Adapter Design Pattern :

Lets say we have two interfaces 'A' and 'B'. Both are completely different. We use ~~an Adapter~~ a class for communication b/w A & B interfaces which we call Adapter class.

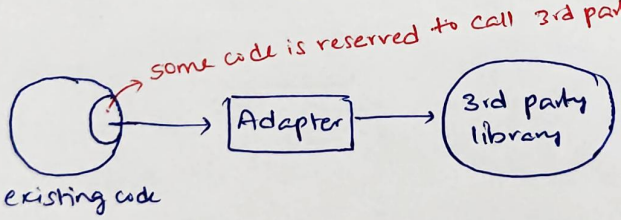
Real-life example : Using 3.5mm jack earphones for a type-c mobile using a dongle.

Need of this pattern :

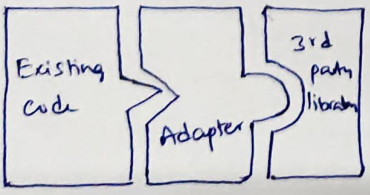
→ When we are developing an application, we may need to integrate Third-party library to our existing code Here our application & 3rd party library are completely different.

→ One way is, we reserve some part of existing code to call the methods of third-party library. But this way we are tightly coupling the third-party library.

→ So, we introduce an Adapter b/w our code and 3rd party



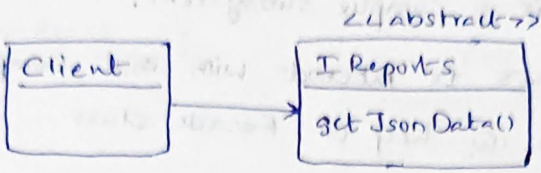
→ Our existing code doesn't know to which 3rd party library we interacting with.



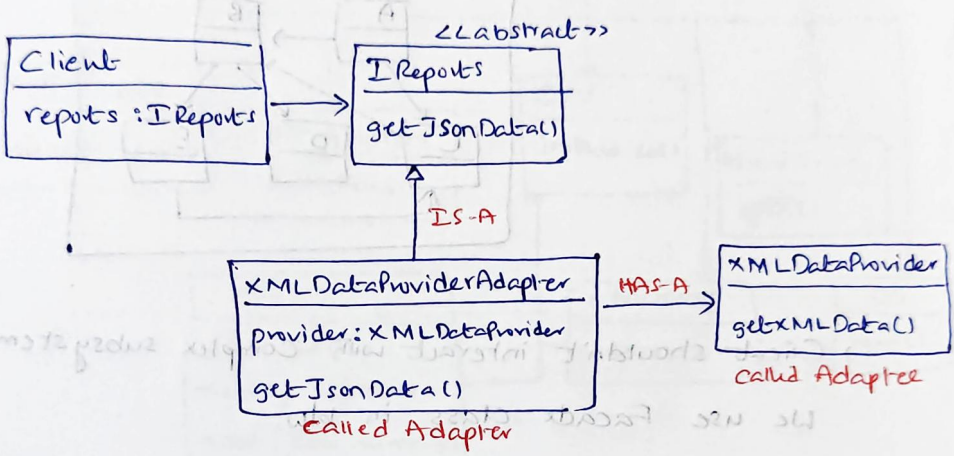
Here existing code and the third party library are completely two different pieces. Therefore we use an adapter to connect them both.

Example :

let Client is fetching some reports in some format



But lets say it doesnt fetch ~~JSON~~ JSON data itself. We have a third party library which fetches in XML format from which we fetch JSON format.

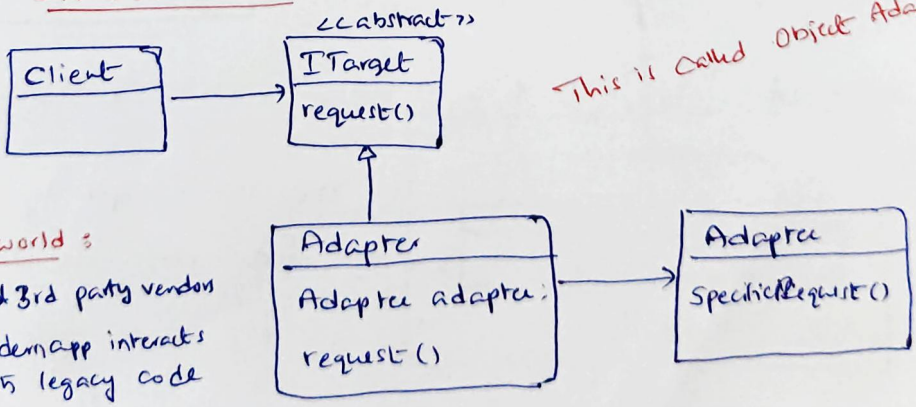


Definition :

Adapter converts the interface of a class into another interface that client expects.

Adapter lets classes, work together that couldn't otherwise because of incompatible interface.

Standard UML :



Real-world :

1. ~~Real~~ 3rd party vendor
2. modern app interacts with legacy code