Trong lập trình hướng đối tượng có 4 tính chất là tính đóng gói (*Encapsulation*), tính kế thừa (*Inheritance*), tính đa hình (*Polymorphism*) và tính trừu tượng (*Abstraction*). Trong chương này bạn sẽ học và hiểu được tính chất đầu tiên trong 4 tính chất trên, đó là tính đóng gói.

Tính đóng gói

Tính đóng là kỹ thuật giúp bạn che giấu được những thông tin bên trong đối tượng. Mục đích chính của tính đóng gói là giúp hạn chế các lỗi khi phát triển chương trình.

Để có cái nhìn trực quan về tính đóng gói, bạn hãy xem ví dụ sau:

```
class Student{
    public String name;
    public int age;
    public double gpa;
}
```

Trong ví dụ trên các thuộc tính của lớp `Student` đều là `public` dẫn tới các thuộc tính của lớp này có thể được truy xuất trực tiếp từ nơi khác. Trên thực tế bạn nên hạn chế tối đa các biến có phạm vi truy cập `public` vì nó vi phạm tính đóng gói trong lập trình hướng đối tượng. Để đảm bảo tính đóng gói cho lớp `Student` bạn nên để phạm vi truy cập của các thuộc tính là `private` và truy xuất tới các thuộc tính này thông qua các phương thức `public` (gọi là các `setter`, `getter`) giống như sau:

```
class Student {
    private String name;
    private int age;
    private double gpa;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
```

```
    public double getGpa() {
        return gpa;
    }

    public void setGpa(double gpa) {
        this.gpa = gpa;
    }
}
```

Với cách làm này thông tin của đối tượng đã được ẩn đi, bạn chỉ có thể giao tiếp với đối tượng thông qua các phương thức. Điều này cũng giống với thực tế. Ví dụ khi bạn gặp một người lạ thì bạn không thể biết được các thuộc tính của người này (số điện thoại, sở thích, ...), kể cả khi bạn hỏi thì người này cũng chưa chắc đã trả lời cho bạn đúng sự thật (giống như phương thức không trả về giá trị thực thuộc tính mà trả về một giá trị khác).

Các lợi ích chính mà tính đóng gói đem lại:

- Hạn chế được các truy xuất không hợp lệ tới các thuộc tính của đối tượng.
- Giúp cho trạng thái của các đối tượng luôn đúng. Ví dụ nếu thuộc tính `gpa` của lớp `Student` là `public` thì sẽ rất khó kiểm soát được giá trị, bạn có thể thay đổi `gpa` thành bất kỳ giá trị nào. Ngược lại, nếu bạn để thuộc tính `gpa` là `private` và cung cấp hàm `setGpa()` giống như sau:

```
public void setGpa(double gpa) {
    if(gpa >= 0 && gpa <= 4) {
        this.gpa = gpa;
    }else {
        System.out.println("gpa is invalid");
    }
}
```

  thì lúc này giá trị của thuộc tính `gpa` sẽ luôn được đảm bảo là không âm và nhỏ hơn hoặc bằng `4` (do muốn thay đổi `gpa` thì phải thông qua hàm `setGpa()`).
- Giúp ẩn đi những thông tin không cần thiết về đối tượng.
- Cho phép bạn thay đổi cấu trúc bên trong lớp mà không ảnh hưởng tới lớp khác. Ví dụ ban đầu bạn thiết kế lớp `Student` giống như sau:

```
class Student{
    private String firstName;
    private String lastName;
    public Student() {
        ...
    }
    public String getFullName() {
        return firstName + lastName;
    }
}
```

  Sau này nếu bạn muốn gộp 2 thuộc tính `firstName` và `lastName` thành `fullName` thì lớp `Student` sẽ giống như sau:

Sau này nếu bạn muốn gộp 2 thuộc tính `firstName` và `lastName` thành `fullName` thì lớp `Student` sẽ giống như sau:

```
class Student{
    private String fullName;
    public Student() {
        ...
    }
    public String getFullName() {
        return fullName;
    }
}
```

Lúc này cấu trúc lớp `Student` đã bị thay đổi nhưng các đối tượng sử dụng lớp này vẫn không cần phải thay đổi do các đối tượng này chỉ quan tâm tới phương thức `getFullName()`. Nếu không có phương thức này thì bạn phải sửa tất cả những chỗ sử dụng thuộc tính `firstName` và `lastName` của lớp `Student`.