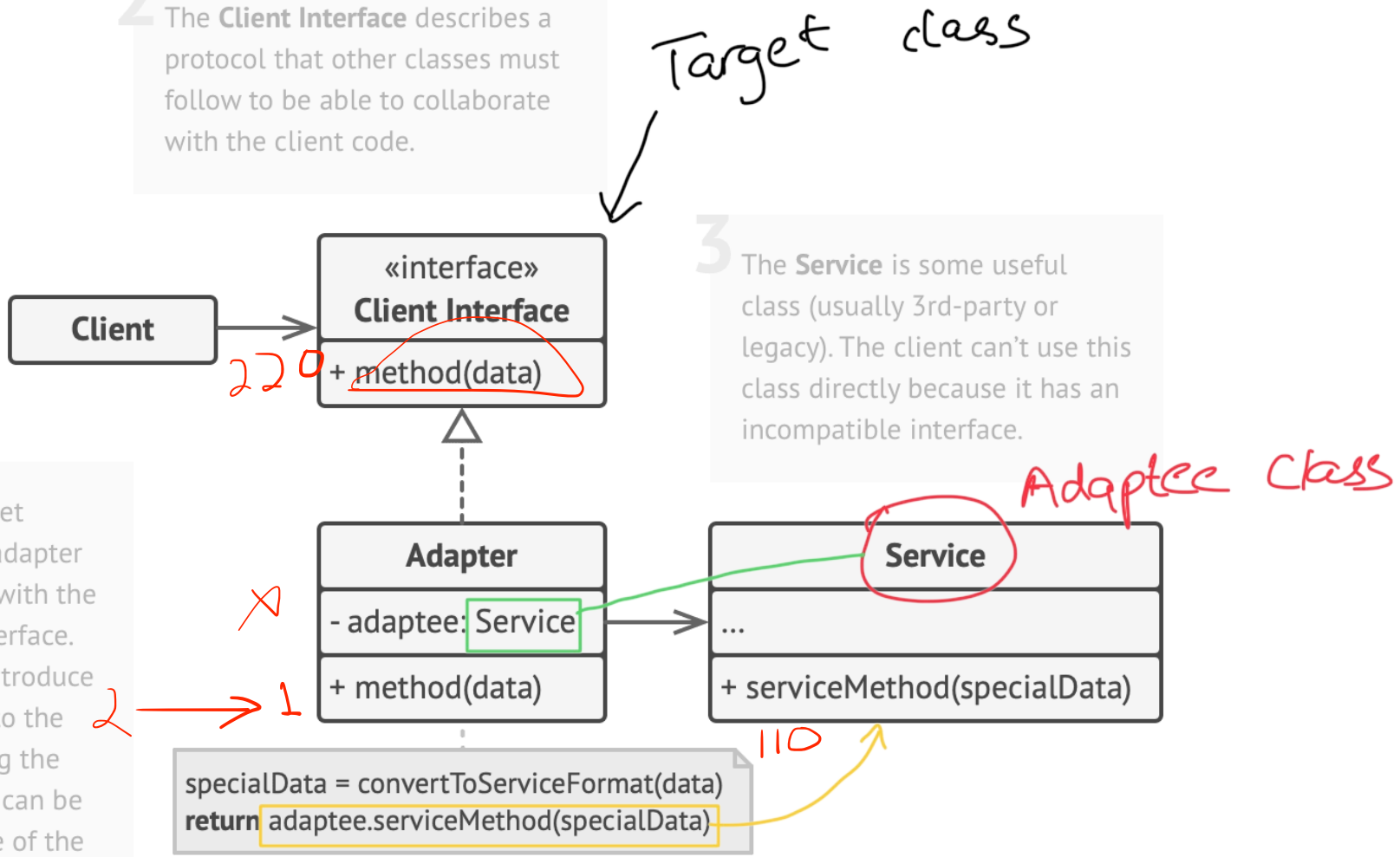
2 The **Client Interface** describes a protocol that other classes must follow to be able to collaborate with the client code.

1 The **Client** is a class that contains the existing business logic of the program.



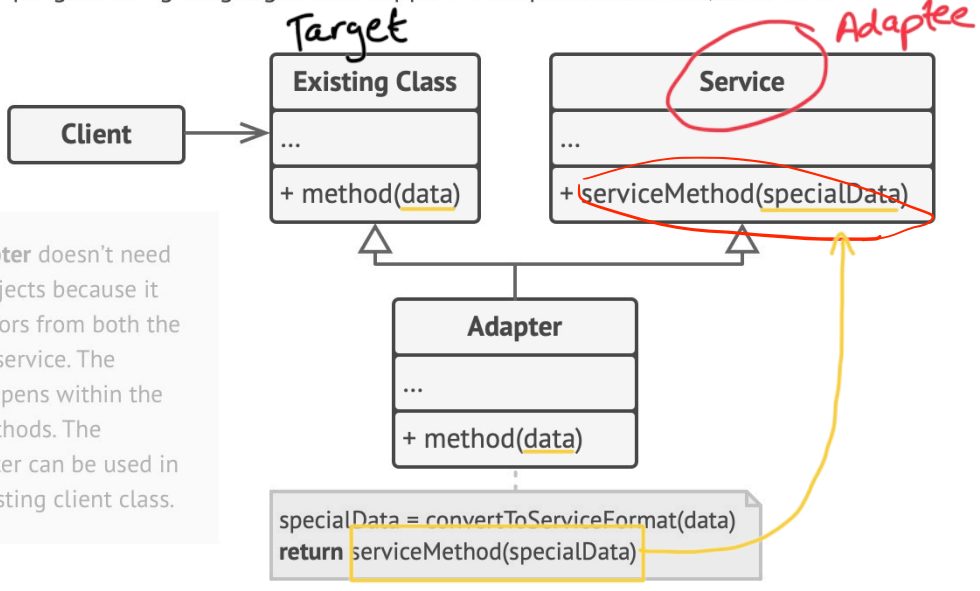
3 The **Service** is some useful class (usually 3rd-party or legacy). The client can't use this class directly because it has an incompatible interface.

5 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.

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 client via the client interface and translates them into calls to the wrapped service object in a format it can understand.

Class adapter

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



1 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.