

NGÔN NGỮ LẬP TRÌNH JAVA

BUỔI 4 QUI ƯỚC VIẾT MÃ TRONG JAVA



GVGD: ThS. Lê Thanh Trọng

NỘI DUNG

- 1. Tiêu chuẩn viết mã**
- 2. Tiêu chí viết mã**
- 3. Kỹ thuật coding**

- 1. Tiêu chuẩn viết mã**
2. Tiêu chí viết mã
3. Kỹ thuật coding

Tiêu chuẩn viết mã

- ❖ Coding Standards gồm những quy định cách viết code của một chương trình phải/nên tuân theo khi tham gia viết mã chương trình/xây dựng dự án
- ❖ Thông thường bao gồm:
 - Đặt tên: lớp, interface, tên biến, phương thức, ...
 - Trình bày: khoảng trắng, tab
 - Khai báo và sử dụng biến
 - Comment mã nguồn: tên người tạo, phiên bản, ngày tạo file, lớp, phương thức, người thay đổi, nội dung thay đổi, ...
 - Độ dài tối đa mỗi dòng code, mỗi file,...

Vai trò tiêu chuẩn viết mã

- ❖ **Nhất quán**, dễ bảo trì, sửa lỗi
- ❖ Để người khác/chính mình hiểu được mã nguồn của mình
- ❖ Thống nhất code giữa các thành viên trong nhóm, hỗ trợ cộng t
- ❖ Giúp cải thiện chất lượng của hệ thống phần mềm tổng thể
- ❖ Chất lượng, thương hiệu

```
public class SortAlgorithm { private static final int SORT_MIN=1; public  
void InsertionSort(int[] data, int firstElement, int lastElement) { int  
lowerBoundary=data[firstElement-1]; data[firstElement-1]=SORT_MIN; for(  
int sortBoundary=firstElement+1; sortBoundary<=lastElement; sortBoundary  
++) { int insertVal=data[sortBoundary]; int insertPos=sortBoundary; while(  
insertVal<data[insertPos-1]) { data[insertPos]=data[insertPos-1];  
insertPos=insertPos-1; }  
data[insertPos]=insertVal; }  
data[firstElement-1]=lowerBoundary; }
```

VS

```
1 package com.gpooderj;  
2  
3 public class SortAlgorithm {  
4  
5     private static final int SORT_MIN = 1;  
6  
7     public void InsertionSort(int[] data, int firstElement, int lastElement) {  
8  
9         int lowerBoundary = data[firstElement - 1];  
10        data[firstElement - 1] = SORT_MIN;  
11  
12        for (int sortBoundary = firstElement + 1; sortBoundary <= lastElement; sortBoundary++)  
13  
14            {  
15                int insertVal = data[sortBoundary];  
16                int insertPos = sortBoundary;  
17  
18                while (insertVal < data[insertPos - 1]) {  
19                    data[insertPos] = data[insertPos - 1];  
20                    insertPos = insertPos - 1;  
21                }  
22  
23                data[insertPos] = insertVal;  
24            }  
25  
26        data[firstElement - 1] = lowerBoundary;  
27    }  
28 }  
29  
30 }
```

1. Tiêu chuẩn viết mã
- 2. Tiêu chí viết mã**
3. Kỹ thuật coding

Tiêu chí quan trọng

- ❖ Khoảng trắng
- ❖ Ngoặc tròn ()
- ❖ Ngoặc nhọn {}
- ❖ Comment
- ❖ Viết hoa
- ❖ Đặt tên

Khoảng trống

❖ Thụt đầu dòng

- 1 đơn vị thụt đầu dòng = 1 tab
- Hoặc, 1 đơn vị thụt đầu dòng = 5 khoảng trắng
- Hai dòng code cách nhau một mức thì sẽ cách nhau một đơn vị thụt đầu dòng

```
1 package com.gpcoder;
2
3 public class SortAlgorithm {
4
5     private static final int SORT_MIN = 1;
6
7     public void InsertionSort(int[] data, int firstElement, int lastElement) {
8
9         int lowerBoundary = data[firstElement - 1];
10        data[firstElement - 1] = SORT_MIN;
11
12        for (int sortBoundary = firstElement + 1; sortBoundary <= lastElement; sortBoundary++)
13
14            int insertVal = data[sortBoundary];
15            int insertPos = sortBoundary;
16
17            while (insertVal < data[insertPos - 1]) {
18                data[insertPos] = data[insertPos - 1];
19                insertPos = insertPos - 1;
20            }
21
22            data[insertPos] = insertVal;
23
24        }
25
26        data[firstElement - 1] = lowerBoundary;
27
28    }
29
30 }
```


Khoảng trống

❖ Dòng trống

- Những dòng code có quan hệ với nhau (cùng thực hiện một công việc) thì gom lại thành một block (không có dòng trống)
- Hai block code thì cách nhau ít nhất một dòng trống
- Đặt khoảng trắng sau dấu phẩy và dấu chấm phẩy
- Đặt khoảng trắng xung quanh các toán tử

Ngoặc (), ngoặc {}

❖ ()

- Thể hiện rõ mục đích thực hiện (độ ưu tiên các toán tử)
- Trình biên dịch thực hiện đúng yêu cầu

❖ {}

- Phải được đặt cùng dòng với các câu if, for, while
- Bao/gói các lệnh liên quan vào một block

Comment

- ❖ Càng đơn giản càng tốt
- ❖ Nên vừa code vừa viết comment
- ❖ Chỉ viết comment khi đoạn code của mình quá phức tạp
- ❖ Tránh
 - Chỉ mô tả là lặp code, chứ không cung cấp thêm thông tin gì cho người đọc
 - Làm code dài hơn
 - Người đọc tốn thời gian đọc nhiều hơn

❖ Camel style

- Các chữ cái đầu mỗi từ được viết hoa, nhưng chữ cái đầu của từ đầu tiên viết thường
- Ví dụ: myProvider, stringBuilder, loadData(), tinhDiemTrungBinh()

Đặt tên

❖ class, interface

- Từ hay cụm danh từ, thể hiện được ý nghĩa
- Dùng Pascal style: SinhVien, FormSinhVien
- Hạn chế viết tắt: SV → SinhVien
- Tên class nên có thêm những từ có hậu tố phía sau để thể hiện rõ hơn mục đích của class đó, chẳng hạn như TimeoutException
- Tên interface nên có thể chữ I đằng trước. Ví dụ: Itestable, ILoadable

❖ Phương thức

- Tên phương thức thể hiện được chức năng
- Camel style
- Ví dụ: tinhDienTich()
- tinh1(), tinh2() → tinhLuong(), tinhTienDien()

Đặt tên

❖ Quy ước chung

- Tên có ý nghĩa và thể hiện được mục đích của file/ biến/ phương thức,...
- Phân biệt hoa, thường
- Không nên dài quá 20 ký tự hoặc có thể ít hơn nhưng phải đảm bảo đầy đủ về mặt ý nghĩa
- Tránh đặt những tên tương tự nhau, vd: SinhVien, SinhViens
- Tên chứa từ viết tắt cũng nên được hạn chế, diemTB → diemTrungBinh
- Tránh kết hợp nhiều ngôn ngữ khác nhau (Tiếng Anh + Tiếng Việt)
- Không trùng với các từ khóa
- Không được bắt đầu bằng số (có thể là chữ cái, hoặc \$, _)
- Không được chứa khoảng trắng, các ký tự toán học

Biến

- ❖ Mang ý nghĩa
- ❖ Không nên đặt tên biến quá dài, hay quá ngắn
- ❖ Sử dụng Camel style
 - `int soLuongSinhVien;`
- ❖ Không dùng tiền tố
- ❖ Chỉ sử dụng tên biến ngắn khi các biến tồn tại ngắn (như trong vòng lặp for)
- ❖ Trong một số trường hợp, tên biến cần phải thể hiện rõ kiểu dữ liệu của biến đó, ví dụ: biến có kiểu là List thì nên đặt tên là ***studentList***

Biến static, enum, hằng

❖ Các từ được viết hoa và phân cách bằng dấu gạch dưới (_)

❖ Ví dụ

- `final float PI = 3.14f`
- `static int MAX_SOLUONG= 100`
- `enum ShoeSize {
 SMALL, MEDIUM, LARGE, EXTRA_LARGE }`

Package

- ❖ Tất cả đều là chữ thường
- ❖ Mang ý nghĩa
- ❖ Có thể được chia nhỏ: `com.company.[nameofpackage]`

Viết phương thức

- ❖ Chữ cái đầu tiên của từ đầu tiên trong tên phương thức phải viết thường và là một động từ
- ❖ Mỗi phương thức chỉ thực hiện một chức năng
- ❖ Một đoạn code xuất hiện ở nhiều nơi trong chương trình thì gom các đoạn code đó thành một phương thức → Tiết kiệm thời gian bảo trì, sửa lỗi

Phương thức

- ❖ Kích thước của một phương thức: nên từ 50 đến 150 dòng code là hợp lý (Steve McConnell, Chapter 7.4 – Code Complete, Second Edition, 2004)
- ❖ if, while, for không nên lồng nhau hơn 3 mức
- ❖ Chỉ import thư viện sử dụng cần thiết. Không sử dụng import tất cả
 - Ví dụ: sử dụng `import java.util.List` thay cho `import java.util.*`;

NỘI DUNG

1. Tiêu chuẩn viết mã
2. Tiêu chí viết mã
3. **Kỹ thuật coding**

Kỹ thuật coding

- ❖ Các thuộc tính không nên đề phạm vi public
- ❖ Các phương thức nên khai báo tường minh cho tất cả các ngoại lệ unchecked có thể xảy ra
- ❖ Nếu lớp có gọi phương thức clone() thì lớp nên hiện thực giao diện Cloneable và override phù hợp cho phương thức clone()
- ❖ Sử dụng phương thức equal() thay vì == khi so sánh 2 đối tượng
- ❖ Hạn chế việc import với *, thay vào đó import cụ thể lớp cần sử dụng
- ❖ Nếu thường xuyên sử dụng thao tác ghép chuỗi thì nên dùng lớp StringBuffer thay vì String

Kỹ thuật coding

- ❖ Các biến của lớp nên khai báo là null
- ❖ Khi không cần sử dụng nữa thì các biến tham chiếu đối tượng nên được chủ động gán giá trị null
- ❖ Các thuộc tính static nên được khởi tạo trong các khối khởi tạo static
- ❖ Các phương thức public nên được đồng bộ (synchronized)
- ❖ Xây dựng lớp nên chú ý việc cho các lớp sau để mở rộng/kê thừa
- ❖ Luôn sử dụng khối finally (xử lý ngoại lệ) khi có thể