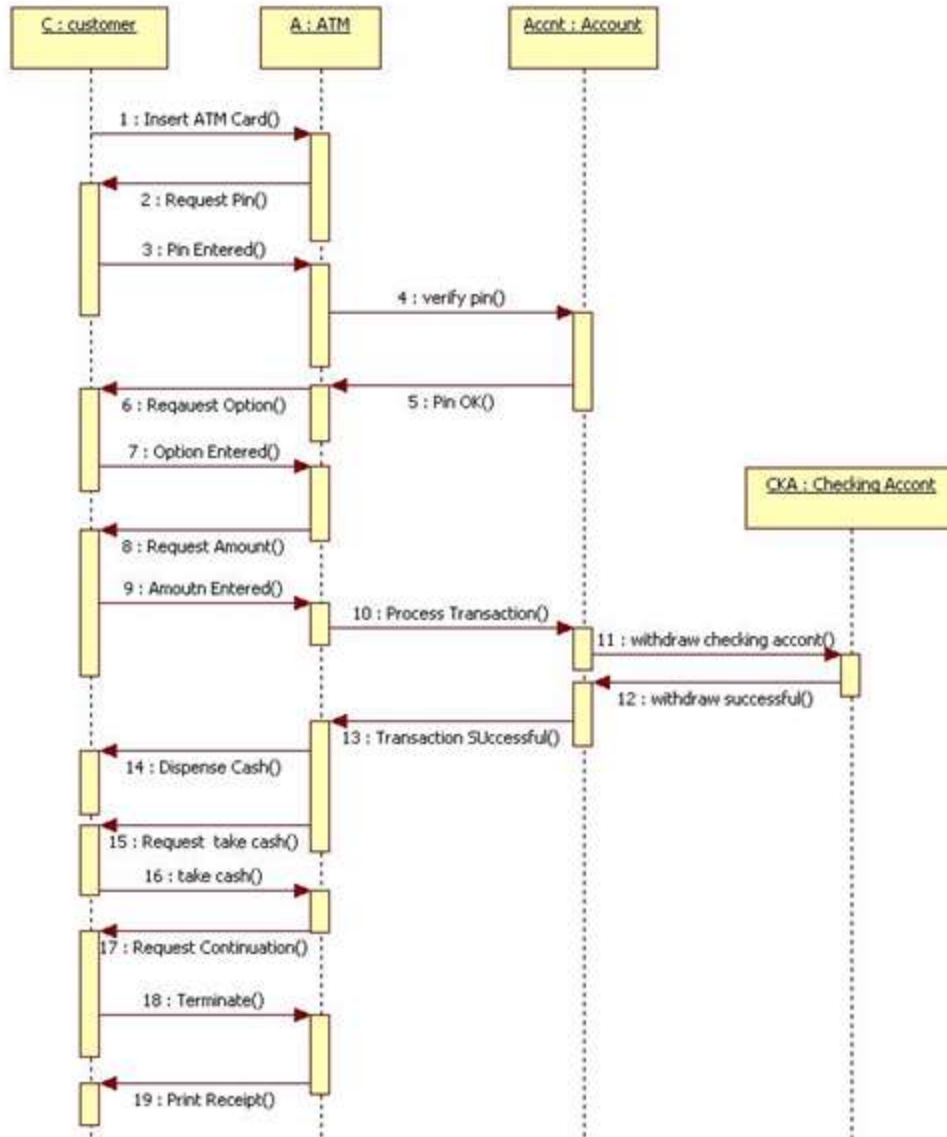
Multiplicity	Ý nghĩa
n (Mặc định)	Nhiều
0..0	Không
0..1	Không hoặc 1
0..n	Không hoặc nhiều
1..1	Chính xác 1
1..n	Một hoặc nhiều

## SEQUENCE DIAGRAM

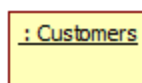
- Biểu đồ tuần tự minh họa các đối tượng tương tác với nhau ra sao.
- Chúng tập trung vào các chuỗi thông điệp, có nghĩa là các thông điệp được gửi và nhận giữa một loạt các đối tượng như thế nào.
- Biểu đồ tuần tự có hai trục: trục nằm dọc chỉ thời gian, trục nằm ngang chỉ ra một tập hợp các đối tượng. Một biểu đồ tuần tự cũng nêu bật sự tương tác trong một cảnh kịch (scenario) – một sự tương tác sẽ xảy ra tại một thời điểm nào đó trong quá trình thực thi của hệ thống.

Từ các hình chữ nhật biểu diễn đối tượng có các đường gạch rời (dashed line) thẳng đứng biểu thị đường đời đối tượng, tức là sự tồn tại của đối tượng trong chuỗi tương tác. Trong khoảng thời gian này, đối tượng được thực thể hóa, sẵn sàng để gửi và nhận thông điệp. Quá trình giao tiếp giữa các đối tượng được thể hiện bằng các đường thẳng thông điệp nằm ngang nối các đường đời đối tượng. Mỗi tên ở đầu đường thẳng sẽ chỉ ra loại thông điệp này mang tính đồng bộ, không đồng bộ hay đơn giản. Để đọc biểu đồ tuần tự, hãy bắt đầu từ phía bên trên của biểu đồ rồi chạy dọc xuống và quan sát sự trao đổi thông điệp giữa các đối tượng xảy ra dọc theo tiến trình thời gian.

## CẤU TRÚC

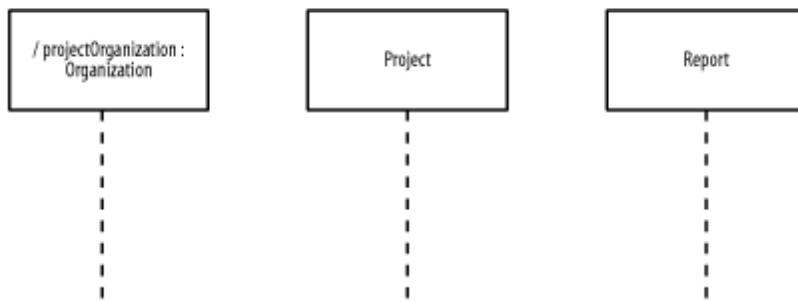


- Một lược đồ tuần tự bao gồm 4 thành tố chính
  - Object: các đối tượng khác nhau liên quan đến lược đồ Object mô tả một đối tượng trong hệ thống. Để phân biệt với Class, Object có dấu ":" phía trước tên của nó



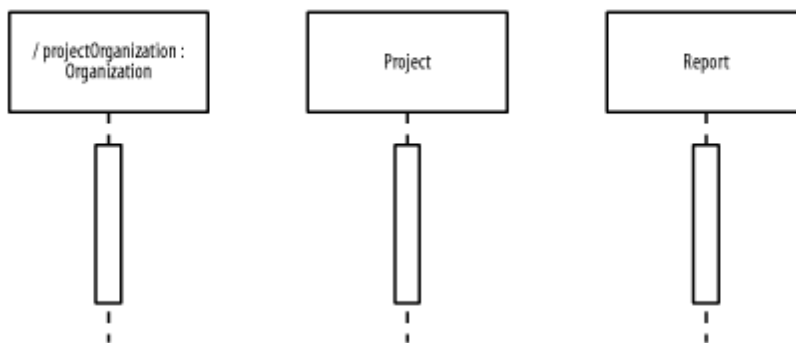
- Lifeline: thể hiện sự tồn tại của đối tượng trên trục thời gian

**Figure 6-10. Lifelines**



- Activation – focus of control: thể hiện bằng hình chữ nhật nằm trên lifeline. Độ dài của focus of control cho biết thời gian mà đối tượng tồn tại.

**Figure 6-11. Activations**



- Message: thể hiện sự liên lạc giữa các đối tượng



## COLLABORATION DIAGRAM

Biểu diễn mối quan hệ tương tác giữa:  
Các đối tượng

Đối tượng và tác nhân

Nhấn mạnh đến vai trò của các đối tượng trong tương tác

Biểu diễn các hoạt động như các luồng công việc hoặc tiến trình khác nhau trong hệ thống được xây dựng

### ĐẶC ĐIỂM

Đối tượng được đặt một cách tự do trong không gian

Không có đường lifeline cho mỗi đối tượng

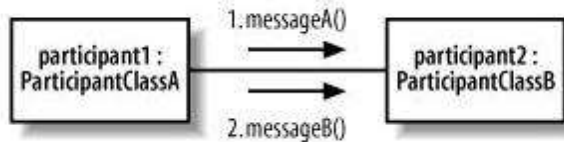
Các message được đánh số theo thứ tự thời gian

### CÁC THÀNH PHẦN

Đối tượng (participants): hình chữ nhật, luôn xuất hiện tại vị trí xác định, tên đối tượng:tên lớp

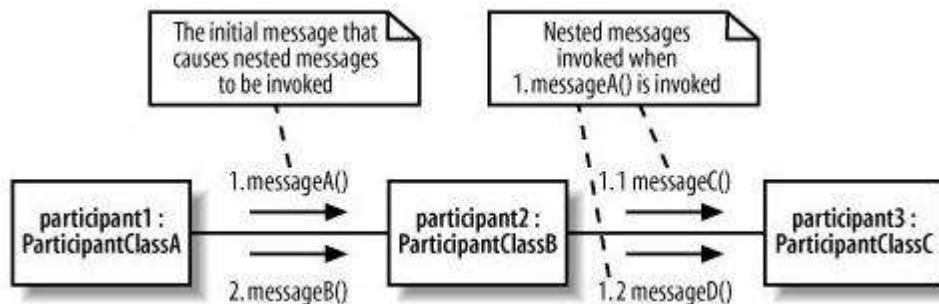
Liên kết (links): đường nối 2 đối tượng có tương tác, không có chiều

Thông điệp (messages): mũi tên từ đ.tượng gửi sang đ.tượng nhận, đánh số theo thứ tự xuất hiện. Các thông điệp thường đính kèm theo các label (nhãn) để chỉ ra thứ tự các thông điệp được gửi đi, điều kiện, giá trị trả về

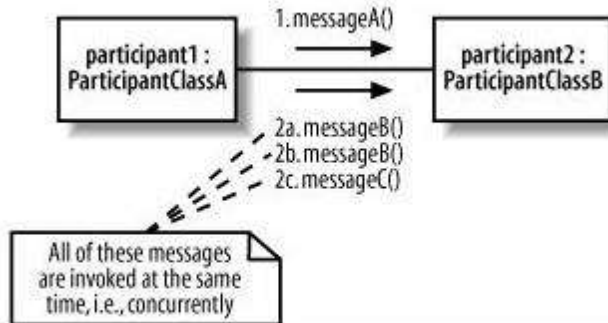


### CÁC LOẠI THÔNG ĐIỆP (MESSAGE)

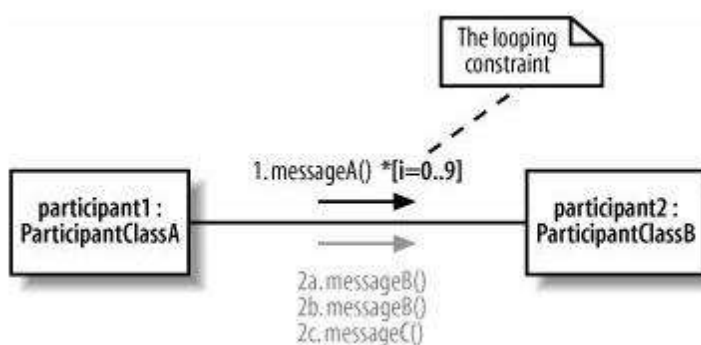
Thông điệp lồng nhau (nested messages): là thông điệp được gọi (invoke) bởi đối tượng theo yêu cầu của 1 thông điệp trước đó



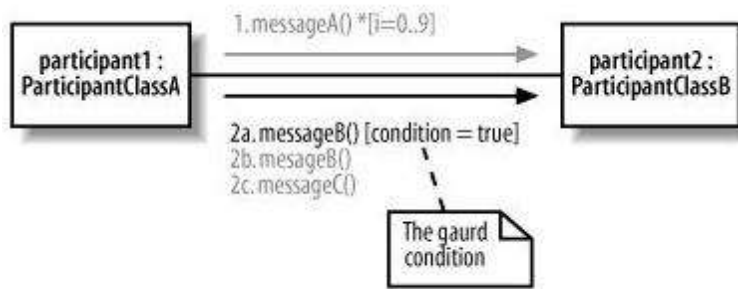
Thông điệp xảy ra đồng thời: ký hiệu bằng cách thêm ký tự (a,b,c,...) sau số chỉ thứ tự thực hiện của nó



Gọi một thông điệp nhiều lần (vd: thông điệp được gọi bởi vòng for)



Thông điệp chỉ được thực hiện khi thỏa 1 điều kiện cụ thể



- Nếu nhấn mạnh yếu tố thời gian hay trình tự -> chọn sequence
- Nếu nhấn mạnh đối tượng, quan hệ và các thông điệp tương tác giữa chúng -> chọn collaboration
- Ưu điểm của collaboration: chỉ ra chi tiết về các lệnh gọi hàm

-----o0o-----

Tài liệu tham khảo: <http://fit.mta.edu.vn/files/DanhSach/Huong-Dan-Su-Dung-Rational-Rose-148.pdf>