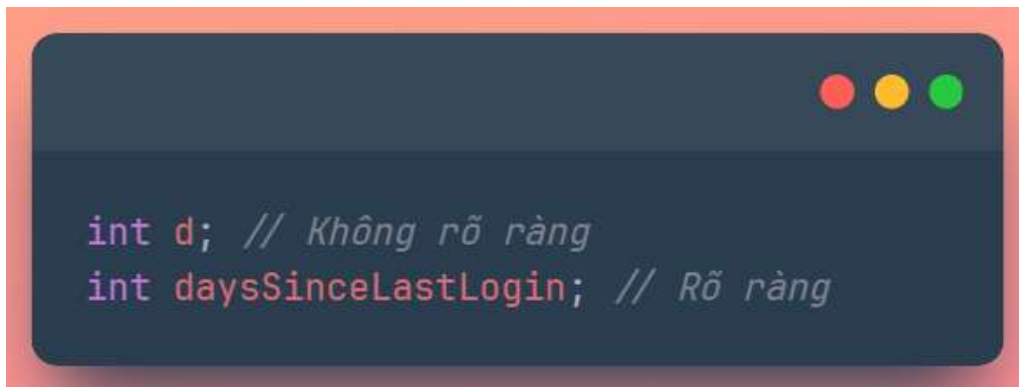


🕒 Cập nhật tháng 8 năm 2024

[Bài đọc] Các tiêu chí cốt lõi của Clean code

1. Tính rõ ràng (Clarity)

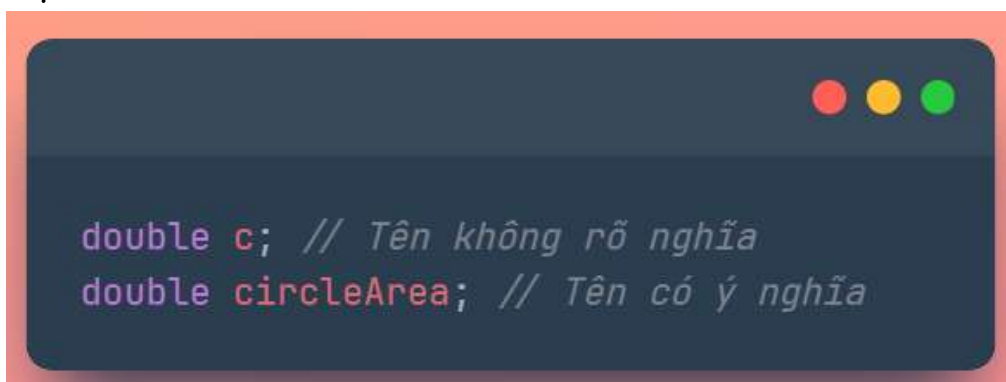
- Clean Code cần **dễ đọc và dễ hiểu** cho bất kỳ ai – kể cả người không viết đoạn mã đó
 - Viết như thể bạn đang giao tiếp với người khác
 - Code càng dễ đọc thì càng dễ bảo trì và mở rộng
- Ví dụ:



```
int d; // Không rõ ràng
int daysSinceLastLogin; // Rõ ràng
```

2. Tên biến và hàm có ý nghĩa (Meaningful Names)

- Tên nên phản ánh đúng chức năng, mục đích của biến, hàm, lớp
- Tránh viết tắt, ký hiệu khó hiểu
- Ví dụ:



```
double c; // Tên không rõ nghĩa
double circleArea; // Tên có ý nghĩa
```

3. Hàm ngắn gọn và đơn nhiệm (Small and Focused Functions)

- Mỗi hàm chỉ nên thực hiện **một nhiệm vụ duy nhất** (Single Responsibility)
- Hàm nên **ngắn gọn**, dễ kiểm soát và dễ test.
- Ví dụ kém:

```
public void handleUser() {  
    validateUser();  
    saveToDatabase();  
    sendEmail();  
}
```

- Clean Code:

```
public void validateUser() { ... }  
public void saveUser() { ... }  
public void sendWelcomeEmail() { ... }
```

4. Không lặp lại (DRY - Don't Repeat Yourself)

- Tránh copy/paste cùng một logic nhiều nơi
- Nên tái sử dụng bằng cách tạo hàm hoặc lớp dùng chung
- Ví dụ:

```
// ❌ Vi phạm DRY  
double area1 = width * height;  
double area2 = width * height;  
  
// ✅ Tuân thủ DRY  
double calculateArea(double w, double h) {  
    return w * h;  
}
```

5. Comment hợp lý (Minimal and Purposeful Comments)

- Code tốt là code tự giải thích
- Comment nên dùng cho:

- Những đoạn logic khó hiểu
 - Giải thích lý do (chứ không phải cách làm)
- Ví dụ:

```
// ❌ Không cần thiết:  
int age; // Biến lưu tuổi  
  
// ✅ Cần thiết:  
// 3s timeout để tránh treo ứng dụng khi API không phản hồi  
int timeout = 3000;
```

6. Format và tổ chức mã (Consistent Formatting)

- Code sạch cần được trình bày rõ ràng, dễ đọc:
 - Cách dòng hợp lý
 - Indent chuẩn
 - Nhóm logic liên quan gần nhau
- Ví dụ:

```
// ❌ Khó đọc  
if(user.isActive()){System.out.println("Active");}else{System.out.println("Inactive");}  
  
// ✅ Dễ đọc  
if (user.isActive()) {  
    System.out.println("Active");  
} else {  
    System.out.println("Inactive");  
}
```

7. Xử lý lỗi rõ ràng (Proper Error Handling)

- Tránh để lỗi xảy ra âm thầm.
- Ưu tiên dùng **try-catch** đúng cách và có log thông tin cần thiết.
- Ví dụ:

```
try {  
    userService.save(user);  
} catch (IOException e) {  
    System.err.println("Failed to save user: " + e.getMessage());  
}
```

8. Tóm tắt

Tiêu chí	Mô tả ngắn
1. Clarity	Code dễ hiểu
2. Tên có ý nghĩa	Rõ chức năng, tránh viết tắt

3. Hàm đơn nhiệm	Mỗi hàm làm 1 việc
4. Không lặp lại	Tái sử dụng code
5. Comment đúng chỗ	Hạn chế, nhưng cần thiết
6. Format rõ ràng	Indent và trình bày logic
7. Xử lý lỗi rõ	Không để lỗi âm thầm

Tài nguyên đọc thêm: <https://www.freecodecamp.org/news/how-to-write-clean-code>

Danh sách các bài học

