Stefan Djokic

ADAPTER

Design Pattern

Simplifed

swipe >>>

Defition:

The adapter pattern converts the interface of a class into another interface clients expect. The adapter lets classes work together that couldn’t otherwise because of incompatible interfaces.

swipe >>>

Stefan from Serbia arrived in the UK.

Stefan has a classic **EU charger** with a **two-part outlet**.

The UK has a standard **three-part** plug.

How will Stefan charge his phone?

Answer: By using an adapter.

Gets one type of input and produces another type of output.

EU interface

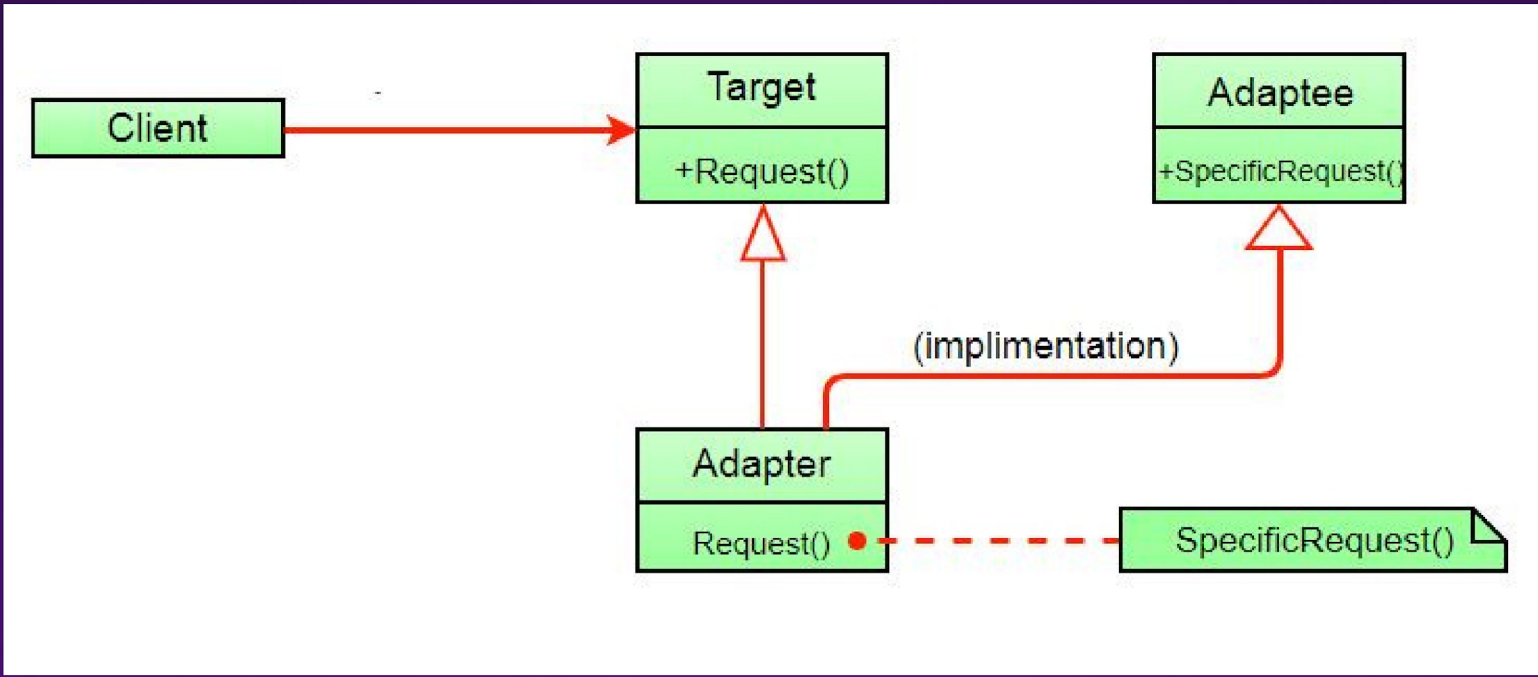
Maps to

UK interface

Adapter

The adapter creates the possibility of connecting 2 different interfaces (2 different sockets).

UML Diagram



Client (user of the code) has something of type Target (which has Request() method). Request is the standard we used to use. We want to use something else - SpecificRequest(). We cannot use it by calling Request() without invoking it in Adapter.

```
public interface ITarget
{
    string GetRequest();
}
```

```
class Adaptee
{
    public string GetSpecificRequest()
    {
        return "Specific request.";
    }
}
```

```
class Adapter : ITarget
{
    private readonly Adaptee _adaptee;

    public Adapter(Adaptee adaptee)
    {
        this._adaptee = adaptee;
    }

    public string GetRequest()
    {
        return $"This is '{this._adaptee.GetSpecificRequest()}'";
    }
}
```

```
static void Main(string[] args)
{
    Adaptee adaptee = new Adaptee();
    ITarget target = new Adapter(adaptee);

    Console.WriteLine(target.GetRequest());
}
```

When to use

- Ado.Net SqlAdapter, MySqlAdapter
- Allow a system to use classes of another system that is incompatible with it.

WANT MORE POSTS LIKE THIS?

CLICK ON THE NOTIFICATION BELL ON MY PROFILE 🔔