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**.

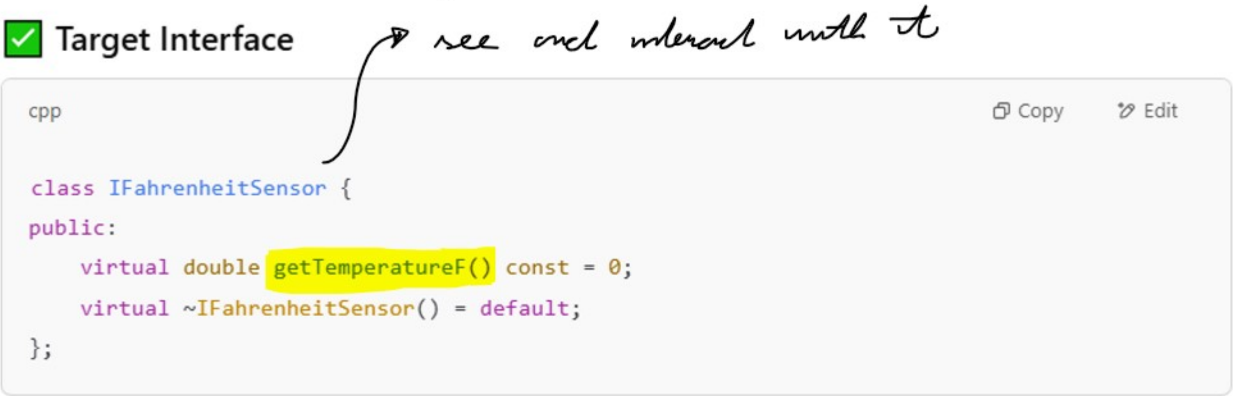
You want to **use the legacy class** without changing its code.

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

Example in C++

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

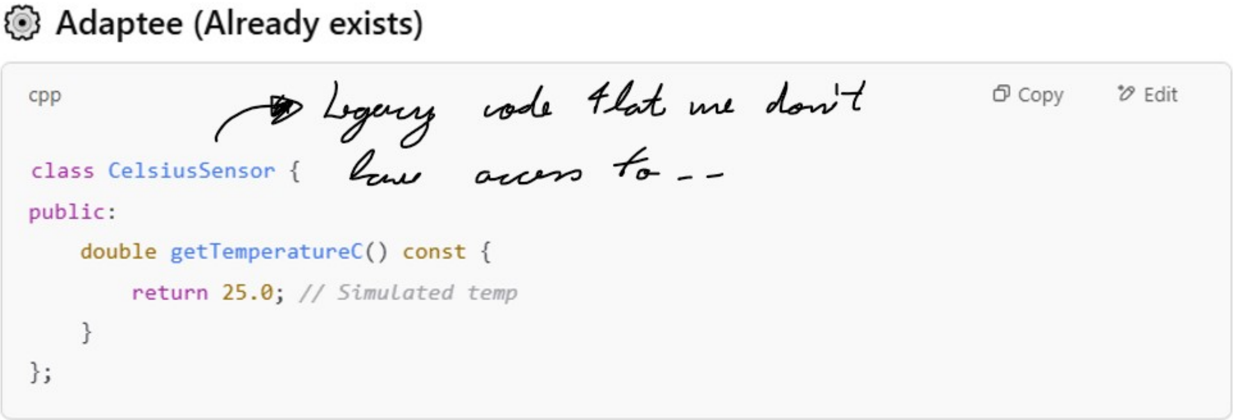
This is what the user will see and interact with it



✅ Target Interface

cpp　Copy　Edit

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

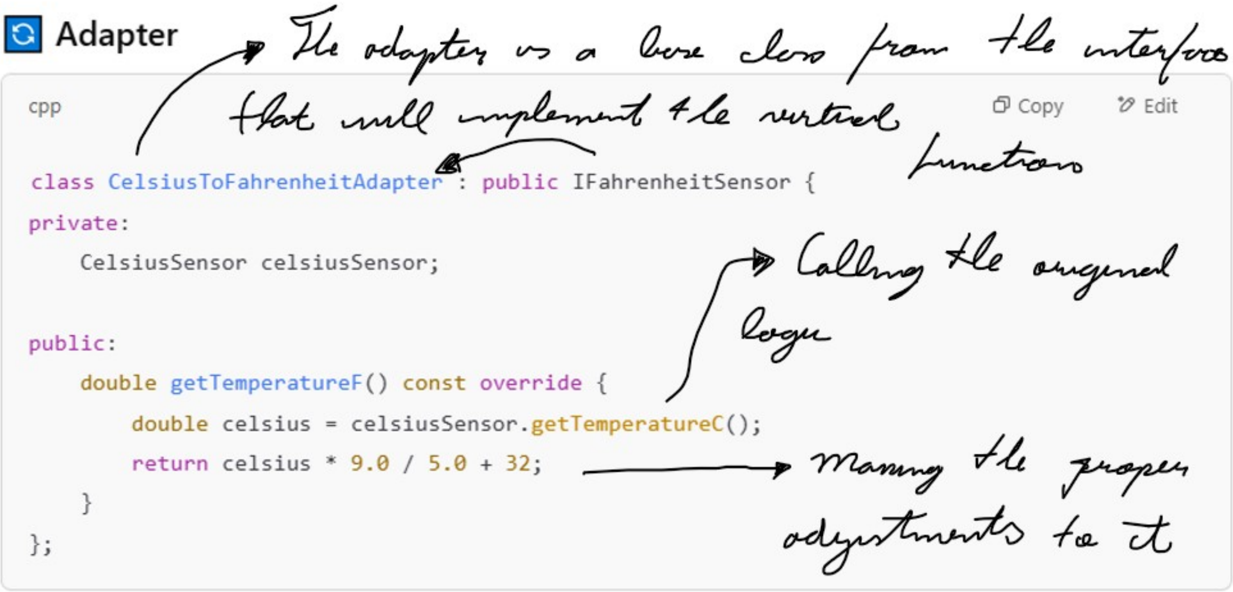


⚙ Adaptee (Already exists)

cpp　Copy　Edit

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

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



🔄 Adapter

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

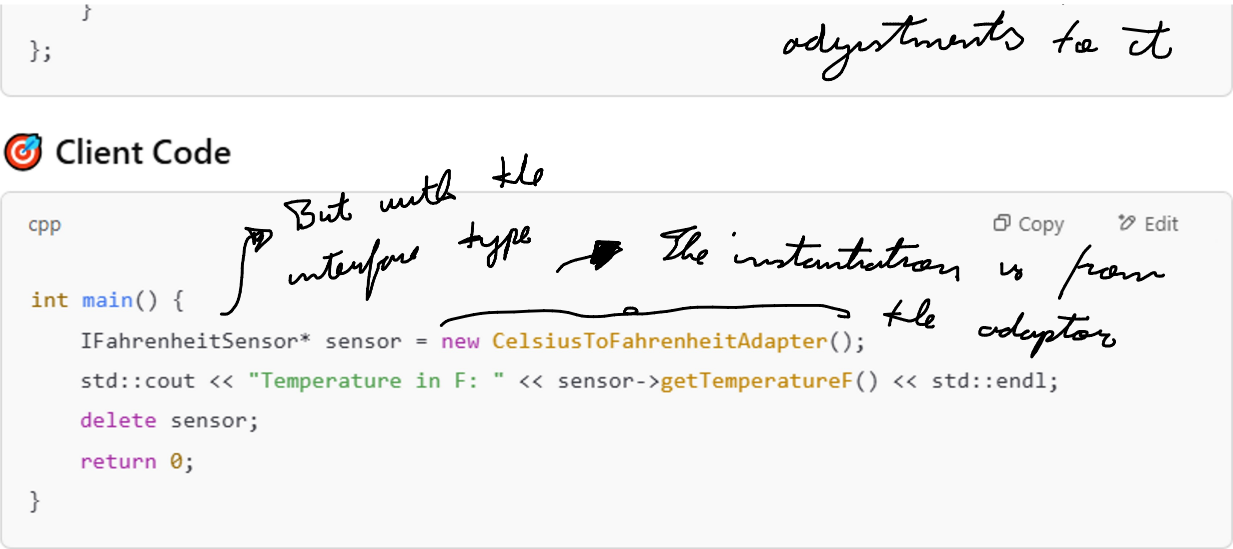
cpp　Copy　Edit

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

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

Calling the original logic

Making the proper adjustments to it



```
    }
};
```

adjustments to it

🎯 Client Code

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

But with the interface type → The instantiation is from the adaptor