Bài 7. Thiết kế. Giao diện

THCS 4: Lập trình cơ bản với Java

Đỗ Thanh Hà, Nguyễn Thị Minh Huyền

Bộ môn Tin học
Khoa Toán - Cơ - Tin học
Trường Đại học Khoa học Tự nhiên

Nội dung

1. Thiết kế chương trình tốt
2. Tìm lỗi (debug)
3. Giao diện lớp (interface)

Nội dung

Chương trình tốt?

- Đúng đắn (không có lỗi)
- Dễ hiểu
- Dễ sửa đổi, mở rộng
- Hiệu năng cao (tốc độ nhanh)

Thiết kế chương trình tốt

- Có tính nhất quán trong phong cách lập chương trình
  - Chương trình dễ viết và dễ hiểu
  - Tuân theo các quy ước về phong cách lập trình
- Quy ước đặt tên
  - Biến: danh từ, chữ cái đầu viết thường, viết hoa các chữ cái đầu các từ tiếp theo (x, *score*, *fileName*)
  - Phương thức: động từ, cách viết như biến (*getSize*(), *print*())
  - Lớp: danh từ, viết hoa chữ cái đầu của tất cả các từ (*Fraction*, *WebPage*)

Thiết kế chương trình tốt

- Có tính nhất quán trong phong cách lập chương trình
  - Chương trình dễ viết và dễ hiểu
  - Tuân theo các quy ước về phong cách lập trình
- Quy ước đặt tên
  - Biến: danh từ, chữ cái đầu viết thường, viết hoa các chữ cái đầu các từ tiếp theo (x, *score*, *fileName*)
  - Phương thức: động từ, cách viết như biến (*getSize*(), *print*())
  - Lớp: danh từ, viết hoa chữ cái đầu của tất cả các từ (*Fraction*, *WebPage*)

Thiết kế lớp tốt

Lớp được thiết kế tốt: dễ hiểu, dễ sử dụng

- Các trường và phương thức được khai báo ngầm định là riêng (private)
- Chỉ khai báo công khai (public) các phương thức khi cần thiết
- Nếu cần truy cập 1 trường dữ liệu, viết các hàm set và get tương ứng

```
public int getA(){
   return a;
}
```

Nội dung

Tìm lỗi

- Quá trình dò tìm và sửa lỗi (*debugging*) trong chương trình
  - Kĩ năng cơ bản khi lập trình
- Các kĩ năng:
  - Thiết kế chương trình: suy nghĩ trước khi viết mã
    - Viết mã giả mô tả chương trình, tập trung vào cấu trúc
  - Kiểm tra:
    - Thiết kế: lưu ý các giá trị biên
    - Cài đặt: sửa tất cả các cảnh báo (warning) của IDE (*Integrated Development Environment*), dùng công cụ trong IDE (VD: Eclipse)

Nội dung

Giao diện lớp

- Mỗi đối tượng tương tác với các đối tượng khác thông qua các phương thức công khai
  - Các phương thức này tạo thành giao diện lớp
  - Người lập trình không biết phương thức được cài đặt như thế nào, chỉ cần biết đặc tả vào/ra của phương thức
    $\Rightarrow$ Tính đóng gói (*encapsulation*) của lập trình hướng đối tượng
- 2 đối tượng thuộc 2 lớp có thể có các hành vi trên giao diện (nguyên mẫu phương thức) giống nhau, nhưng cách thức thực hiện hành vi (cài đặt phương thức) khác nhau.

Kiểu interface

- Các đối tượng thuộc nhiều lớp khác nhau có thể chia sẻ 1 kiểu giao diện (interface) nào đó
  - Các đối tượng này có tất cả các thuộc tính tĩnh và các phương thức khai báo trong giao diện
  - Ngoài các phương thức đã khai báo trong giao diện này, mỗi lớp có thể có các phương thức riêng.
- Kiểu giao diện interface
  - Được khai báo trong một tệp có đuôi .java, tên tệp trùng với tên giao diện;
  - Được biên dịch thành tệp mã byte có đuôi .class như các lớp (class) thông thường;
  - Không có hàm dựng; có thể có các thuộc tính tĩnh
  - Chứa một số bất kì các phương thức trừu tượng, tức là phương thức chỉ có khai báo nguyên mẫu, không định nghĩa thân hàm;
  - Từ Java 8, interface còn chứa các phương thức ngầm định default và các phương thức tĩnh, các phương thức này có cài đặt thân hàm.

Kiểu interface

- Giao diện là kiểu trừu tượng, không thể tạo đối tượng giao diện.
- Lớp sử dụng (cài đặt) giao diện:
  - Một lớp được khai báo là cài đặt một giao diện bắt buộc phải cài đặt (implements) cụ thể tất cả các phương thức trừu tượng của giao diện. Việc cài đặt này được coi là ghi đè (*override*) lên các phương thức của giao diện.
  - Với các phương thức ngầm định (default), lớp cài đặt giao diện cũng có thể ghi đè lên phương thức này.
  - Lớp cài đặt giao diện có thể định nghĩa phương thức khác có chung nguyên mẫu với phương thức tĩnh của giao diện - đây không phải là thao tác ghi đè. Việc gọi phương thức tĩnh được thực hiện thông qua tên lớp hoặc tên giao diện.
  - Chương trình có thể dùng giao diện làm tên kiểu cho 1 biến đối tượng. Có thể ép kiểu giao diện về kiểu lớp thực tế của biến.
  - Một lớp có thể cài đặt nhiều giao diện.

Kiểu interface

- Giao diện là kiểu trừu tượng, không thể tạo đối tượng giao diện.
- Lớp sử dụng (cài đặt) giao diện:
  - Một lớp được khai báo là cài đặt một giao diện bắt buộc phải cài đặt (implements) cụ thể tất cả các phương thức trừu tượng của giao diện. Việc cài đặt này được coi là ghi đè (*override*) lên các phương thức của giao diện.
  - Với các phương thức ngầm định (default), lớp cài đặt giao diện cũng có thể ghi đè lên phương thức này.
  - Lớp cài đặt giao diện có thể định nghĩa phương thức khác có chung nguyên mẫu với phương thức tĩnh của giao diện - đây không phải là thao tác ghi đè. Việc gọi phương thức tĩnh được thực hiện thông qua tên lớp hoặc tên giao diện.
  - Chương trình có thể dùng giao diện làm tên kiểu cho 1 biến đối tượng. Có thể ép kiểu giao diện về kiểu lớp thực tế của biến.
  - Một lớp có thể cài đặt nhiều giao diện.

Phân biệt khái niệm tải bội và ghi đè phương thức

- Tải bội/nạp chồng phương thức (*overload*)
  - 2 phương thức trong cùng một lớp có cùng tên nhưng danh sách kiểu đối số khác nhau
  - Ví dụ
    - Lớp thư viện java.lang.Math có 4 phương thức tĩnh công khai abs: int abs(int a), long abs(long a), float abs(float a), double abs(double a).
    - Mỗi lớp có thể có nhiều hàm dựng với danh sách kiểu đối khác nhau.
- Ghi đè phương thức (*override*)
  - Một lớp cài đặt một giao diện (hoặc kế thừa một lớp) định nghĩa lại phần thân (phần cài đặt) của một phương thức của giao diện hoặc của lớp mà nó kế thừa.

Phân biệt khái niệm tải bội và ghi đè phương thức

- Tải bội/nạp chồng phương thức (*overload*)
  - 2 phương thức trong cùng một lớp có cùng tên nhưng danh sách kiểu đối số khác nhau
  - Ví dụ
    - Lớp thư viện java.lang.Math có 4 phương thức tĩnh công khai abs: int abs(int a), long abs(long a), float abs(float a), double abs(double a).
    - Mỗi lớp có thể có nhiều hàm dựng với danh sách kiểu đối khác nhau.
- Ghi đè phương thức (*override*)
  - Một lớp cài đặt một giao diện (hoặc kế thừa một lớp) định nghĩa lại phần thân (phần cài đặt) của một phương thức của giao diện hoặc của lớp mà nó kế thừa.

Cách khai báo giao diện

- Tên tệp: <tên giao diện>.java

```
/* File name: NameOfInterface.java */
//import statements
public interface NameOfInterface {
   // final, static fields
   // method declarations
}
```

Cách cài đặt giao diện

- Lớp cài đặt một hoặc nhiều giao diện

```
public class NameOfClass implements
    NameOfInterface1, NameOfInterface2 {
// Method implementation of Interface 1
// Method implementation of Interface 2
// other stuffs (fields, methods) of this class
}
```

Ví dụ

- Shape.java
  Khai báo giao diện Shape, các phương thức trừu tượng getPerimeter và getArea, phương thức ngầm định compareTo.
- Square.java
  Có các phương thức riêng. Ghi đè các phương thức trừu tượng.
- Rectangle.java
  Có các phương thức riêng. Ghi đè các phương thức trừu tượng.
- Circle.java
  Có các phương thức riêng. Ghi đè các phương thức trừu tượng và phương thức ngầm định.
- ShapeTest.java
  Tạo các đối tượng cùng kiểu interface Shape. In ra diện tích với hình vuông hay hình chữ nhật, in diện tích hình vuông ngoại tiếp hình tròn.

Lưu ý khi làm việc với giao diện

- Mỗi lớp cài đặt một giao diện bắt buộc phải định nghĩa ghi đè tất cả các phương thức trừu tượng của giao diện.
- Mỗi khi có thay đổi khai báo phương thức trong một lớp giao diện, cần thay đổi cài đặt trong tất cả các lớp cài đặt giao diện đó.
- Do vậy, cần thận trọng mỗi khi thay đổi khai báo phương thức trong một lớp giao diện.
- $\Rightarrow$ Chỉ định nghĩa các giao diện nhỏ, mang tính ổn định cao.