

CÔNG NGHỆ PHẦN MỀM

Khoa Công nghệ Thông tin – Đại học Đà Lạt



CHƯƠNG 5. THIẾT KẾ PHẦN MỀM

NỘI DUNG

- 1. Giới thiệu (Mục tiêu, quy trình)**
- 2. Một số khái niệm**
- 3. Xây dựng sơ đồ lớp (mức thiết kế)**
- 4. Thiết kế dữ liệu**
- 5. Thiết kế giao diện**
- 6. Thiết kế xử lý**

1. Giới thiệu (Mục tiêu, quy trình)

Thiết kế phần mềm là giai đoạn quan trọng trong quá trình phát triển dự án.

Kết quả: bản **Đặc tả Thiết kế** (Design Specifications)



1. Giới thiệu (Mục tiêu, quy trình)

Mục tiêu

- Hiểu biết sâu về:
 - Các yêu cầu phi chức năng
 - Các ràng buộc
 - Sử dụng lại thành phần, các hệ điều hành, các công nghệ phân tán, cơ sở dữ liệu, giao diện người dùng, quản lý các giao dịch

1. Giới thiệu (Mục tiêu, quy trình)

Mục tiêu

- Tạo ra một đầu vào thích hợp và làm xuất phát điểm cho các hoạt động cài đặt tiếp theo
- Có khả năng phân rã việc cài đặt thành các mẩu nhỏ dễ quản lý hơn, được phát triển bởi nhiều nhóm khác nhau và có thể tiến hành đồng thời.

1. Giới thiệu (Mục tiêu, quy trình)

Mục tiêu

- Nắm bắt sớm các giao diện chủ yếu giữa các hệ thống con trong vòng đời của phần mềm.
- Trực quan hóa và suy luận thiết kế bằng cách sử dụng một hệ thống các ký pháp chung.
- Tạo ra một sự trừu tượng hóa liên tục của việc cài đặt của hệ thống.

1. Giới thiệu (Mục tiêu, quy trình)

Quy trình

- Xây dựng sơ đồ lớp.
- Xây dựng sơ đồ trạng thái
- Thiết kế dữ liệu
- Thiết kế giao diện
- Thiết kế xử lý

=> **Đặc Tả Thiết Kế (Design Specifications)**

2. Một số khái niệm

Thiết kế phần mềm là quá trình thiết kế cấu trúc phần mềm dựa trên những tài liệu đặc tả.

Thiết kế dữ liệu: Cấu trúc dữ liệu được sử dụng để cài đặt hệ thống phải được thiết kế một cách chi tiết và cụ thể.

Thiết kế xử lý: Chi tiết các hàm xử lý phục vụ cho các chức năng của hệ thống.

2. Một số khái niệm

Thiết kế giao diện: với mỗi hệ thống con, các giao diện của nó với những hệ thống con khác phải được thiết kế và tư liệu hóa.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Tổng quan

Các lớp được xem có vị trí trọng tâm của bất kỳ hệ thống hướng đối tượng.

Đa số các sơ đồ UML phổ biến đều là sơ đồ lớp.

Câu hỏi ôn lại: Kể tên các sơ đồ UML đã học?

3. Xây dựng sơ đồ lớp (mức thiết kế)

Tổng quan

Cấu trúc hệ thống được tạo thành từ các đối tượng.

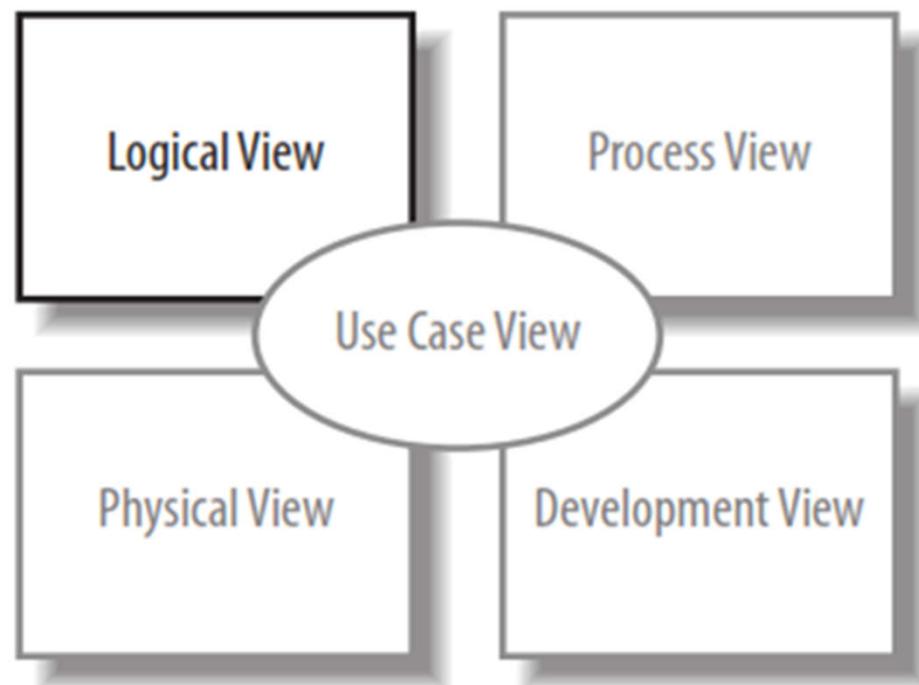
Các lớp mô tả các dạng đối tượng khác nhau

Sơ đồ lớp mô tả các lớp này và các mối quan hệ giữa chúng.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Tổng quan

Các lớp hình thành nên phần **Logical View**.

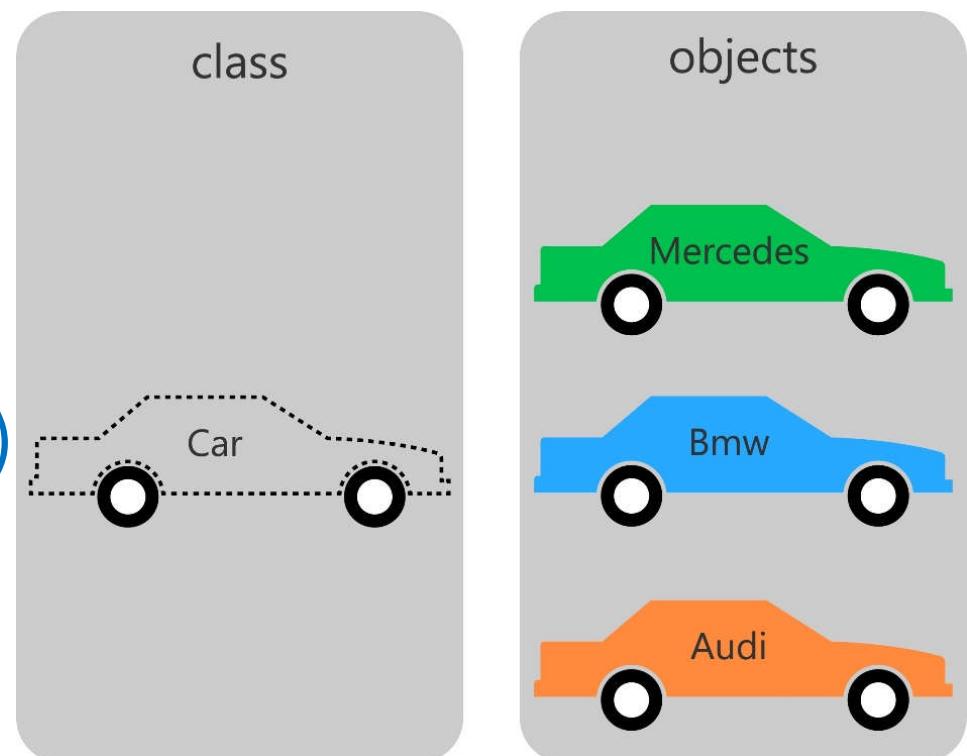


3. Xây dựng sơ đồ lớp (mức thiết kế)

Lớp

Một lớp là một bản mẫu (template) tạo ra các đối tượng cung cấp giá trị khởi tạo cho:

- Các trạng thái (thuộc tính)
- Các thực thi hành vi (phương thức).

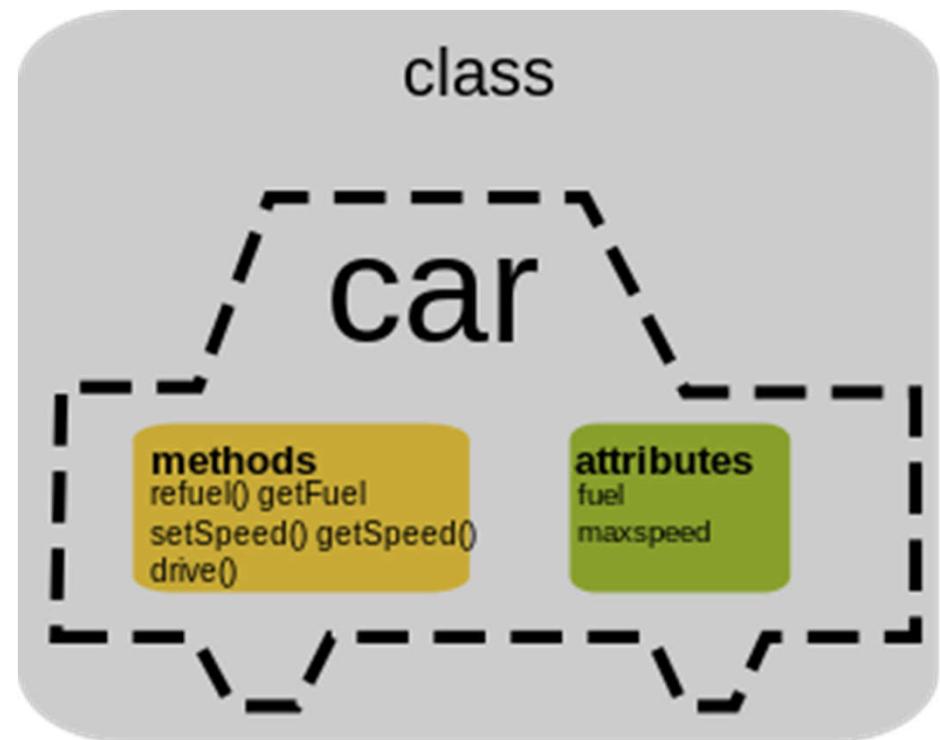


3. Xây dựng sơ đồ lớp (mức thiết kế)

Lớp

Một lớp là một bản mẫu (template) tạo ra các đối tượng cung cấp giá trị khởi tạo cho:

- Các trạng thái (thuộc tính)
- Các thực thi hành vi (phương thức).



3. Xây dựng sơ đồ lớp (mức thiết kế)

Trừu tượng hóa

Trừu tượng hóa là việc mô tả định nghĩa lớp có chứa các thông tin chi tiết theo một bối cảnh nào đó cho trước.

Đôi khi, chúng ta có thể loại bỏ một số thông tin chi tiết không cần thiết hoặc thêm vào thông tin cho phù hợp với việc trừu tượng hóa.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Đóng gói

Một đối tượng của 1 lớp bao gồm các thuộc tính và các phương thức.

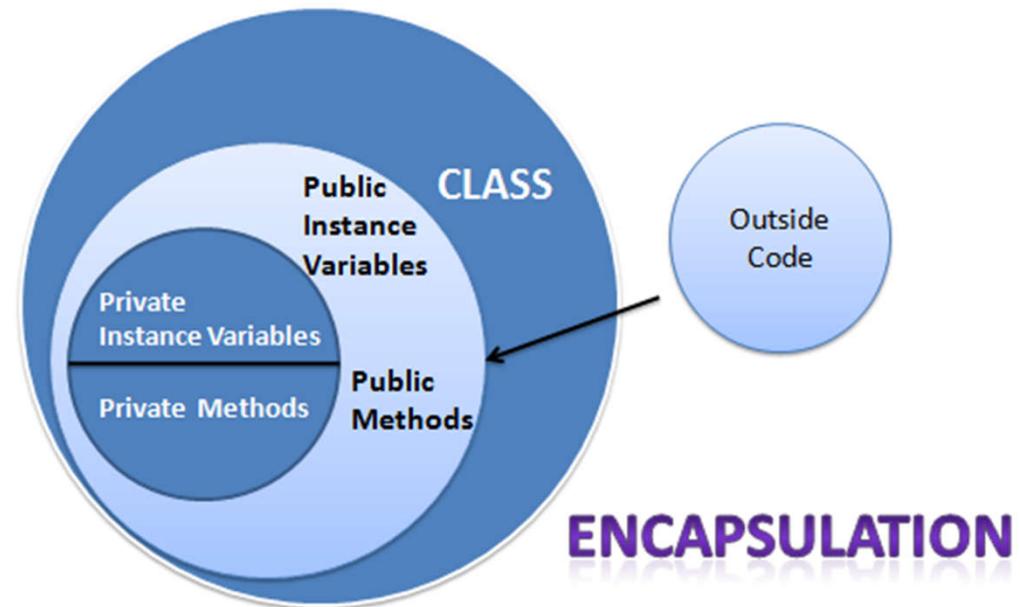
Tính đóng gói là tính chất che dấu các hoạt động bên trong 1 đối tượng với thế giới bên ngoài.

Đóng gói là một tính năng hữu dụng và mạnh nhất của cách tiếp cận hướng đối tượng trong việc phát triển hệ thống.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Đóng gói

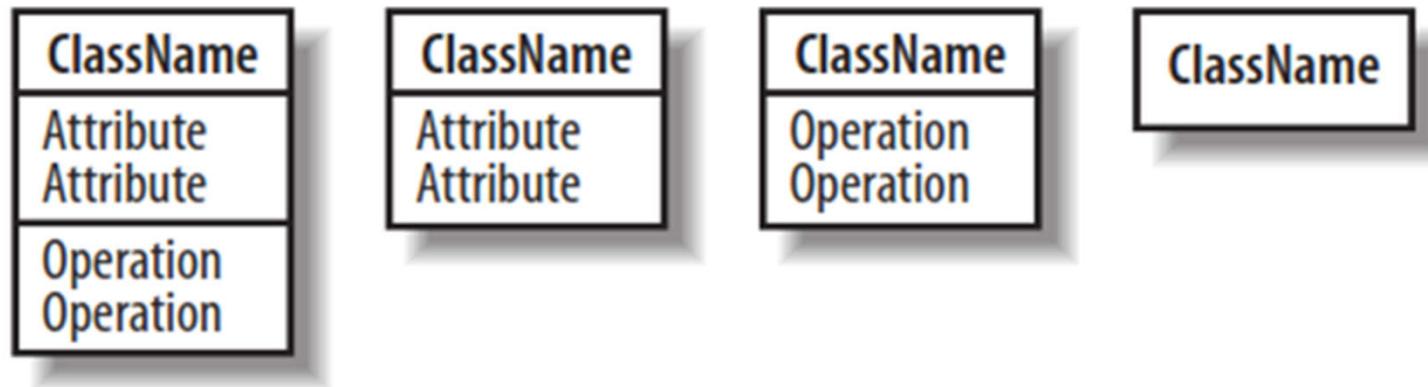
Đóng gói rất quan trọng bởi vì một lớp có thể thay đổi cách hoạt động bên trong mà toàn bộ hệ thống đều không thể nhìn thấy.



3. Xây dựng sơ đồ lớp (mức thiết kế)

Lớp trong UML

Lớp trong sơ đồ UML được biểu diễn là hình chữ nhật gồm 3 phần. Phần trên chứa tên lớp, phần giữa chứa thuộc tính và phần dưới chứa các phương thức.



Bốn cách biểu diễn một lớp dùng ký hiệu UML

3. Xây dựng sơ đồ lớp (mức thiết kế)

Lớp trong UML

Các thuộc tính và phương thức là tùy chọn, có thể không cần mô tả.

Nếu các **thuộc tính và phương thức không được mô tả, không có nghĩa là lớp trống không chứa thuộc tính hay phương thức nào, chỉ là sơ đồ muốn đơn giản hóa với các thông tin chứa bên trong.**

3. Xây dựng sơ đồ lớp (mức thiết kế)

Lớp trong UML

Câu hỏi: Tên một lớp thường là:

- a. Danh từ
- b. Tính từ
- c. Động từ
- d. Trạng từ

3. Xây dựng sơ đồ lớp (mức thiết kế)

Lớp trong UML

Ví dụ: mô tả mức tổng quát 2 lớp BlogAccount và BlogEntry trong hệ thống CMS.

BlogAccount: định nghĩa các thông tin liên quan đến tài khoản người dùng

BlogEntry: định nghĩa các thông tin chứa trong 1 entry được tạo bởi người dùng



3. Xây dựng sơ đồ lớp (mức thiết kế)

Phạm vi truy cập

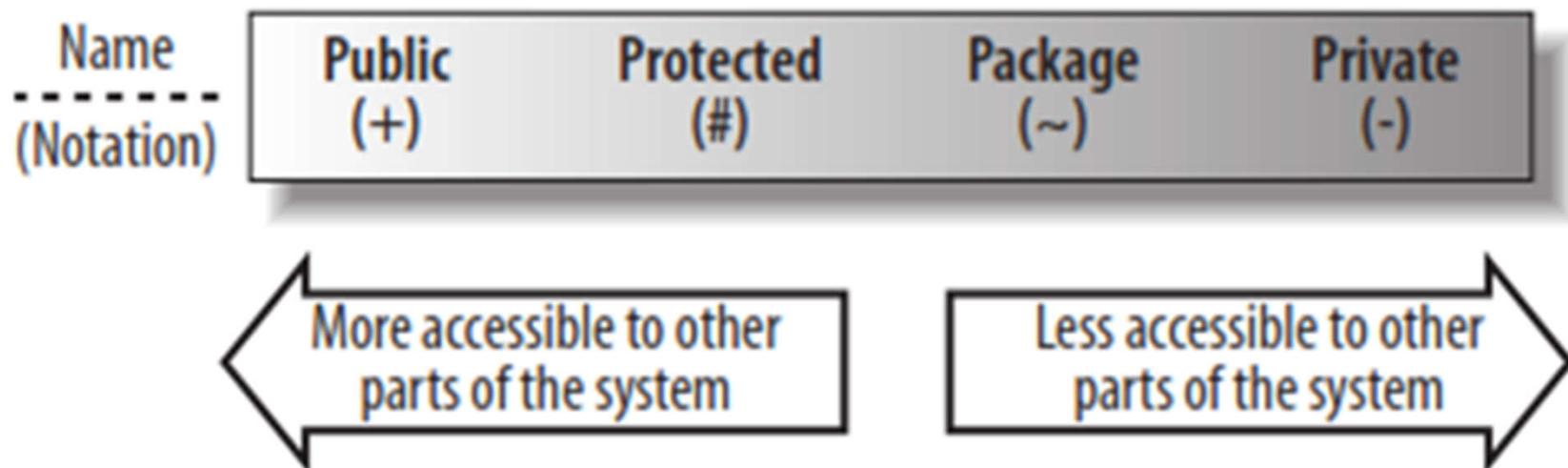
Phạm vi truy cập là cách thức một lớp cho phép các lớp khác truy xuất các thuộc tính và phương thức của nó

Khi dùng các đặc tính phạm vi truy cập, lập trình viên có thể kiểm soát các thuộc tính, các phương thức và cả toàn bộ các lớp để thực thi hiệu quả quá trình đóng gói.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Phạm vi truy cập

Có 4 dạng phạm vi truy cập được áp dụng cho 1 phần tử của sơ đồ UML



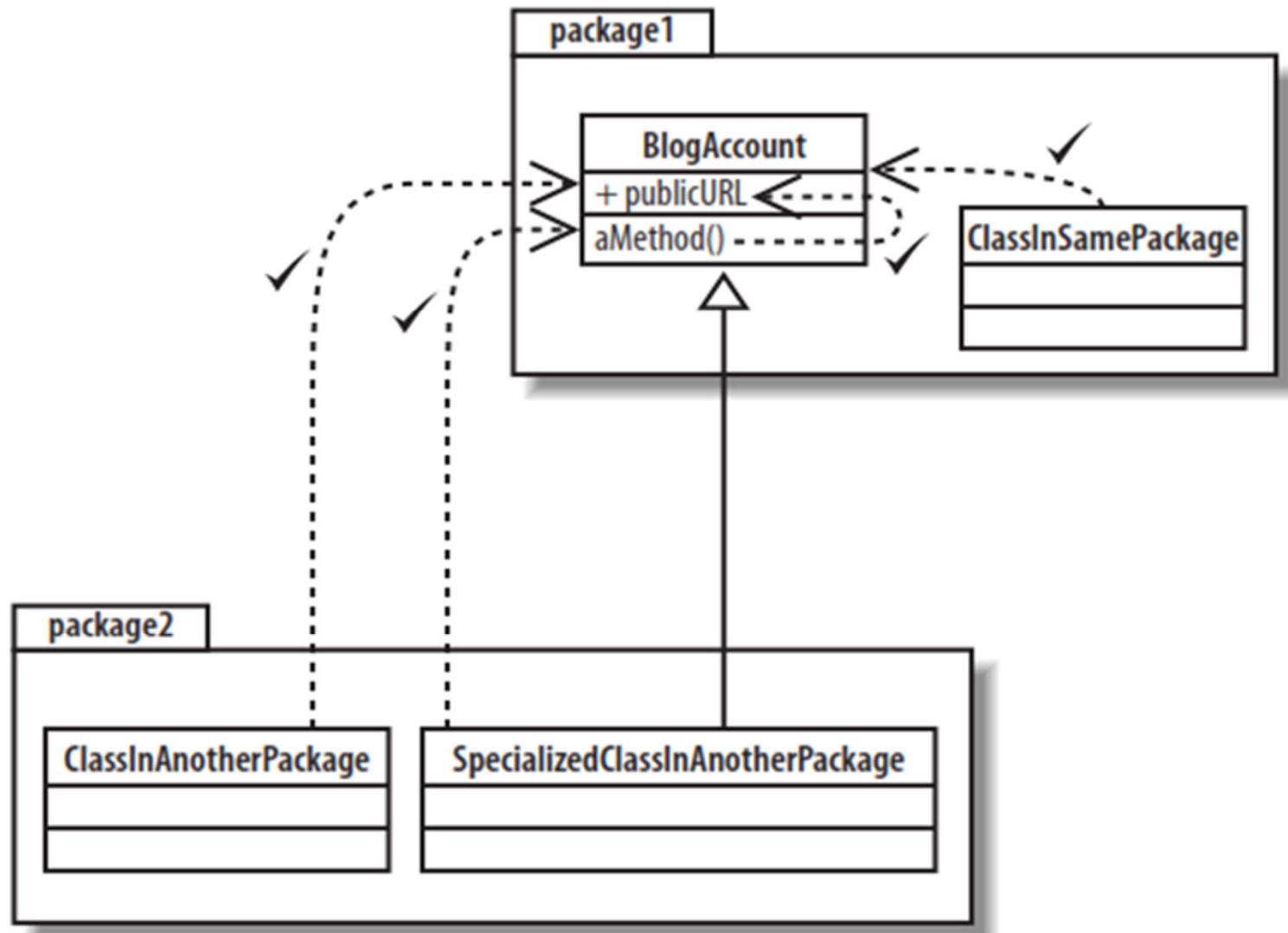
3. Xây dựng sơ đồ lớp (mức thiết kế)

Phạm vi truy cập public

Phạm vi truy cập Public (mô tả bằng dấu +) cho phép tất cả thuộc tính, phương thức của 1 lớp được truy xuất trực tiếp bởi tất cả các lớp khác trong sơ đồ mô hình.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Phạm vi truy cập public



3. Xây dựng sơ đồ lớp (mức thiết kế)

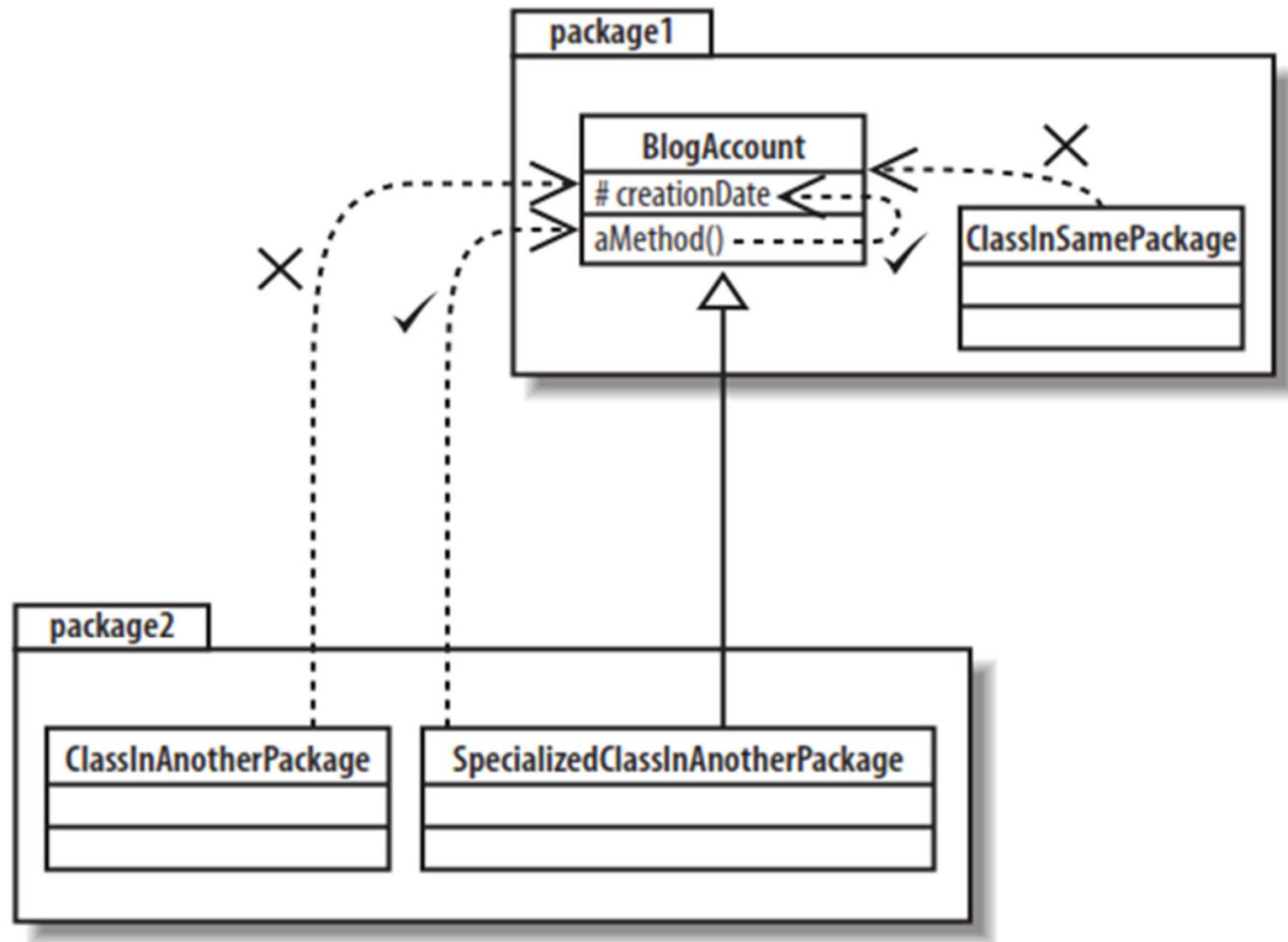
Phạm vi truy cập protected

Các thuộc tính hay phương thức được gán Protected (ký hiệu #).

Các phần tử Protected của lớp **BlogAccount** có thể được truy cập bởi các phương thức là một phần của lớp **BlogAccount** và các phương thức của các lớp khác nếu kế thừa lớp **BlogAccount**.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Phạm vi truy cập protected



3. Xây dựng sơ đồ lớp (mức thiết kế)

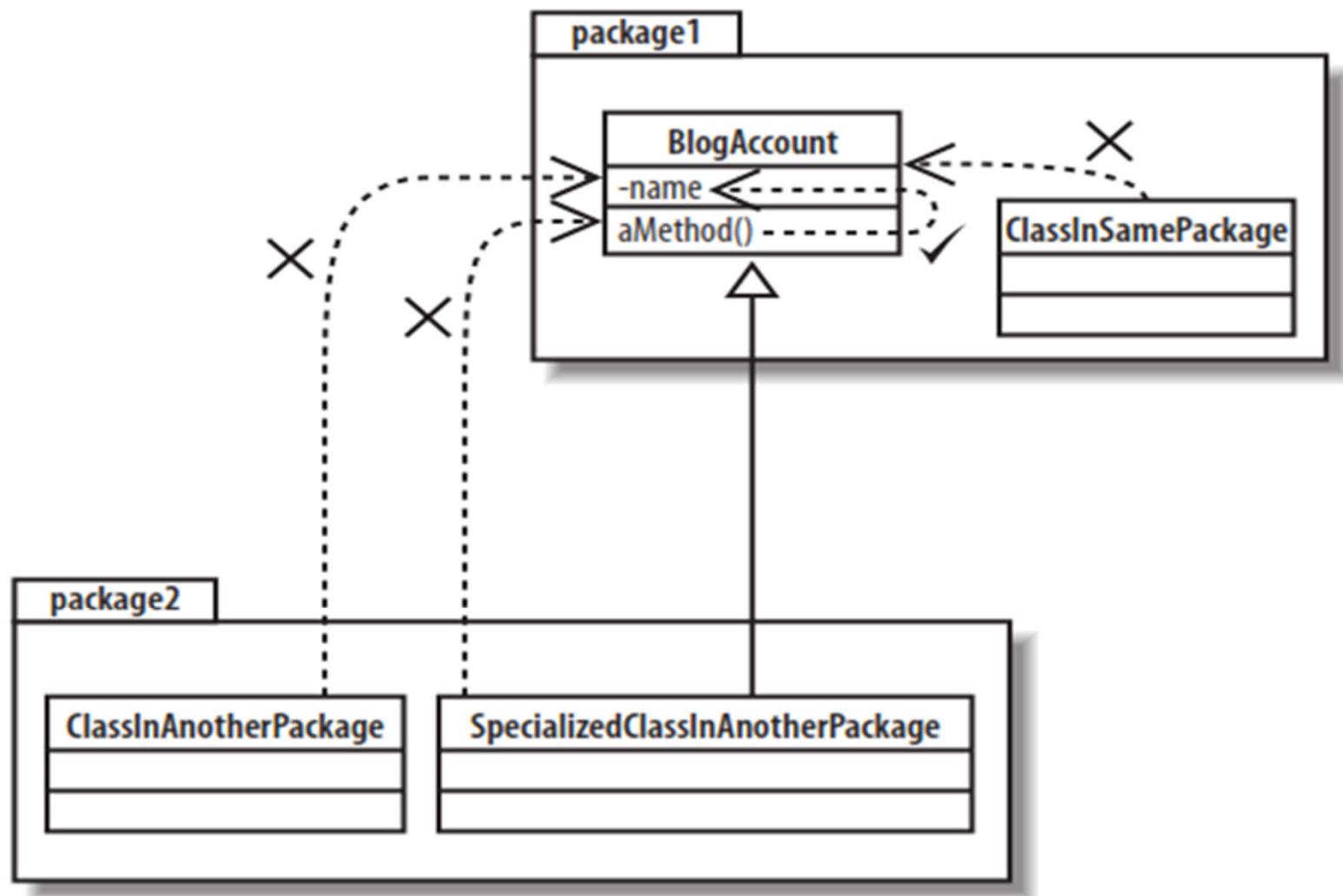
Phạm vi truy cập private

Phạm vi truy cập Private là dạng truy xuất hạn chế nhất, được biểu diễn bằng dấu trừ (-) trước thuộc tính hay phương thức.

Chỉ có lớp chứa phần tử Private mới có thể nhìn thấy và tương tác với dữ liệu được lưu trữ ở thuộc tính Private hay truy xuất đến một phương thức Private.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Phạm vi truy cập private



3. Xây dựng sơ đồ lớp (mức thiết kế)

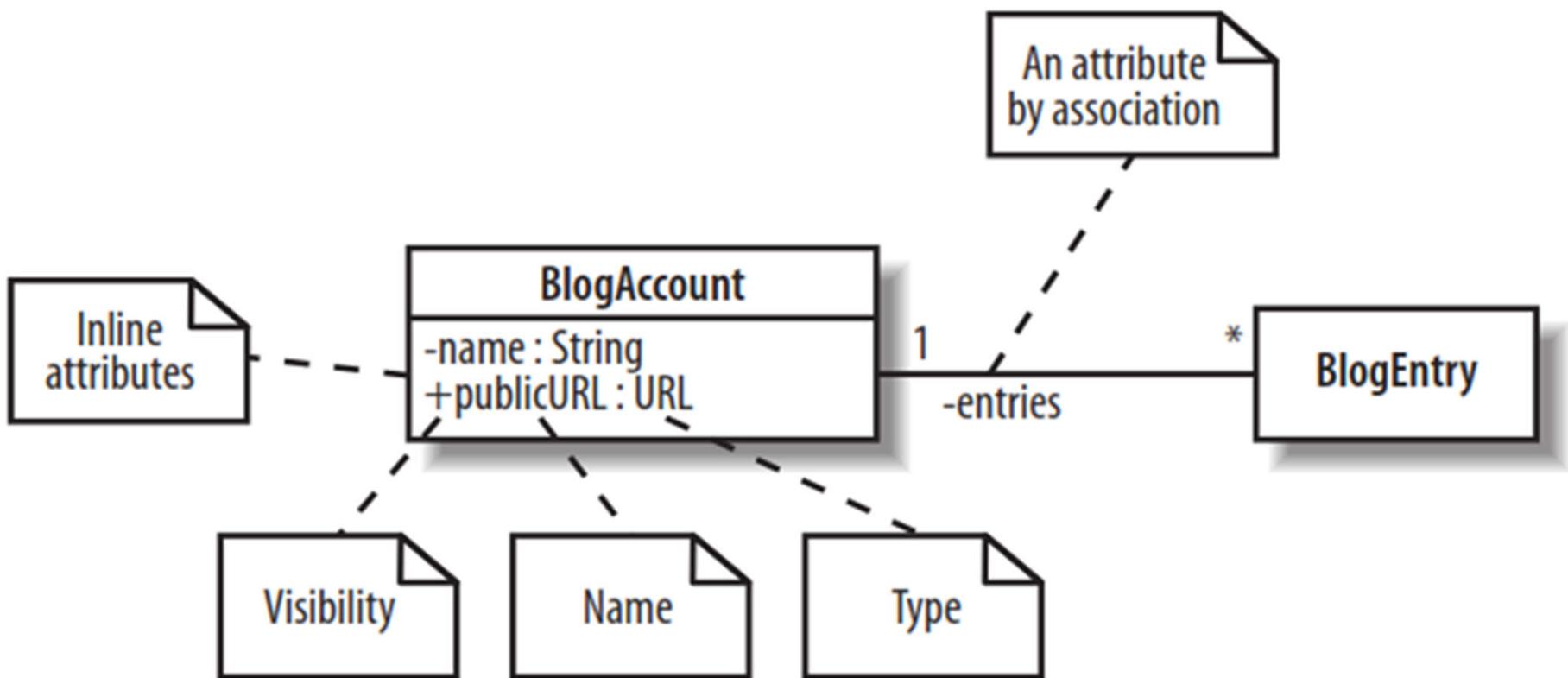
Trạng thái lớp: Các thuộc tính

Các thuộc tính là các mẫu thông tin thể hiện trạng thái của đối tượng.

Các thuộc tính này có thể biểu diễn trên mô hình lớp bằng cách đặt vào phần thuộc tính của 1 lớp hoặc là một mối quan hệ giữa các lớp.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Trạng thái lớp: Các thuộc tính



3. Xây dựng sơ đồ lớp (mức thiết kế)

Tên và kiểu

Tên của một thuộc tính có thể là tập các ký tự, không thể có 2 thuộc tính trùng tên trong một lớp.

Kiểu dữ liệu thuộc tính có thể khác nhau tùy thuộc vào mục đích biểu diễn thông tin của lớp.

Kiểu dữ liệu có thể là 1 lớp như kiểu **String** hoặc là kiểu nguyên thủy, chẳng hạn như kiểu **int**, **double**, ...

3. Xây dựng sơ đồ lớp (mức thiết kế)

Tên và kiểu

Ví dụ

```
public class BlogAccount
{
    // The two inline attributes from Figure 4-11.
    private String name;
    private URL publicURL;
    // The single attribute by association, given the name 'entries'
    BlogEntries[] entries;
    // ...
}
```

3. Xây dựng sơ đồ lớp (mức thiết kế)

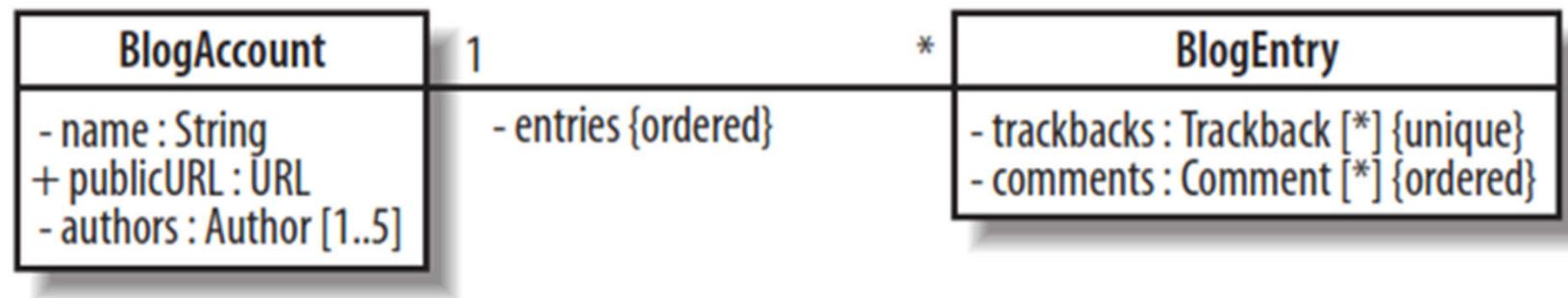
Bội số quan hệ

Một thuộc tính có thể biểu diễn bất kỳ số lượng đối tượng nào theo kiểu dữ liệu của nó.

Bội số quan hệ cho phép đặc tả một thuộc tính thể hiện như một tập hợp các đối tượng, và có thể áp dụng cho thuộc tính trong lớp hoặc thuộc tính quan hệ.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Bộ số quan hệ



Thuộc tính trackbacks: tập các đối tượng, * có thể chứa bất kỳ số lượng đối tượng nào, {unique} các đối tượng này phải là duy nhất.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Bộ số quan hệ

Thuộc tính comments: tập các đối tượng, * có thể chứa bất kỳ số lượng đối tượng nào, {ordered} các đối tượng này phải sắp theo thứ tự.

Thuộc tính author: tập các đối tượng, [1..5] có thể chứa số lượng đối tượng Author từ 1 đến 5.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Bộ số quan hệ

Thuộc tính entries: là một quan hệ giữa lớp BlogAccount và BlogEntry chứa 2 thuộc tính bộ số quan hệ đặc tả ở mỗi đầu.

- Dấu * ở lớp **BlogEntry** là bất kỳ số lượng đối tượng **BlogEntry** nào cũng được lưu ở **entries** bên trong 1 lớp **BlogAccount**.
- Số 1 ở đầu ngược lại chỉ mỗi đối tượng **BlogEntry** ở thuộc tính **entries** liên kết với 1 chỉ và chỉ đối tượng **BlogAccount**.

3. Xây dựng sơ đồ lớp (mức thiết kế)

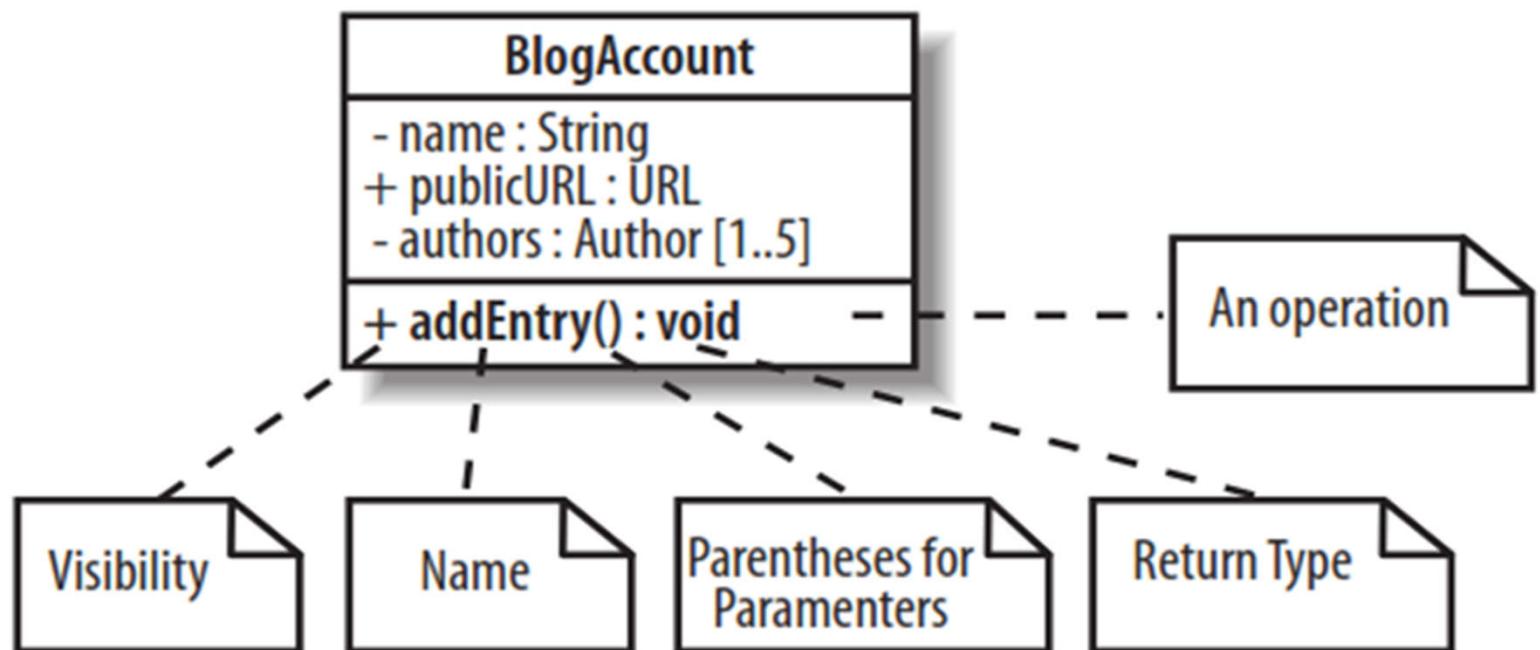
BỘI SỐ QUAN HỆ

Câu hỏi: Liệt kê các dạng bội số quan hệ khác?

3. Xây dựng sơ đồ lớp (mức thiết kế)

Hành vi lớp: Các phương thức

Các phương thức gồm: phạm vi truy cập, tên lớp, cặp dấu ngoặc tròn đóng mở chứa các tham số của phương thức và kiểu trả về.

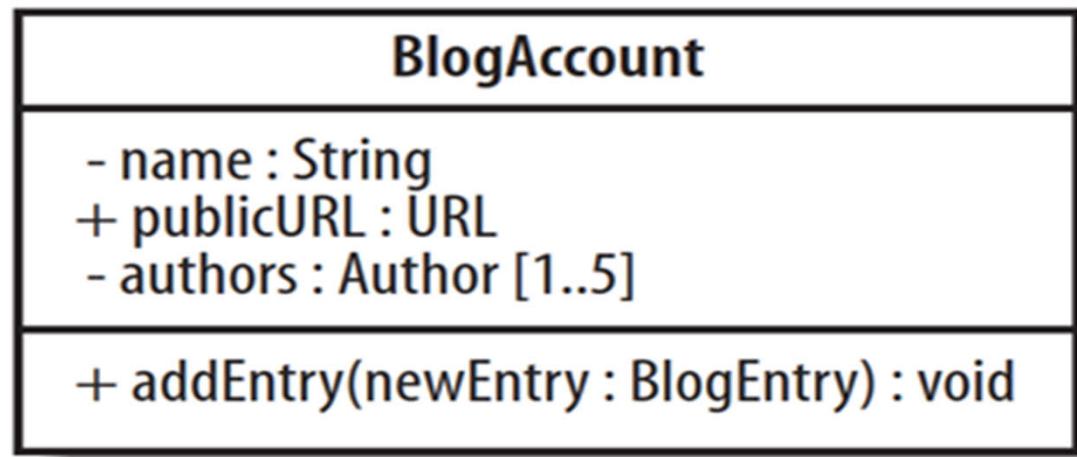


3. Xây dựng sơ đồ lớp (mức thiết kế)

Tham số

Các tham số được dùng để mô tả thông tin đầu vào được cung cấp cho một phương thức cho phép nó hoàn thành công việc của nó.

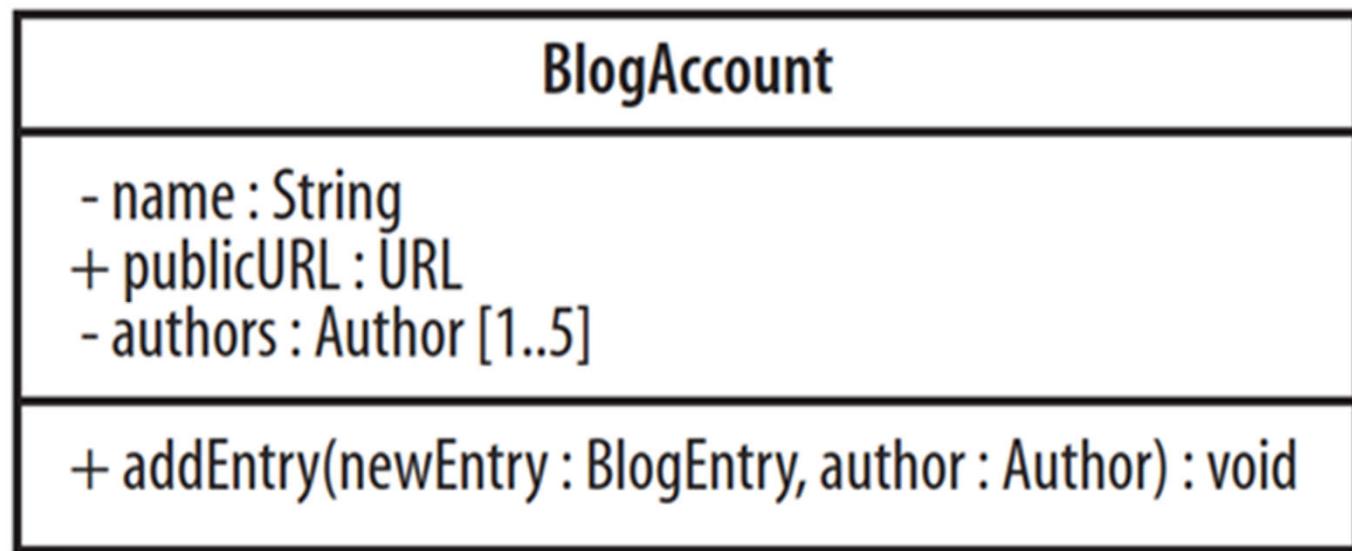
Ví dụ phương thức **addEntry(...)** cần được cung cấp đối tượng **BlogEntry** để thêm vào tài khoản như hình



3. Xây dựng sơ đồ lớp (mức thiết kế)

Tham số

Để thêm 1 tham số author với kiểu dữ liệu lớp **Author**, chúng ta thêm tương tự như hình.



3. Xây dựng sơ đồ lớp (mức thiết kế)

Kiểu trả về

Một phương thức đều có 1 kiểu trả về, **kiểu trả là kiểu sau dấu hai chấm cuối cùng khi mô tả 1 phương thức ở sơ đồ lớp** như hình.

Kiểu trả về có thể là **kiểu nguyên thủy** như int, string, double, ... hay các **kiểu lớp đối tượng** như Author, BlogEntry, List, ...

BlogAccount
- name : String + publicURL : URL - authors : Author [1..5]
+ addEntry(newEntry : BlogEntry, author : Author) : boolean

3. Xây dựng sơ đồ lớp (mức thiết kế)

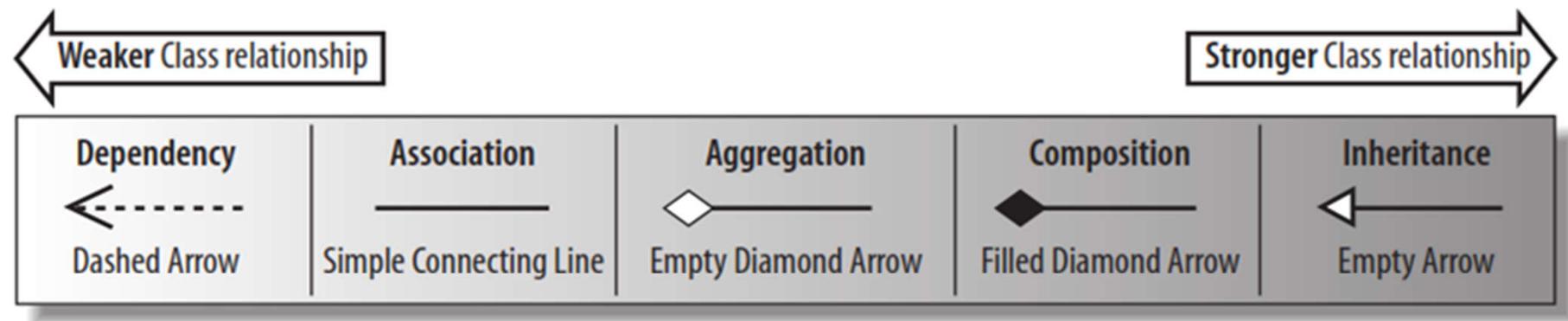
Quan hệ giữa các lớp

Chúng ta có 5 quan hệ lớp bao gồm:

- Quan hệ phụ thuộc (Dependency)
- Quan hệ liên kết (Association)
- Quan hệ kết tập (Aggregation)
- Quan hệ hợp thành (Composition)
- Quan hệ tổng quát hóa hay quan hệ thừa kế (Inheritance, Generalization).

3. Xây dựng sơ đồ lớp (mức thiết kế)

Quan hệ giữa các lớp



When objects of one class **work briefly** with objects of another class

When objects of one class **work with** objects of another class **for some prolonged amount of time**

When one class **owns but shares a reference** to objects of another class

When one class **contains** objects of another class

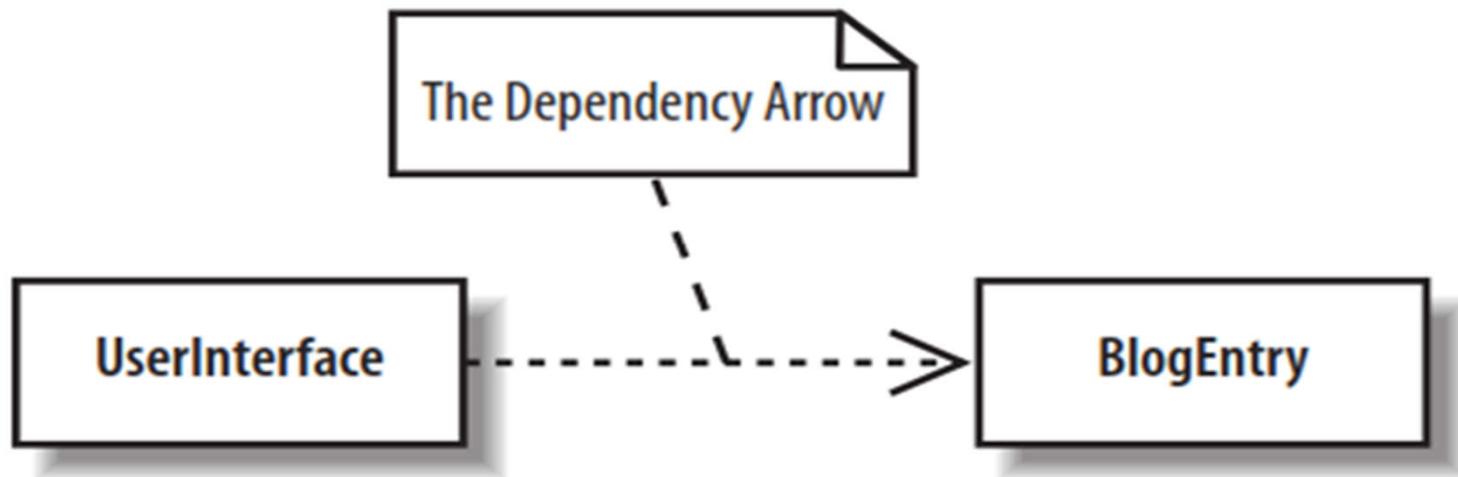
When one class is a **type of** another class

3. Xây dựng sơ đồ lớp (mức thiết kế)

Quan hệ phụ thuộc

Một quan hệ phụ thuộc giữa hai lớp là mối quan hệ khi một lớp cần biết thông tin về lớp khác để sử dụng các đối tượng của lớp đó.

Nếu lớp **UserInterface** cần truy xuất đối tượng lớp **BlogEntry**, mối quan hệ phụ thuộc này được biểu diễn như hình.



3. Xây dựng sơ đồ lớp (mức thiết kế)

Quan hệ phụ thuộc

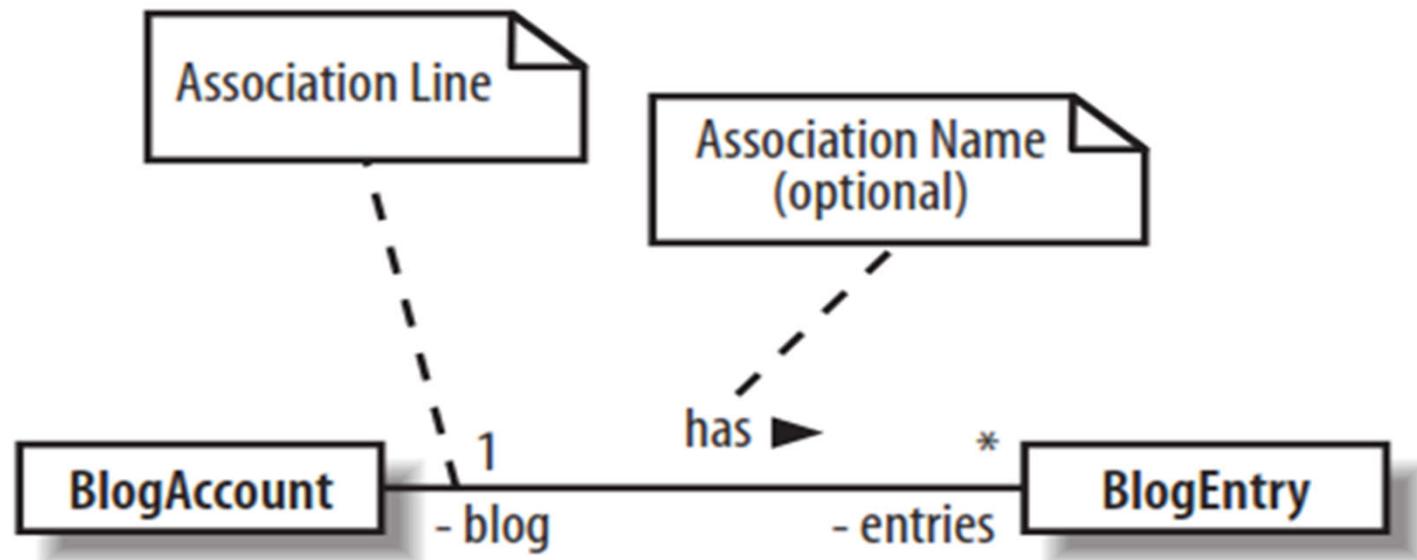
Hai lớp **UserInterface** và **BlogEntry** làm việc cùng nhau để khi hiển thị nội dung các bài viết blog cho người dùng. Hai lớp đối tượng là phụ thuộc nếu chúng làm việc cùng nhau trong thời gian thực thi.

Một quan hệ phụ thuộc chỉ xảy ra khi các đối tượng của 1 lớp có thể làm việc cùng nhau; vì vậy quan hệ phụ thuộc được xem là **quan hệ trực tiếp yếu nhất** tồn tại giữa hai lớp.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Quan hệ liên kết

Quan hệ liên kết lại chứa một **tham khảo (reference)** tới một đối tượng, hay các đối tượng, của lớp khác trong **dạng một thuộc tính**.



3. Xây dựng sơ đồ lớp (mức thiết kế)

Quan hệ liên kết

Lớp **BlogAccount** liên kết với không hoặc nhiều hơn đối tượng của lớp **BlogEntry** (dấu *); lớp **BlogEntry** cũng liên kết với 1 và chỉ 1 đối tượng lớp **BlogAccount** (số 1).

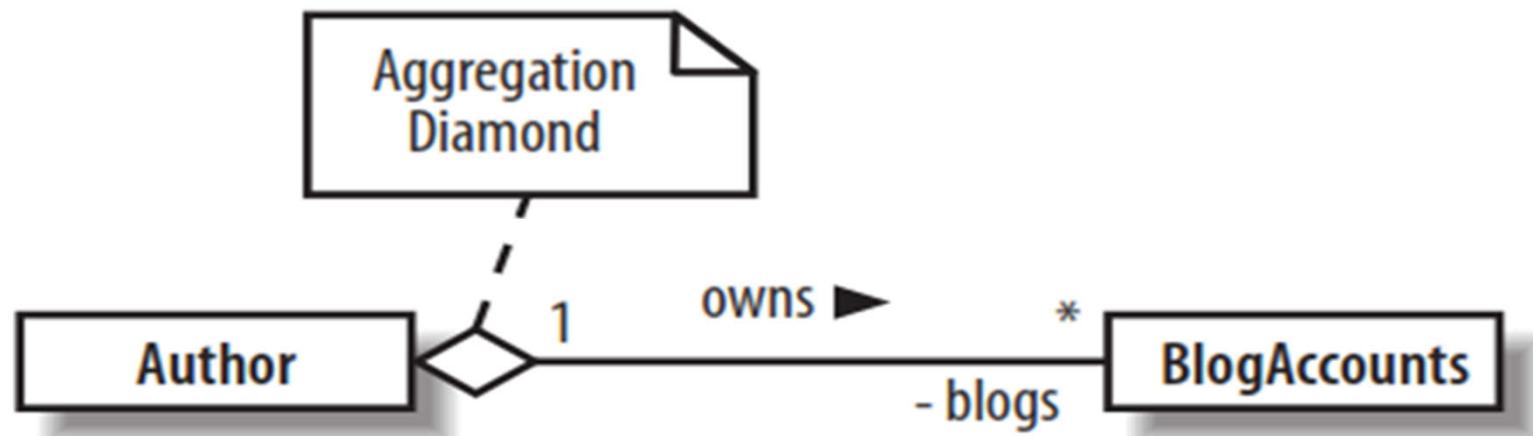
Sự điều hướng thường xảy ra giữa một quan hệ liên kết để mô tả lớp chứa **thuộc tính hỗ trợ quan hệ**.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Quan hệ kết tập

Quan hệ kết tập là quan hệ được dùng để chỉ ra một lớp có thể sở hữu nhưng có thể chia sẻ các đối tượng của một lớp khác.

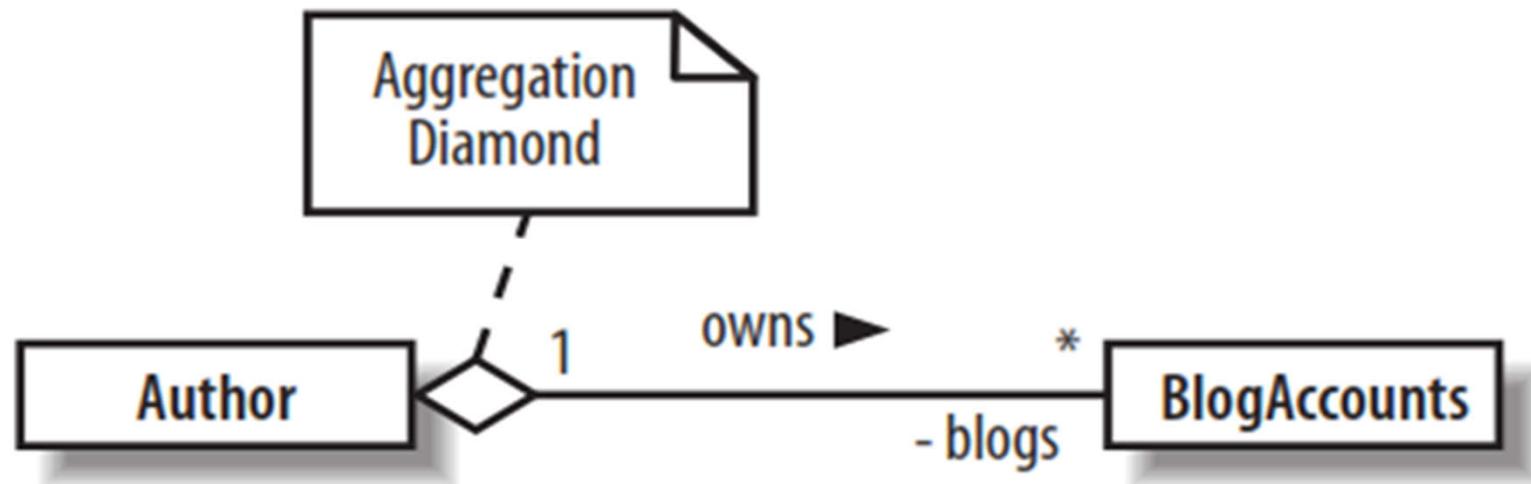
Quan hệ kết tập được mô tả là **một hình kim cương rỗng** ở một đầu của đường thẳng nét liền kết nối giữa hai lớp.



3. Xây dựng sơ đồ lớp (mức thiết kế)

Quan hệ kết tập

Một tác giả sở hữu các blog của mình và có thể chia sẻ chúng với các tác giả khác, tuy nhiên **tác giả đó vẫn là người sở hữu các blog** và có thể **xóa** chúng đi nếu muốn.

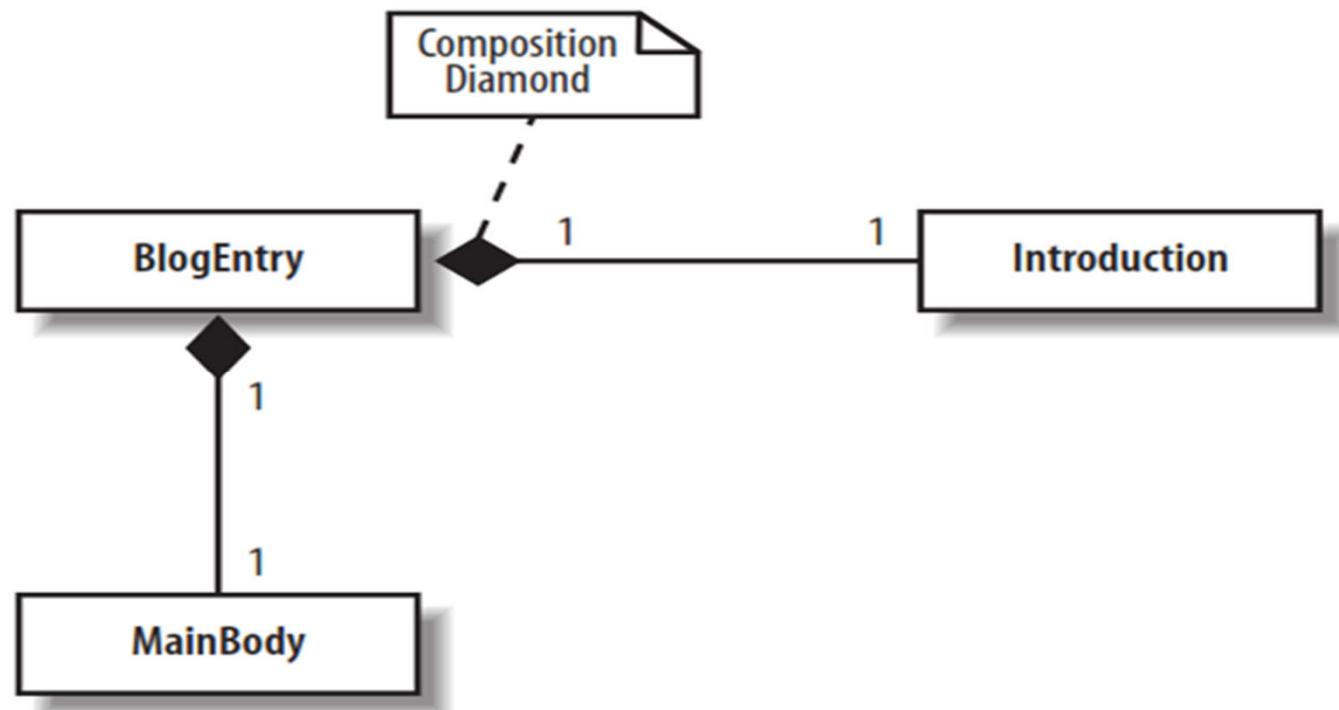


3. Xây dựng sơ đồ lớp (mức thiết kế)

Quan hệ hợp thành

Quan hệ hợp thành mạnh hơn quan hệ kết tập, mặc dù chúng khá giống nhau.

Quan hệ hợp thành dùng **hình kim cương đen** trên đường thẳng nét liền nối giữa hai lớp.

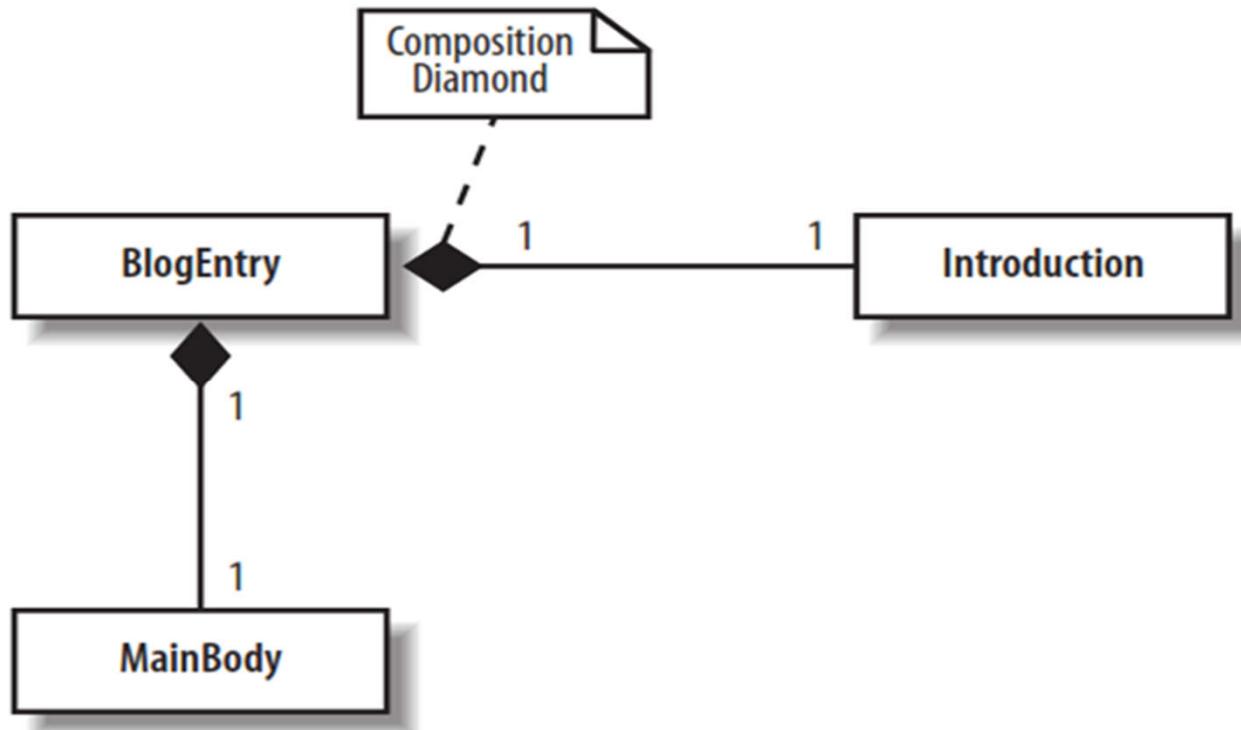


3. Xây dựng sơ đồ lớp (mức thiết kế)

Quan hệ hợp thành

Lời giới thiệu và phần thân chính là các phần cấu thành nên 1 blog và không chia sẻ với các thành phần khác. Nếu blog bị xóa, các thành phần này cũng bị xóa.

Quan hệ hợp thành chính là hợp các thành phần con để tạo thành một lớp.



3. Xây dựng sơ đồ lớp (mức thiết kế)

Quan hệ tổng quát hóa

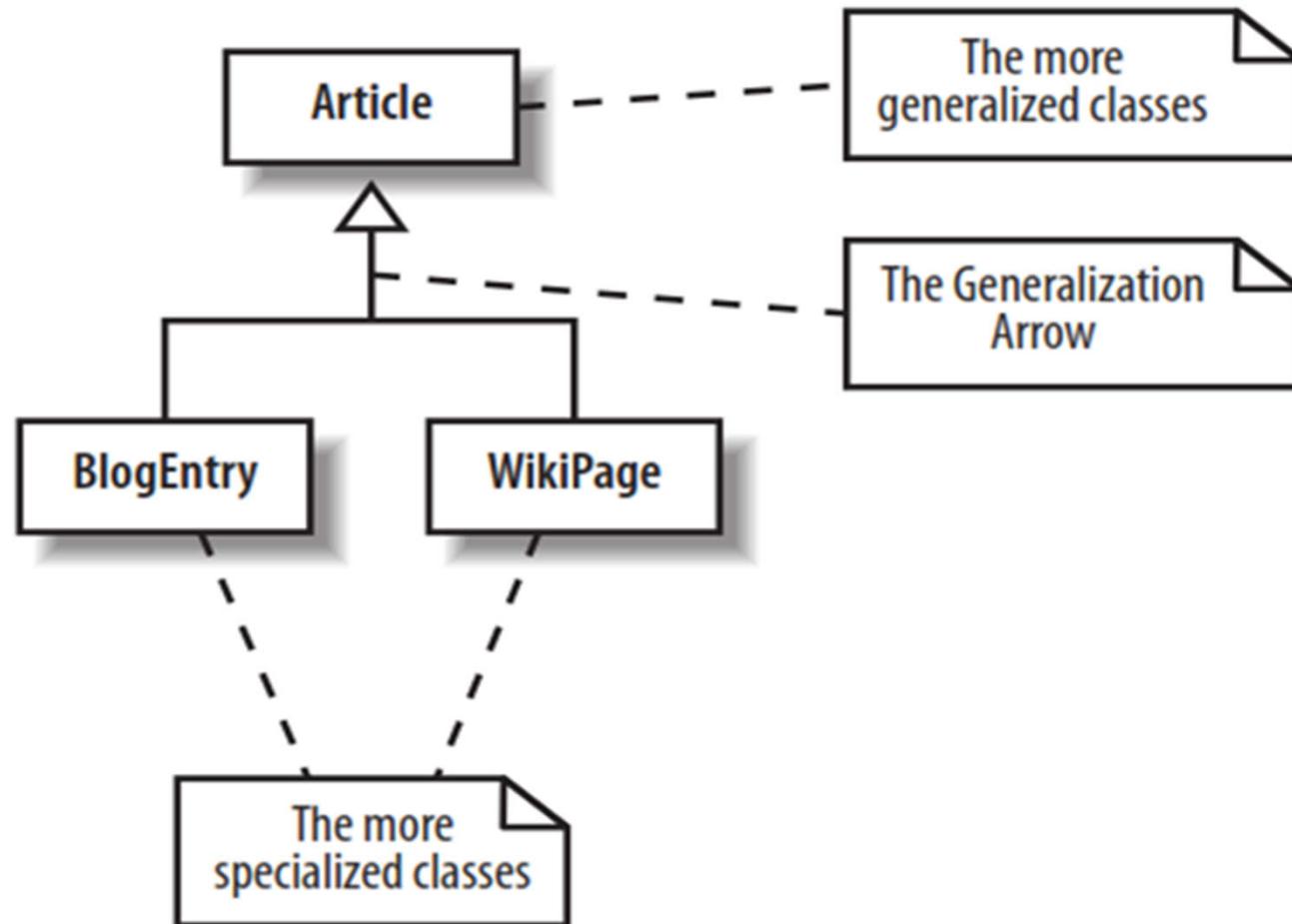
Quan hệ tổng quát hóa (quan hệ kế thừa) mô tả một lớp là một dạng của lớp nào đó.

Thuật ngữ "có một" (has a) và "một dạng của" (a type of) là cách một mối quan hệ giữa hai lớp là quan hệ kết tập hay quan hệ tổng quát hóa.

- Nếu một lớp là 1 phần của 1 đối tượng của lớp khác, mối quan hệ này có thể là: quan hệ liên kết, quan hệ kết tập, quan hệ hợp thành.
- Nếu một lớp là 1 dạng của lớp khác thì đây chính là **quan hệ tổng quát hóa**.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Quan hệ tổng quát hóa



3. Xây dựng sơ đồ lớp (mức thiết kế)

Quan hệ tổng quát hóa

Lớp **Article** (bài viết) được xem là lớp cha (lớp cơ bản) hay siêu lớp.

Các lớp chuyên biệt hơn sẽ kế thừa hai lớp **BlogEntry**, **WikiPage** và được xem là con hay lớp kế thừa.

Lớp kế thừa sẽ nhận toàn bộ thuộc tính và phương thức của lớp tổng quát hóa (nếu phạm vi truy cập của các phần tử này là public hoặc protected) và có thể tự thêm các thuộc tính và phương thức của riêng mình.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Ràng buộc

Để hạn chế cách thức một lớp có thể thực thi chúng ta có thể dùng các ràng buộc.

Các ràng buộc này thường không xuất hiện ở một sơ đồ UML đơn giản.

Người ta thường dùng ngôn ngữ OCL (Object Constraint Language) để định nghĩa các ràng buộc này.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Ràng buộc

Có 3 dạng ràng buộc dùng OCL (Object Constraint Language):

- *Ràng buộc bắt biến*: là một ràng buộc luôn đúng, nếu không hệ thống sẽ trong trạng thái không hợp lệ. Ràng buộc bắt biến được định nghĩa cho các thuộc tính lớp.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Ràng buộc

Có 3 dạng ràng buộc dùng OCL (Object Constraint Language):

- *Ràng buộc tiền điều kiện*: là một ràng buộc được định nghĩa trên 1 phương thức và được kiểm tra trước khi phương thức thực thi. Ràng buộc tiền điều kiện thường được dùng để kiểm tra các tham số đầu vào của 1 phương thức.

3. Xây dựng sơ đồ lớp (mức thiết kế)

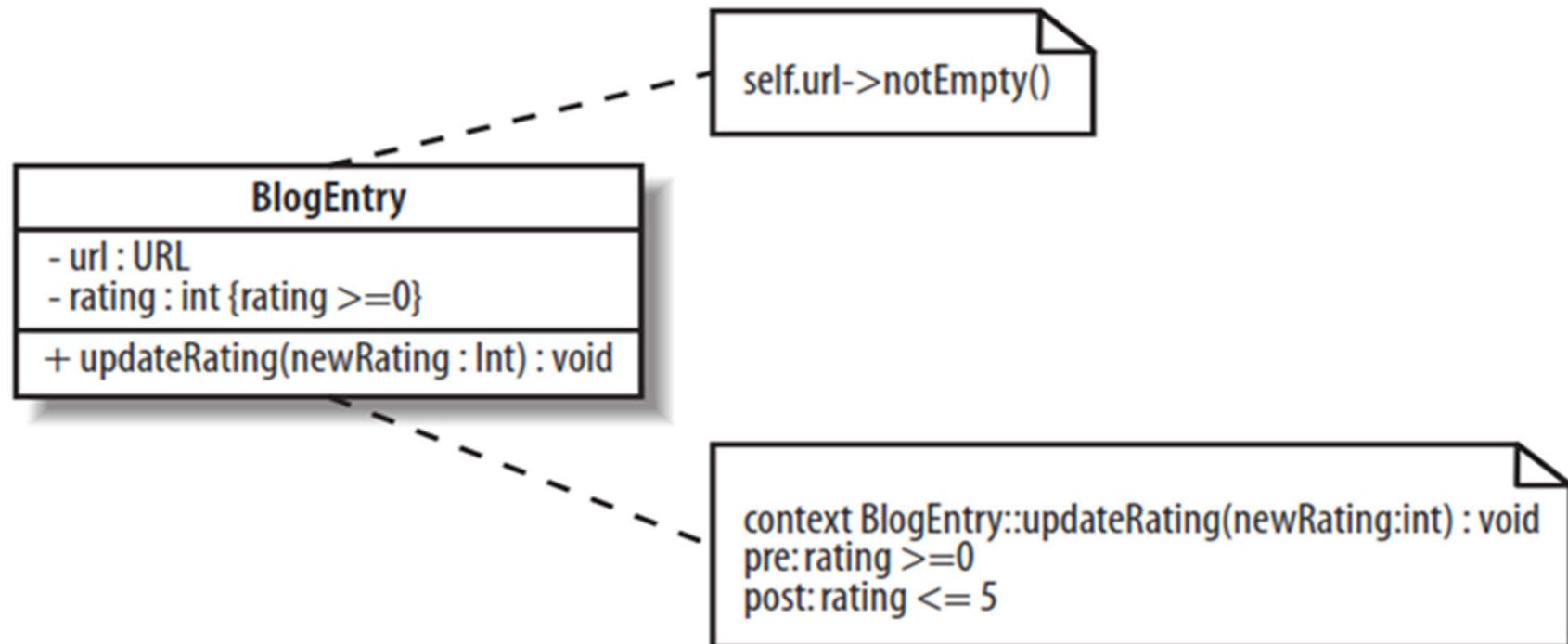
Ràng buộc

Có 3 dạng ràng buộc dùng OCL (Object Constraint Language):

- *Ràng buộc hậu điều kiện*: là một ràng buộc định nghĩa trên 1 phương thức và được kiểm tra sau khi phương thức thực thi. Ràng buộc hậu điều kiện thường dùng để mô tả cách các giá trị bị thay đổi bởi một phương thức.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Ràng buộc



3. Xây dựng sơ đồ lớp (mức thiết kế)

Ràng buộc

Các ràng buộc bắt biến:

- Thuộc tính **url** bị ràng buộc không rỗng
- Thuộc tính **rating** bị ràng buộc phải lớn hơn hoặc bằng 0

Ràng buộc hậu điều kiện:

- Phương thức **updateRating(...)**, **rating <=5**

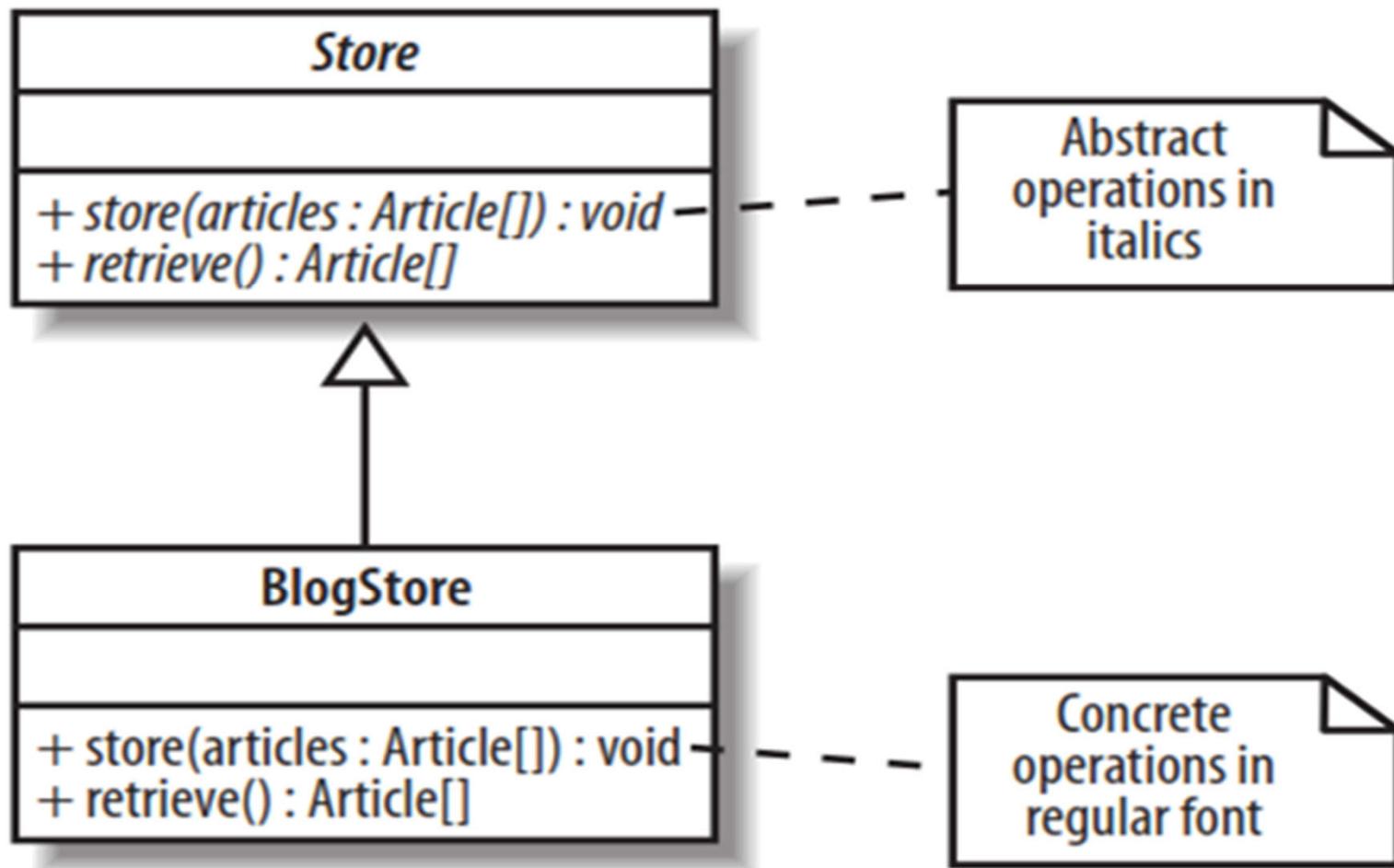
3. Xây dựng sơ đồ lớp (mức thiết kế)

Lớp trừu tượng

Nếu dùng tổng quát hóa để định nghĩa một lớp chung có thể tái sử dụng, và không cần phải thực thi tất cả phương thức của lớp đó. Lớp đó gọi là **lớp trừu tượng**.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Lớp trừu tượng



3. Xây dựng sơ đồ lớp (mức thiết kế)

Lớp trừu tượng

Một lớp **Store** chứa 2 phương thức **store(...)** và **retrieve(...)** dùng để lưu trữ và lấy thông tin các bài viết.

Nếu thiết lập lớp Store này là lớp trừu tượng thì chúng ta không cần viết mã thực thi 2 lớp này, thay vào đó để nội dung trống.

Các lớp nào kế thừa lớp Store này sẽ chịu trách nhiệm thực thi nội dung 2 phương thức.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Giao diện (Interfaces)

Nếu muốn định nghĩa các phương thức mà các lớp cụ thể (concrete class) có thể thực thi, nhưng không dùng trừu tượng thì chỉ có thể dùng **giao diện**.

Một giao diện là 1 tập các phương thức và có không có các thực thi tương ứng, tương tự như lớp trừu tượng chỉ chứa các phương thức trừu tượng.

3. Xây dựng sơ đồ lớp (mức thiết kế)

Giao diện (Interfaces)

Một giao diện **EmailSystem** dùng kiểu ký hiệu mẫu hoặc một ký hiệu vòng tròn trong UML.



Stereotype Notation

Or

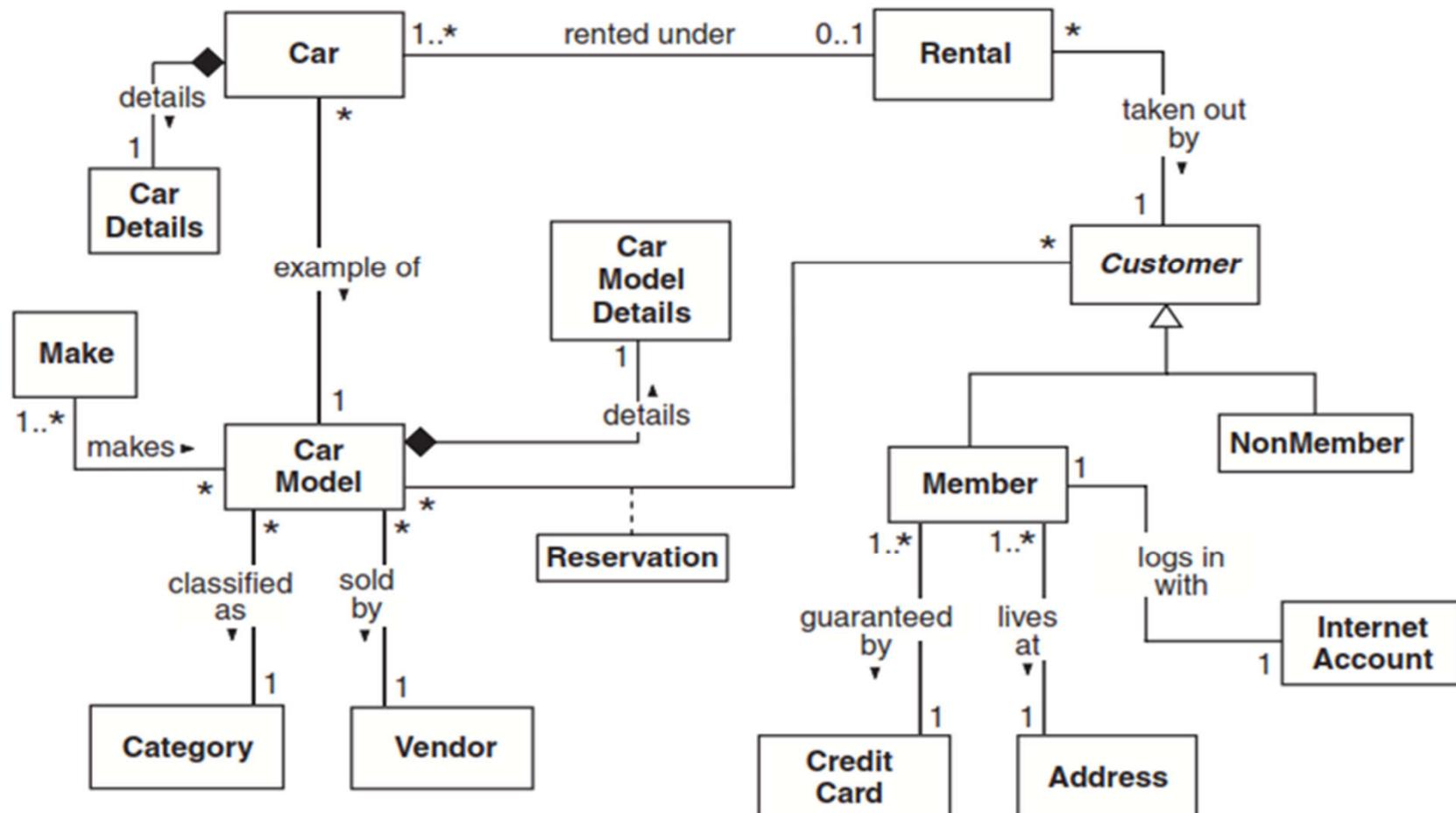


"Ball" Notation

3. Xây dựng sơ đồ lớp (mức thiết kế)

Mô hình hóa sơ đồ lớp cho hệ thống phần mềm

Sơ đồ lớp mức độ phân tích tổng quát của hệ thống phần mềm thuê xe hơi.



3. Xây dựng sơ đồ lớp (mức thiết kế)

Mô hình hóa sơ đồ lớp cho hệ thống phần mềm

Tùy theo quy mô của dự án hệ thống phần mềm, chúng ta xây dựng sơ đồ lớp các mức khác nhau:

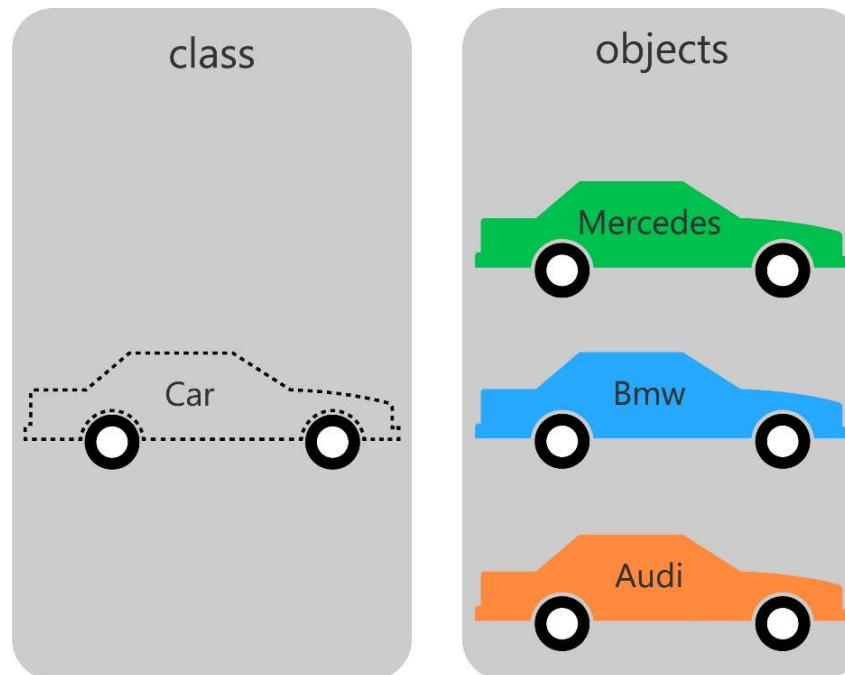
- **Xây dựng mô hình tổng quát:** mô hình chỉ chứa tên các module lớn hoặc tên lớp và mối quan hệ giữa chúng
- **Xây dựng mô hình chi tiết:** mô hình chứa tên lớp, thuộc tính, các mối quan hệ, các ràng buộc, ... nhằm giúp lập trình viên hình dung tổng thể để tiến hành xây dựng hệ thống phần mềm

3. Xây dựng sơ đồ lớp (mức thiết kế)

Mô hình hóa sơ đồ lớp cho hệ thống phần mềm

Các bước chính:

- Dựa vào nội dung yêu cầu, xác định các lớp
 - Một lớp thường là các đối tượng có đặc điểm chung.

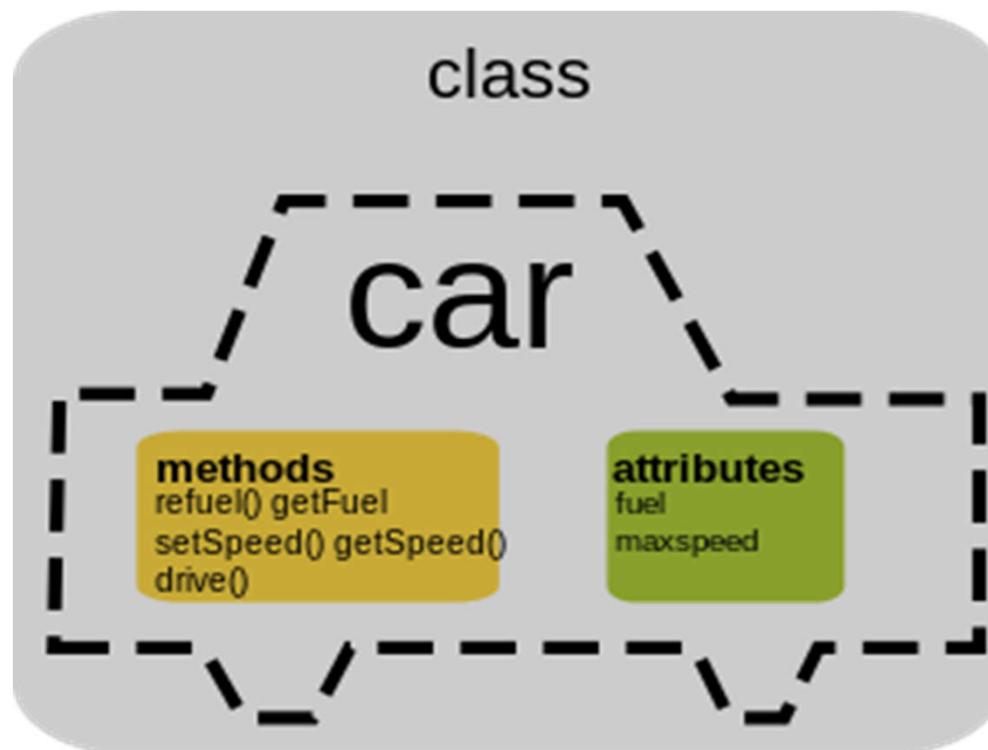


3. Xây dựng sơ đồ lớp (mức thiết kế)

Mô hình hóa sơ đồ lớp cho hệ thống phần mềm

Các bước chính:

- Mỗi lớp xác định các thuộc tính, phương thức

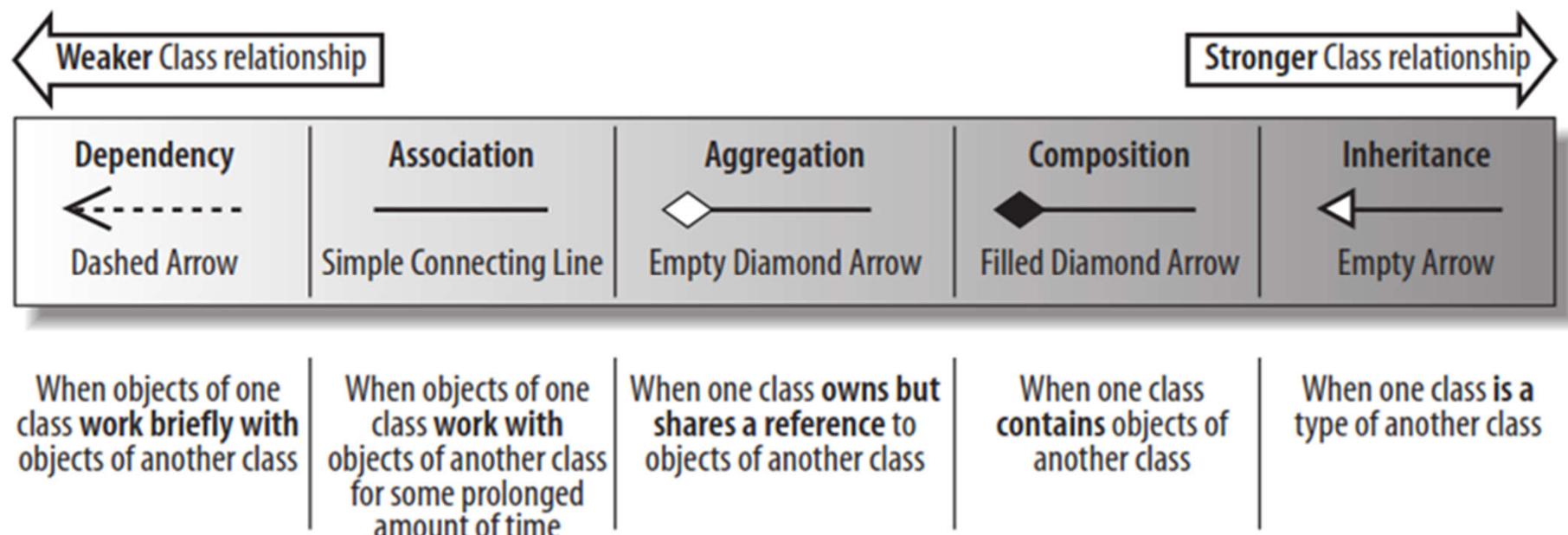


3. Xây dựng sơ đồ lớp (mức thiết kế)

Mô hình hóa sơ đồ lớp cho hệ thống phần mềm

Các bước chính:

- Xác định các mối quan hệ giữa các lớp kèm theo các thuộc tính quan hệ.



4. Thiết kế dữ liệu

Mục tiêu chính của thiết kế dữ liệu là mô tả cách thức tổ chức lưu trữ các dữ liệu của phần mềm. Có hai dạng lưu trữ chính mà người thiết kế cần phải cân nhắc và lựa chọn. Đó là:

- Lưu trữ dưới dạng tập tin
- Lưu trữ dưới dạng cơ sở dữ liệu



4. Thiết kế dữ liệu

Lưu trữ dưới dạng tập tin thích hợp với một số phần mềm đặc thù (cờ tướng, trò chơi nhỏ, v.v.). Các phần mềm này chú trọng rất nhiều vào xử lý, hình thức giao diện và không chú trọng nhiều đến việc lưu trữ lại các thông tin được tiếp nhận trong quá trình sử dụng phần mềm.

Lưu trữ dạng cơ sở dữ liệu rất thông dụng và tỏ ra hiệu quả hơn so với lưu trữ trực dữ liệu tập tin.

4. Thiết kế dữ liệu

Kết quả của thiết kế dữ liệu

Cách thức tổ chức lưu trữ dữ liệu được mô tả bằng 2 loại thông tin:

- **Thông tin tổng quát:** tổng quan về các thành phần lưu trữ
 - Danh sách các bảng dữ liệu
 - Danh sách các liên kết
- **Thông tin chi tiết**
 - Danh sách các thuộc tính của từng thành phần
 - Danh sách các miền giá trị toàn vẹn

4. Thiết kế dữ liệu

Kết quả của thiết kế dữ liệu

Sơ đồ logic là sơ đồ cho phép thể hiện hệ thống các bảng dữ liệu cùng với mối quan hệ liên kết giữa chúng.

Bảng (hình chữ nhật)

Tên bảng

Liên kết (mũi tên, xác định duy nhất một)



4. Thiết kế dữ liệu

Kết quả của thiết kế dữ liệu



Một phần tử bảng A sẽ xác định duy nhất một phần tử bảng B, ngược lại một phần tử bảng B có thể tương ứng với nhiều phần tử bảng A.

4. Thiết kế dữ liệu

Kết quả của thiết kế dữ liệu

Ví dụ hệ thống quản lý sách thư viện



Lưu trữ các dữ liệu trong phần mềm quản lý mượn sách của thư viện được tổ chức thành ba bảng: DOC_GIA, MUON_SACH, SACH cùng với hai liên kết giữa chúng.

4. Thiết kế dữ liệu

Kết quả của thiết kế dữ liệu

Bảng thuộc tính: mô tả chi tiết thành phần trong sơ đồ logic

STT	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
...

4. Thiết kế dữ liệu

Kết quả của thiết kế dữ liệu

Bảng thuộc tính

Bảng thuộc tính mô tả chi tiết các thành phần dữ liệu và dùng trong báo cáo về thiết kế dữ liệu.

Cách trình bày gọn hơn là **dạng lược đồ quan hệ** gồm:

- Tên bảng và các thuộc tính đi kèm
- Các thuộc tính khóa chính được gạch chân nét liền
- Các thuộc tính khóa ngoại được gạch chân nét đứt.

4. Thiết kế dữ liệu

Kết quả của thiết kế dữ liệu

Ví dụ

DOC_GIA (MaDG, LoaiDG, HoTen, NgaySinh, NgayLapThe,
GioiTinh, DiaChi, SoDT)

SACH (MaSach, TenSach, TheLoai, NgayNhap, TacGia,
NhaXB, NamXB, NgonNgu)

MUON_SACH (MaDG, MaSach, NgayMuon, NgayTra)

4. Thiết kế dữ liệu

Quá trình thiết kế

Quá trình thiết kế dữ liệu bao gồm 3 bước:

- **Thiết kế với tính đúng đắn**

- Đảm bảo đầy đủ, chính xác ngữ nghĩa các thông tin công việc
- Các thông tin cho yêu cầu chất lượng sẽ không được xét đến

- **Thiết kế với yêu cầu chất lượng**

- Đảm bảo tính đúng đắn và thỏa mãn thêm các yêu cầu khác.
- Bảo đảm tính đúng đắn khi cải tiến sơ đồ logic

- **Thiết kế với yêu cầu hệ thống**

- Đảm bảo tính đúng đắn và các yêu cầu chất lượng khác nhưng thỏa mãn thêm các yêu cầu hệ thống.

4. Thiết kế dữ liệu

Thiết kế dữ liệu với tính đúng đắn

Các bước thực hiện:

- Bước 1: Chọn một yêu cầu và xác định sơ đồ logic.
- Bước 2: Bổ sung thêm một yêu cầu và xem xét lại sơ đồ
 - Nếu sơ đồ logic vẫn đáp ứng được => bước 3
 - Nếu sơ đồ logic không đáp ứng được thì bổ sung vào sơ đồ thuộc tính mới (ưu tiên 1) hoặc thành phần mới (ưu tiên 2) cùng với các thuộc tính và liên kết tương ứng.
- Bước 3: Quay lại bước 2 cho đến khi đã xem xét đầy đủ các yêu cầu.

4. Thiết kế dữ liệu

Thiết kế dữ liệu với tính đúng đắn

Các lưu ý:

- Với mỗi yêu cầu phải xác định rõ cần lưu trữ các thông tin gì và tìm cách bổ sung các thuộc tính để lưu trữ các thông tin này.
- Chỉ xem xét tính đúng đắn.

4. Thiết kế dữ liệu

Thiết kế dữ liệu với tính đúng đắn

Các lưu ý:

- Cần chọn các yêu cầu theo thứ tự từ đơn giản đến phức tạp (thông thường yêu cầu tra cứu là đơn giản nhất).
- Với yêu cầu phức tạp có thể phải bổ sung vào sơ đồ logic nhiều thành phần mới. Khóa của các thành phần phải dựa trên ngữ nghĩa tương ứng trong thế giới thực

4. Thiết kế dữ liệu

Thiết kế dữ liệu với tính đúng đắn

Ví dụ: xét phần mềm với các yêu cầu sau

1. Lập thẻ độc giả.
2. Nhận sách.
3. Cho Mượn/Trả sách.

4. Thiết kế dữ liệu

Thiết kế dữ liệu với tính đúng đắn

Ví dụ:

Bước 1: Xét yêu cầu 1

- Lập bảng DOC_GIA.
- Chi tiết bảng: **DOC_GIA(MaDG, LoaiDG, HoTen, NgaySinh, NgayLapThe, GioiTinh, DiaChi, SoDT)**
- Sơ đồ logic:

DOC_GIA

4. Thiết kế dữ liệu

Thiết kế dữ liệu với tính đúng đắn

Ví dụ:

Bước 2: Xét yêu cầu 1, 2 với thông tin mới: Tên sách, Tác giả, Nhà xuất bản, Năm xuất bản, Thể loại, Ngày nhập, Ngôn ngữ.

- Tạo thêm bảng SACH.
- Chi tiết bảng: **SACH(MaSach, TenSach, TheLoai, NgayNhap, TacGia, NhaXB, NamXB, NgonNgu)**
- Sơ đồ logic

DOC_GIA

MUON_SACH

4. Thiết kế dữ liệu

Thiết kế dữ liệu với tính đúng đắn

Ví dụ:

Bước 3: Xét yêu cầu 1, 2, 3 với thông tin mới: Ngày mượn, Ngày trả, Tiền phạt.

- Tạo thêm bảng MUON_SACH.
- Chi tiết bảng MUON_SACH(MaDG, MaSach, NgayMuon, NgayTra, TienPhat)
- Sơ đồ logic



4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính tiến hóa

Tính tiến hóa cho phép thay đổi các mô tả quy định khi đang sử dụng phần mềm.

Để bảo đảm tính tiến hóa, sơ đồ logic sẽ còn bổ sung cập nhật lại nhiều thành phần qua các bước thiết kế chi tiết.

Trong thiết kế dữ liệu, chúng ta phải xem xét đến các thuộc tính có giá trị rời rạc.

4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính tiến hóa

Thuộc tính có giá trị rời rạc là các thuộc tính mà miền giá trị chỉ bao gồm một số giá trị nhất định.

Các giá trị này thông thường thuộc về tập hợp có độ biến động rất ít trong quá trình sử dụng phần mềm

4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính tiến hóa

Ví dụ:

Trong bảng **DOC_GIA**: LoaiDG là thuộc tính có khả năng thêm mới là rất thấp.

Trong bảng **SACH**: NgonNgu là thuộc tính có khả năng thêm mới là rất thấp

Để đảm bảo tính tiến hóa, chúng ta sẽ tiến hành tách các thuộc tính rời rạc này thành các thành phần độc lập.

4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính tiến hóa

Ví dụ:

Kết quả khi tách các thuộc tính rời rạc:



4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính tiến hóa

Ví dụ:

Chi tiết các bảng:

DOC_GIA(MaDG, MaLDG, HoTen, NgaySinh, NgayLapThe, GioiTinh, DiaChi, SoDT)

LOAI_DG(MaLDG, TenLDG, GhiChu)

SACH(MaSach, TenSach, TacGia, TheLoai, MaNN, NhaXB, NamXB, NgayNhap)

NGON_NGU(MaNN, TenNN)

MUON_SACH(MaDG, MaSach, NgayMuon, NgayTra, TienPhat)

4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính hiệu quả về tốc độ xử lý

Phạm vi xem xét:

- Xem xét việc tăng tốc độ thực hiện phần mềm bằng cách bổ sung thêm các thuộc tính vào các bảng dùng lưu trữ các thông tin đã tính toán trước
- Các thông tin này phải được tự động cập nhật khi có bất kỳ thay đổi thông tin gốc liên quan

4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính hiệu quả về tốc độ xử lý

Các bước:

- Bước 1: Chọn một yêu cầu và xem xét cần bổ sung thông tin gì trên bộ nhớ phụ để tăng tốc độ thực hiện của xử lý liên quan (các thông tin xử lý phải đọc mà không cần thực hiện việc tính toán)
- Bước 2: Quay lại bước 1 cho đến khi đã xem xét đầy đủ các yêu cầu

4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính hiệu quả về tốc độ xử lý

Các lưu ý:

- Sau mỗi bước, phải lập bảng danh sách các thuộc tính tính toán cùng với thông tin liên quan
- Nếu thông tin gốc hay bị thay đổi, việc bổ sung thuộc tính tính toán để tăng tốc độ thực hiện sẽ mất ý nghĩa.
- Việc tăng tốc độ truy xuất có thể sẽ dẫn đến việc lưu trữ không tối ưu.
- Thứ tự xem xét các yêu cầu theo thứ tự từ đầu đến cuối

4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính hiệu quả về tốc độ xử lý

Ví dụ: Bổ sung thêm các yêu cầu

1. Mỗi thẻ độc giả có thời hạn 03 năm.
2. Cho biết trạng thái sách là đang được mượn hay không.

Sơ đồ logic ví dụ trước:



4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính hiệu quả về tốc độ xử lý

Bước 1: Xét yêu cầu 1, thêm thuộc tính Ngày hết hạn cho bảng DOC_GIA:

**DOC_GIA(MaDG, LoaiDG, HoTen, NgaySinh, NgayLapThe,
NgayHetHan, GioiTinh, DiaChi, SoDT)**

Bước 2: Xét yêu cầu 2 thêm thuộc tính Tình trạng mượn
cho bảng SACH

**SACH(MaSach, TenSach, TacGia, TheLoai, NgonNgu,
NhaXB, NamXB, NgayNhap, TinhTrangMuon)**

4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính hiệu quả về lưu trữ thông tin

Tính hiệu quả xem xét việc lưu trữ có tối ưu hay không.

Vấn đề đặt ra: xây dựng sơ đồ logic nhằm lưu trữ đầy đủ thông tin yêu cầu + dung lượng lưu trữ ít nhất có thể.

Cần đặc biệt chú ý các thành phần dữ liệu tương ứng sẽ được phát sinh nhiều theo thời gian.

4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính hiệu quả về lưu trữ thông tin

Các bước:

Bước 1: Lập danh sách bảng cần tối ưu hóa việc lưu trữ

- Xem xét và xác định các công việc có tần suất thực hiện thường xuyên và bổ sung vào danh sách các bảng được sử dụng tương ứng của công việc này.
- Xem xét các bảng mà khóa của bảng bao gồm nhiều thuộc tính và bổ sung bảng này vào danh sách được chọn

4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính hiệu quả về lưu trữ thông tin

Các bước:

Bước 2: Tối ưu hóa lưu trữ các bảng có khối lượng dữ liệu
lưu trữ lớn = việc tối ưu hóa lưu trữ từng thuộc tính
trong bảng

- Xác định các thuộc tính mà việc lưu trữ chưa tối ưu. Ưu tiên xem xét các thuộc tính có kiểu chuỗi.
- Tối ưu hóa việc lưu trữ tùy theo từng trường hợp cụ thể

4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính hiệu quả về lưu trữ thông tin

Các bước:

Bước 3: Tối ưu hóa các bảng mà khóa của bảng bao gồm nhiều thuộc tính

- Phân rã bảng đang xét thành hai bảng. Trong đó, một bảng chứa các thuộc tính mà giá trị được lặp lại nhiều lần trong cùng một lần thực hiện công việc tương ứng trong thế giới thực. Bảng này cần có khóa riêng (sẽ được bảng còn lại sử dụng để tham chiếu đến)

4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính hiệu quả về lưu trữ thông tin

Lưu ý:

- Việc phân rã giúp cho việc lưu trữ được tối ưu hơn, tuy nhiên tốc độ truy xuất có thể sẽ chậm hơn và việc thực hiện xử lý sẽ khó khăn hơn (do thuật giải sẽ trở nên phức tạp hơn).

=> Cho nên cần cân nhắc trước khi thực hiện việc phân rã

4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính hiệu quả về lưu trữ thông tin

Ví dụ:

Bước 1: Lập danh sách các bảng cần xem xét để tối ưu hóa lưu trữ

MUON_SACH(MaDG, MaSach, NgayMuon, NgayTra, TienPhat): việc mượn sách có tần suất thực hiện thường xuyên, khóa của bảng MUON_SACH gồm có 02 thuộc tính

4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính hiệu quả về lưu trữ thông tin

Ví dụ:

Bước 2: Xác định các thuộc tính mà việc lưu trữ chưa tối ưu

Bảng **SACH**(MaSach, TenSach, TheLoai, NgayNhap, TacGia, NhaXB, NamXB, NgonNgu): có các thuộc tính lưu trữ chưa tối ưu: TheLoai, TacGia, NhaXB.

4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính hiệu quả về lưu trữ thông tin

Ví dụ:

Bước 3:

Phân rã bảng **MUON_SACH** thành 2 bảng: **MUON_SACH** và **CT_MUON** (chi tiết mượn):

- **MUON_SACH**(MaCTM, TienPhat)
- **CT_MUON**(MaCTM, MaDG, MaSach, NgayMuon, NgayTra)

4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính hiệu quả về lưu trữ thông tin

Ví dụ:

Bước 3:

Phân rã bảng **SACH** thành các bảng: **SACH, THE_LOAI, TAC_GIA, NHA_XB**:

- **SACH**(MaSach, TenSach, MaTG, MaTL, MaNN, MaNXB, NamXB, NgayNhap, TinhTrangMuon)
- **TAC_GIA**(MaTG, HoTen, DiaChi, SoDT)
- **THE_LOAI**(MaTL, TenTL)
- **NHA_XB**(MaNXB, TenNXB, NgayThanhLap)

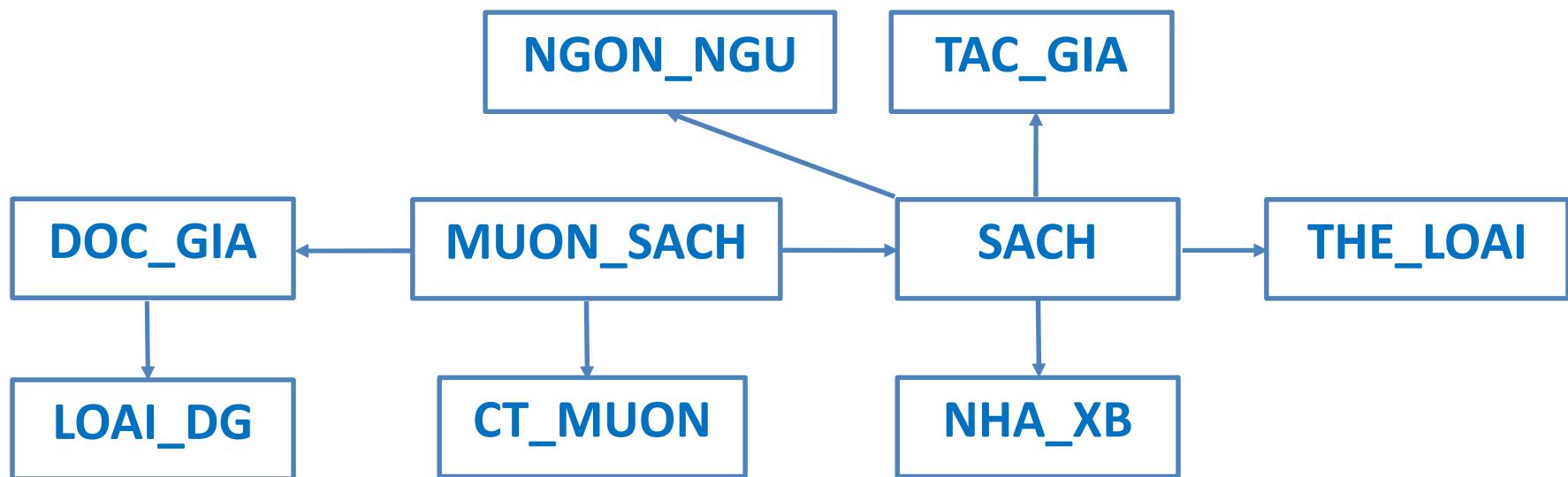
4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu chất lượng

Tính hiệu quả về lưu trữ thông tin

Ví dụ:

Sơ đồ logic



4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu hệ thống

Cải tiến phần mềm bằng việc thêm vào các yêu cầu hệ thống:

- Phân quyền
- Cấu hình phần cứng
- Môi trường phần mềm
- Nhiều yêu cầu khác

=> Đảm bảo tính đúng đắn và các yêu cầu chất lượng

4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu hệ thống

Ví dụ: Yêu cầu phân quyền cho hệ thống phần mềm quản lý thư viện.

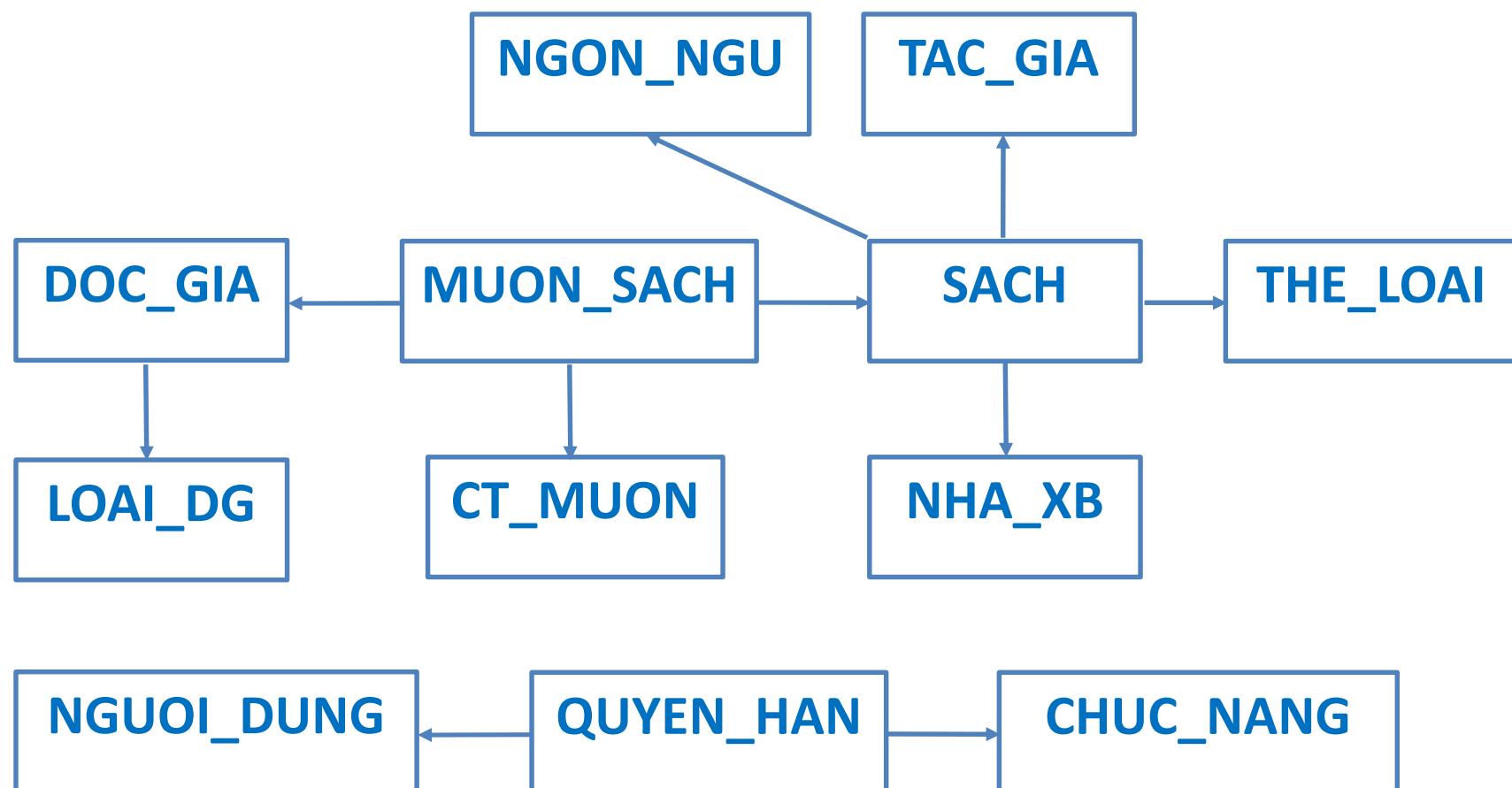
Bổ sung các bảng:

- **NGUOI_DUNG(MaND, TenND)**
- **CHUC_NANG(MaCN, TenCN, GhiChu)**
- **QUYEN_HAN(MaND, MaCN)**

4. Thiết kế dữ liệu

Thiết kế dữ liệu với yêu cầu hệ thống

Ví dụ: Sơ đồ logic



5. Thiết kế giao diện

Khái niệm

Thiết kế giao diện là quá trình xây dựng giao diện để người dùng có thể tương tác với chương trình ứng dụng thông qua các chức năng, đồng thời thông tin từ chương trình có thể được biểu diễn hiển thị ở giao diện.

Giao diện chương trình được thiết kế để sử dụng ở màn hình các thiết bị như máy tính, laptop, các thiết bị di động, hay nhiều thiết bị điện tử khác.

5. Thiết kế giao diện

Người dùng

Người dùng không quan tâm đến cấu trúc bên trong hệ thống mà chỉ **quan tâm đến giao diện sử dụng**.

Họ dựa vào **giao diện để đánh giá một cách chủ quan về chất lượng hệ thống**.

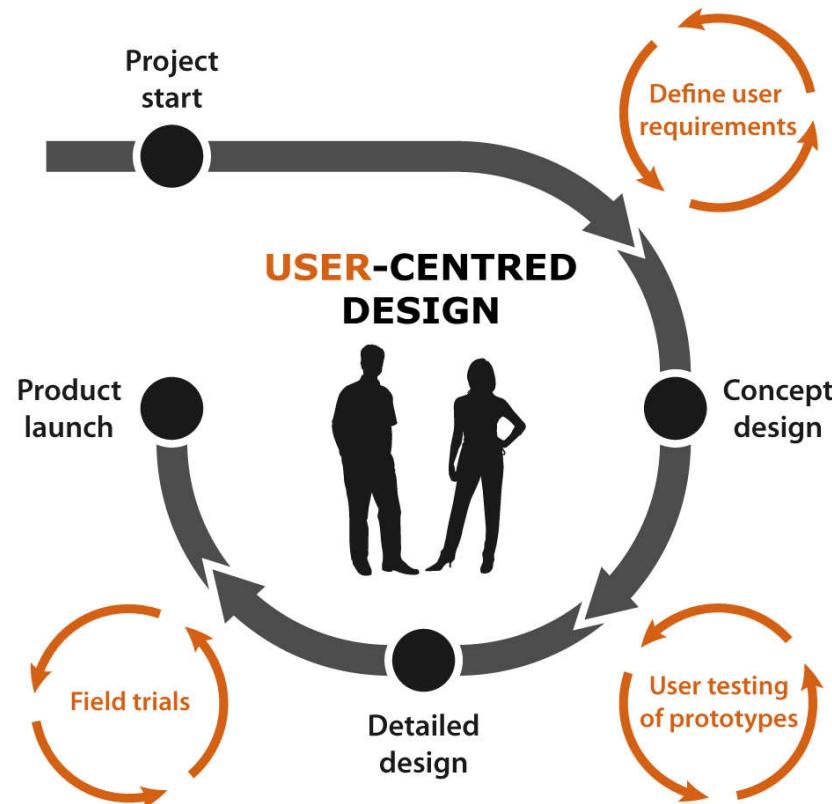
Người dùng đánh giá giao diện sử dụng chưa phù hợp thì => sự thất bại của toàn bộ dự án

5. Thiết kế giao diện

Người dùng

Giao diện phải được thiết kế để người dùng cảm thấy thoải mái, thuận tiện và sử dụng các chức năng hiệu quả.

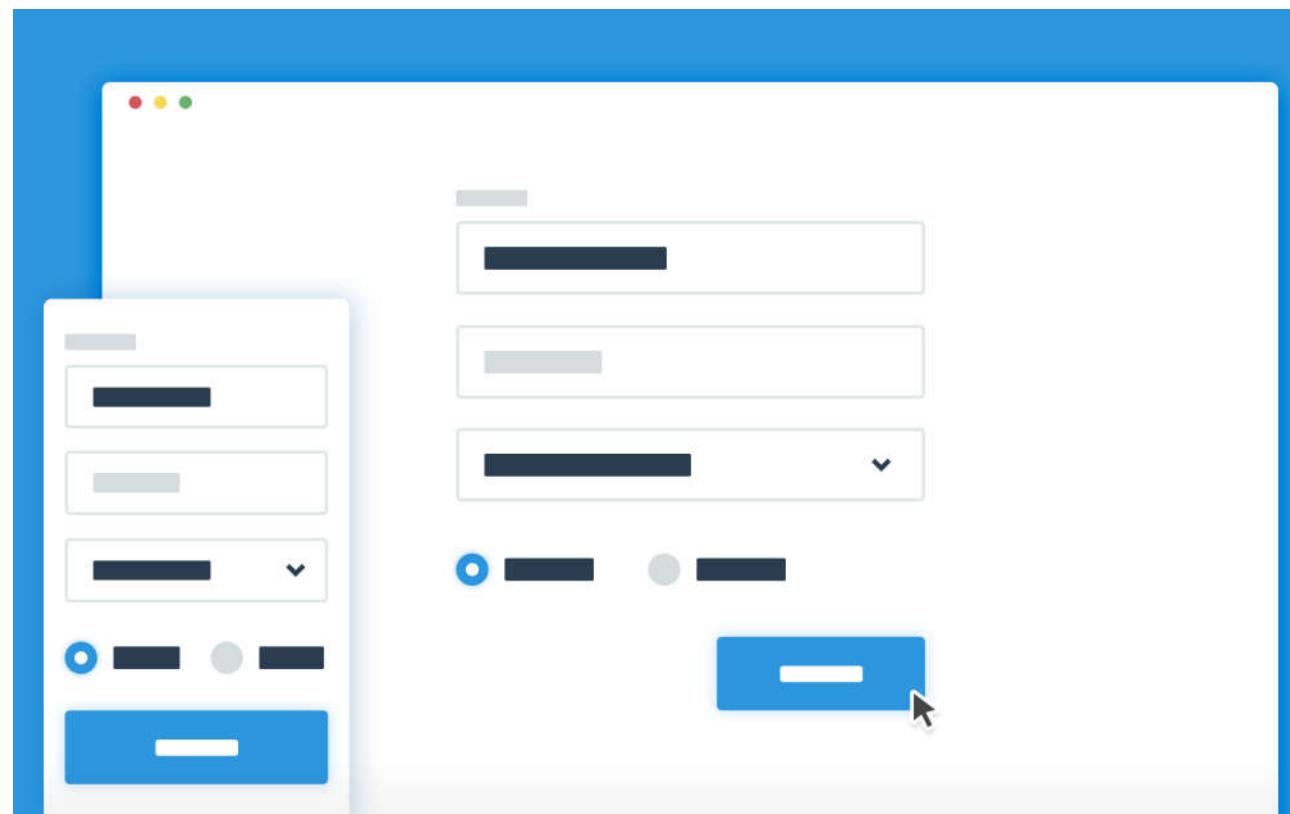
Người dùng được xem là **trọng tâm của việc thiết kế**.



5. Thiết kế giao diện

Người dùng

Bố cục và phong cách giao diện cũng ảnh hưởng đến người dùng theo nhiều cách khác nhau.



5. Thiết kế giao diện

Người dùng

Để thiết kế giao diện đạt nhu cầu người dùng cần:

- Nhận thức được sự tác động của người dùng với giao diện chương trình
- Nắm vững các nguyên tắc thiết kế, phân loại khả năng tương tác giữa người dùng và thiết kế hiển thị giao diện
- Thể hiện nội dung thông tin phù hợp với trình độ của các nhóm người dùng khác nhau

5. Thiết kế giao diện

Người dùng

Ngoài ra, **khả năng trí nhớ tức thời** của con người bị hạn chế => vì vậy giao diện thiết kế cũng không nên biểu diễn quá nhiều dữ liệu.

Chú ý đến các giới hạn vật lý, tinh thần của người dùng.

Con người luôn có thể gây ra lỗi cho dù hệ thống có thể hiện giao diện và kèm theo hướng dẫn tốt đến đâu.

5. Thiết kế giao diện

Các nguyên tắc thiết kế giao diện

Các nguyên tắc mang tính cơ bản ảnh hưởng đến thiết kế và thực thi của tất cả giao diện, bao gồm giao diện phần mềm và giao diện Web.

5. Thiết kế giao diện

Nguyên tắc thẩm mỹ

- Cung cấp sự tương phản có ý nghĩa giữa các thành phần trên màn hình
- Thực hiện gom nhóm các thành phần giao diện có cùng chức năng
- Canh chỉnh các thành phần giao diện và các nhóm giao diện
- Cung cấp thể hiện 3 chiều nhằm mang tính trực quan nếu có thể
- Sử dụng hiệu ứng màu sắc và đồ họa đơn giản, hợp lý

5. Thiết kế giao diện

Nguyên tắc rõ ràng

Giao diện phải trực quan, các khái niệm và ngôn ngữ phải minh bạch, đơn giản với người sử dụng.

Các thành phần giao diện phải dễ hiểu, có sự liên quan đến khái niệm và chức năng của con người trong thế giới thực

5. Thiết kế giao diện

Nguyên tắc tương thích

Thiết kế giao diện phải tương thích với người dùng, công việc, nhiệm vụ, và sản phẩm

- Tương thích người dùng
- Tương thích công việc, nhiệm vụ
- Tương thích sản phẩm

5. Thiết kế giao diện

Nguyên tắc dễ hiểu

Giao diện thiết kế phải dễ hiểu với người dùng theo khía cạnh

- Nên nhìn cái gì
- Nên làm gì
- Khi nào làm một việc gì đó
- Nơi nào để làm việc đó
- Tại sao phải làm việc đó
- Cách để làm việc đó

Một hệ thống phải dễ hiểu, theo trật tự rõ ràng, có ý nghĩa. Các bước hoàn thành một nhiệm vụ phải đơn giản. Các chú thích phải được diễn giải ngắn gọn.

5. Thiết kế giao diện

Nguyên tắc cấu hình

Giao diện thiết kế phải cho phép người dùng tùy ý cá nhân hóa và tùy biến cấu hình.

=> tăng khả năng kiểm soát chương trình, thúc đẩy vai trò cá nhân trong việc hiểu, cho phép việc sử dụng giao diện dựa mức độ kinh nghiệm của người dùng.

Điều này mang đến sự thỏa mãn tốt hơn với người dùng.

5. Thiết kế giao diện

Nguyên tắc bền vững

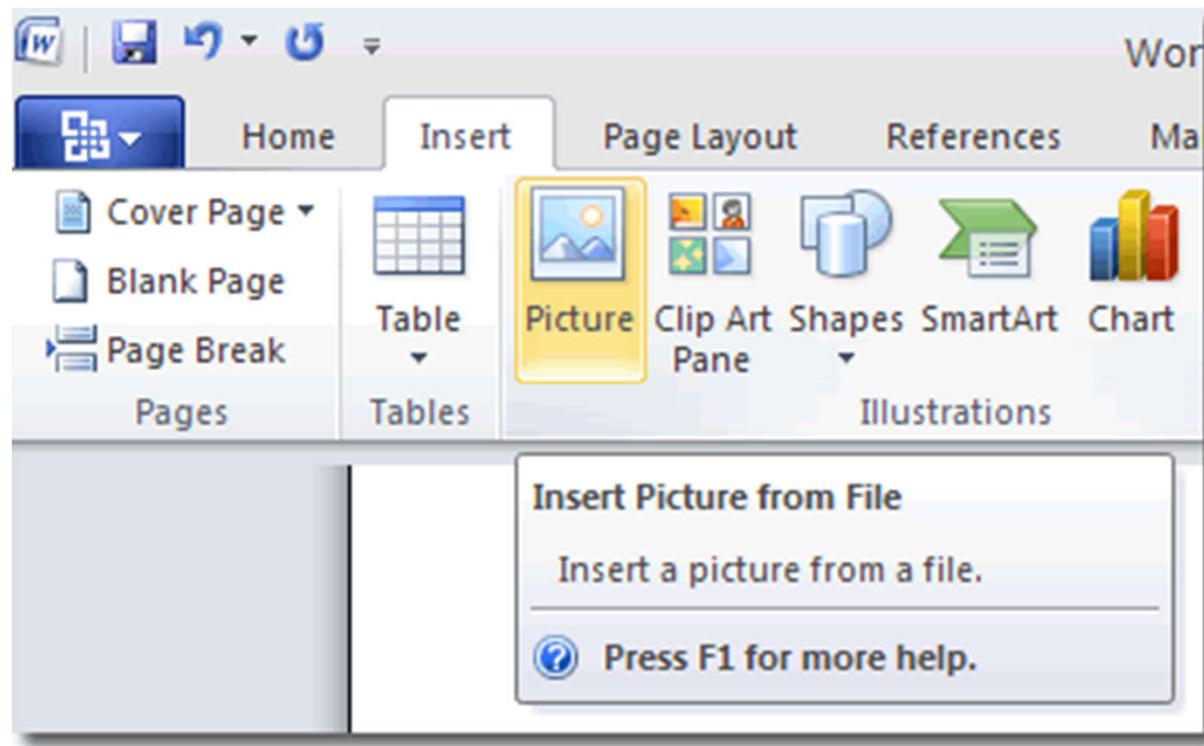
Một giao diện phải có cái nhìn, hoạt động nhất quán xuyên suốt để người dùng dễ sử dụng.

- Các thành phần nên giống nhau như phong cách giao diện (màu sắc, font chữ, ...), tính năng và cách hoạt động tương tự nhau
- Cùng một chức năng phải luôn có một kết quả giống nhau.
- Các tính năng và vị trí các thành phần giao diện phải không thay đổi.
- Các giao diện thiết kế không nhất quán sẽ gây ức chế, bối rối lớn với người dùng.

5. Thiết kế giao diện

Nguyên tắc bền vững

Ví dụ: với một chương trình soạn thảo văn bản thì các chức năng soạn thảo đơn giản phải luôn ở trên đầu chương trình và không nên bị thay đổi vị trí, màu sắc hay biểu tượng ở các module khác trong chương trình.



5. Thiết kế giao diện

Nguyên tắc kiểm soát

Người dùng phải kiểm soát được các tương tác với chương trình thông qua:

- Các chức năng phải cho ra kết quả từ các yêu cầu rõ ràng
- Các chức năng có thời gian thực thi nhanh
- Các chức năng phải được ngắt hay trì hoãn không thực hiện trong trường hợp người dùng muốn
- Người dùng không nên bị dừng tương tác bởi các lỗi sinh ra trong hệ thống

5. Thiết kế giao diện

Nguyên tắc hiệu quả

Người thiết kế phải chú ý số lượng các chuyển động mắt và tay của người dùng. **Các chuyển động này không nên được lãng phí.**

Người dùng hầu hết sử dụng thị giác để nhận biết thông tin, vì vậy các thành phần giao diện phải có tính dự đoán, hiển nhiên và ngắn gọn.

Việc chuyển tiếp thủ công giữa các điều khiển hệ thống hay các màn hình phải ngắn nhất có thể.

5. Thiết kế giao diện

Nguyên tắc hiệu quả

Tránh việc dùng thường xuyên chuột, bàn phím khi thực hiện thao tác.

Phải luôn dự đoán nhu cầu người dùng. Ở mỗi bước trong 1 quy trình, phải trình bày tất cả các thông tin và công cụ cần thiết để hoàn thành quy trình đó.

Hạn chế hỏi người dùng tìm kiếm, thu thập thông tin hay công cụ cần thiết.

- Ví dụ, khi người dùng nhập vào ngày, tháng, năm sinh thì chương trình phải tự tính tuổi của người đó.

5. Thiết kế giao diện

Nguyên tắc thân thiện

Phải thể hiện trên giao diện các nội dung, khái niệm thông qua **ngôn ngữ quen thuộc** với người dùng, giữ giao diện tự nhiên, bắt chước các hành vi người dùng.

Các khái niệm dễ hiểu sẽ làm người dùng làm quen hệ thống nhanh hơn.

5. Thiết kế giao diện

Nguyên tắc mềm dẻo

Một hệ thống phải mềm dẻo thể hiện các nhu cầu khác nhau của người dùng, cho phép một mức độ và dạng nền tảng dựa trên:

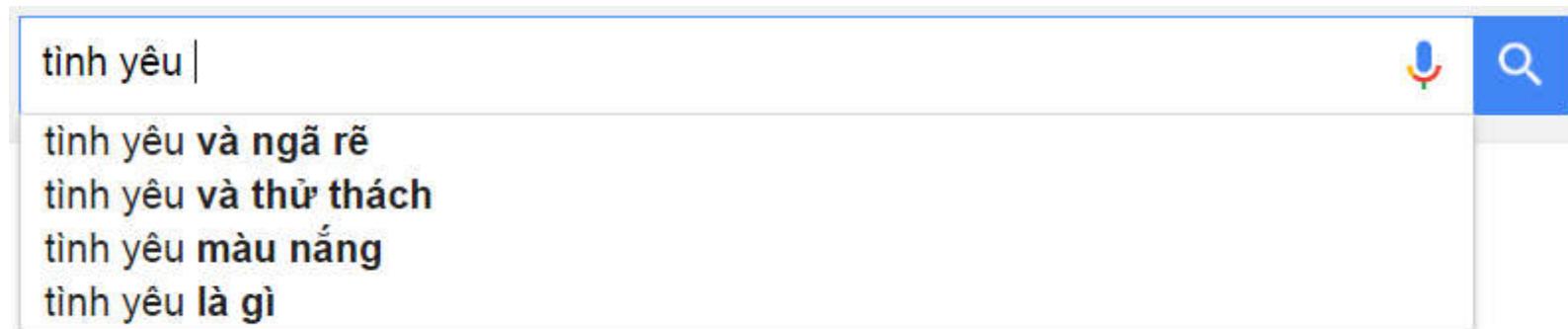
- Kỹ năng và kiến thức của người dùng
- Kinh nghiệm người dùng
- Sở thích người dùng
- Thói quen người dùng
- Các điều kiện hiện tại

5. Thiết kế giao diện

Nguyên tắc dự đoán

Giao diện chương trình phải có tính dự đoán tự nhiên dựa trên từng chức năng để:

- Cung cấp các thành phần giao diện nhận biết được và rõ ràng
- Cung cấp các gợi ý cho kết quả hoặc dữ liệu mà người dùng mong muốn



Dự án các kết quả mà người dùng có thể muốn tìm kiếm ở Google Search

5. Thiết kế giao diện

Nguyên tắc phục hồi

Người dùng phải có chức năng lưu trữ thông tin tự động lúc đang làm việc khi gặp sự cố.

Việc phục hồi giúp người dùng tránh được lỗi gấp trước đó hoặc lấy lại được thông tin bị mất, giảm thiểu thời gian và công sức phải thực hiện công việc từ đầu.



5. Thiết kế giao diện

Biểu diễn bố cục giao diện

Bố cục giao diện thường được biểu diễn theo các tính chất sau:

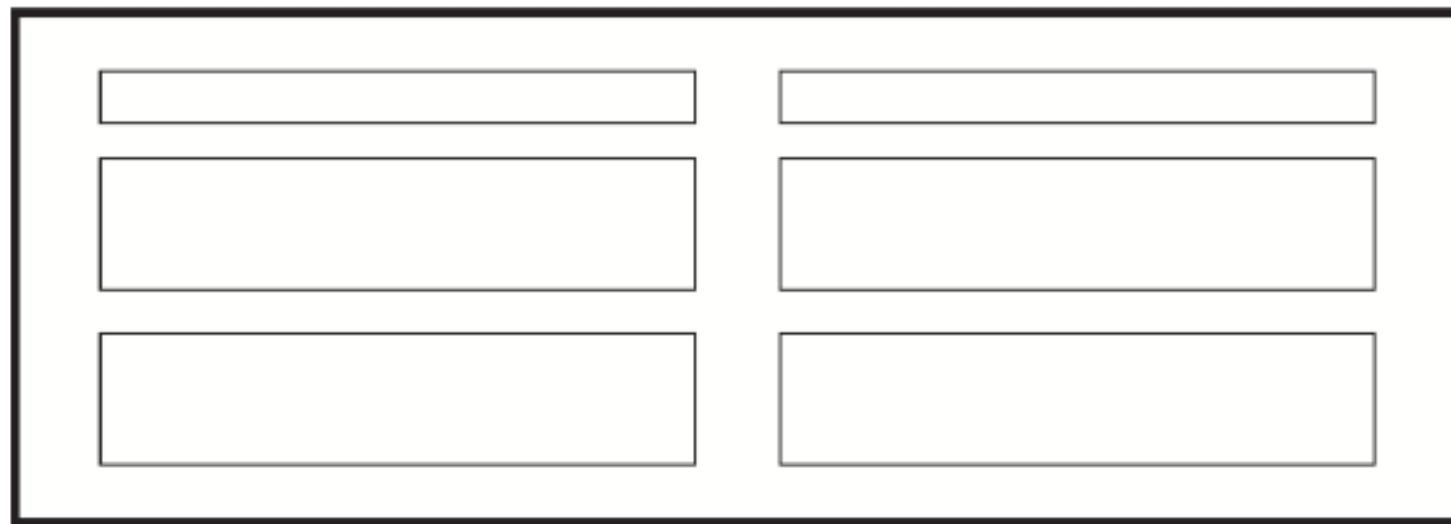
- Tính cân bằng
- Tính đối xứng
- Tính đều đặn
- Tính dự đoán
- Tính liên tục
- Tính kinh tế
- Tính thống nhất
- Tính cân xứng theo tỉ lệ
- Tính đơn giản
- Tính gom nhóm

5. Thiết kế giao diện

Tính cân bằng

Bố cục cân bằng có các thành phần giao diện tương đối bằng nhau theo các phía trái, phải, trên, dưới.

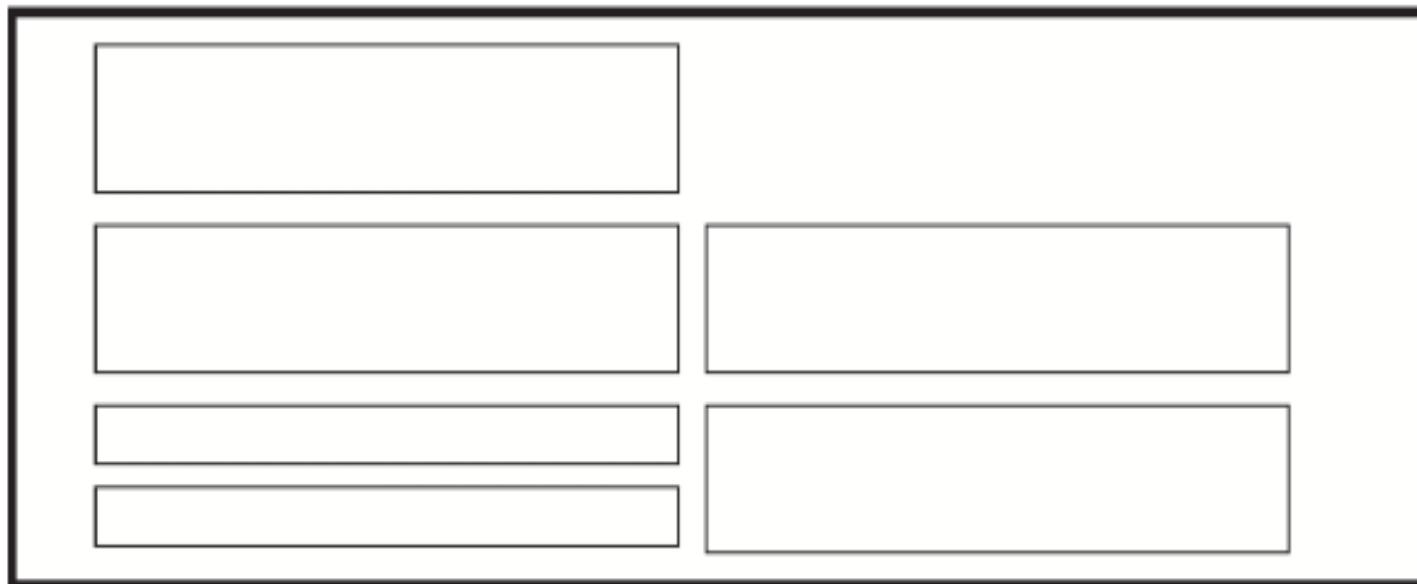
Hình sau là giao diện cân bằng, các thành phần giao diện đối xứng với nhau qua trọng tâm ở giữa



5. Thiết kế giao diện

Tính cân bằng

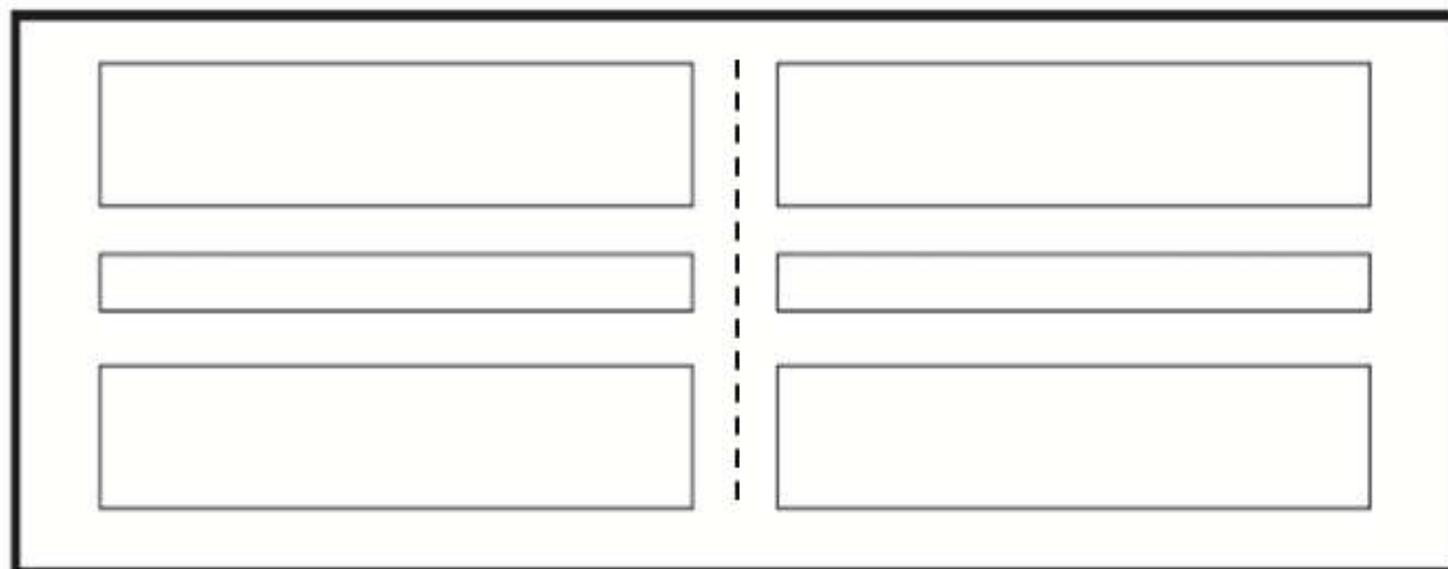
Hình dưới là giao diện không cân bằng với các số lượng thành phần bên trái là 4, bên phải là 2, hơn nữa các thành phần 2 bên không đối xứng nhau từng đôi một nếu xét theo chiều ngang.



5. Thiết kế giao diện

Tính đối xứng

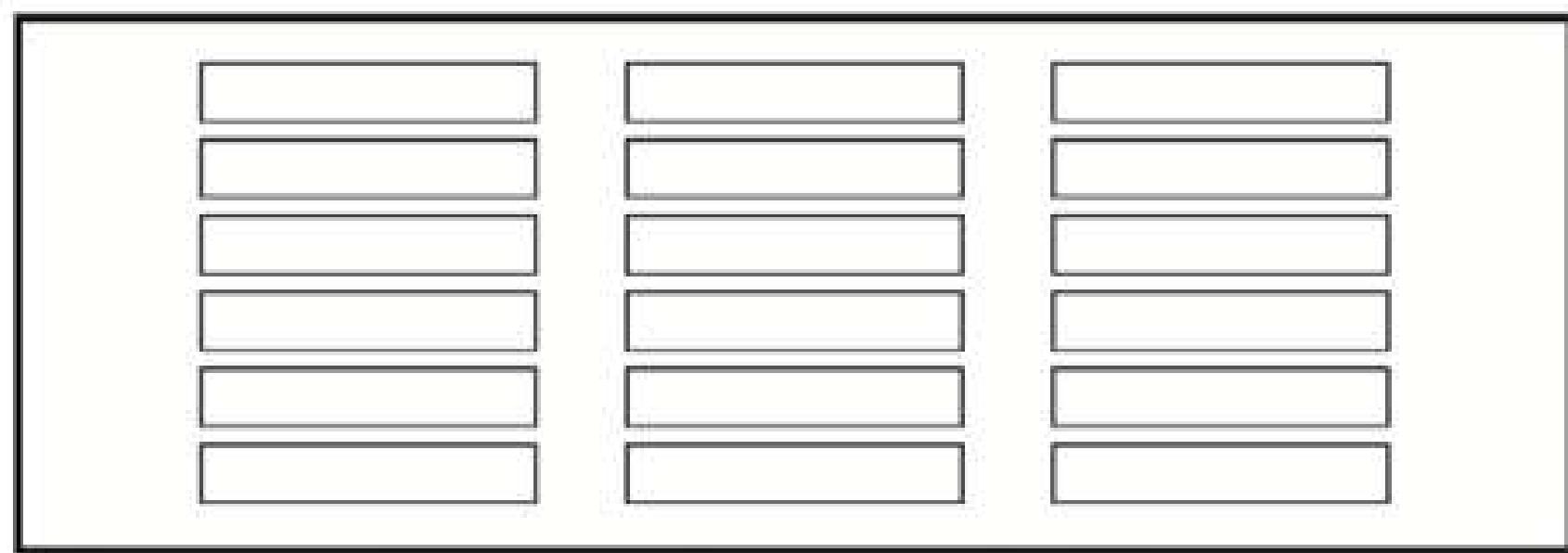
Nếu kẻ một **đường dọc nét đứt ở giữa màn hình làm đường trực** thì nếu các thành phần giao diện phải đối xứng nhau thông qua đường này thì gọi giao diện này là **giao diện đối xứng**.



5. Thiết kế giao diện

Tính đều đặn (regularity)

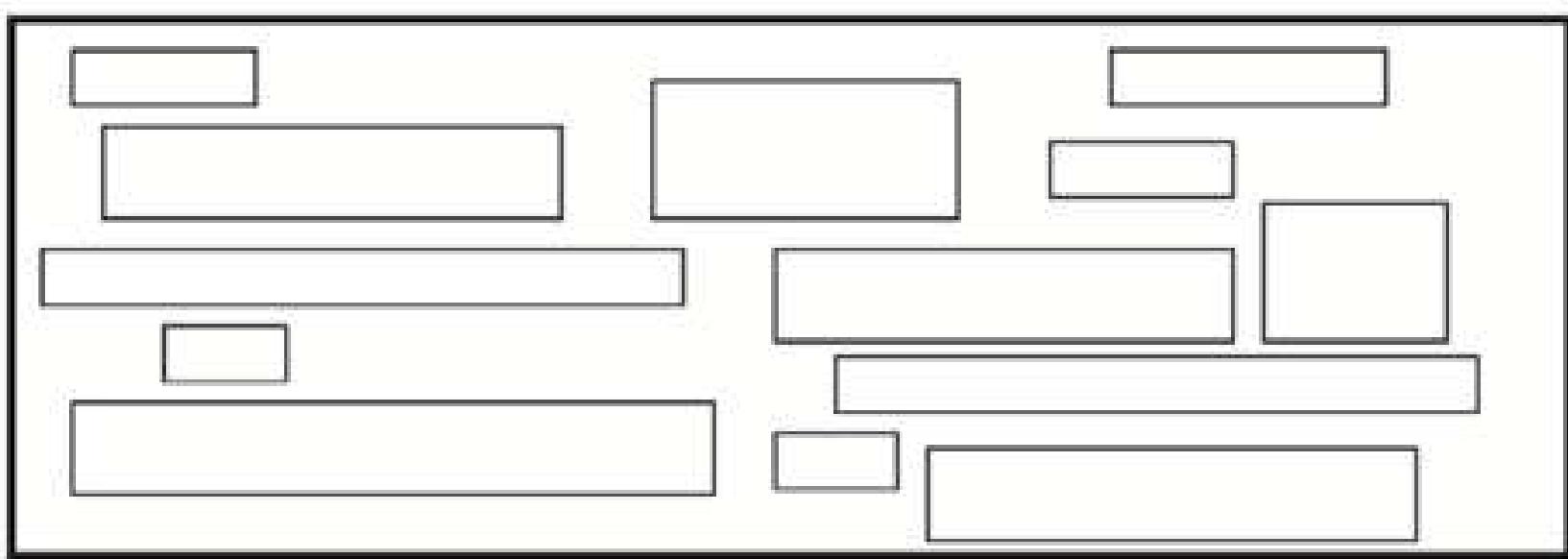
Để tạo một bố cục đều đặn, phải thiết lập các tiêu chuẩn và các điểm canh lề ngang, dọc đồng nhất và sử dụng các thành phần kích thước, hình dạng, màu sắc và khoảng trống tương tự nhau.



5. Thiết kế giao diện

Tính đều đặn (regularity)

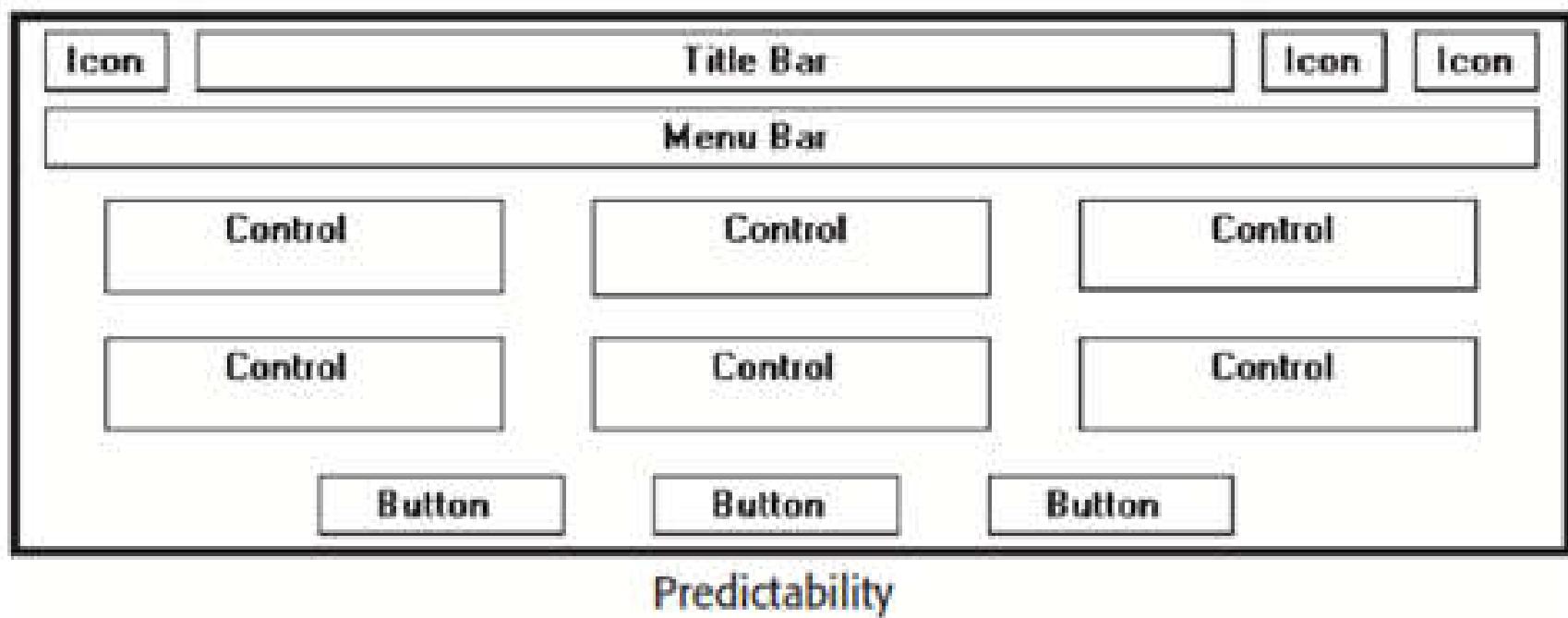
Giao diện không đều đặn, với kích thước các thành phần khác nhau, được bố trí lộn xộn.



5. Thiết kế giao diện

Tính dự đoán

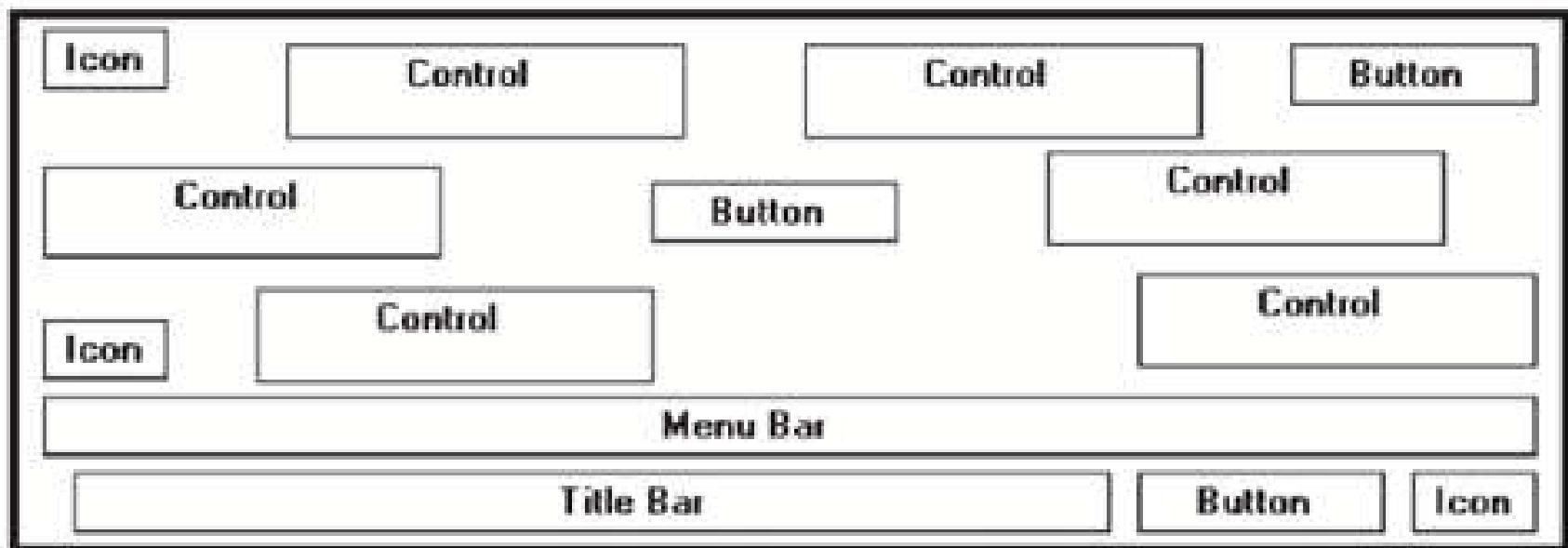
Giao diện phải được tạo theo trật tự hoặc thứ tự sắp xếp nhất định.



5. Thiết kế giao diện

Tính dự đoán

Giao diện tự phát là giao diện được thiết kế không theo kế hoạch vì vậy bố cục không theo thứ tự.

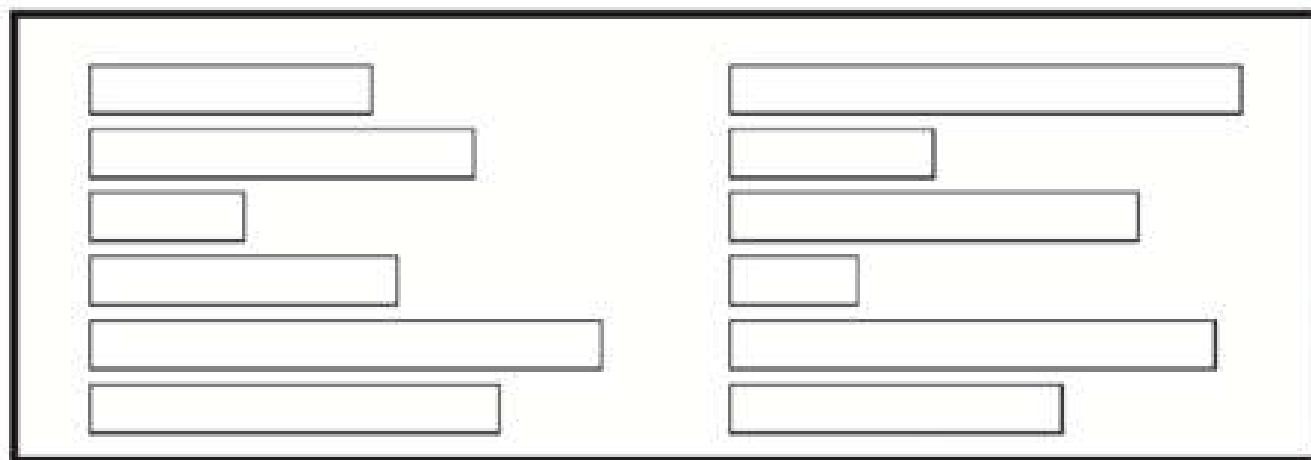


5. Thiết kế giao diện

Tính liên tục

Giao diện có tính liên tục chứa **các thành phần được sắp xếp** giúp người dùng có thể theo dõi theo một **trật tự rõ ràng, hợp lý và hiệu quả**.

Hình dưới là giao diện có tính liên tục giao diện liên tục hướng thị giác người dùng theo trật tự từ trên xuống dưới và từ trái sang phải khi thực hiện các thao tác tương tác.



5. Thiết kế giao diện

Tính liên tục

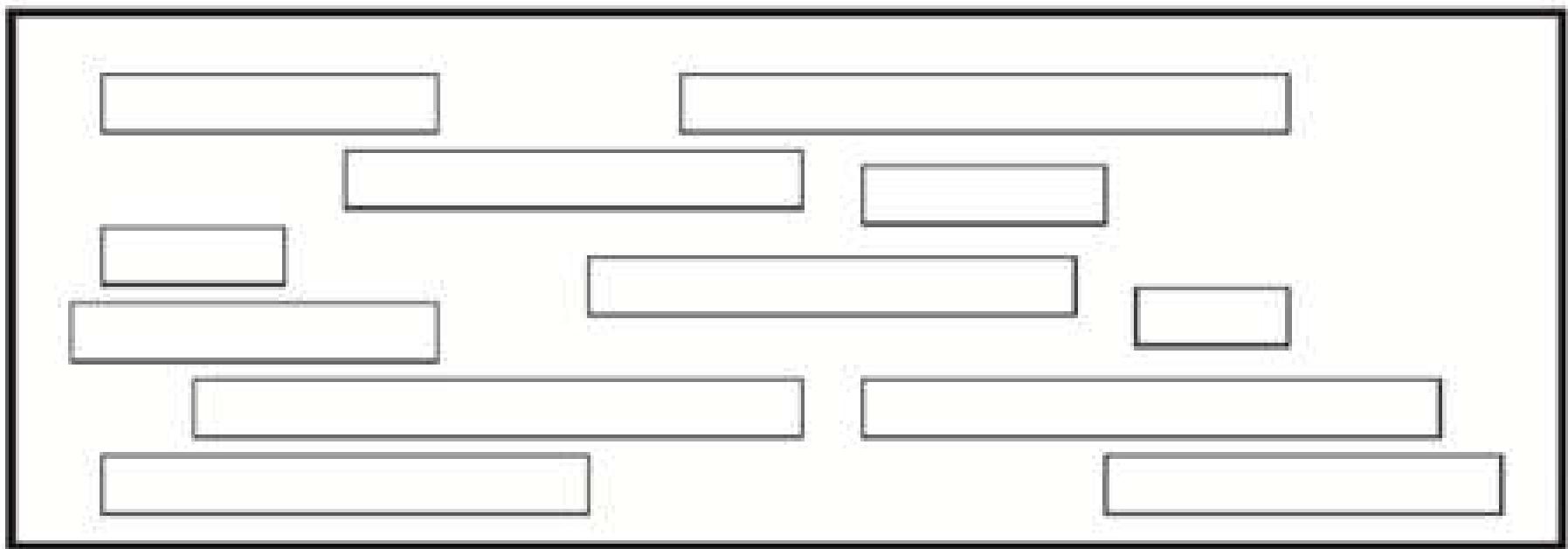
Thị giác người sử dụng thường bị cuốn hút bởi:

- Các đối tượng nổi bật về ánh sáng so với đối tượng ít nổi bật hơn
- Các thành phần riêng biệt so với các nhóm thành phần
- Hiệu ứng đồ họa hơn so với văn bản
- Màu sắc hơn so với trắng, đen
- Vùng tối hơn so với vùng sáng
- Thành phần có kích lớn hơn so với thành phần có kích thước nhỏ
- Hình dạng bất thường so với hình dạng bình thường

5. Thiết kế giao diện

Tính liên tục

Hình dưới là giao diện ngẫu nhiên làm người dùng bối rối không biết phải thao tác theo thứ tự như thế nào.

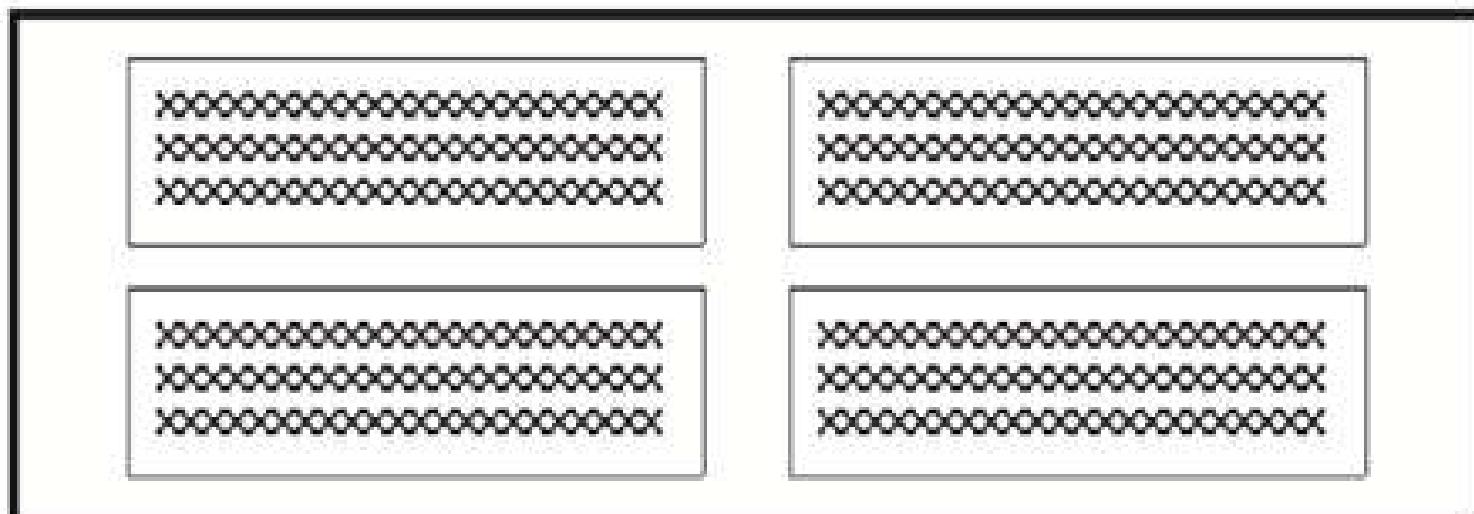


5. Thiết kế giao diện

Tính kinh tế

Giao diện phải mang tính kinh tế khi sử dụng phong cách, kỹ thuật hiển thị và màu sắc ít nhất có thể.

Trong hình, giao diện kinh tế sử dụng **phong cách đồng nhất**, đơn giản để hiển thị các thành phần dữ liệu.

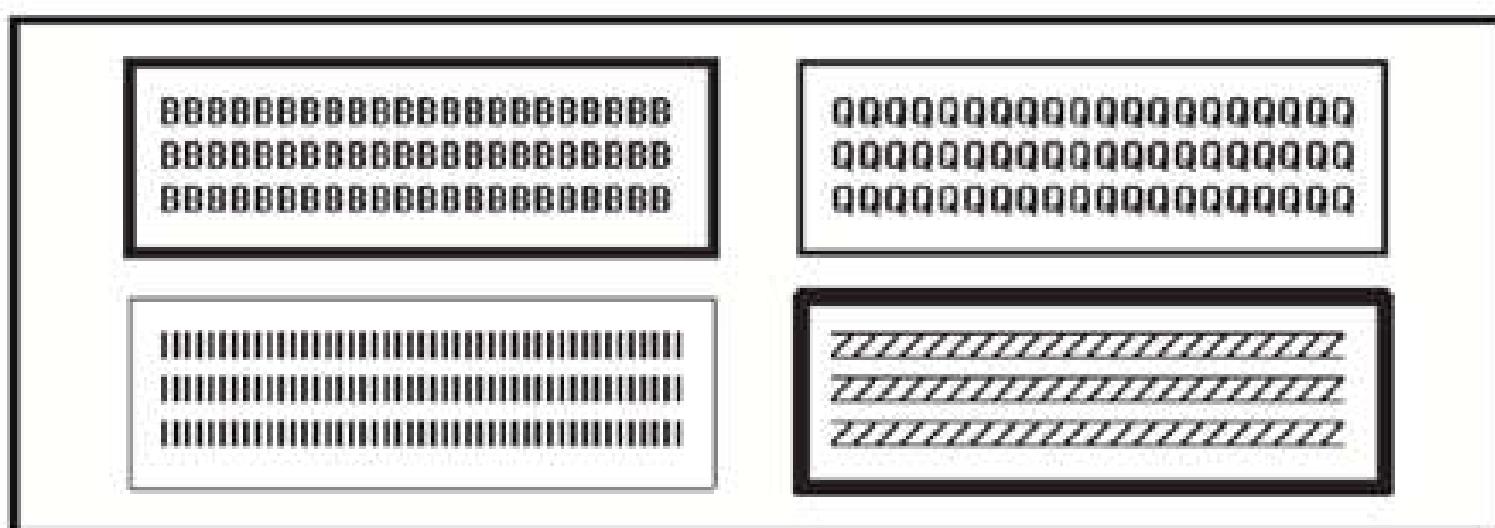


5. Thiết kế giao diện

Tính kinh tế

Giao diện phức tạp sử dụng nhiều phong cách hiển thị khác nhau (độ đậm nhạt đường viền, văn bản).

Với thiết kế giao diện phức tạp, chi phí bỏ ra cho các nhân viên thiết kế có thể tăng thêm mà chương trình phần mềm vẫn không đạt hiệu quả nhưng mong muốn.

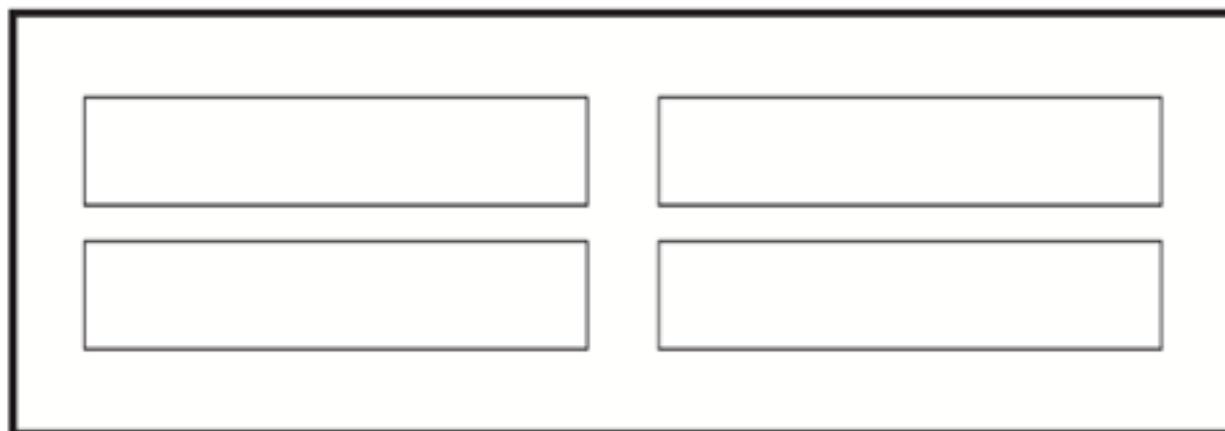


5. Thiết kế giao diện

Tính thống nhất

Giao diện mang tính thống nhất khi sử dụng thông tin có kích thước, màu sắc, hình dạng tương tự nhau và ít có khoảng trống giữa các thành phần hơn so với khoảng trống với các lề màn hình.

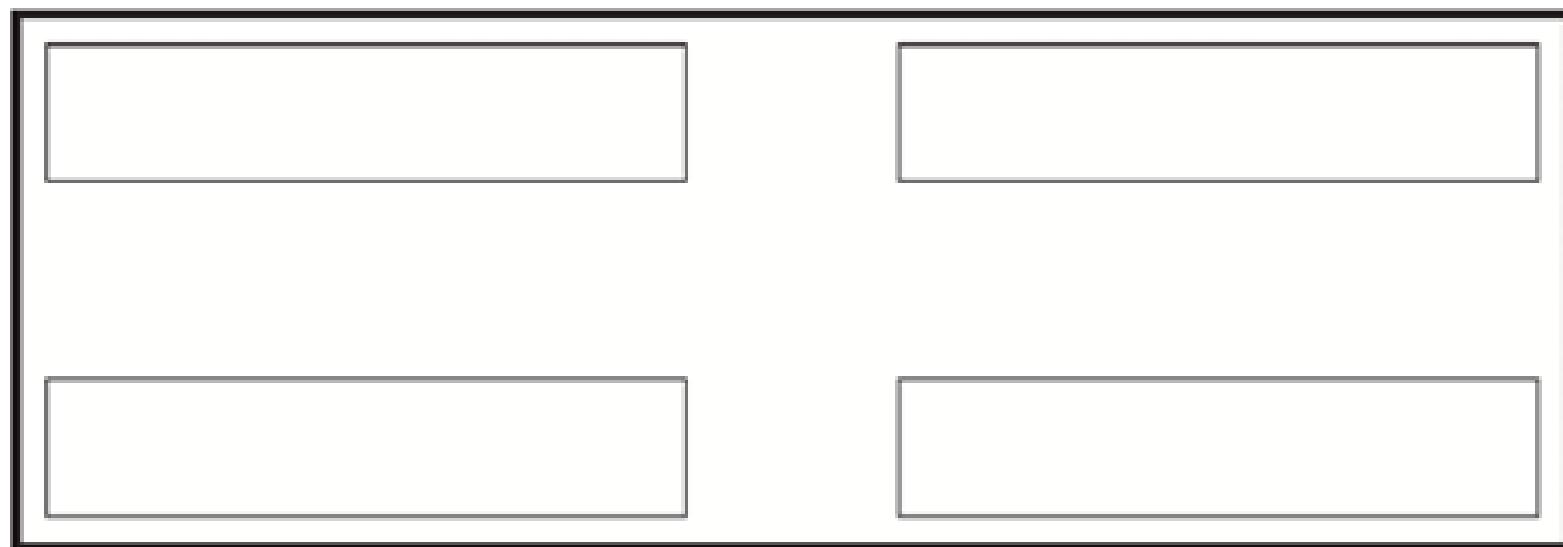
Giao diện có tính thống nhất: cùng đối tượng thành phần (ô chữ nhật với kích thước giống nhau), khoảng cách giữa các thành phần nhỏ hơn so với khoảng cách giữa các thành phần với lề màn hình.



5. Thiết kế giao diện

Tính thống nhất

Trái lại, **giao diện phân mảnh** thì khoảng cách giữa các thành phần rộng hơn rất nhiều với khoảng cách giữa các thành phần với lề màn hình.



5. Thiết kế giao diện

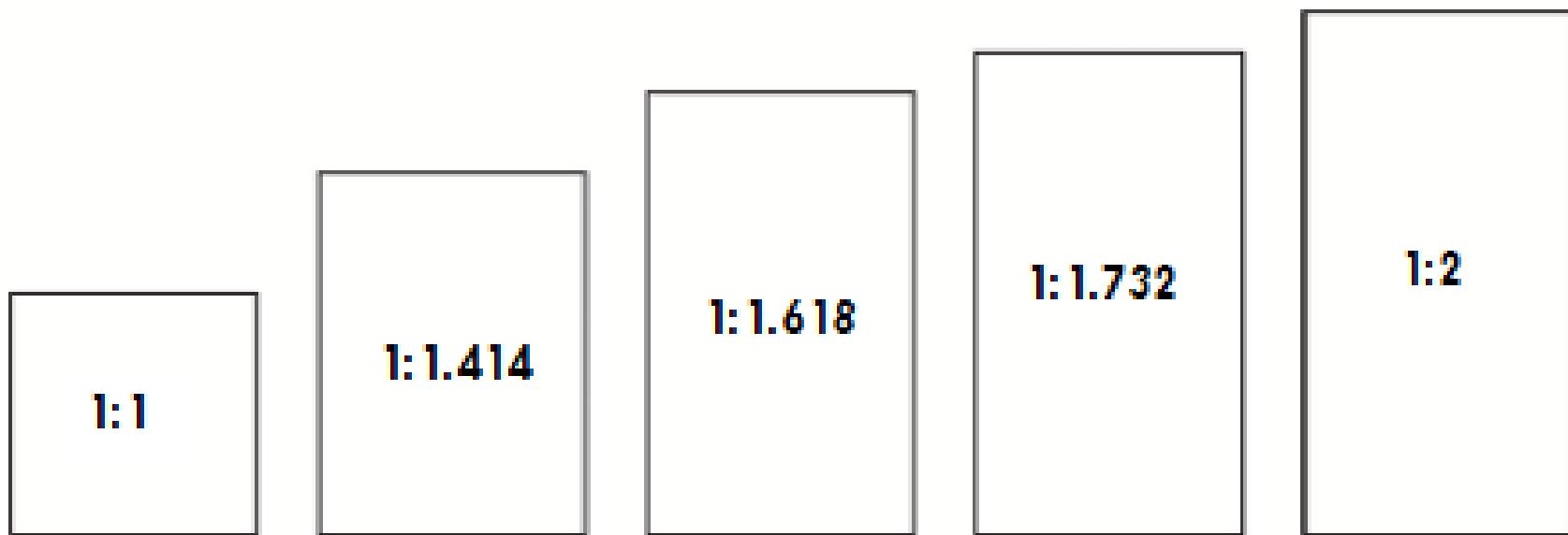
Tính cân xứng theo tỷ lệ

Khi thiết kế giao diện, chúng ta có thể chọn một số kích thước chuẩn cho các thành phần giao diện. Một số tỉ lệ chuẩn được Marcus định nghĩa:

- Hình vuông (1:1)
- Tỉ lệ căn bậc 2 (1:1.414)
- Hình chữ nhật vàng (1:1.618)
- Tỉ lệ căn bậc 3 (1:1.732)
- Tỉ lệ 1:2

5. Thiết kế giao diện

Tính cân xứng theo tỷ lệ



5. Thiết kế giao diện

Tính đơn giản

Người thiết kế phải tối ưu số lượng thành phần sử dụng trên giao diện và giảm thiểu các điểm canh lề, đồng thời xây dựng một chuẩn lưới canh lề dọc, ngang để giao diện đều, đơn giản, rõ ràng.



5. Thiết kế giao diện

Tính gom nhóm

Giao diện thiết kế phải gom các thành phần cho cùng chức năng và ý nghĩa vào cùng 1 nhóm để người dùng dễ dàng nhận biết và tương tác tốt hơn với giao diện.

Nhà thiết kế sử dụng các đường viền để khoanh vùng các nhóm và giữa các nhóm khác nhau đều có 1 khoảng cách nhất định.

5. Thiết kế giao diện

Biểu diễn thông tin

Các thành phần có cùng ý nghĩa được gom thành các nhóm, mỗi nhóm có đường viền bao quanh, giữa các nhóm có khoảng trắng phù hợp, cân xứng.

TIFF File format	
Version #	42
Byte order	11
NewSubFile Type	1
Image size	
Horizontal	1888
Vertical	1656
Units	Pixel
	Inch
Spatial Configuration	
Samples per pixel	1
Bits per sample	1
Planar config.	1
Image resolution	
Horizontal res.	300
Vertical res.	300
Units	dpi

5. Thiết kế giao diện

Biểu diễn thông tin

Thông tin từ hệ thống được hiển thị tới người dùng sử dụng thông qua giao diện thiết kế.

Thông tin khác nhau có kiểu dữ liệu khác nhau vì vậy biểu diễn thông tin có thể chuyển thành nhiều cách hiển thị như dạng đồ họa, âm thanh.

5. Thiết kế giao diện

Biểu diễn thông tin

Thông tin được thể hiện bằng 2 loại:

- Thông tin tĩnh: là dạng được khởi tạo ở đầu của mỗi phiên làm việc trong chương trình. Những thông tin này không thay đổi trong suốt phiên làm việc đó và có thể là ở dạng số hoặc dạng văn bản.
- Thông tin động: là những thông tin thay đổi liên tục trong cả phiên sử dụng với sự quan sát của người dùng

5. Thiết kế giao diện

Biểu diễn thông tin

Một số nhân tố có tác động đến việc hiển thị thông tin:

- Sở thích của người sử dụng về việc hiển thị thông tin, một phần thông tin hay quan hệ dữ liệu
- Tốc độ thay đổi dữ liệu thông tin nhanh hay chậm
- Các thao tác cần làm của người dùng để thay đổi dữ liệu
- Thể hiện thông tin ở các kiểu dữ liệu khác nhau
- Màu sắc trong thiết kế

5. Thiết kế giao diện

Biểu diễn thông tin

Khi thể hiện bằng thông tin bằng màu sắc, nên chú ý các nguyên tắc:

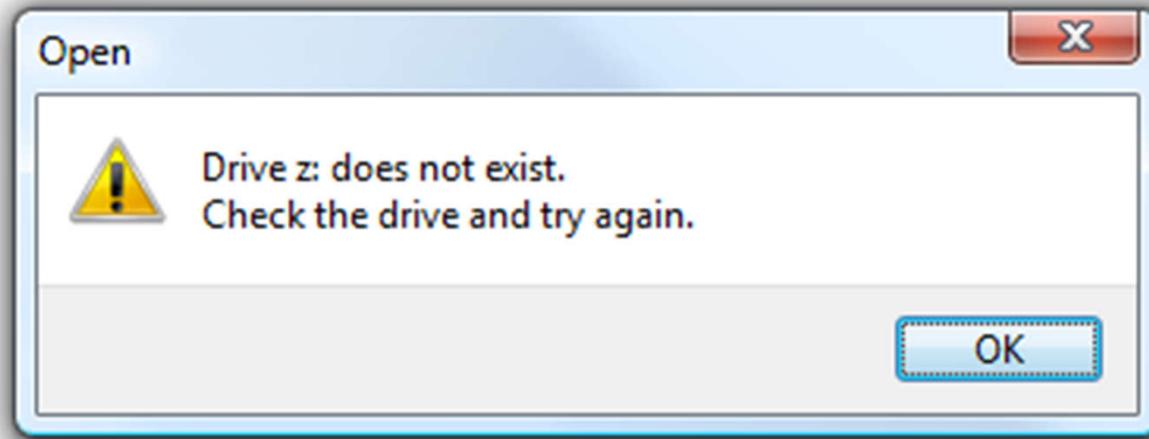
- Thay đổi màu khi thay đổi trạng thái của hệ thống
- Giới hạn số lượng màu được sử dụng và không nên lạm dụng việc sử dụng màu.
- Sử dụng màu để hỗ trợ cho những nhiệm vụ mà người dùng đang cố gắng thực hiện.
- Sử dụng màu một cách thống nhất và cẩn thận.
- Cẩn thận khi sử dụng các cặp màu.

5. Thiết kế giao diện

Biểu diễn thông tin

Các thông tin dạng thông báo lỗi phải có cấu trúc, ngắn gọn, xúc tích và thống nhất cho tất cả các giao diện trong hệ thống.

Người thiết kế phải đánh giá đúng kỹ năng và kinh nghiệm của người dùng khi thiết kế thông báo lỗi.



5. Thiết kế giao diện

Quy trình thiết kế giao diện

Người thiết kế phải lập ra một số danh sách như danh sách các biến cố và danh sách các thành phần giao diện.

Danh sách các biến cố

STT	Điều kiện kích hoạt	Xử lý	Ghi chú
...

5. Thiết kế giao diện

Quy trình thiết kế giao diện

Danh sách các thành phần giao diện

STT	Tên	Kiểu	Ý nghĩa	Miền giá trị	Giá trị mặc định	Ghi chú
...

5. Thiết kế giao diện

Quy trình thiết kế giao diện

Ví dụ: thiết kế chức năng tiếp nhận học sinh mới của phần mềm Quản Lý Học Sinh.

Tiếp nhận học sinh

Họ tên: Giới tính:

Ngay sinh: Địa chỉ:

Lớp:

Qui định: Họ tên phải có. Tuổi từ 15-20. Trường có 20 lớp và 3 khối. Khối 10 có 8 lớp, Khối 11 có 7 lớp. Khối 12 có 5 lớp

5. Thiết kế giao diện

Quy trình thiết kế giao diện

Ví dụ: lập danh sách các thành phần giao diện

STT	Tên	Kiểu	Ý nghĩa	Miền giá trị	Giá trị mặc định	Ghi chú
1	Lb_Tieu_de	Label	Tiêu đề màn hình
2	Lb_Hoten	Label	Tiêu đề họ tên
3	Txt_Hoten	TextBox	Text box nhập họ tên
4	Ch_Phai	Checkbox	...			
5	...					
6	...					

5. Thiết kế giao diện

Quy trình thiết kế giao diện

Ví dụ: tiến hành xây dựng giao diện nhập

Tiếp nhận học sinh

Họ tên	<input type="text"/>	Nam <input checked="" type="checkbox"/>
Ngày sinh	<input type="text"/>	Lớp <input type="text"/>
Địa chỉ	<input type="text"/>	
Ghi		

Danh sách học sinh đã tiếp nhận

STT	Mã HS	Tên HS	Giới tính	Ngày sinh
...

5. Thiết kế giao diện

Khảo sát người dùng và phân tích, đánh giá giao diện

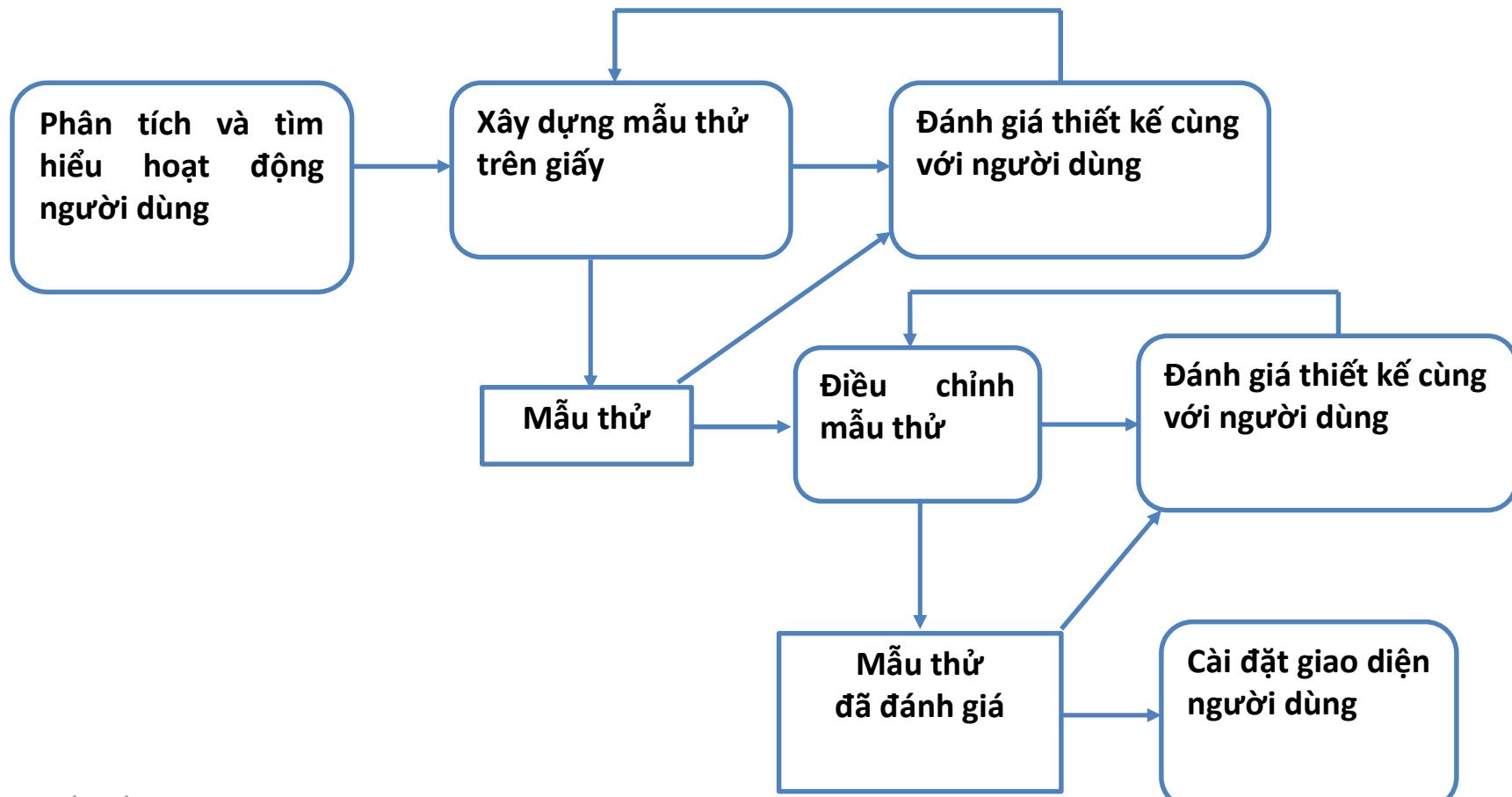
Để thiết kế giao diện hiệu quả, thông thường thì phải cần một quy trình lặp lại với sự cộng tác giữa người dùng và người thiết kế. Quy trình này gồm các hoạt động chính như:

- Khảo sát người sử dụng
- Lập mẫu thử hệ thống
- Đánh giá giao diện

5. Thiết kế giao diện

Khảo sát người dùng và phân tích, đánh giá giao diện

Mô hình chung về quá trình khảo sát người, phân tích và đánh giá giao diện



6. Thiết kế xử lý

Khái niệm

Thiết kế xử lý là bước xây dựng mô tả các hàm xử lý và hằng, biến, kiểu dữ liệu liên quan tương ứng với các chức năng yêu cầu của dự án.

Từ danh sách các yêu cầu ở phần trước, chúng ta có thể lập ra một số danh sách xử lý tùy thuộc vào yêu cầu và quy mô của dự án.

Thiết kế xử lý giúp định hình được nội dung xây dựng mã nguồn cần phải làm để phân chia công việc hiệu quả.

6. Thiết kế xử lý

Các danh sách thiết kế xử lý

Danh sách các kiểu dữ liệu xử lý

Danh sách này liệt kê các kiểu dữ liệu cần thiết để xử lý trong chương trình, kèm theo là ý nghĩa và ghi chú tương ứng.

STT	Kiểu dữ liệu	Ý nghĩa	Ghi chú
...

6. Thiết kế xử lý

Các danh sách thiết kế xử lý

Danh sách các thuộc tính kiểu dữ liệu X

Danh sách này liệt kê các thuộc tính của dữ liệu nào đó với các thông tin chi tiết hơn.

STT	Thuộc tính	Kiểu	Ràng buộc	Giá trị khởi động	Ghi chú
...

6. Thiết kế xử lý

Các danh sách thiết kế xử lý

Danh sách các hàm, phương thức xử lý

Các dự án phải lập danh sách các hàm xử lý vì chúng là các hàm quan trọng trong việc tạo sự tương tác kết nối giữa dữ liệu và giao diện chương trình.

6. Thiết kế xử lý

Các danh sách thiết kế xử lý

Danh sách các hàm, phương thức xử lý

STT	Hàm	Tham số	Kết quả trả về	Thuật giải	Ý nghĩa	Ghi chú
1	InsertStudent	StudentName (string), Birthday (datetime), Address (string)	Kiểu bool với 2 giá trị, "true" là nhập thành công, "false" là nhập không thành công	Lấy các tham số, kiểm tra kiểu dữ liệu và nhập vào hệ thống	Nhập 1 sinh viên vào cơ sở dữ liệu	
2	DeleteStudent	StudentID (int)	Kiểu bool với 2 giá trị, "true" là xóa thành công, "false" là xóa không thành công	Lấy tham số StudentID, kiểm tra dữ liệu và thực hiện thao tác xóa	Xóa 1 sinh viên dựa theo StudentID	

6. Thiết kế xử lý

Các danh sách thiết kế xử lý

Danh sách các hàm, phương thức xử lý

Phương thức **InsertStudent** thì cần 3 tham số đầu vào là **StudentName**, **Birthday**, **Address** và kết quả trả về là kiểu **bool**.

```
bool InsertStudent(string StudentName, DateTime Birthday,  
string Address)
```

```
{
```

```
//Kiểm tra các dữ liệu đầu vào  
if (nhập thành công) trả về true  
else trả về false
```

```
}
```

6. Thiết kế xử lý

Các danh sách thiết kế xử lý

Danh sách các biến

Các biến lập trong danh sách này là biến toàn cục, dùng chung cho hệ thống.

STT	Biến	Kiểu	Ý nghĩa	Ghi chú

6. Thiết kế xử lý

Các danh sách thiết kế xử lý

Danh sách các hằng

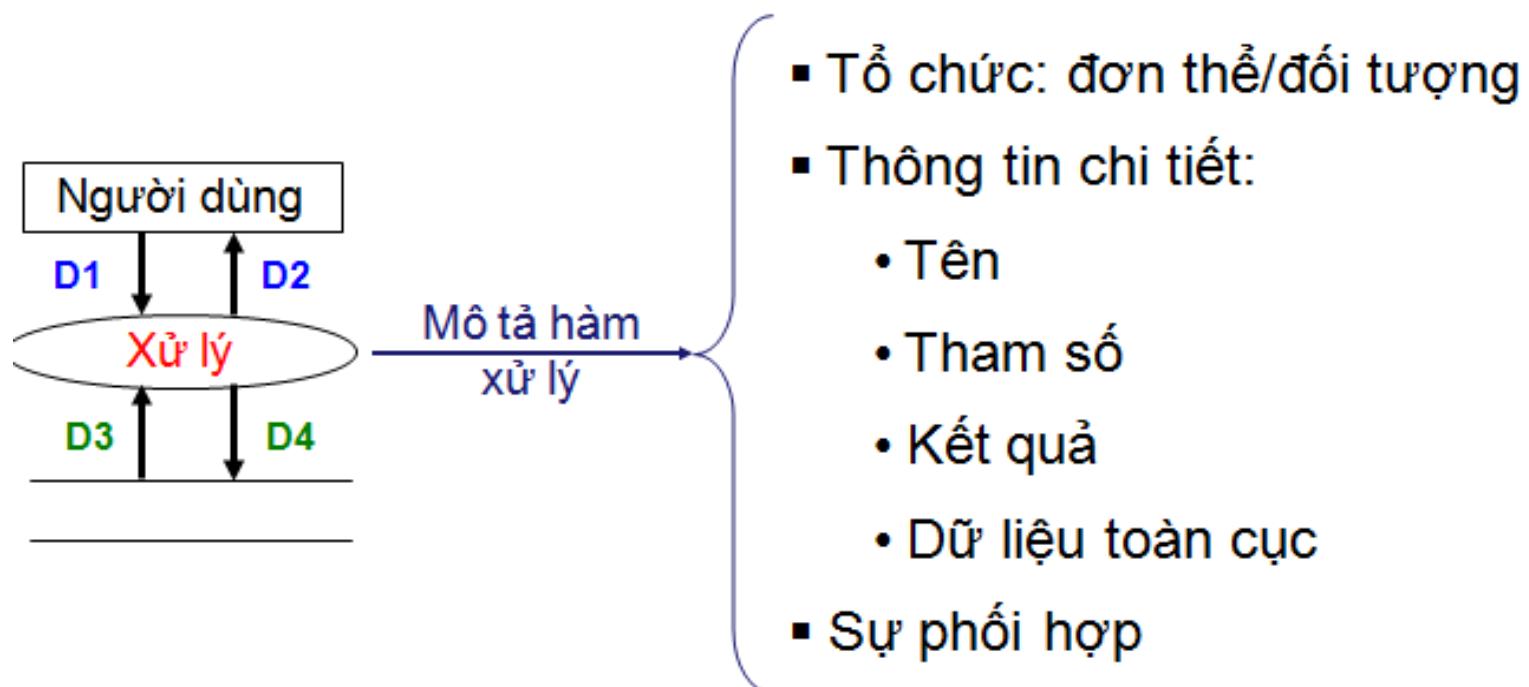
Tương tự danh sách các biến, chúng ta cũng có thể lập danh sách các hằng số (biến không đổi) cho chương trình. Các hằng cũng được xem là dạng biến toàn cục, sử dụng chung cho toàn bộ chương trình phần mềm

STT	Hằng	Kiểu	Giá trị	Ý nghĩa	Ghi chú

6. Thiết kế xử lý

Mô tả hàm xử lý

Sau khi lập danh sách các xử lý, chúng ta phải mô tả chi tiết các xử lý và các sơ đồ phối hợp này để có cái nhìn tổng quát hơn.



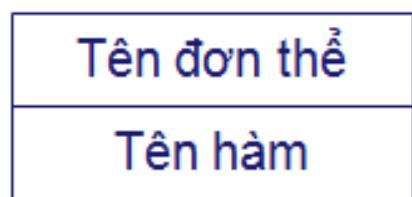
6. Thiết kế xử lý

Mô tả hàm xử lý

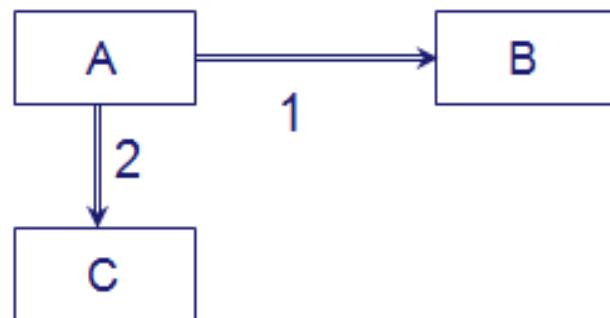
Yêu cầu cho xử lý phải đảm bảo tính đúng đắn, dễ bảo trì, tái sử dụng và di chuyển tốt. Các ký hiệu:



Hàm xử lý



Hàm của đơn thể



A có gọi đến B, C theo thứ tự, không chuyển tham số, không nhận kết quả



A gọi đến B có chuyển tham số, không nhận kết quả

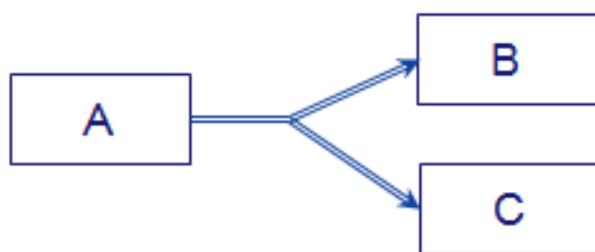
6. Thiết kế xử lý

Mô tả hàm xử lý

Yêu cầu cho xử lý phải đảm bảo tính đúng đắn, dễ bảo trì, tái sử dụng và di chuyển tốt. Các ký hiệu:



A gọi đến B không chuyển tham số, nhưng nhận kết quả



A gọi đến B hoặc C



A gọi đến B nhiều lần (ít nhất là 0 lần)



A gọi đến B nhiều lần (ít nhất là 1 lần)

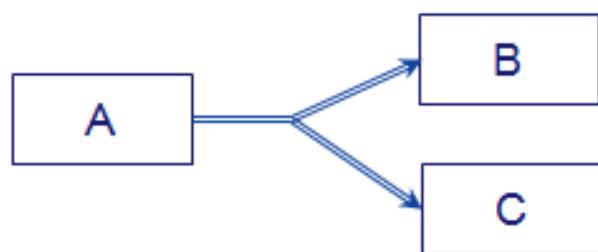
6. Thiết kế xử lý

Mô tả hàm xử lý

Chú ý nếu có n biến cố cần xử lý thì tương ứng phải có n sơ đồ phối hợp.



A gọi đến B không chuyển tham số, nhưng nhận kết quả



A gọi đến B hoặc C



A gọi đến B nhiều lần (ít nhất là 0 lần)



A gọi đến B nhiều lần (ít nhất là 1 lần)

6. Thiết kế xử lý

Mô tả hàm xử lý

Ví dụ: Xét đến màn hình tiếp nhận một học sinh mới

Tiếp nhận học sinh				
Họ tên	<input type="text"/>		Nam	<input checked="" type="checkbox"/>
Ngày sinh	<input type="text"/>		Lớp	<input type="text"/> 
Địa chỉ	<input type="text"/>			
<input type="button" value="Ghi"/>				
Danh sách học sinh đã tiếp nhận				
STT	Mã HS	Tên HS	Giới tính	Ngày sinh
...

6. Thiết kế xử lý

Mô tả hàm xử lý

Ví dụ: Xét đến màn hình tiếp nhận một học sinh mới
Đầu tiên, mô tả các biến cố (các trường hợp kết quả có thể) cho màn hình trên và lập danh sách chi tiết biến cố.

- **Biến cố 0:** Khởi động màn hình
- **Biến cố 1:** Kiểm tra tuổi học sinh hợp lệ (tuổi từ 15 đến 20)
- **Biến cố 2:** Khi chọn một lớp học trên combobox
- **Biến cố 3:** Kiểm tra dữ liệu hợp lệ và ghi

6. Thiết kế xử lý

Mô tả hàm xử lý

Ví dụ:

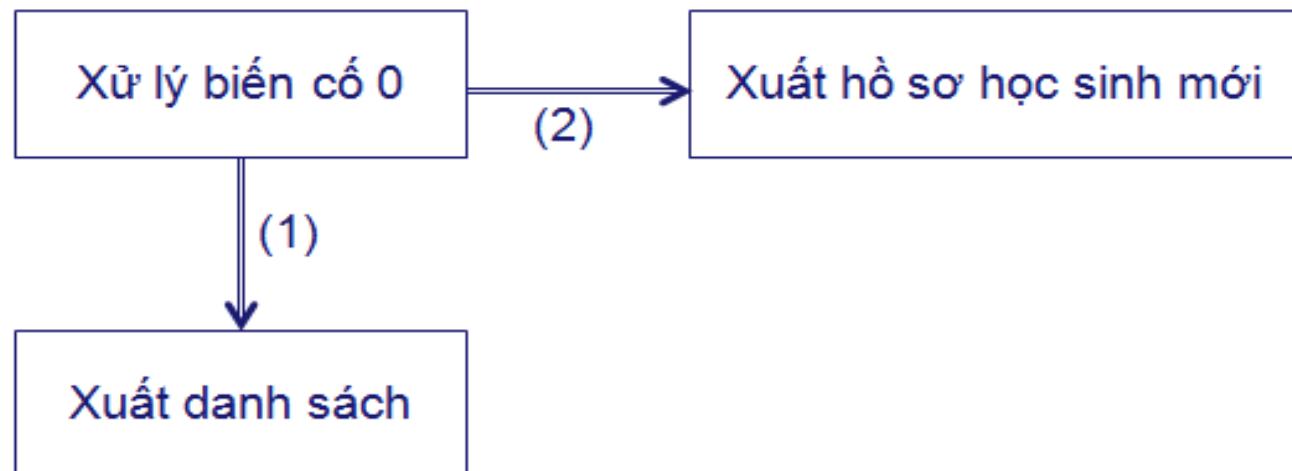
Biến cố	Điều kiện kích hoạt	Xử lý	Ghi chú
0	Khởi động màn hình	Đọc danh sách lớp, danh sách học sinh, tham số Xuất danh sách lớp, danh sách học sinh, hồ sơ học sinh mới	
1	Kết thúc nhập ngày sinh	Kiểm tra ngày sinh hợp lệ và xuất thông báo lỗi nếu không hợp lệ	Tuổi theo qui định từ 15 đến 20
2	Kết thúc chọn lớp	Ghi nhận vị trí của lớp được chọn trong danh sách lớp	Chuẩn bị ghi tên hồ sơ
3	Nhút nút ghi	Kiểm tra hồ sơ hợp lệ. Nếu hợp lệ thì nhập hồ sơ học sinh và ghi hồ sơ học sinh. Xuất thông báo	Mã và tên phải khác rỗng

6. Thiết kế xử lý

Mô tả hàm xử lý

Ví dụ:

Xét mô tả sơ đồ chi tiết từng biến cố. Các kỹ thuật thiết kế có thể dùng là phân rã/tích hợp, tham số hóa và đổi tượng hóa. Với biến cố 0 phân tích ở trên, lập sơ đồ phối hợp như sau:

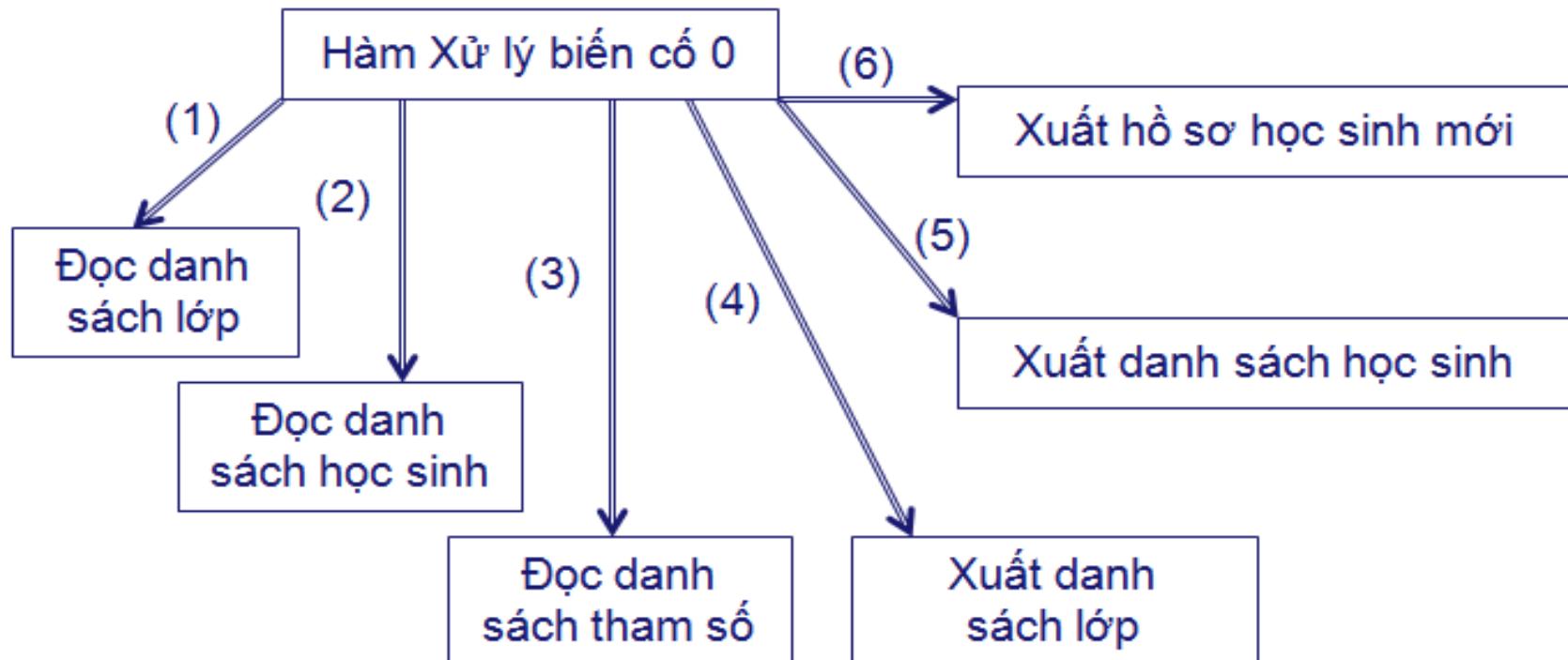


6. Thiết kế xử lý

Mô tả hàm xử lý

Ví dụ:

Khi phân rã hàm từ biến cố 0, chúng ta lại có sơ đồ:

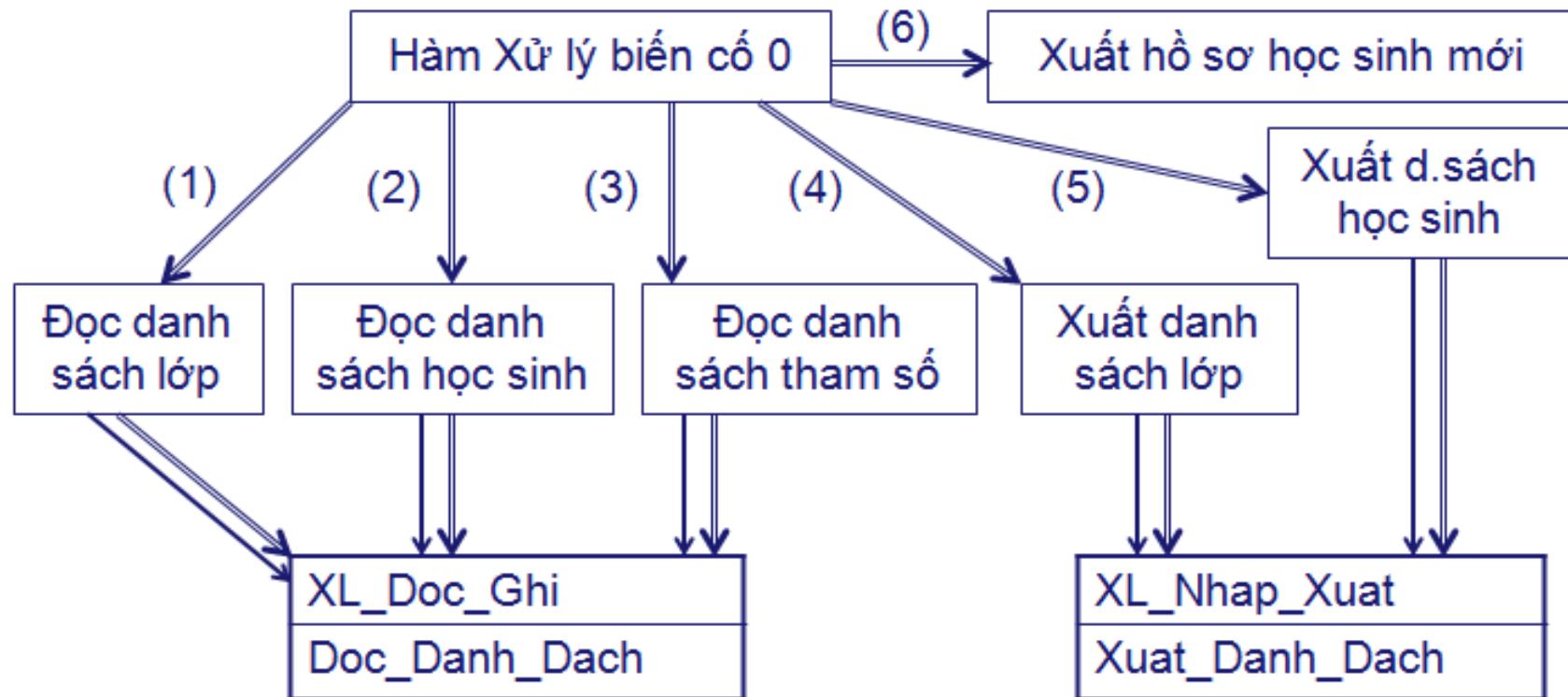


6. Thiết kế xử lý

Mô tả hàm xử lý

Ví dụ:

Chi tiết hơn nữa là:



6. Thiết kế xử lý

Mô tả hàm xử lý

Tùy thuộc vào kiểu thiết kế xử lý, chúng ta có các mô tả biến cố khác nhau.

Vì vậy, từ một hàm xử lý chúng ta có nhiều cách để thể hiện mô hình xử lý miễn sao đảm bảo đúng chức năng của hàm xử lý đó.



THẢO LUẬN





Question?