6-24 - Adapters

quinta-feira, 13 de março de 2025 07:16

- It's super handy when you need to make **incompatible interfaces work together**
- think of it as a "translator" between two systems.
- Use when
  - ✅ You want to **reuse an existing class**, but its interface doesn't match your needs.
  - ✅ You can't modify the original class (e.g., it's from a third-party library).
  - ✅ You want to **decouple your code** from external interfaces.

The Problem

You have:

- A **client** expecting a specific interface.
- A **legacy class** (or library) with a **different interface**.

You want to **use the legacy class** without changing its code.

Client --> Target Interface
▲
|
Adapter --> Adaptee

Example in C++

Let's say your client expects to use IFahrenheitSensor, but you only have a CelsiusSensor.

✅ Target Interface

This is what the user will see and interact with it

cpp Copy Edit

```cpp
class IFahrenheitSensor {
public:
    virtual double getTemperatureF() const = 0;
    virtual ~IFahrenheitSensor() = default;
};
```

The contract, every derived class must implement this!!

↳ will be used in the high level implementations but not in the concrete usage (here need to define which specialization to use!!!

⚙️ Adaptee (Already exists)

cpp Copy Edit

```cpp
class CelsiusSensor {
public:
    double getTemperatureC() const {
        return 25.0; // Simulated temp
    }
};
```

Legacy code that we don't have access to --

🔄 Adapter

The adapter is a derived class from the interface that will implement the virtual functions

cpp Copy Edit

```cpp
class CelsiusToFahrenheitAdapter : public IFahrenheitSensor {
private:
    CelsiusSensor celsiusSensor;

public:
    double getTemperatureF() const override {
        double celsius = celsiusSensor.getTemperatureC();
        return celsius * 9.0 / 5.0 + 32;
    }
};
```

Use the interface as contract to define what we need to implement in this adapter (low level)

Calling the original logic

Making the proper adjustments to it

🎯 Client Code

cpp Copy Edit

But with the interface type

The instantiation is from

Concrete

```cpp
int main() {
    IFahrenheitSensor* sensor = new CelsiusToFahrenheitAdapter();
    std::cout << "Temperature in F: " << sensor->getTemperatureF() << std::endl;
    delete sensor;
    return 0;
}
```

But uses interface type → The instantiation is from the adaptor

call from the interface and not the specialized class

Concrete implementation

we can change the adapter easelly

[Padrão de Projeto Adapter: Melhore sua Orientação a Objetos e Testes Unitários! Design Pattern (GOF)]

Tem de ter o problema para resolver com ele, não só implementar de cara

command.php | SalesReportGenerator.php

```php
<?php

namespace App\Service;

use DateTime;
use Dompdf\Dompdf;

class SalesReportGenerator
{
    public function generate(): void
    {
        $dompdf = new Dompdf();
        $dompdf->loadHtml( str: 'conteúdo do relatório');
        $dompdf->setPaper( size: 'A4', orientation: 'landscape');
        $dompdf->render();

        $filename = (new DateTime())->getTimestamp() . '.pdf';

        file_put_contents($filename, $dompdf->output());
    }
}
```

Couple

we need to program for abstractions !!

to stability

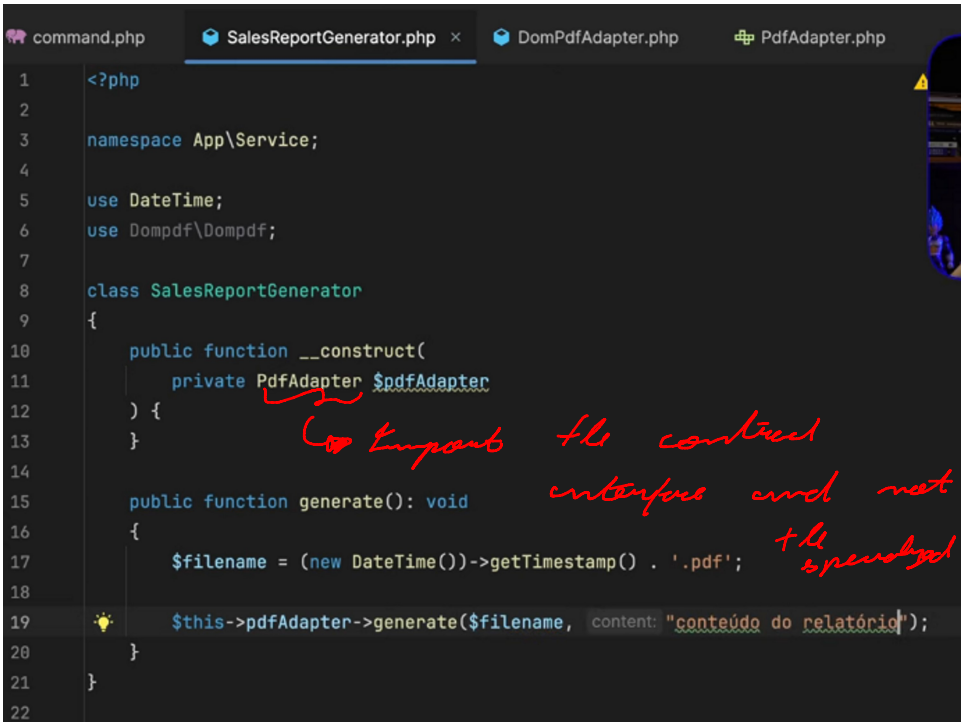
High level class should not be associated to a low level class

ADAPTER

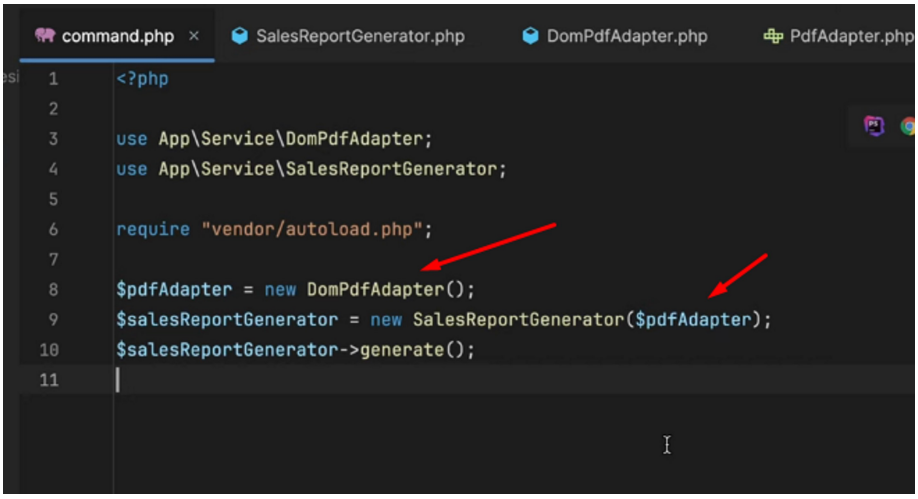
Every time we see a hgh level class depending on a low level implementation. It's wrong and adater can be applied

We have to create na interface so the original class can receive it and use it

Create na adaptor for the low level dependency



Import the contract and the high level class receives the contract only. We don't know how this is implemented or how to maintain this. It just work



In the highest level, the one that uses the SalesReportGenerator in this case, we need to pass the specialized one. to show which type of generator to use. But Since we have a contract.... the inner implementation (that has the actual logic doesn't have to worry with this kind of information because it receives na interface)

Impementação concreata que tem de conter qual tipo de generator usar

Melhora manutenção do código. Fazr objetos fake e passar para dentro na hora de testar. instancia uma classe nova e joga para dentro da classe.

classes de baixo nivel tem de servir o código e não ao contrario