# Chap 2: Tên có ý nghĩa

## Sử dụng tên bộc lộ mục đích

# Tránh sai lệch thông tin

Tránh đặt những gợi ý sai lầm làm lu mờ ý nghĩa thực sự của code. Ví dụ như viết tắt từ này có thể hiểu nhầm sang từ khác.   
ví dụ:

## Làm cho sự riêng biệt trở lên rõ ràng

Cách đặt tên biến theo kiểu a1, a2 , .. aN tạo ra sự phân biệt nhưng đó thực sự là một cách đặt tên biến tồi.

## Sử dụng tên đọc được

Với các tên không thể phát âm gây khó chịu và rắc rối khi trao đổi giữa các thành viên

## Sử dụng tên tìm kiếm được

Đặt tên một biến có thể tìm kiếm giúp bạn rút ngắn thời gian tìm kiếm tên biến, hàm và lớp đó một cách dễ dàng.

## Avoid Encodings

## Avoid Mental Mapping ( Tránh ánh xạ tinh thần)

## Class name

Tên class nên đặt là danh từ hoặc cụm danh từ

## Method name

Tên phương thức nên đặt là động từ hoặc cụm động từ

## Don’t be cute (Không đặt tên cute, chơi chữ)

## Pick one Word per Concept( Chọn một từ cho mỗi khái niệm)

## Thêm ngữ cảnh có ý nghĩa

## trong phương thức làm việc với địa chỉ , các tên biến như firstName, lastName, state,… có thể làm cho bạn dễ hiểu đó là các biến để lưu thông tin về địa chỉ. Tuy nhiên khi đứng một mình state liệu bạn có dám chắc là nó để lưu thông tin về địa chỉ. Do đó bạn có thể gắn thêm các thông tin về ngữ cảnh cho nó để làm cho người đọc thực sự hiểu nó hơn. Như là addrFirstName, addrLastName.

## 

## Không chơi chữ

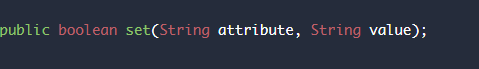
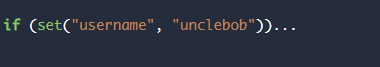
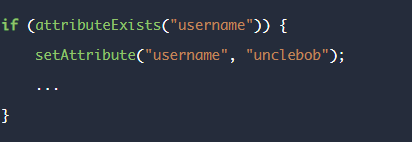
## Use Solution Domain name ( dùng tên của miền giải pháp)

## Use Problem Domain name(Dùng tên của miền nghiệp vụ)

## Don’t Add Gratuitous Context ( Đừng thêm những bối cảnh vô căn cứ)

## Final Words

# Chap 3: Function

1. Small !!!  
   Nguyên tắc đầu tiên viết hàm là hàm đó nên nhỏ và nguyên tắc thứ 2 là hàm đó nên nhỏ hơn nữa.   
   Block and Indenting : Những khối lệnh if, else, while nên được chứa trong một dòng và những dòng này nên đặt thành một lời gọi hàm. Điều này không chỉ giữ cho function nhỏ mà còn cho biết thêm thông tin cụ thể việc nó đang làm bằng một cái tên mô tả độc đáo
2. Do one thing !!! (Làm một việc)   
   Hàm nên được rõ ràng và chỉ nên đảm nhiệm làm 1 thứ.
3. One Level of Abstraction per Function  
   Để chắc chắn rằng 1 hàm làm 1 việc , bạn chri nên gọi 1 level hàm trong 1 hàm. Nghĩa là bạn không nên gọi hàm trong hàm trong một hàm.
4. Switch Statement
5. Sử dụng tên mô tả những gì function làm.  
   Không ngại bởi vì nó là cái tên dài. Tên mô tả dài tốt hơn là một tên ngắn bí ẩn. Tên mô tả dài tốt hơn là một đoạn bình luận mô tả. Sử dụng một quy tắc đặt tên dùng những từ ngữ dễ đọc để làm tên hàm và dùng những từ khóa để mô tả chức năng của nó làm gì.
6. Số lượng tham số lý tưởng trong 1 hàm là 0, 1 sau đó là 2, 3 tham số nên tránh nếu có thể. Nhiều hơn 3 tham số cần sự lí giải đặc biệt.
7. Have No Side Effects(Không có tác dụng phụ)   
   Hàm của bạn hứa hẹn làm một điều , nhưng đôi khi có những phản ứng không mong muốn. Đôi khi nó làm thay đổi bất thường các biến trong class của chính nó. Đôi khi nó để các thông số thông qua và vào hàm hoặc hệ thống toàn cục. Trong cả 2 trường hợp nó sinh ra nhưng hiểu lầm quanh co và tai hại dẫn đến những sự phụ thuộc và liên kết kỳ lạ.  
   Ví dụ: appendFooter (s): Có phải là hàm này nối thêm s vào Footer vào cái gì không? Hay là nối thêm footer vào s? s là input hay output?  
   Sẽ tốt hơn khi gọi appendFooter là : report.appendFooter();
8. Command Querry Separation: Phân tách các lệnh truy vấn  
   Các hàm nên làm một cái gì đó hoặc trả lời một cái gì đó, nhưng không phải cả hai. Hoặc hàm sẽ thay đổi trạng thái của một đối tượng hoặc nó sẽ trả về một số thông tin về đối tượng đó. Làm cả 2 thường dẫn đến nhầm lẫn.   
   Ví dụ  
     
   Hàm này sẽ set giá trị cho 1 thuộc tính và trả về true nếu nó thành công và false nếu không có thuộc tính tồn tại.  
     
   Vấn đề ở đây là : Không biết set dùng theo cách:   
   - Nếu “username” đã tồn tại giá trị “unclebob” trước chưa  
   - Set thuộc tính “username” với gái trị “unclebob”  
   Điều này gây ra nhầm lẫn :  
     
   Ta có thể giải quyết vấn đề này bằng cách đặt lại tên và phân tách các câu lệnh truy vấn để sự mơ hồ không xảy ra:   
   
9. Bạn nên viết hàm như thế nào ?

Viết phần mềm cũng như viết các thứ khác. Khi bạn viết báo hoặc tạp chí. Bạn viết các ý tưởng của bạn đầu tiên, sau đấy chỉnh sửa nó cho tới khi đọc có vẻ ổn. Bản thảo đầu tiên có thể là vụng về và vô tổ chức, vì vậy bạn cơ cấu lại nó và tinh chỉnh nó cho đến khi nó có thể đọc theo cách bạn muốn nó đọc.

Khi viết 1 hàm, chúng quá dài và phức tạp. Có rất nhiều indenting và các vòng lặp lồng nhau. Nó có quá nhiều đối số. Các tên tùy ý, có các mã lặp lại. Và cũng có các đoạn mã unit tests để test chỗ code lôm côm này.   
Vì vậy, sau đó bạn xem lại và tinh chỉnh đoạn mã đó, tách các hàm nhỏ hơn, thay đổi tên, loại bỏ trùng lặp. Bạn cần thu nhỏ các phương thức và sắp xếp lại chúng, đôi khi bạn cần phá vỡ toàn bộ các lớp đã xây dựng từ trước đó.  
Tổng kết :  
Khi sử dụng switch-case, cân nhắc sử dụng ABSTRACT FACTORY.

* Cân nhắc và tìm cách giảm đối số truyền vào của 1 hàm, xuống 1 đến 2 là tốt để có thể test và dễ hiểu.
* Tên hàm cũng cực kì quan trọng, tránh để double-take. Tên đẹp nhất là cum verb/noun(cho loại hàm 1 đối số).
* Nguyên tắc mang tính chất tham khảo, tùy mục đích mà bạn có thể áp dụng cho không. Nếu các project không có các đoạn unit tests hoặc không coi trọng clean, bạn có thể tạo ra các hàm với nhiều đối số, cờ để thuận tiện nhất cho project