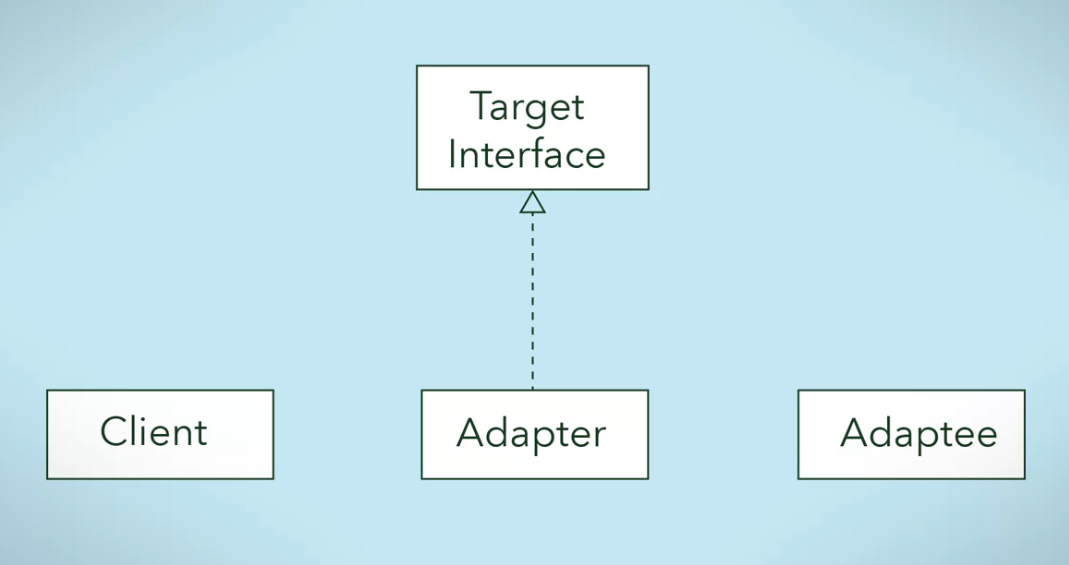
* De ex, vrem sa conectam un Phone la o baterie externa. Avem nevoie de un USB, numit adaptor, pentru a face asta
* Sau de ex, Bateria la telefon are nevoie de 3V, dar priza ofera 120V. Incarcatorul e un adaptor ce face conexiunea intre bateria si priza, desi ele nu sunt compatibile, dar adapter le face cumva sa poata lucra impreuna.
* In Software, rezultatele oferite de o librarie ar putea sa nu fie cele asteptate de alta, care are nevoie sa interactioneze cu ea. Un Adapter va face ca sistemele sa poata lucra intre ele, prin a lura raspunsurile de la un sistem si a le face asa ca sa fie ca cele asteptate de celalat.
* **Adapter** – face ca 2 interfete sau siteme ce nu sunt conectate sau compatibile sa lucreze intre ele.
* **Adapter are 3 componente**:

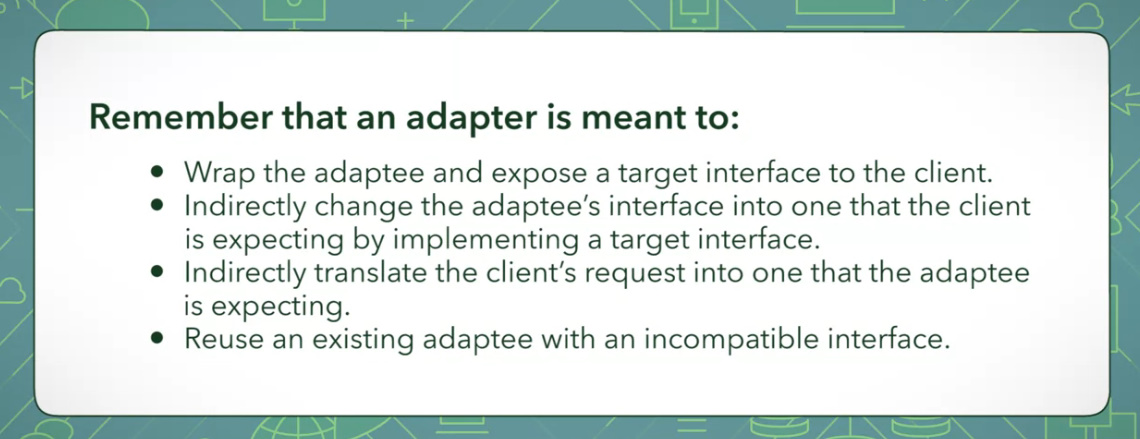


**Client** – sistemul/clasa ce primeste raspunsul

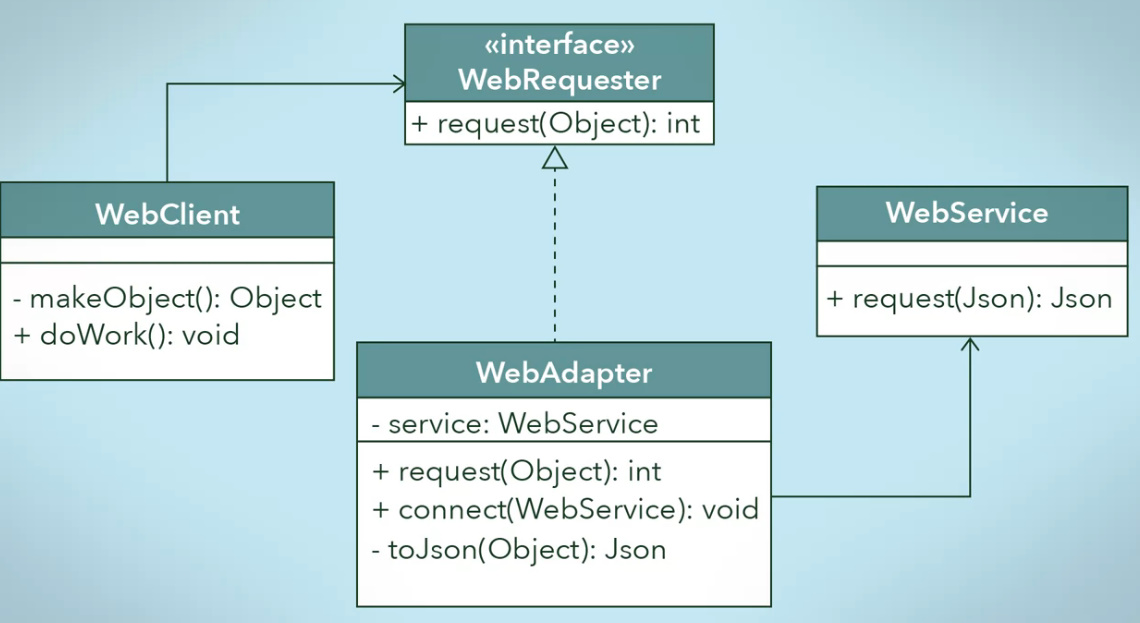
**Adaptee** – sistemul/clasa ce ofera output

**Adapter** – clasa care face ca client sa primesca ceea ce asteapta de la Adaptee. Implementeaza o interfata, cu care Client va lucra

Client va da un request catre Adapter, adapter il va face asa ca Adaptee sa il inteleaga si il trimite la Adaptee. Adaptee ofera un output catre Adapter si Adpter face asa ca Client sa il inteleaga si sa fie asa cum il astepta.

****

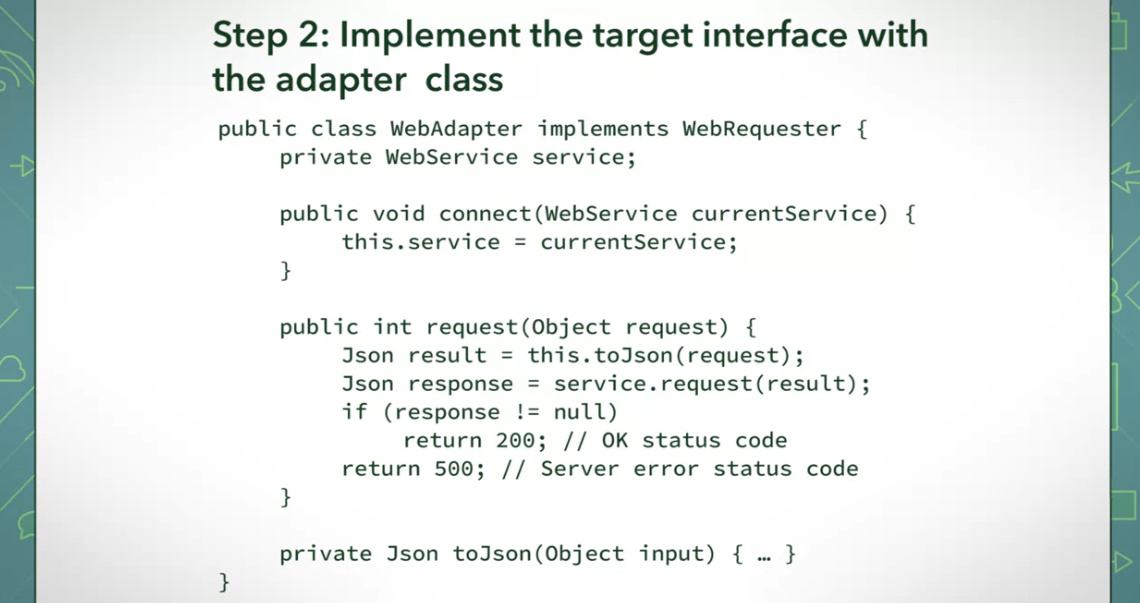
**Exemplu**

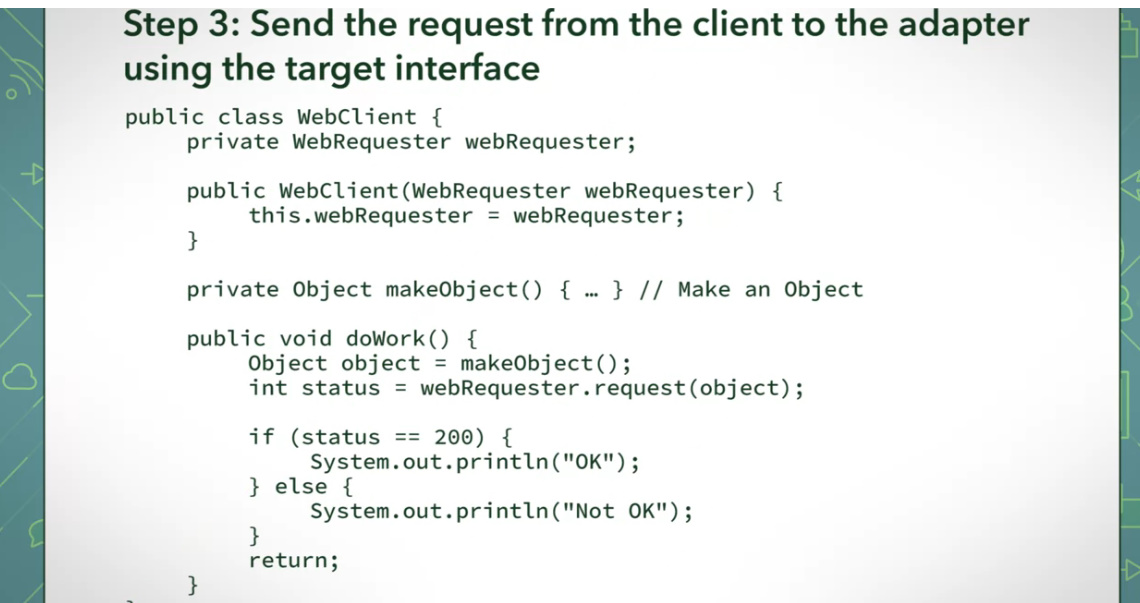


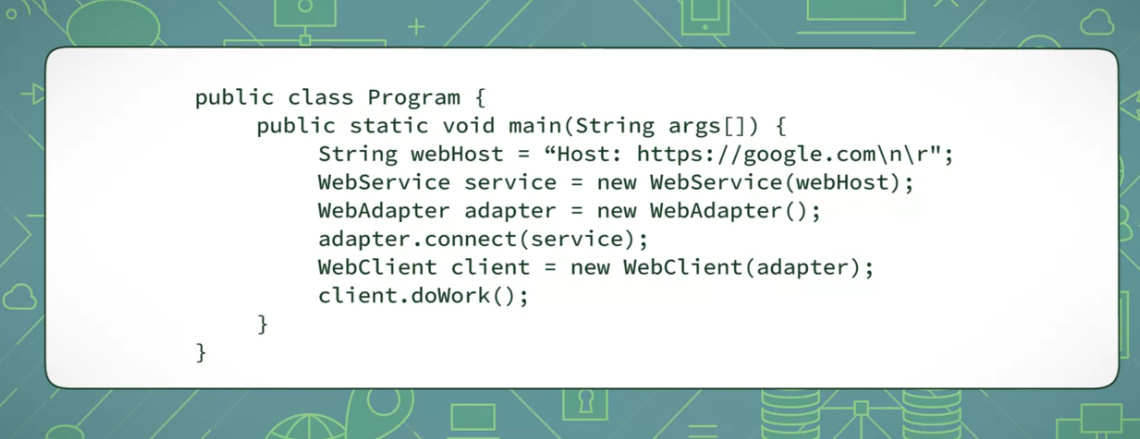
Un WebClient vrea sa comunica cu un WebService. WebClient returneaza un POJO, dar WebService vrea un JSON, nu POJO.

WebAdapter va avea grija sa preia POJO si sa il transforme in JSON, ca sa il trimita la WebService









E o idee rea sa modificam propriu zis sistemele sau clasele ca ele sa fie compatibile. De ex, mai sus nici nu stim implementarea la WebService si posibil nici sa nu avem acces la ea, asa cum e o librarie poate.