

Bài 29: Tính đóng gói dữ liệu


- ✓ Mục đích sử dụng
- ✓ Các getter/setter
- ✓ Các phương thức private
- ✓ Minh họa & bài tập thực hành

Mục đích sử dụng

- Tính đóng gói dữ liệu là 1 trong 4 tính chất của ngôn ngữ lập trình hướng đối tượng Java
- 3 tính chất còn lại là: kế thừa, đa hình, trừu tượng
- Tính đóng gói dữ liệu là kỹ thuật gói gọn dữ liệu và các chức năng vận hành trên dữ liệu đó vào trong một đơn vị đơn nhất, là một lớp
- Nói cách khác, tính đóng gói dữ liệu quy định rằng, dữ liệu của lớp nào chỉ lớp đó trực tiếp truy cập và sử dụng
- Các lớp khác muốn sử dụng phải thông qua các phương thức public



Biểu hiện của tính đóng gói

- 
- Các thuộc tính(dữ liệu) được khai báo với access private
 - Có các phương thức getter/setter public để thao tác với các thuộc tính private tương ứng

Ví dụ

```
public class Student {  
    // các thuộc tính:  
    private String studentId;  
    private String fullName;  
    private float avgGrade;  
    private String address;  
    private String email;  
  
    // constructors  
    public Student() {  
        //...  
    }  
    // getter  
    public String getStudentId() {  
        return studentId;  
    }  
    // setter  
    public void setStudentId(String studentId) {  
        if (!studentId.isEmpty()) { // nếu mã sv không rỗng  
            this.studentId = studentId; // cập nhật mã  
        }  
    }  
}
```

Getter và setter

- Để đọc dữ liệu ra ta dùng getter.
- Kiểu của getter luôn cùng kiểu với kiểu của thuộc tính cần đọc
- Tên của getter có dạng getX với X là tên thuộc tính cần đọc
- Getter không nhận tham số
- Để thay đổi dữ liệu của đối tượng ta dùng setter
- Setter thường có kiểu là void với định dạng setX. X là tên thuộc tính cần thay đổi giá trị
- Setter luôn nhận vào một tham số cùng kiểu với kiểu của thuộc tính cần thay đổi giá trị

Getter và setter

- Trong setter có thể có các hành động kiểm tra, so khớp điều kiện trước khi tiến hành cập nhật giá trị để đảm bảo giá trị luôn hợp lệ
- Một thuộc tính có thể có đầy đủ cả getter và setter hoặc có thể chỉ có setter hoặc getter
- Ví dụ:

```
public String getStudentId() {  
    return studentId;  
}  
  
public void setStudentId(String studentId) {  
    if (!studentId.isEmpty()) { // nếu mã sv không rỗng  
        this.studentId = studentId; // cập nhật mã  
    }  
}
```

Các phương thức private

- Phương thức private cũng chỉ có thể được sử dụng nội bộ trong một lớp
- Khi ta loại bỏ các phương thức này sẽ không ảnh hưởng đến bên ngoài
- Thường sử dụng private methods để thực hiện các xử lý nội bộ, helper methods...
- Nếu một nhiệm vụ nào đó có thể chia nhỏ hơn thì ta thường chia nhỏ nó dùng các private method để xử lý
- Khi sinh phương thức trong công cụ IntelliJ mặc định sẽ là private method. Ta chú ý đổi access modifier để sử dụng hợp ngữ cảnh

Ví dụ private method

```
public void setFullName(String fullName) {  
    if (isValidName(fullName)) { // nếu tên sv k rỗng  
        this.fullName = fullName; // cập nhật tên  
    } else {  
        System.out.println("Tên không hợp lệ");  
    }  
}  
  
private boolean isValidName(String fullName) {  
    // nếu tên rỗng, không hợp lệ  
    if(fullName.length() == 0) {  
        return false;  
    }  
    // nếu tên chứa kí tự số hoặc kí tự đặc biệt  
    // thì không hợp lệ  
    if(fullName.matches(".*[^a-zA-Z\\s]+.*")) {  
        return false;  
    }  
    return true; // mặc định tên hợp lệ  
}
```




Ví dụ

- Ví dụ sinh code getter/setter
- Ví dụ sinh code private method

Nội dung tiếp theo

Tìm hiểu về enum