

PHÂN TÍCH THIẾT KẾ HƯỚNG ĐỐI TƯỢNG

Chương 5

Ngôn ngữ mô hình UML



Fundamental



Nội dung

- 1 Giới thiệu tổng quan về UML**
- 2 Một số Case tool hỗ trợ UML**
- 3 Một số biểu đồ UML cơ bản**
- 4 Cài đặt (ánh xạ) biểu đồ**
- 5 Giới thiệu Visual Paradigm**



1. Tổng quan về UML

- ❖ UML (Unified Model Language) là một ngôn ngữ dùng cho phân tích thiết kế hướng đối tượng (OOAD – Object Oriented Analysis and Design)
- ❖ Được duy trì và phát triển bởi OMG (Object Management Group), do **Jacobson, Booch, Rumbaugh** sáng lập. Ngoài ra còn có hàng trăm các tập đoàn lớn khác bảo trợ phát triển.
- ❖ UML 2.0 có 13 loại biểu đồ để thể hiện các khung nhìn khác nhau (View) về hệ thống.
- ❖ Các biểu đồ UML cho ta cái nhìn rõ hơn về hệ thống (cả cái nhìn tĩnh và động)

1. Tổng quan về UML....



- ❖ Hiện nay UML được sử dụng rất phổ biến trong các dự án phần mềm.
- ❖ UML thể hiện phương pháp phân tích hướng đối tượng nên không lệ thuộc ngôn ngữ LT.
- ❖ Có rất nhiều công cụ phần mềm hỗ trợ phân tích thiết kế dùng UML.
- ❖ Nhiều công cụ có thể sinh ra mã từ UML và ngược lại (từ mã thành UML-Reverse Eng)
- ❖ UML không phải là ngôn ngữ lập trình !.
- ❖ Phiên bản mới nhất của UML là 2.1.2 (omg.org)



UML dùng để làm gì ?

- ❖ UML là một ngôn ngữ dùng để:
 1. Trực quan hóa (Visualizing)
 2. Đặc tả (Specifying)
 3. Xây dựng (Constructing)
 4. Viết tài liệu (Documenting)



Trực quan hóa-Visualizing

- ❖ Dùng tập các ký hiệu đồ họa phong phú để biểu diễn hệ thống đang được nghiên cứu.
- ❖ Hệ thống ký hiệu đều có ngũ nghĩa chặt chẽ, có thể hiểu bởi nhiều công cụ khác nhau.
- ❖ Giúp cho các nhà thiết kế, nhà lập trình khác biệt về ngôn ngữ đều có thể hiểu được.



UML là ngôn ngữ cho đặc tả - specifying

- ❖ UML giúp xây dựng các mô hình chính xác, đầy đủ và không nhập nhằng.
- ❖ Tất cả các công đoạn từ phân tích, thiết kế cho đến triển khai đều có các biểu đồ UML biểu diễn.
- ❖ Use case (dùng cho phân tích); Class, Sequence, Activity... (cho thiết kế); Component, Deployment (cho triển khai).



Xây dựng - Constructing

- ❖ Các mô hình của UML có thể kết nối với nhiều ngôn ngữ lập trình. Tức là có thể ánh xạ các mô hình UML về một ngôn ngữ lập trình như C++, Java...
- ❖ Việc chuyển các mô hình trong UML thành Code trong ngôn ngữ lập trình → Forward engineering
- ❖ Việc chuyển ngược trở lại code trong một ngôn ngữ lập trình thành UML → Reverse Engineering.
- ❖ Cần công cụ để chuyển đổi “xuôi” & “ngược”



UML là ngôn ngữ giúp viết tài liệu

- ❖ Giúp xây dựng tài liệu đặc tả - requirements
- ❖ Tài liệu kiến trúc (architecture)
- ❖ Tài liệu thiết kế
- ❖ Source code
- ❖ Tài liệu để kiểm thử - Test
- ❖ Tài liệu mẫu - Prototype
- ❖ Tài liệu triển khai – Deployment



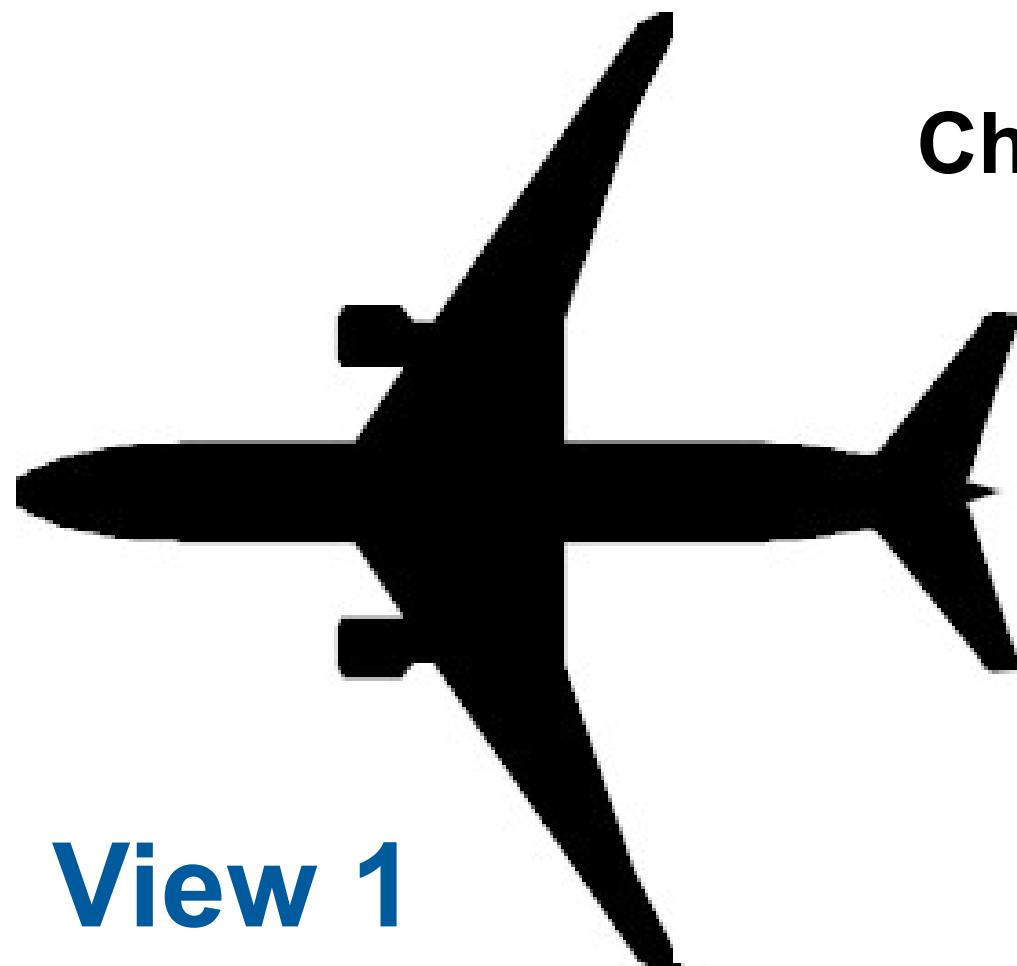
2. Một số Case tool (Công cụ) hỗ trợ UML

- ❖ Rational Rose (của hãng Rational) <http://www-128.ibm.com/developerworks/downloads/r/rsd/?>
- ❖ Visual Paradiagm <http://www.visual-paradigm.com>
- ❖ Microsoft Visio www.microsoft.com
- ❖ Power designer <http://www.sybase.com>
- ❖ Visual Case <http://www.visualcase.com>
- ❖ Pacestar UML Diagrammer www.peacestar.com
- ❖



3. Một số biểu đồ UML cơ bản

Chiều cao ? ? ?



View 1



3. Một số biểu đồ UML cơ bản

Chiều cao phía sau ?





3. Một số biểu đồ UML cơ bản



View 3



3. Một số biểu đồ UML cơ bản

1

- ❖ Biểu đồ ca sử dụng
Use Case
Diagram



3. Một số biểu đồ UML cơ bản



1

❖ Biểu đồ ca
sử dụng
**Use Case
Diagram**

2

Class
Diagram

- Mô tả các chức năng của hệ thống dựa trên quan điểm người sử dụng.
- Mô tả sự tương tác giữa người dùng và hệ thống.
- Cho biết hệ thống được sử dụng như thế nào ?

Component

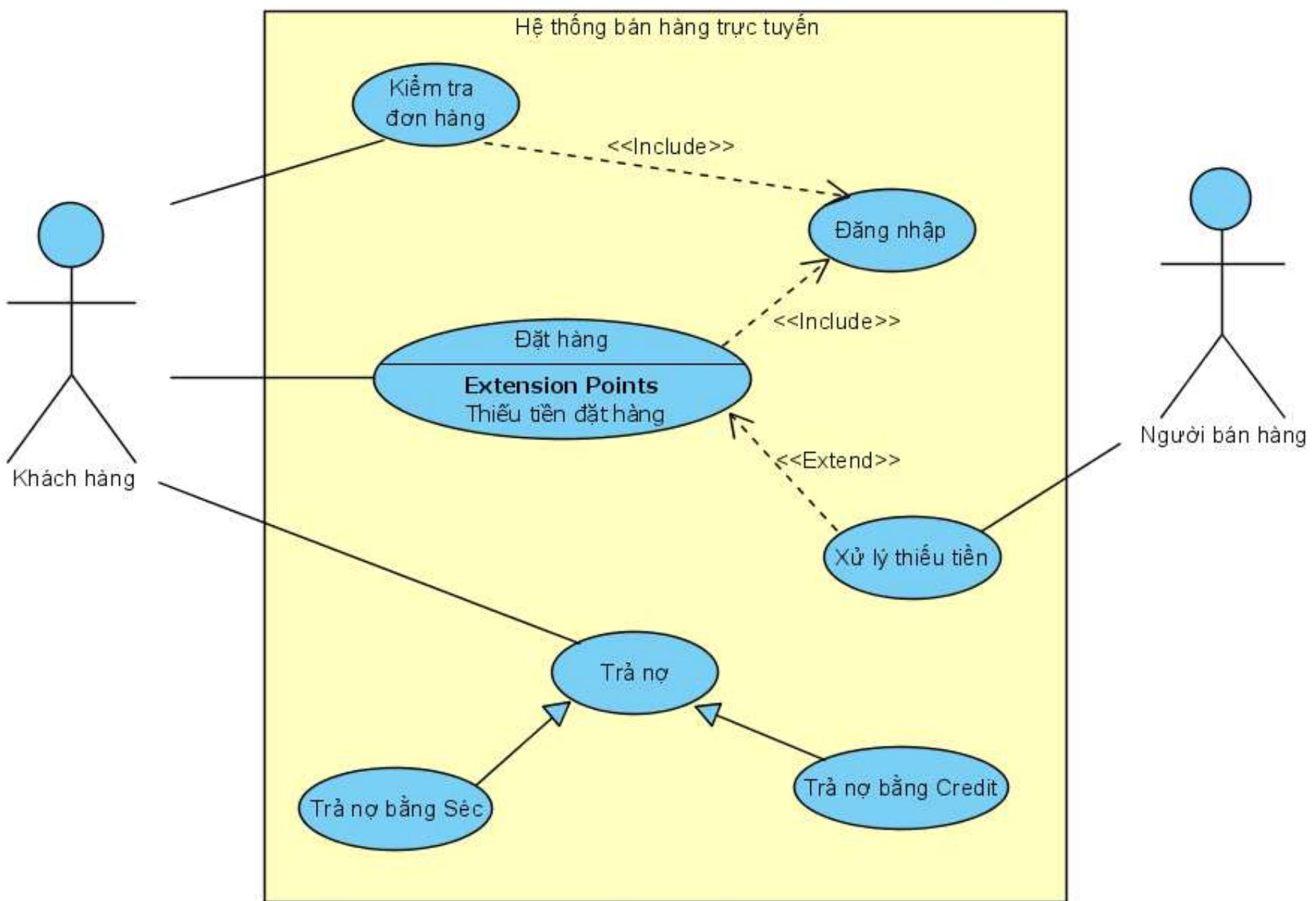
Deployment

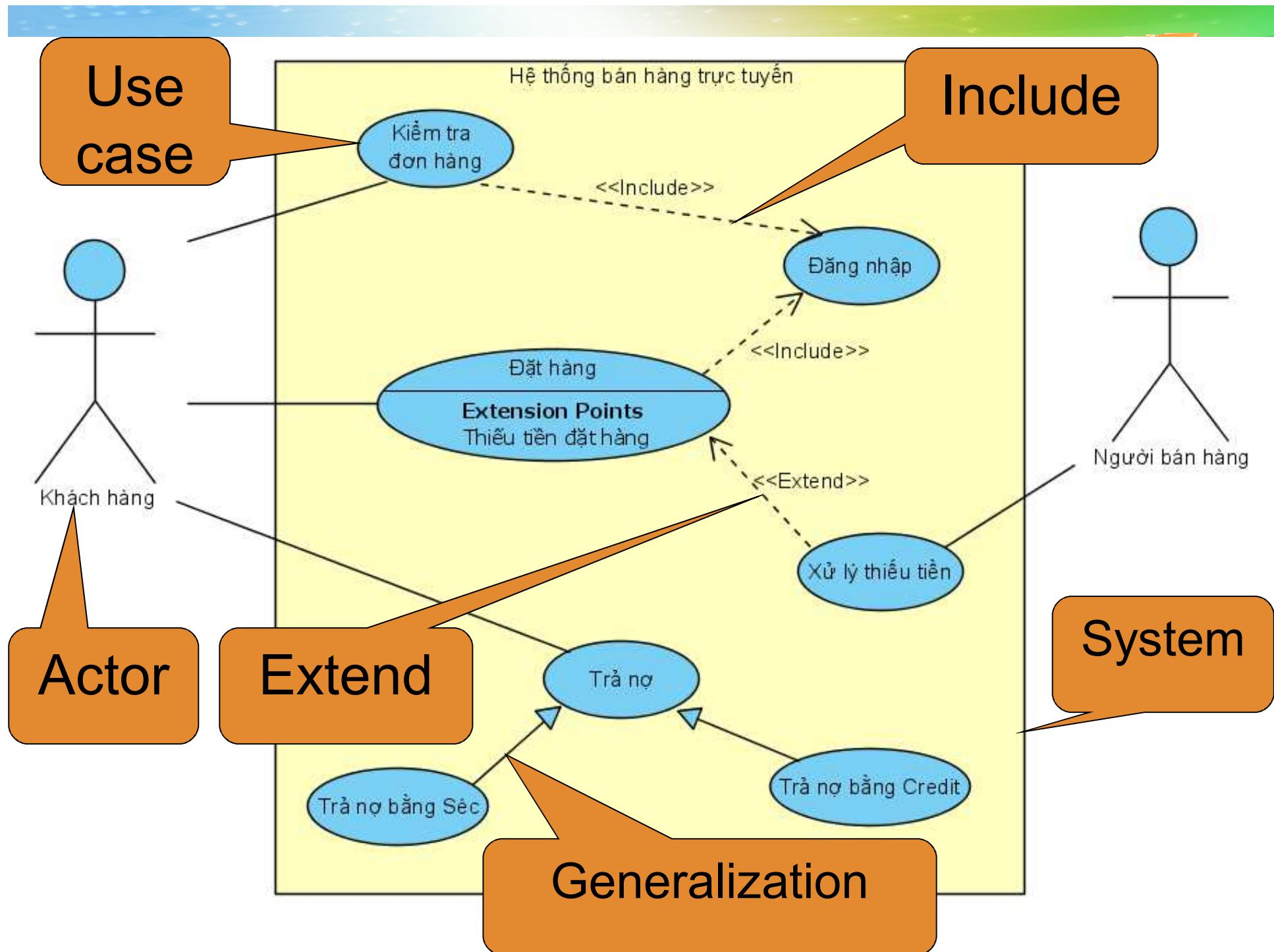
Communication

Collaboration

Timing

State





Chú ý:



- ❖ Khi nào thì vẽ quan hệ <Include> (bao hàm)
→ Use case A được gọi là Include B nếu trong xử lý của A có gọi đến B ít nhất 1 lần !

- ❖ Minh họa thông qua Code

```
Class B { public void X () { .... } }
```

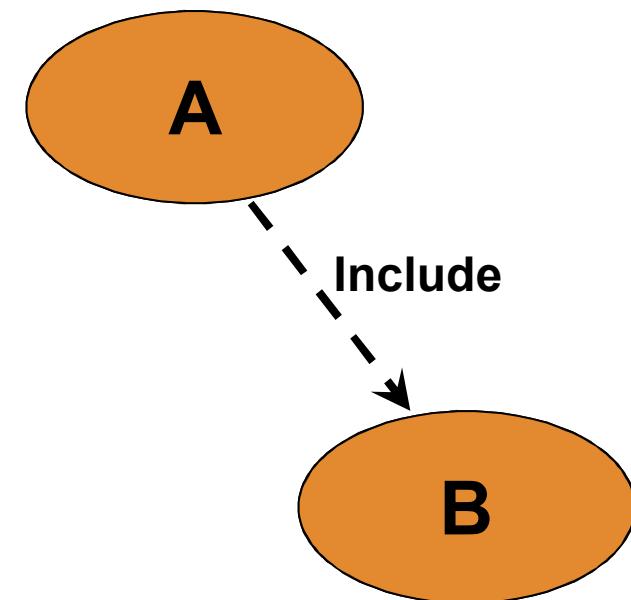
```
Class A {
```

```
    Public void Y () {
```

```
        B objB = new B(); objB.X (); ...
```

```
}
```

```
}
```



Chú ý:

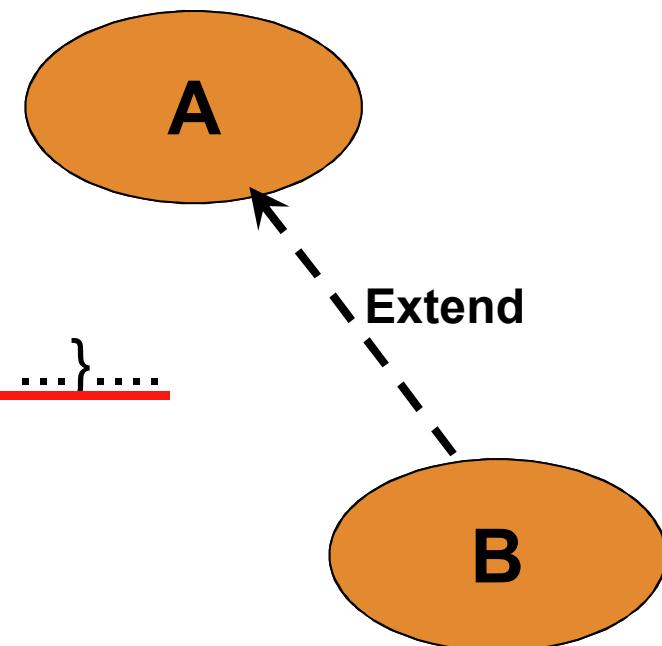


- ❖ Khi nào thì vẽ quan hệ <Extend> (mở rộng)
→ **Use case B được gọi là Exntend A nếu use case B được gọi bởi A nếu thỏa mãn điều kiện nào đó.**

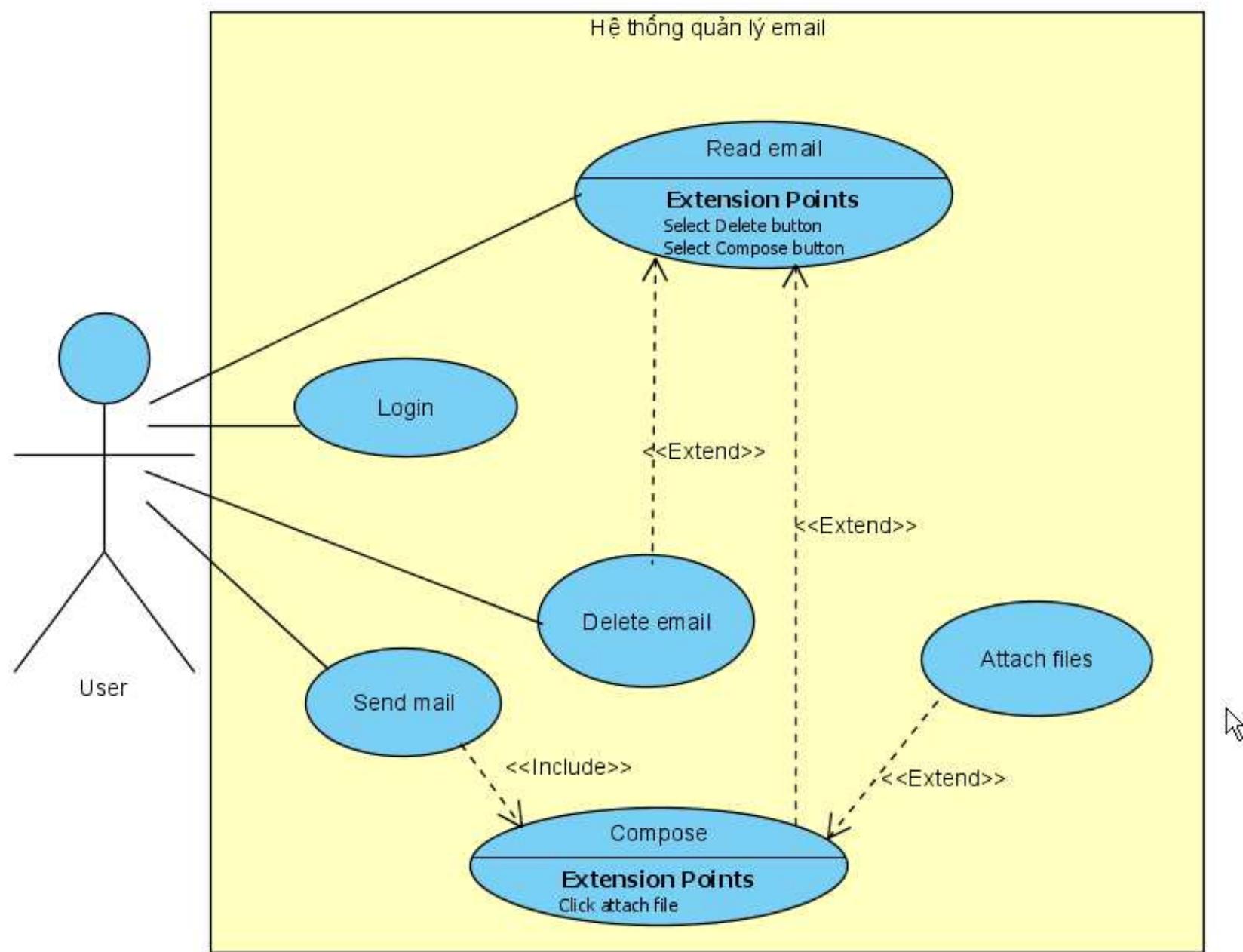
- ❖ Minh họa thông qua Code

Class B { public void InÁn () { } }

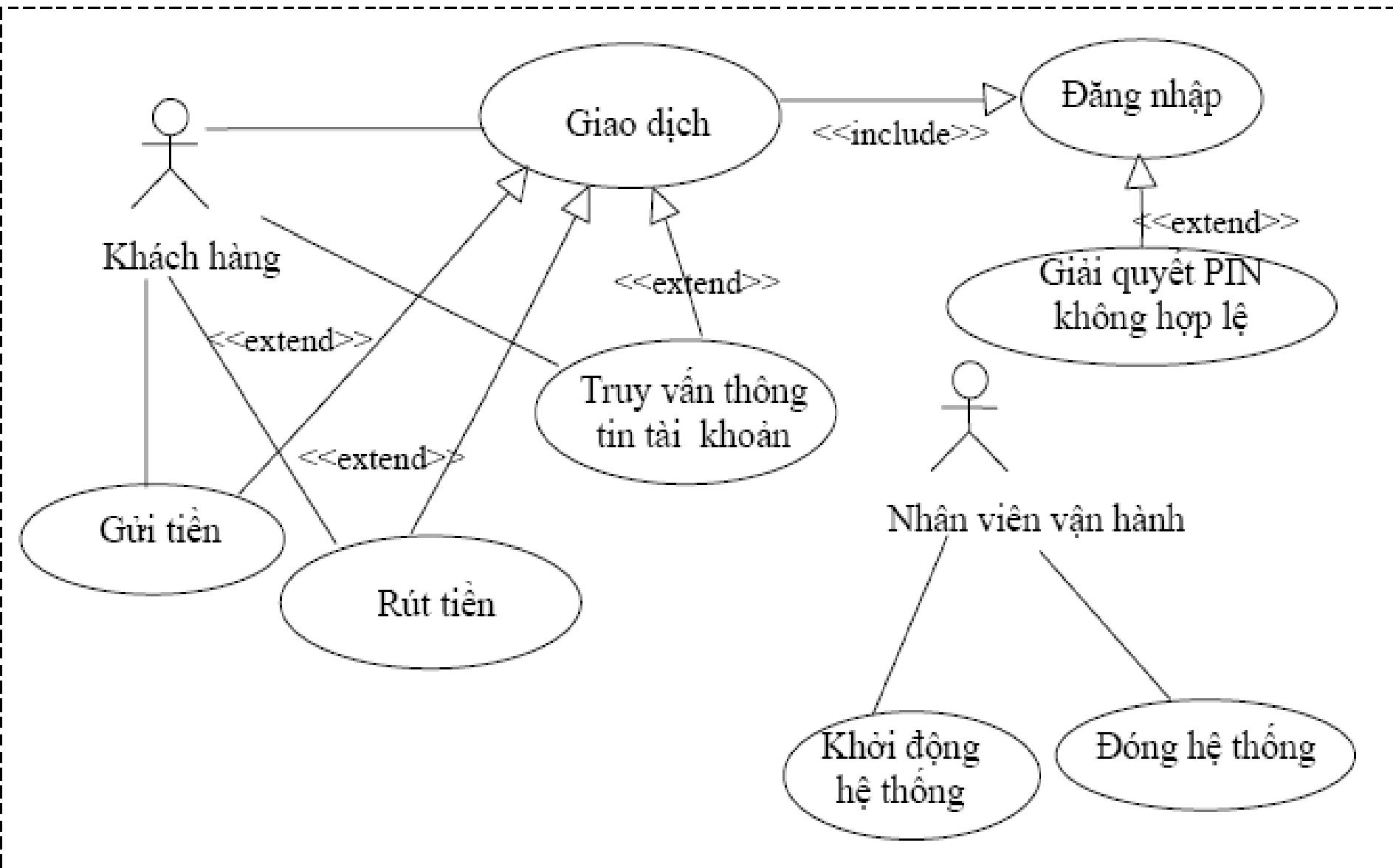
```
Class A {  
    Public void XemDSSV () {  
        ... If (Click_Nút_InÁn)  
        { B objB = new B(); objB.InÁn(); ....  
    }  
}
```



Một số hình vẽ đúng

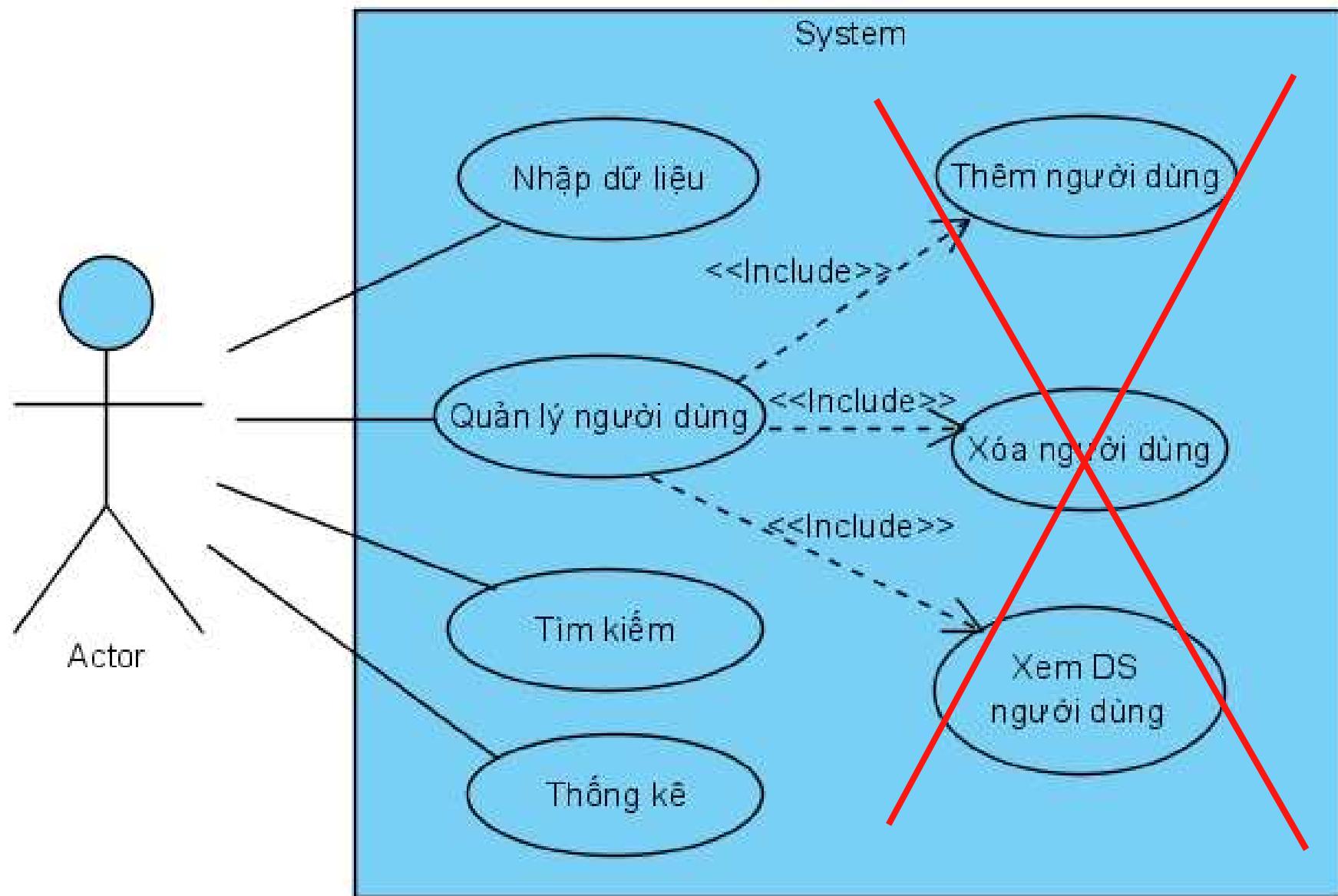


Một số hình vẽ đúng



Mô hình use case của hệ thống máy ATM

Một số hình vẽ sai



Vẽ quan hệ tổng quát hóa (thừa kế)



- ❖ Khi nào thì vẽ quan hệ <Generalization> (tổng quát hóa)
→ *Use case A được gọi là Generalization B nếu B là một trường hợp riêng của A !*
- ❖ Nếu A Generalization B thì code có dạng như thế nào

Class A {

.....

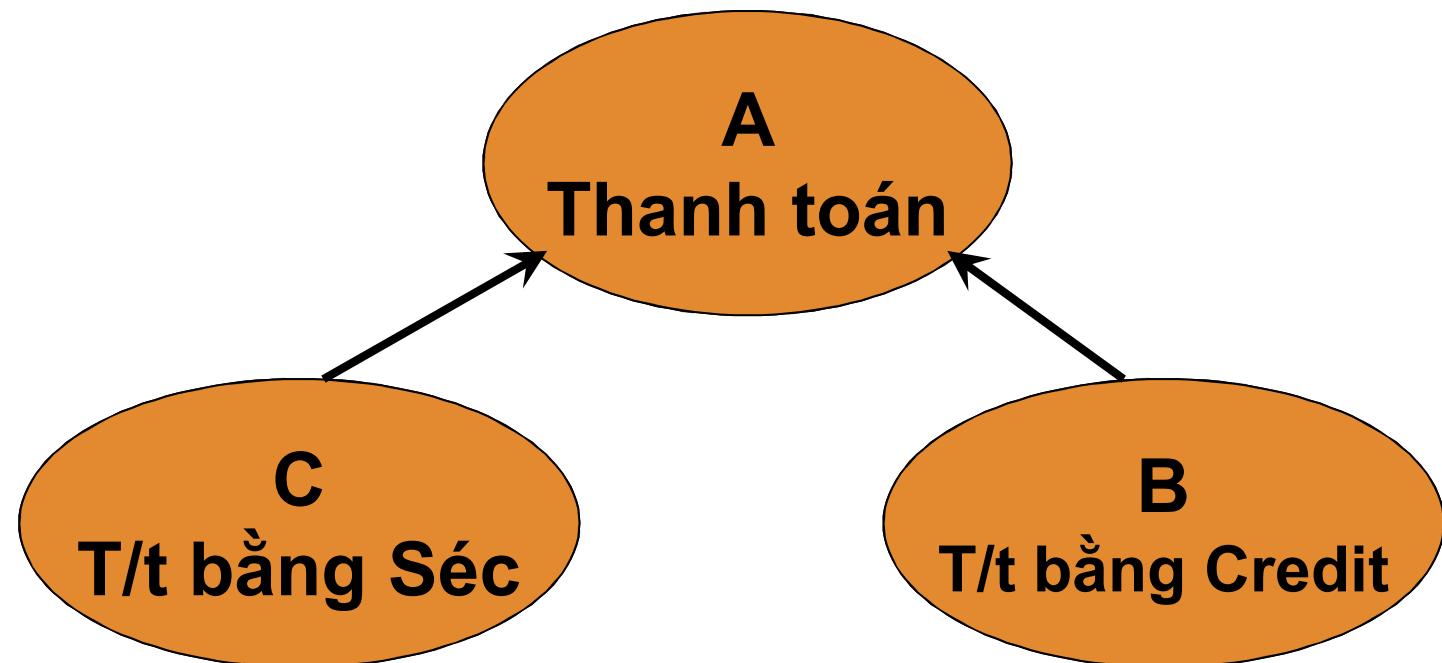
}

Class B : A

{

.....

}



3. Một số biểu đồ UML cơ bản



1

❖ Biểu đồ ca
sử dụng
**Use Case
Diagram**

2

❖ Biểu đồ
Lớp
**Class
Diagram**

- Là biểu đồ quan trọng nhất
 - Mô tả các đối tượng và mối quan hệ của chúng trong hệ thống.
 - Mô tả các thuộc tính và các hành vi (Behavior) của đối tượng.
 - Có biểu đồ lớp mức phân tích và mức cài đặt.

Component

Deployment

Communication

Collaboration

Timing

State

Hai dạng lớp: phân tích và thiết kế



Analysis

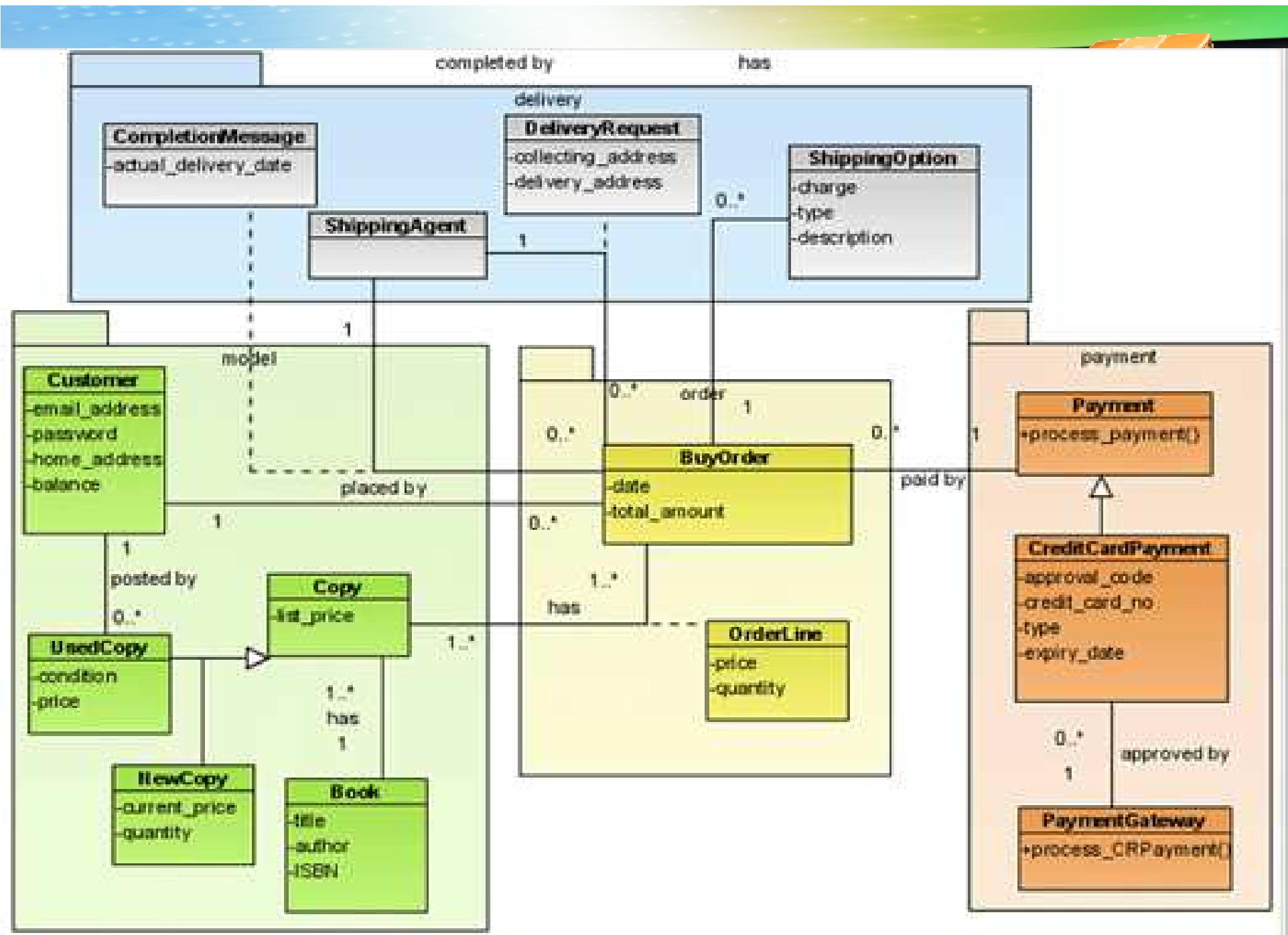
Order
Placement Date
Delivery Date
Order Number
Calculate Total
Calculate Taxes

Design

Order
<ul style="list-style-type: none">- deliveryDate: Date- orderNumber: int- placementDate: Date- taxes: Currency- total: Currency <pre># calculateTaxes(Country, State): Currency # calculateTotal(): Currency getTaxEngine() {visibility=implementation}</pre>

Bỏ qua các chi tiết
không cần thiết

Phải đầy đủ & chi
tiết các thành phần

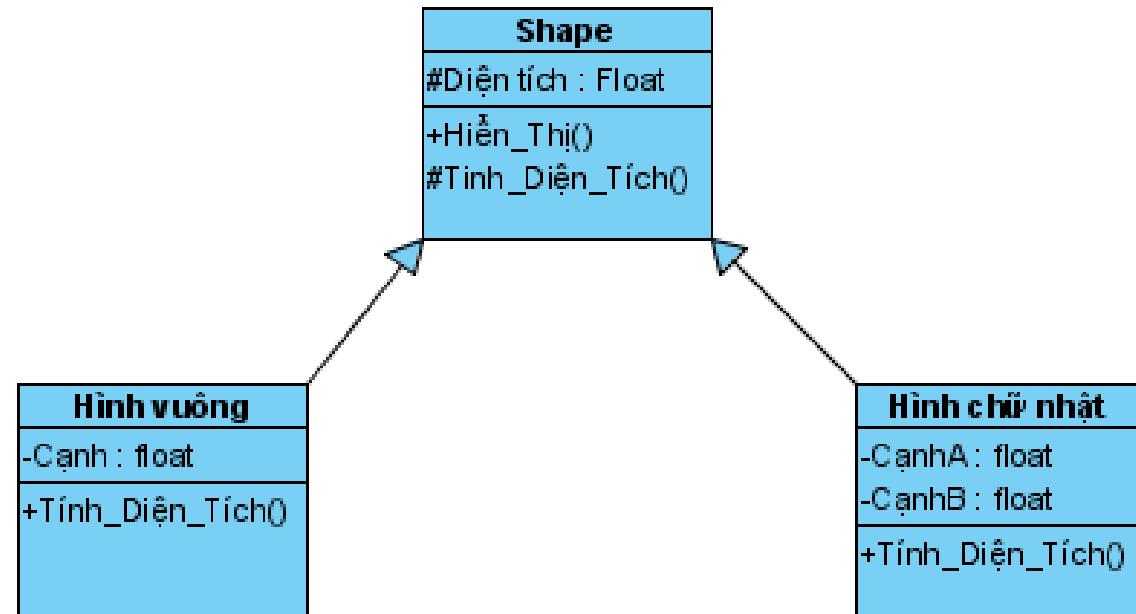




Các quan hệ trong biểu đồ lớp

- ❖ Quan hệ **Generalization**: Thể hiện rằng một lớp A kế thừa từ một lớp B (Hay A là trường hợp riêng của B; B là tổng quát của A)
- ❖ Gọi là quan hệ **Là một (Is a)**

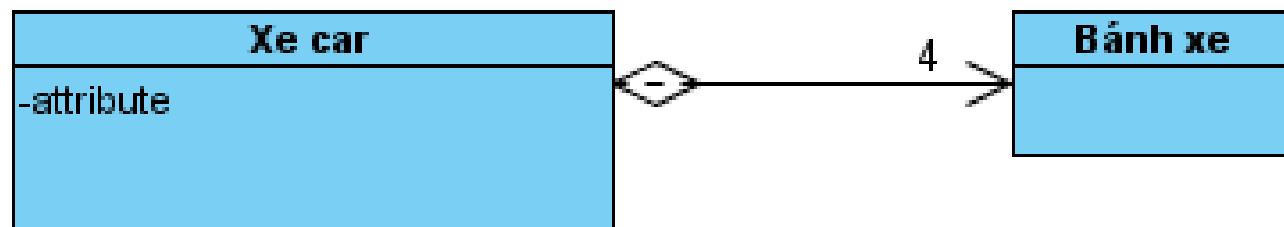
- ❖ Thể hiện:





Các quan hệ trong biểu đồ lớp (2)

- ❖ Quan hệ **Aggregation**: Thể hiện rằng một lớp A nào đó bao gồm lớp B. Lớp B này có thể tồn tại độc lập mà không cần lớp A.
- ❖ Còn gọi là mối quan hệ: **Có một (Has a)**
- ❖ Thể hiện:

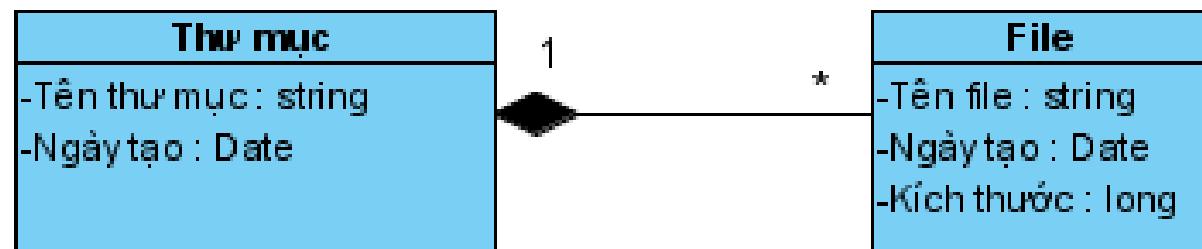




Các quan hệ trong biểu đồ lớp (3)

❖ Quan hệ **Composition**: thể hiện rằng một lớp A bao hàm lớp B. Nhưng lớp B không thể tồn tại độc lập (Tức không thuộc lớp nào). Tức là, nếu có B thì phải suy ra được A.

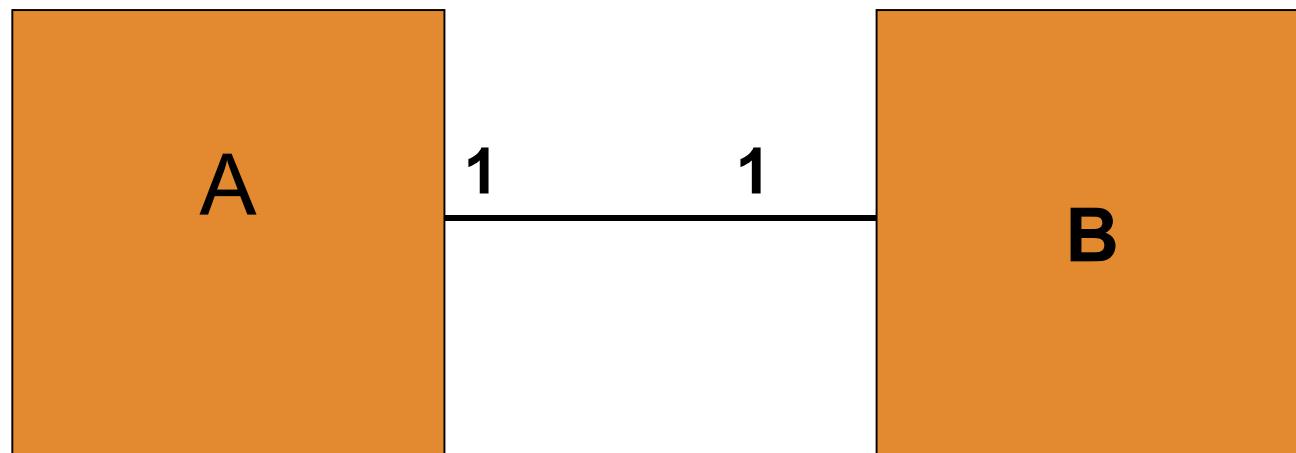
❖ Thể hiện:





Ứng số (Multiplicity)

- ❖ Thể hiện rằng ứng với mỗi lớp A thì có (chứa, dạy, có, mua, đặt,...) bao nhiêu phần tử lớp B?



Một phần tử lớp A có 1 phần tử lớp B



Ứng số (Multiplicity)

- ❖ Thể hiện rằng ứng với mỗi lớp A thì có (chứa, dạy, có, mua, đặt,...) bao nhiêu phần tử lớp B?



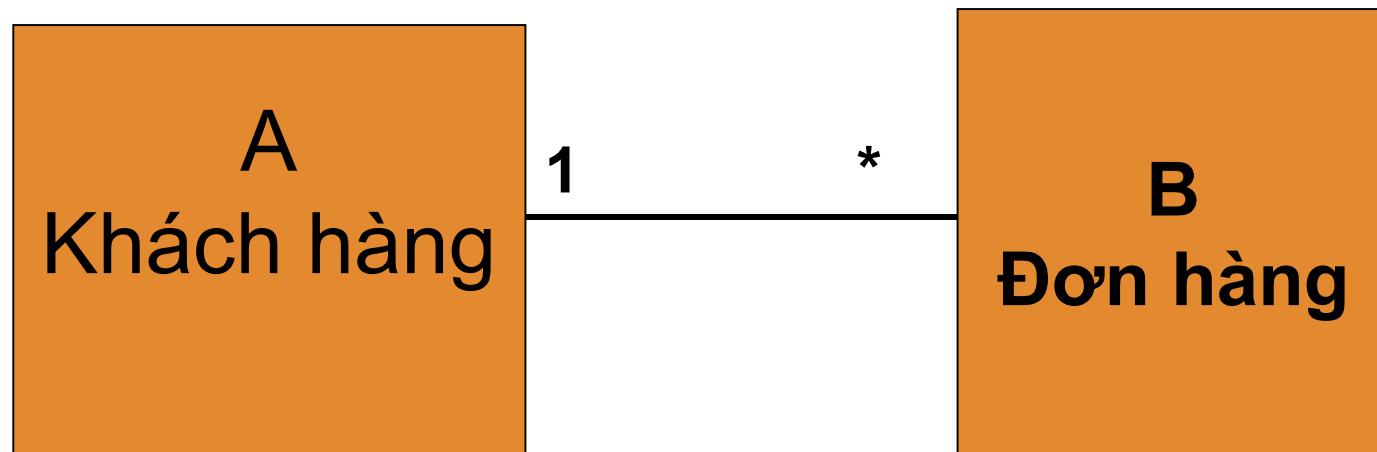
Một phần tử lớp A có tối đa 3 phần tử lớp B

Mỗi phần tử lớp B có đúng 1 phần tử lớp A



Ứng số (Multiplicity)

- ❖ Thể hiện rằng ứng với mỗi lớp A thì có (chứa, dạy, có, mua, đặt,...) bao nhiêu phần tử lớp B?



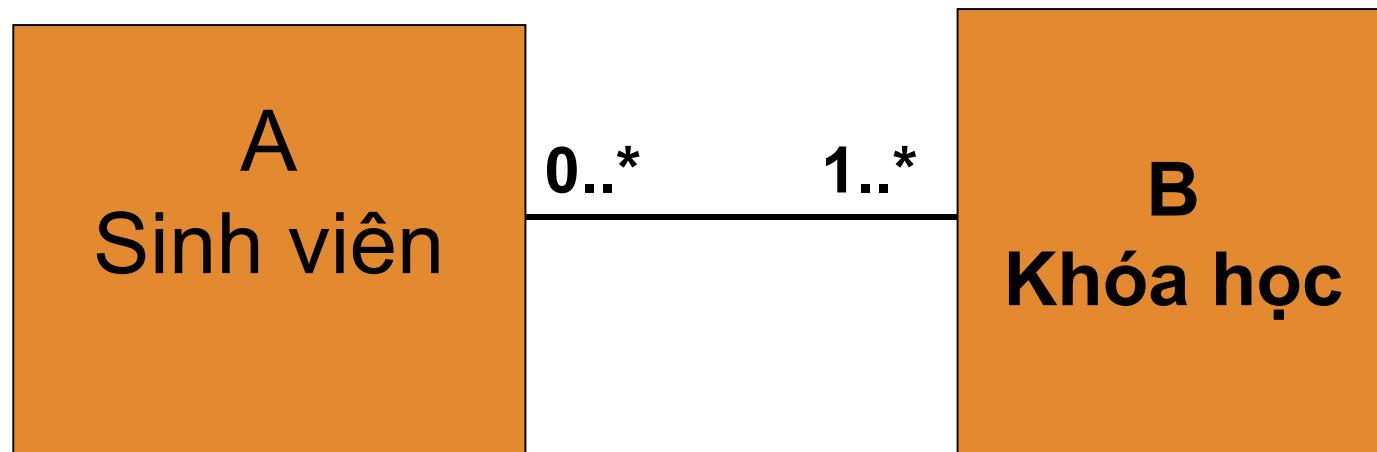
Một phần tử lớp A có nhiều phần tử lớp B

Mỗi phần tử lớp B có đúng 1 phần tử lớp A



Ứng số (Multiplicity)

- ❖ Thể hiện rằng ứng với mỗi lớp A thì có (chứa, dạy, có, mua, đặt,...) bao nhiêu phần tử lớp B?



Mỗi sinh viên tham gia ít nhất 1 khóa học

Mỗi khóa học có thể có 0 hoặc nhiều sv tham gia

3. Một số biểu đồ UML cơ bản



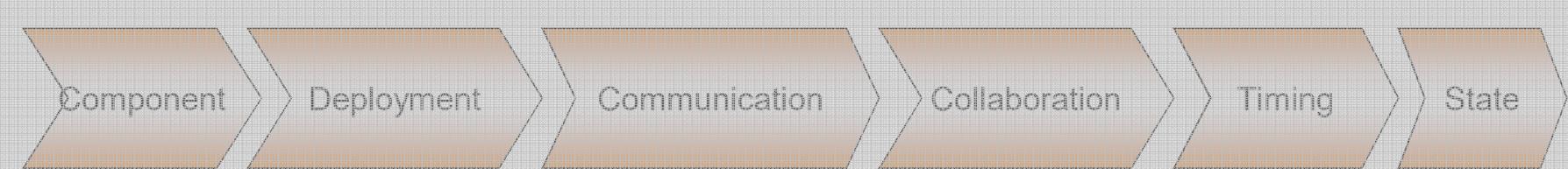
- Mô tả sự tương tác của các đối tượng theo trình tự về thời gian.
- Có sự liên kết chặt chẽ với biểu đồ lớp.
- Mỗi biểu đồ tuần tự mô tả một tình huống xử lý.

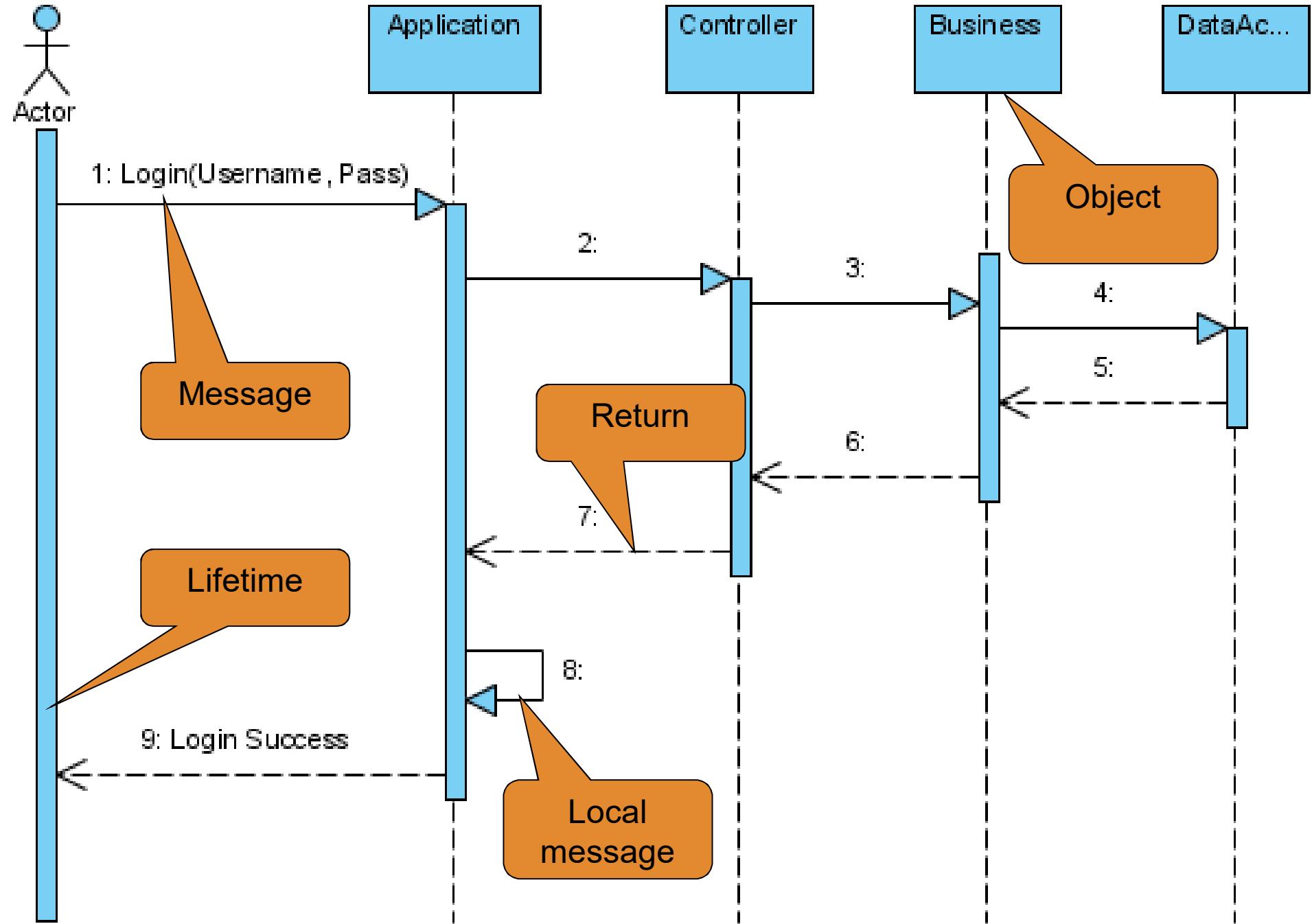
3

Biểu đồ
Tuần tự
**Sequence
Diagram**

4

Biểu đồ
Hoạt động
**Activity
Diagram**

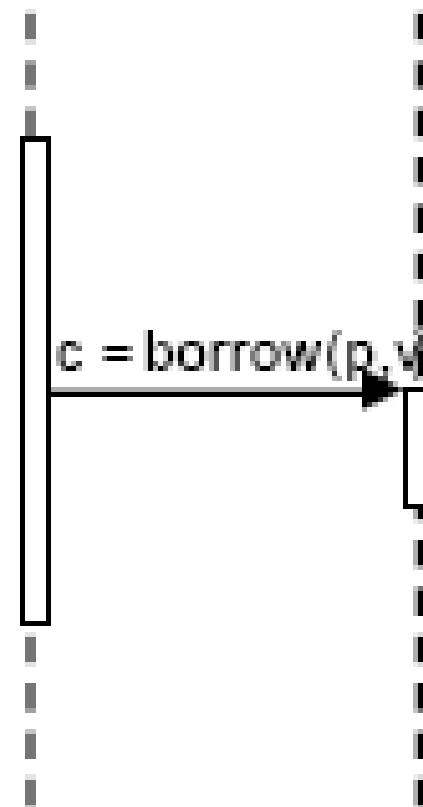
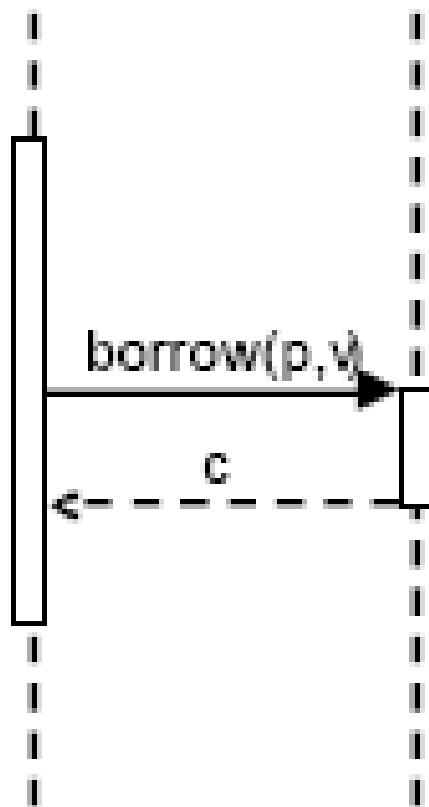




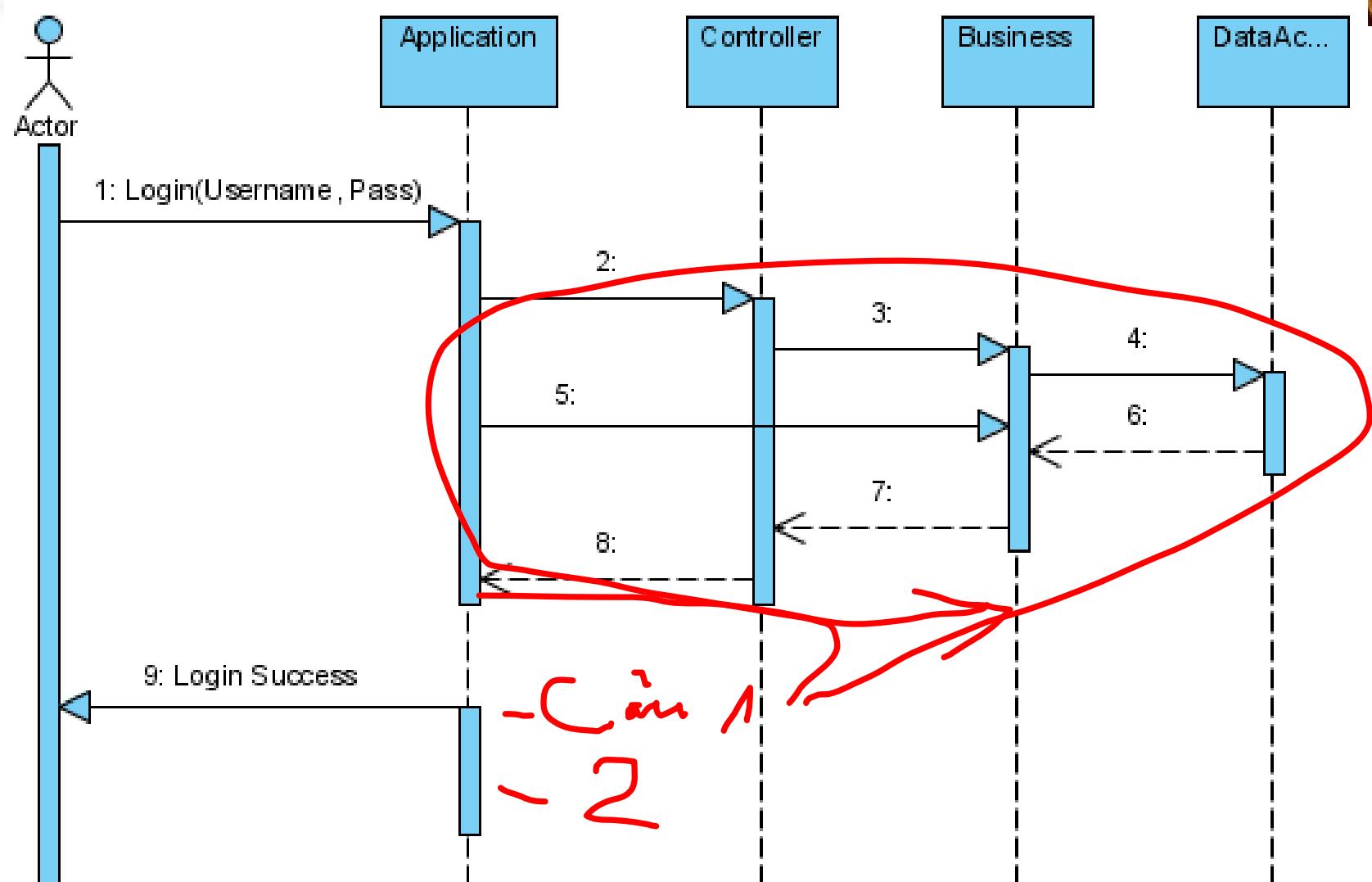
Vẽ biểu đồ tuần tự



- ❖ Chú ý: có thể vẽ một trong 2 dạng



Ví dụ vẽ sai !



3. Một số biểu đồ UML cơ bản



❖ Biểu đồ ca
sử dụng
**Use Case
Diagram**

- Mô tả các luồng công việc, qui trình nghiệp vụ.
- Tương tự như sơ đồ khối (Flowchart).
- Hỗ trợ việc mô tả các xử lý song song.

Biểu đồ
Hoạt động
**Activity
Diagram**

Component

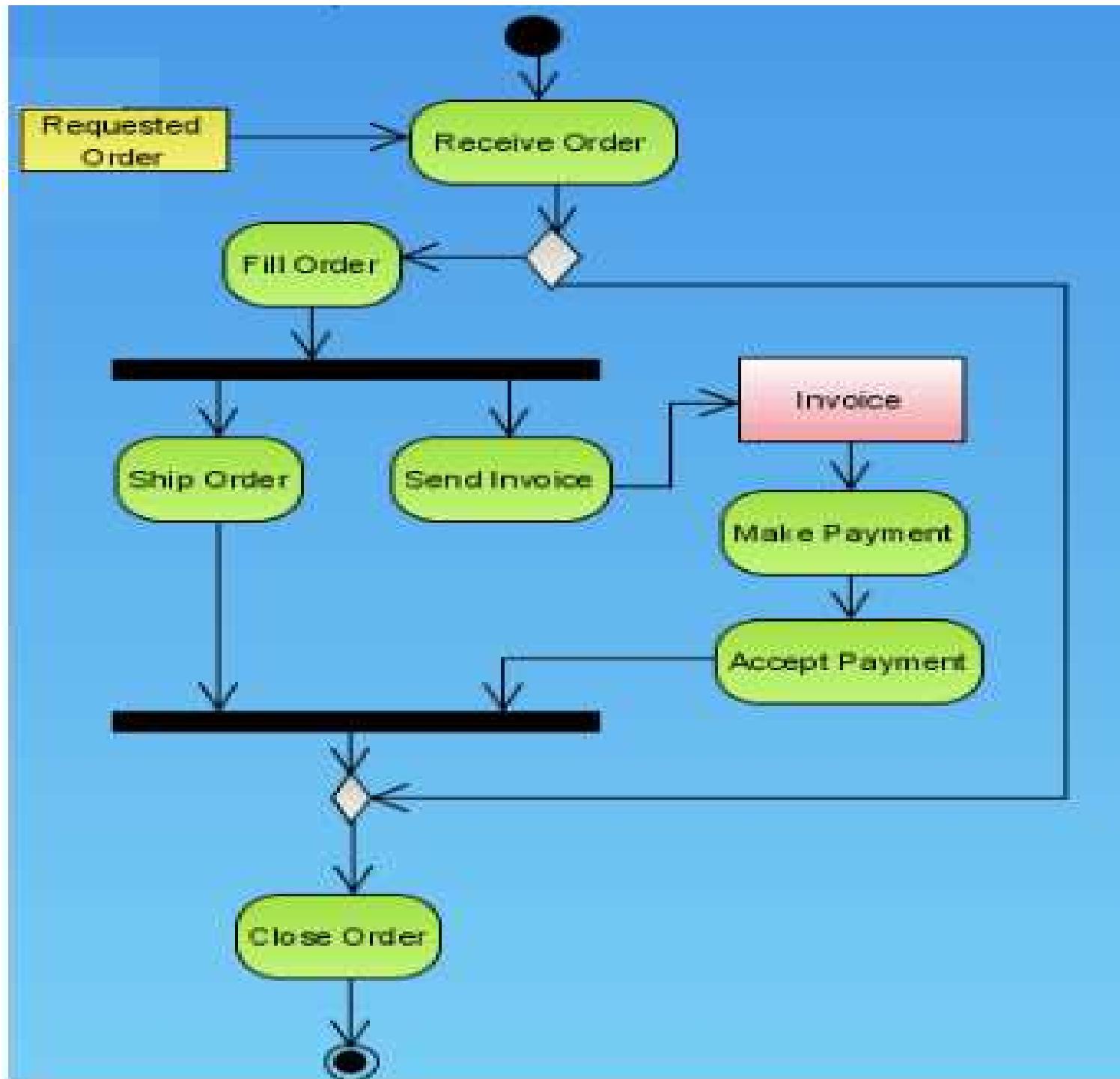
Deployment

Communication

Collaboration

Timing

State

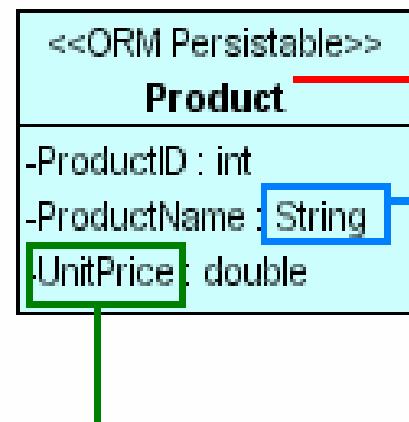


Một số biểu đồ khác



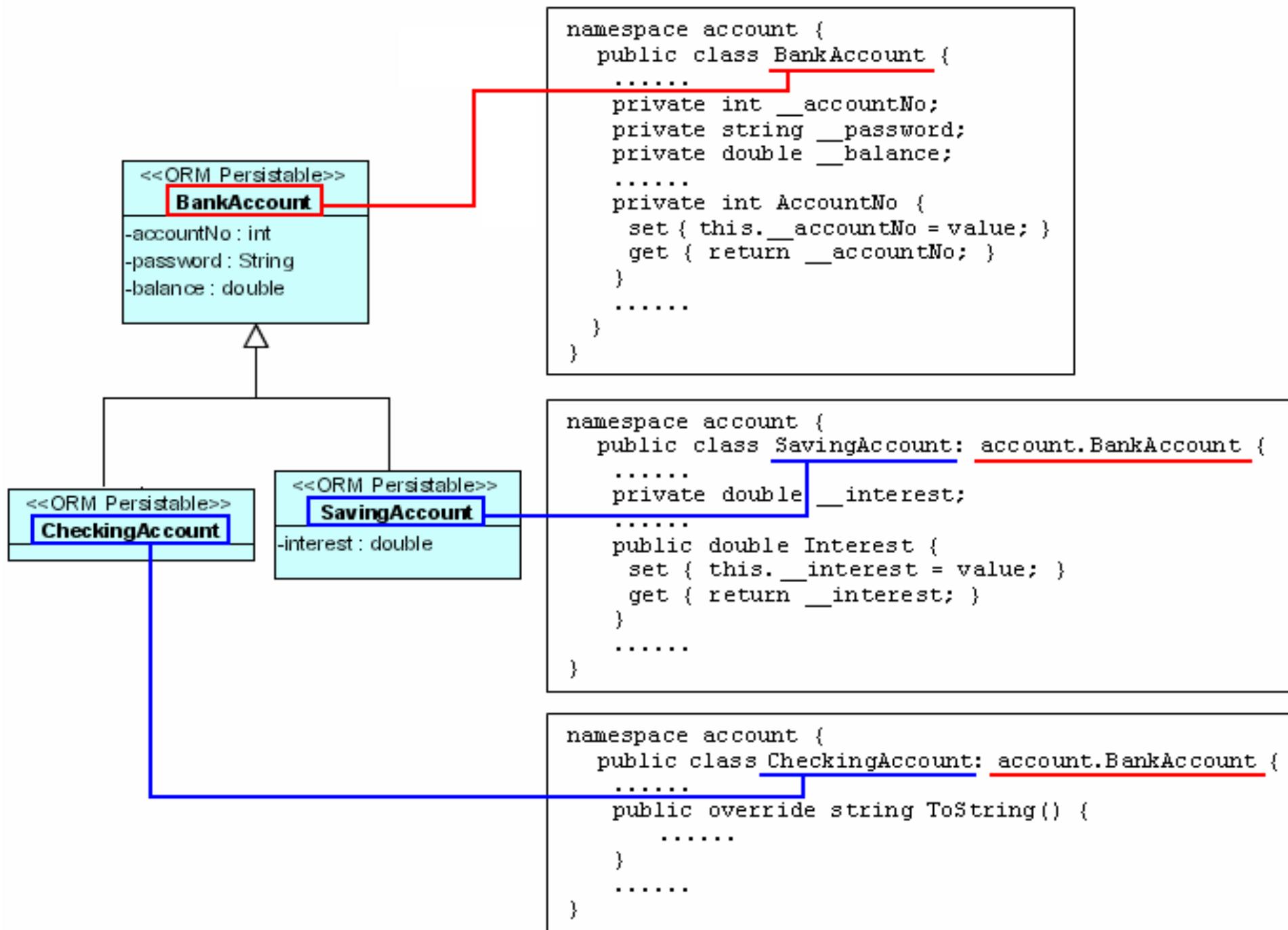
- ❖ Biểu đồ truyền thông: Communication diagram*
- ❖ Biểu đồ tương tác: Interaction Diagram
- ❖ Biểu đồ thời gian – Timming diagram*
- ❖ Biểu đồ trạng thái – State Diagram
- ❖ Biểu đồ đối tượng – Object Diagram
- ❖ Biểu đồ gói - Package Diagram
- ❖ Biểu đồ cấu trúc kết hợp – Composite Structured*
- ❖ Biểu đồ thành phần – Component Diagram
- ❖ Biểu đồ triển khai – Deployment Diagram

Ánh xạ biểu đồ sang Code



```
/// <summary>
/// ORM-Persistable Class
/// </summary>
public class Product {
    .....
    private int _ProductID;
    private string _ProductName;
    private double _UnitPrice;
    .....
}
```

• Mapping Classes, Attributes and Data Type



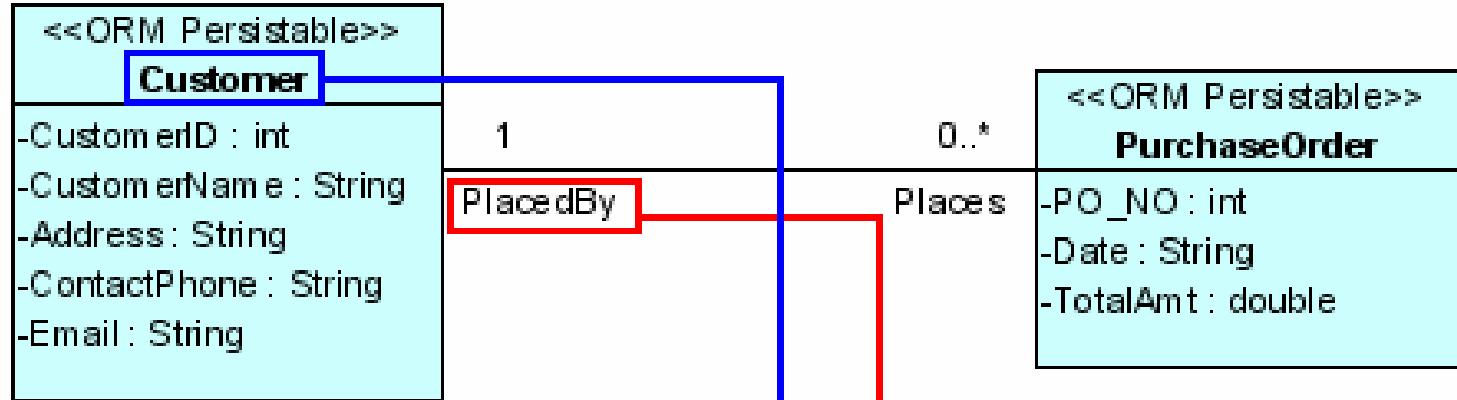
Mapping generalization

Ánh xạ khách hàng- đơn hàng



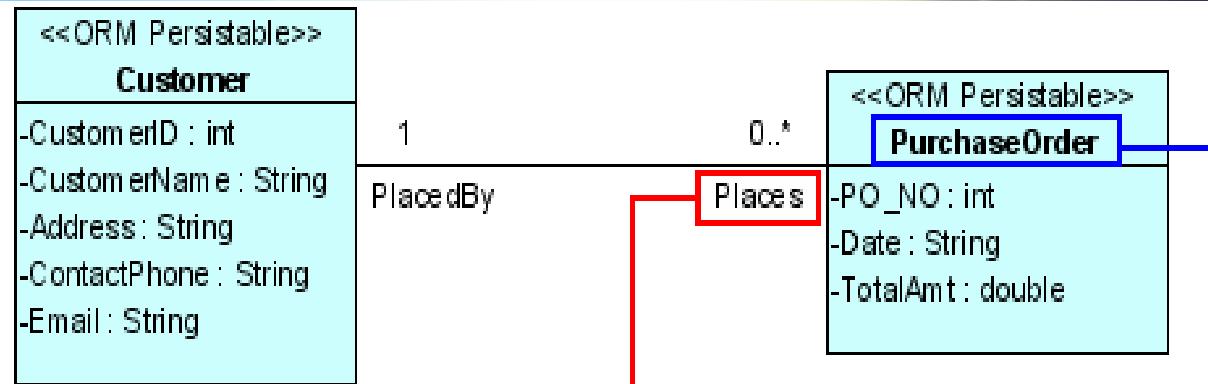
<<ORM Persistable>> Customer			<<ORM Persistable>> PurchaseOrder
-CustomerID : int -CustomerName : String -Address : String -ContactPhone : String -Email : String	1	0..*	Places PlacedBy

The Classes with association and multiplicity



```
public class PurchaseOrder {  
    .....  
    private int __PO_NO;  
    private string __Date;  
    private double __TotalAmt;  
    private shoppingcart.Customer __PlacedBy;  
    .....  
    public shoppingcart.Customer PlacedBy {  
        set { value.Places.Add(this); }  
        get { return __PlacedBy; }  
    }  
    .....  
}
```

Mapping Association and Multiplicity



```

public class Customer {
    .....
    private int _CustomerID;
    private string _CustomerName;
    private string _Address;
    private string _ContactPhone;
    private string _Email;
    .....
    public readonly
        shoppingcart.PurchaseOrderSetCollection Places:
    .....
}

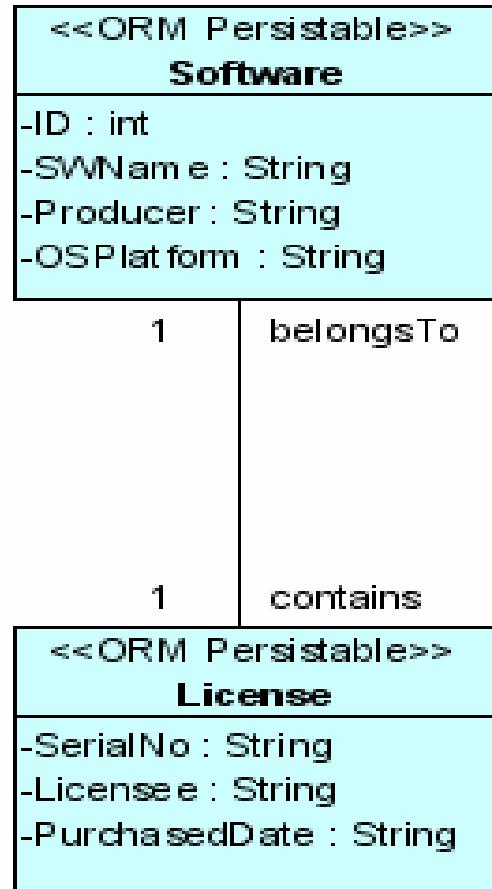
```

```

public class PurchaseOrderSetCollection
: Orm.Util.ORMSet {
    .....
    public void Add(PurchaseOrder value) {
        .....
    }
    public void Remove(PurchaseOrder value) {
        .....
    }
    public bool Contains(PurchaseOrder value) {
        .....
    }
    .....
}

```

Mapping the One-to-many Associations



Mapping



```

public class Software{
    private int _ID;
    private string _SWName;
    private string _Producer;
    private string _OSPlatform;
    private License _contains;
    .....
    public License Contains{
        set {
            .....
        }
        get {
            .....
        }
    }
}

```

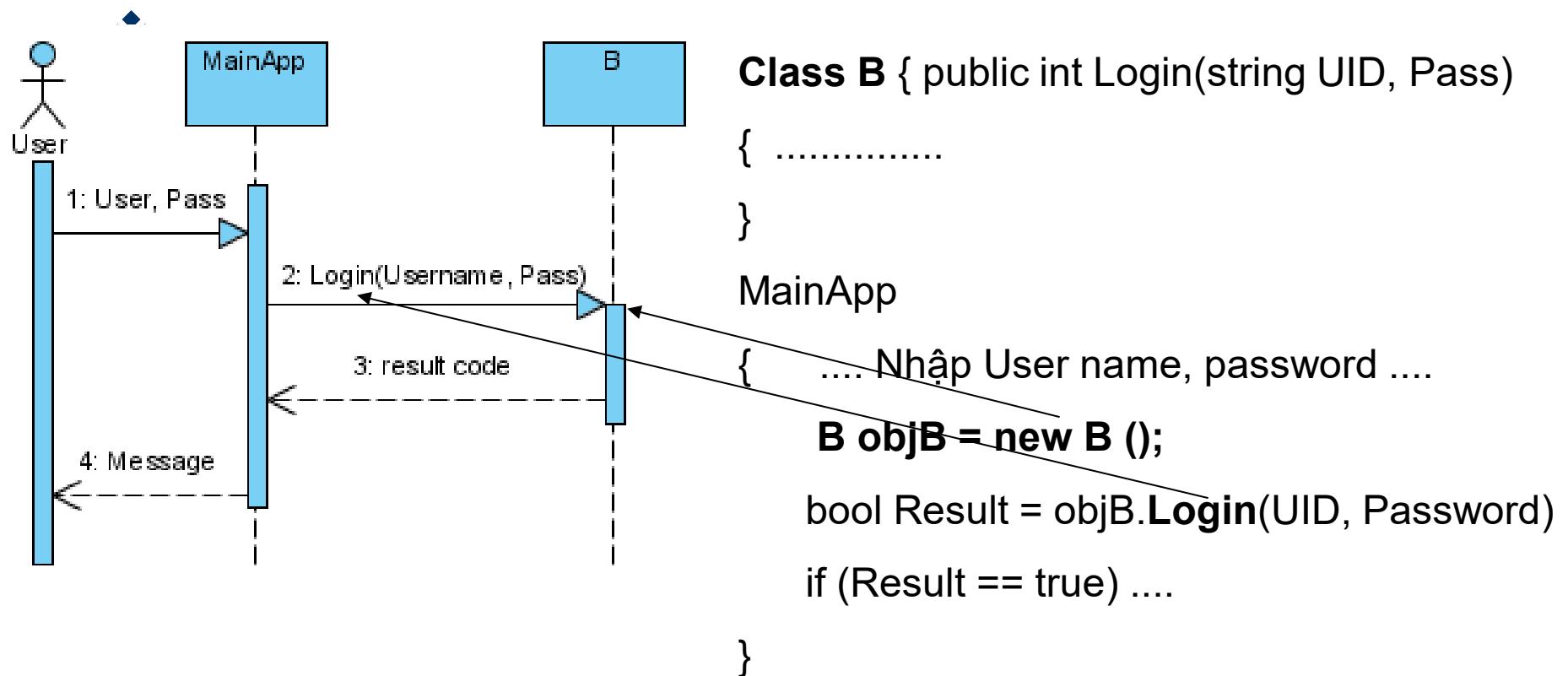
```

public class License{
    private string _SerialNo;
    private string _Licensee;
    private string _PurchasedDate;
    private Software _belongsTo;
    .....
    public Software BelongsTo{
        set {
            .....
        }
        get {
            .....
        }
    }
}

```

Mapping the One-to-one Relationship

Ánh xạ biểu đồ tuần tự sang Code





Thank You !

Aptech Computer Education