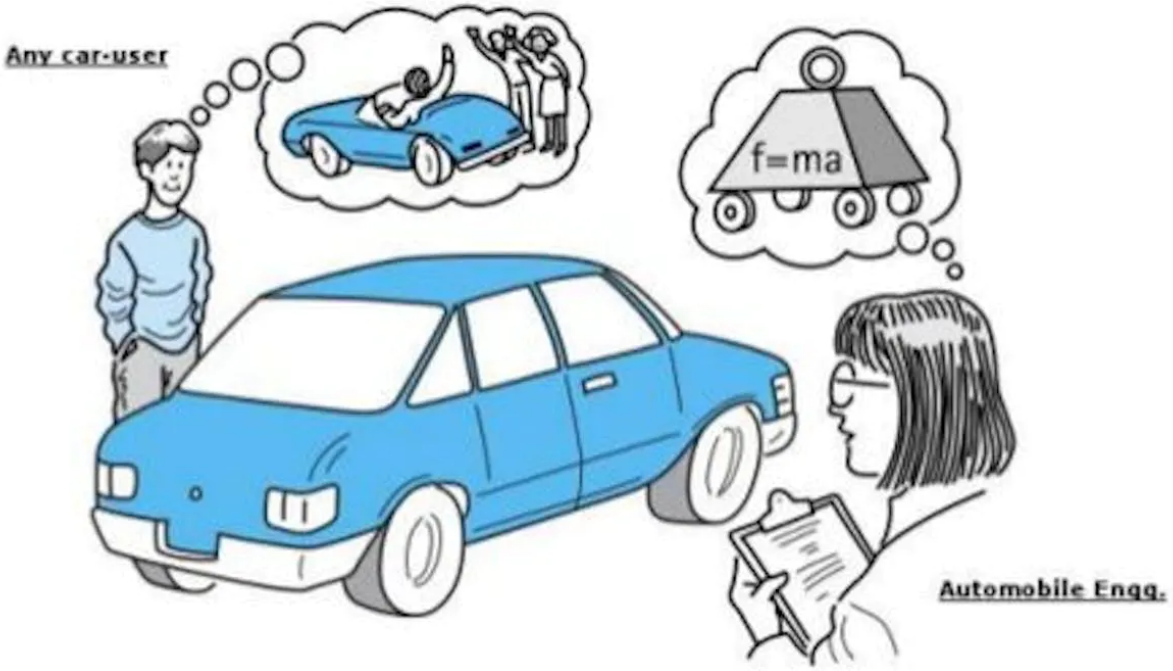


An abstraction includes the essential details relative to the perspective of the viewer

Tránh hiểu sai về Abstraction trong OOP

Tong Hoang Vu

Aug 20, 2022 · 7 min read

TABLE OF CONTENTS

Dân lập trình hẳn không còn xa lạ với OOP và 4 tính chất của nó. Tuy vậy, có một tính chất quan trọng nhưng thường bị hiểu sai, đó chính là tính trừu tượng (Abstraction).

> *Tính trừu tượng giúp loại bỏ những thứ phức tạp, không cần thiết của đối tượng và chỉ tập trung vào những gì cốt lõi, quan trọng.*
>
> *Ví dụ quản lý sinh viên thì chỉ cần quan tâm đến những thông tin như: mã SV, họ tên, ngày sinh, giới tính.*
>
> *Chứ không cần phải quản lý thêm thông tin về: chiều cao, cân nặng, sở thích.*

Trên đây là một định nghĩa hay gặp khi nói đến Abstraction, nhưng nó không hề đúng. Nếu bạn vẫn chưa biết được sai ở đâu, hãy tiếp tục đọc bài viết.

1. Phân tích điểm chưa đúng

Định nghĩa như trên không hề chính xác và còn gây hiểu nhầm, khi cho rằng thực hiện Abstraction là chỉ lựa chọn những thông tin cần thiết đưa vào class (hoặc loại bỏ những thứ không cần thiết).

Có hai vấn đề ở đây:

> Chưa mô tả được cách thức đạt được Abstraction (theo định nghĩa trên)
>
> Không thể hiện được lợi ích khi sử dụng Abstraction

1.1. Chưa mô tả cách đạt được Abstraction

Nhiều trang web hướng dẫn giới thiệu định nghĩa tính trừu tượng như trên xong, sau đó đi tiếp qua abstract class và interface luôn. Người học không cảm thấy giữa định nghĩa và việc dùng hai từ khóa `abstract`, `interface` có liên quan gì với nhau, dẫn đến sự mơ hồ.

Ngoài ra, nếu theo định nghĩa trên, thì không cần dùng `abstract` và `interface` cũng đã đạt được tính trừu tượng rồi. Vậy hai từ khóa đó để làm gì? Ví dụ như class sau.

COPY

```
// Đạt được abstraction theo định nghĩa
// Mà không dùng abstract class hay interface
class Person {
  // Chỉ có các thông tin cần thiết
  private int age;
  private String name;
}
```

Như vậy lúc nào cũng phải có abstraction, vì khi code không thể nào liệt kê tất cả thuộc tính của đối tượng được. Nên nhớ các tính chất khác của OOP đều là tùy chọn, không bắt buộc sử dụng. Ví dụ một class có thể không tuân theo tính đóng gói, không có kế thừa cũng như đa hình.

1.2. Không thể hiện được lợi ích của Abstraction

Vấn đề thứ hai, có 2 lợi ích chính khi dùng tính trừu tượng:

- Đơn giản hóa việc sử dụng chức năng của đối tượng
- Giảm sự phụ thuộc (coupling) giữa các đối tượng

Nếu giải thích Abstraction như định nghĩa đầu bài thì không thể hiện được 2 lợi ích trên. Vì chỉ có một class duy nhất, lúc nào cũng phải dùng class này, nên không thể áp dụng vào được.

2. Định nghĩa chuẩn cho abstraction

Như phần trên, chúng ta biết rằng Abstraction không phải là việc "chỉ lựa chọn những thông tin, chức năng cần thiết". Vậy đâu mới là định nghĩa đúng của Abstraction, và nên hiểu như thế thế nào?

> *Abstraction là việc ẩn đi các chi tiết triển khai bên trong, chỉ hiển thị những gì cần thiết ra bên ngoài.*
>
> *Nói cách khác, người dùng chỉ cần biết có thể thực hiện những gì (what), còn việc thực hiện ra sao thì không cần quan tâm (how).*

Khái niệm như trên vẫn còn khá trừu tượng, không sao, abstract mà. Hãy xem tiếp ví dụ thực tế bên dưới.

2.1. Câu chuyện về cái remote TV

Ví dụ này mình hay dùng để giải thích cho tính trừu tượng.

Bây giờ, mình đưa cho bạn cái remote không có vỏ (phần ở trên) và yêu cầu bạn bật qua kênh 7. Tất nhiên là cuối cùng bạn cũng sẽ tìm được nút nào là kênh 7, nhưng việc đó khá mất thời gian.

Tiếp theo, mình lắp lại phần vỏ cho remote, và yêu cầu bạn chuyển qua kênh 5. Easy game, right?

Như vậy, cái remote không có vỏ vẫn là cái remote, vẫn sử dụng được bình thường. Tuy nhiên, nếu có thêm lớp vỏ bên ngoài nữa thì việc sử dụng sẽ dễ dàng hơn.

2.2. Bản chất của Abstraction

Bản chất của tính trừu tượng là tạo thêm một layer đơn giản hơn để sử dụng, nhằm ẩn đi những thứ phức tạp bên trong.

Như ví dụ về cái remote TV ở trên, phần vỏ chính là abstraction layer. Trên vỏ có những nút kí hiệu, màu sắc giúp người dùng dễ sử dụng. Còn khi bấm nút bên trong

hoạt động như thế nào, dòng điện chạy ra sao không cần quan tâm.

Như vậy, abstraction phải là quá trình **ẩn đi** thông tin và cách triển khai chức năng, chứ không phải việc **lựa chọn** những gì cần thiết để tạo thành đối tượng.

2.3. Sự mơ hồ trong định nghĩa

Hãy xét tiếp một định nghĩa khác như sau, nguồn từ Guru99.

> *Abstraction is the concept of object-oriented programming that **"shows" only essential attributes and "hides" unnecessary information**.*
>
> *The main purpose of abstraction is **hiding the unnecessary details** from the users.*
>
> *Abstraction is **selecting data** from a larger pool to show only relevant details of the object to the user.*

Định nghĩa của Guru99 khác hoàn toàn với định nghĩa sai ở đầu bài. Theo mình nó vẫn chỉ ở mức tạm ổn, nhưng còn hơi mơ hồ:

- Làm người đọc ngầm hiểu chỉ có một đối tượng duy nhất. Không phân biệt ra abstraction layer và implementation.
- Chưa nêu ra cách "show", "hide", "select data" như thế nào. Người học sẽ thắc mắc có phải "show" là dùng `public`, còn "hide" là dùng `private`? Nếu tách biệt ra 2 phần như mục 1 thì sẽ dễ hiểu hơn.

3. Mở rộng vấn đề

3.1. Trường hợp Data abstraction

Như trên, khi class ẩn dữ liệu khỏi bên ngoài thì cũng là thực hiện data abstraction. Bạn có thể thắc mắc, như vậy thì chỉ có một class thôi, làm sao phân biệt abstraction layer và implementation?

Thật ra lúc này, cả abstraction layer và implementation đều nằm chung một class:

> Implementation là toàn bộ mọi thứ của class đó
>
> Abstraction layer là phần public cho bên ngoài sử dụng

Ví dụ như đoạn code sau, một phần class `Person` gồm các public field và public method chính là abstraction layer.

COPY

```
class Person {
  public String name = "Vu";
  private int birthYear = 2001;

  public int getAge() {
    return 2022 - birthYear;
  }
}
```

3.2. Abstraction với các dynamic language

Các ngôn ngữ dynamic như JavaScript,... không có từ khóa `abstract` và `interface`. Như vậy, abstraction có thể thực hiện được hay không?

Câu trả lời là vẫn có abstraction trong các ngôn ngữ này. Tuy nhiên sẽ có đôi chút khác biệt với Java hay C# (static language). Ở đây mình chọn JavaScript để so sánh nhé.

Đầu tiên, JavaScript không có access modifier thực sự, chỉ có các convention phân biệt (dùng prefix `#` cho các private member). Nhưng vậy cũng đủ để có Data abstraction, vì vẫn phân biệt được abstraction layer và phần implementation.

Thứ hai, JavaScript có thể dùng object bất kì làm abstraction layer (do là weak language và có duck typing). Java thì phải khai báo rõ ràng abstraction layer là một interface hay abstract class, JavaScript thì không cần.

COPY

```
function use(duck) {
  // Tham số duck là abstraction layer
  // Có thể gọi method bất kì
  duck.quack()
}

// Implementation thực tế
use({
  quack() {
    console.log('I am a duck')
  }
})
```

Phần implementation thực sự đã được ẩn đi trong ngữ cảnh hàm `use()`. Hàm này chỉ biết rằng tham số `duck` sẽ gọi được method `quack()`, nhưng không biết được `quack()` hoạt động như thế nào.

Như vậy, qua quá trình "tam sao thất bản", mỗi người giải thích theo một cách khác nhau đã làm sai lệch đi định nghĩa của tính trừu tượng trong OOP. Hi vọng các bạn đọc được bài viết này sẽ hiểu được chính xác và không còn bị mơ hồ khi nói về chủ đề này nữa.

Bài viết có tham khảo kiến thức từ các nguồn sau, khuyến khích đọc thêm nhé.

https://www.digitalocean.com/community/tutorials/what-is-abstraction-in-oops

https://www.journaldev.com/33191/what-is-abstraction-in-oops

https://www.guru99.com/java-data-abstraction.html

Bonus thêm post mình tìm được trên Stack Overflow cũng nói về điều này, nhưng không được quan tâm nhiều.

https://stackoverflow.com/questions/64520515/what-is-the-correct-definition-of-abstract-in-oop

oop Java abstraction

WRITTEN BY

Tong Hoang Vu

+ Follow

Code từ trong trứng





1

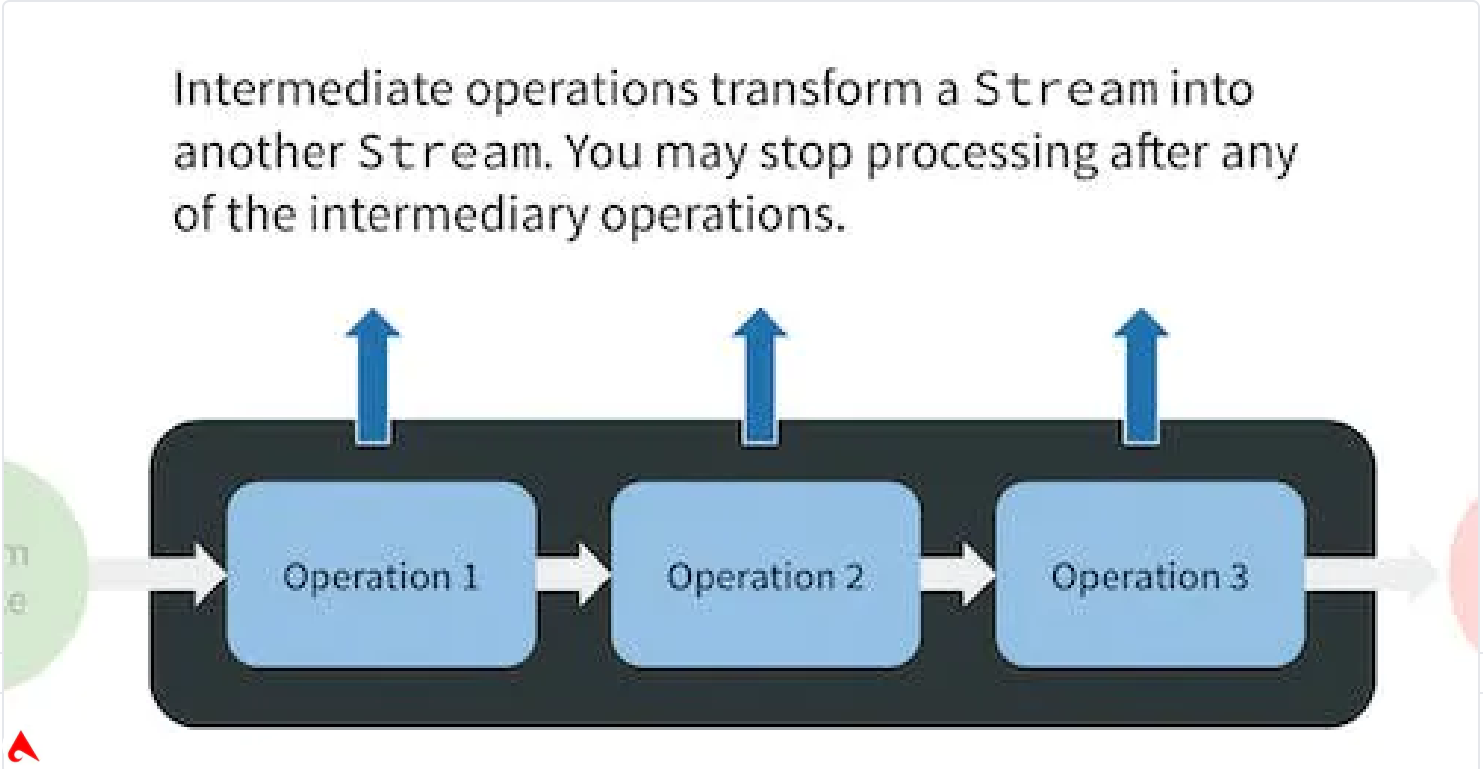
MORE ARTICLES

Tong Hoang Vu

Khoa học là phải thuần túy!

Nay tui ngoi lên nói nhanh vấn đề này thôi. Chả là gần đây xem lại được mấy clip dạng này trên Faceb...

Tong Hoang Vu



Lấy ra stream từ nguồn dữ liệu

Bài trước mình đã giới thiệu về phần lý thuyết của Java stream. Các bạn cũng biết là để xử lý stream...

Comments (1)

+ Write a comment

MB

mben

delphi developer

Dec 3

please read my Vision about the truth Here: from this site << medium >> please search for this title here:

abstraction in oop the truth that no one talks about

--<< sorry i'm new here and i'm not able to embed links here >>---

thanks a lot in advance...

Reply Like