Viết code theo kiểu suy đoán(Speculatively General)

Viết code theo kiểu suy đoán trong lập trình gọi là một **code mùi(code smell)** — tức là một **dấu hiệu** cho thấy có thể bạn đang **thiết kế hệ thống quá phức tạp so với nhu cầu hiện tại**, vì bạn **dự đoán tương lai** và **tạo ra những tính năng chung chung chưa cần thiết**. hiểu đơn giản là khi bạn viết mã không phục vụ yêu cầu hiện tại, mà để "phòng hờ" cho những gì *có thể* xảy ra trong tương lai."

Hãy trở lại bài toán **99 Bottles of Beer**

Giải pháp tiếp theo này đi theo một hướng khác. Nó thực hiện tốt nhiều việc nhưng không thể cưỡng lại việc đắm chìm vào sự phức tạp không cần thiết. Hãy xem mã bên dưới:

```
class BottlesOfBeer:
  def song(self):
    return self.verses(99, 0)

  def verses(self, high, low):
    return '\n'.join(
      self.verse(verse_number) for verse_number in sorted(range(low, high+1), reverse=True)
    )

  def verse(self, number):
    return self.verse_for(number).text()

  def verse_for(self, number):
    def no_more(verse):
      return (
        'No more bottles of beer on the wall, '
        'no more bottles of beer.\n'
        'Go to the store and buy some more, '
        '99 bottles of beer on the wall.\n'
      )
```

```
  def last_one(verse):
    return (
      '1 bottle of beer on the wall, '
      '1 bottle of beer.\n'
      'Take it down and pass it around, '
      'no more bottles of beer on the wall.\n'
    )

  def penultimate(verse):
    return (
      '2 bottles of beer on the wall, '
      '2 bottles of beer.\n'
      'Take one down and pass it around, '
      '1 bottle of beer on the wall.\n'
    )

  def default(verse):
    return (
        f'{verse.number} bottles of beer on the wall, '
        f'{verse.number} bottles of beer.\n'
        f'Take one down and pass it around, '
        f'{verse.number - 1} bottles of beer on the wall.\n'
      )

  lyric_functions = [no_more, last_one, penultimate, default]
  chosen_lyric = (
    lyric_functions[0] if number == 0 else
    lyric_functions[1] if number == 1 else
    lyric_functions[2] if number == 2 else
    lyric_functions[3]
  )
  return Verse(number, chosen_lyric)



class Verse:
  def __init__(self, number, lyrics):
    self.number = number
    self.lyrics = lyrics
  def text(self):
    return self.lyrics(self)
```

Trong đoạn code, mục đích ban đầu **rất đơn giản**: in ra bài hát “99 Bottles of Beer” với một số đoạn đặc biệt khi còn lại 2, 1, và 0 chai.

Tuy nhiên, tác giả đã **tổng quát hóa quá mức cần thiết**, ví dụ như:

1. Việc sử dụng lớp Verse

```
class Verse:
    def __init__(self, number, lyrics):
        self.number = number
        self.lyrics = lyrics
    def text(self):
        return self.lyrics(self)
```

Vấn đề:

Nhưng ở đây, người viết đã tạo:

- Một lớp `Verse`
- Truyền vào số chai và function tạo lyrics
- Gọi method `text()` để in

→ **Dự đoán rằng trong tương lai sẽ cần tách dữ liệu và logic ra như vậy** (có thể để mở rộng? kế thừa? tái sử dụng?). Nhưng hiện tại thì chưa cần. Việc này **gây phức tạp không cần thiết**.

Chỉ cần viết một hàm như sau là đủ

```
def generate_verse(number):
    if number == 0:
        return "No more bottles of beer..."
    elif number == 1:
        return "1 bottle of beer..."
    ...
```

Mục tiêu thực tế là chỉ cần in ra đoạn nhạc tương ứng với 99, 98, ..., 0 chai bia.

Chưa có dấu hiệu cần tái sử dụng, kế thừa

Bạn tạo class khi:

- Có nhiều thuộc tính liên quan cần gói gọn
- Có nhiều hành vi (method) tương tác với dữ liệu đó
- Hoặc bạn muốn tái sử dụng, kế thừa

Nhưng ở đây `Verse` chỉ có 2 thành phần:

- Một con số
- Một function đơn giản trả về chuỗi

Và `text()` thì chỉ gọi đúng 1 dòng:

```
return self.lyrics(self)
```

Không đáng để tạo class. Nếu bạn không **chắc chắn** sẽ cần mở rộng như vậy, thì đó là **suy đoán**

2. Việc chọn lyrics qua `lyrics_functions` list

```
lyric_functions = [no_more, last_one, penultimate, default]
chosen_lyric = (
  lyric_functions[0] if number == 0 else
  lyric_functions[1] if number == 1 else
  lyric_functions[2] if number == 2 else
  lyric_functions[3]
)
```

Vấn đề:

- Việc lưu các function vào list rồi chọn qua index tạo ra một dạng “tổng quát hóa" **khó hiểu** hơn việc viết thẳng `if...elif...else`.
- Cách này **giống như bạn đang chuẩn bị để xử lý 100 kiểu câu hát khác nhau**, nhưng thực tế chỉ có **4 trường hợp** rõ ràng.

Việc này chỉ thực sự có ích **khi số lượng function cần chọn lớn hoặc có cấu trúc lặp đi lặp lại**, ví dụ:

```
lyric_functions[number]  # khi có hàng trăm trường hợp
```

Nhưng trong trường hợp này, chỉ có 4 trường hợp nhỏ và đã biết trước, nên viết như sau **vừa đủ, dễ đọc hơn**:

```
if number == 0:
    chosen_lyric = no_more
elif number == 1:
    chosen_lyric = last_one
elif number == 2:
    chosen_lyric = penultimate
else:
    chosen_lyric = default
```

3. **Đặt tên phương thức `verse_for()`**

```
def verse_for(self, number):
```

→ **Vấn đề**:
Cái tên `verse_for()` làm người đọc tưởng rằng: hàm này có thể được mở rộng/phân nhánh thành hàng tá loại `verse` khác nhau trong tương lai. Nhưng hiện tại nó chỉ dùng để chọn 1 trong 4 đoạn văn bản đơn giản.

Tóm lại: Đâu là phần “Speculatively General”?

Vị trí	Tổng quát hóa gì?	Có cần thiết không?
`Verse` class	Đóng gói logic và data	Không – quá phức tạp cho tác vụ in chuỗi
List `lyric_functions`	Tạo mảng function để chọn	Không – dùng `if-elif` đơn giản rõ ràng hơn
Hàm `verse_for()`	Tên gợi ý khả năng mở rộng lớn	Không – hiện tại chỉ có 4 case đơn giản

---

Phiên bản #4
Được tạo 19 tháng 4 2025 04:44:55 bởi Đỗ Ngọc Tú
Được cập nhật 19 tháng 4 2025 05:41:25 bởi Đỗ Ngọc Tú