



SUMMER SALE



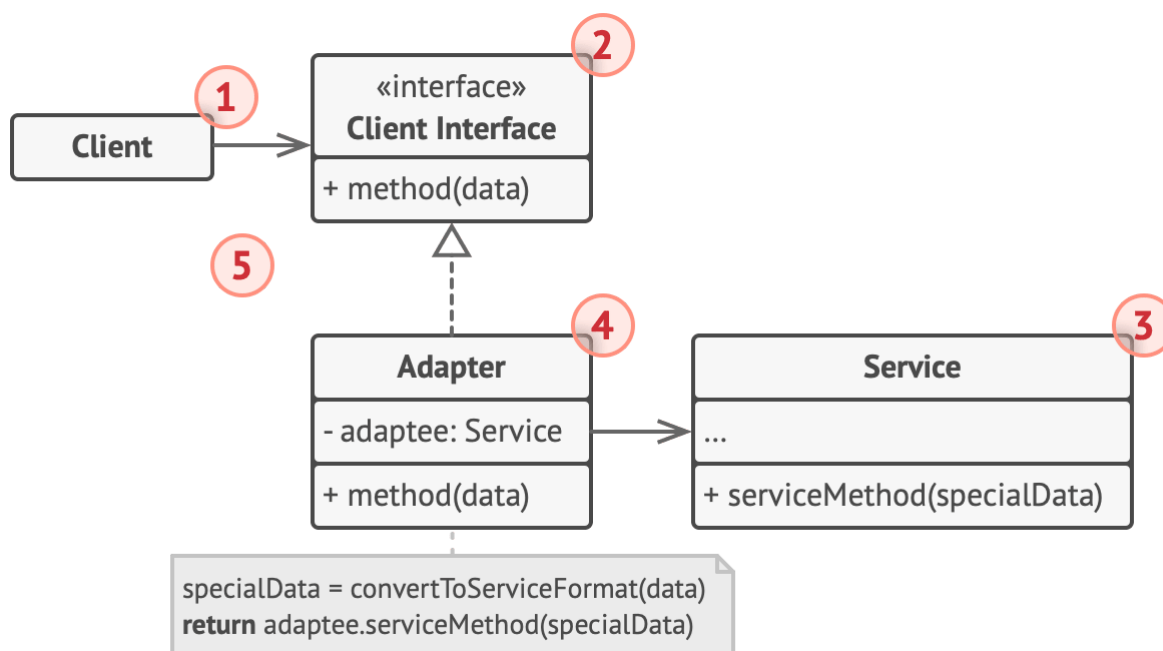
charge your laptop. The power plug and sockets standards are different in different countries.

That's why your US plug won't fit a German socket. The problem can be solved by using a power plug adapter that has the American-style socket and the European-style plug.

Structure

Object adapter

This implementation uses the object composition principle: the adapter implements the interface of one object and wraps the other one. It can be implemented in all popular programming languages.



1. The **Client** is a class that contains the existing business logic of the program.
2. The **Client Interface** describes a protocol that other classes must follow to be able to collaborate with the client code.
3. The **Service** is some useful class (usually 3rd-party or legacy). The client can't use this class directly because it has an incompatible interface.
4. The **Adapter** is a class that's able to work with both the client and the service: it implements the client interface, while wrapping the service object. The adapter receives calls from the



SUMMER SALE

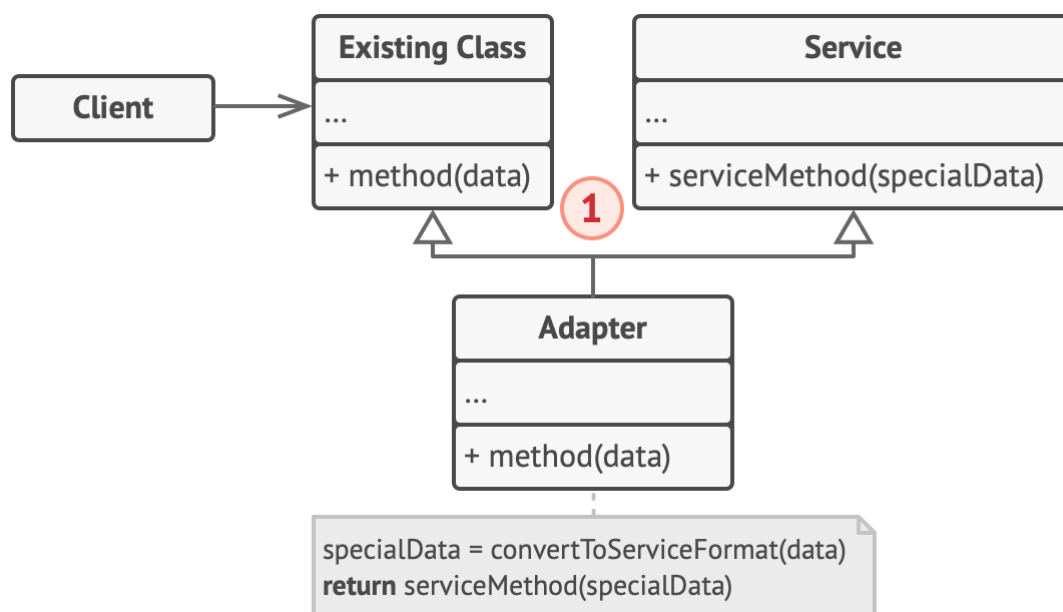


in a format it can understand.

- The client code doesn't get coupled to the concrete adapter class as long as it works with the adapter via the client interface. Thanks to this, you can introduce new types of adapters into the program without breaking the existing client code. This can be useful when the interface of the service class gets changed or replaced: you can just create a new adapter class without changing the client code.

Class adapter

This implementation uses inheritance: the adapter inherits interfaces from both objects at the same time. Note that this approach can only be implemented in programming languages that support multiple inheritance, such as C++.



- The **Class Adapter** doesn't need to wrap any objects because it inherits behaviors from both the client and the service. The adaptation happens within the overridden methods. The resulting adapter can be used in place of an existing client class.

Pseudocode

This example of the **Adapter** pattern is based on the classic conflict between square pegs and round holes.