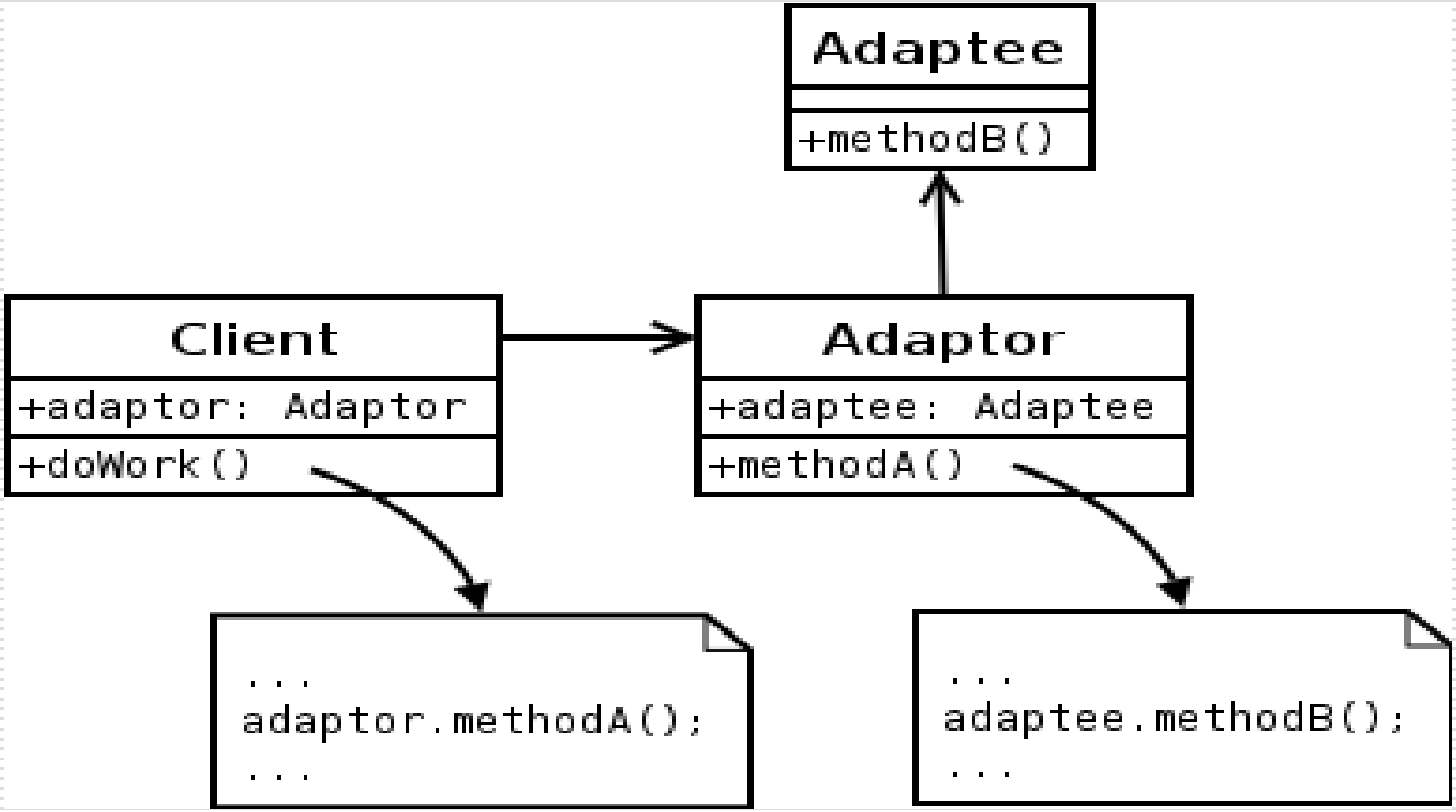
Object Adapter Pattern

Structural Pattern

Intent

- To allow objects to work together that have otherwise incompatible interfaces
- To convert the interface of an Object to another that the client expects*
- Also known as a Wrapper

Description



Example

```
class Customer{
        void setRestaurantData(String address, String city, String state, String zip){
                // do some calculation
        }
} – Standard customer

interface CustomerAdapter{
        void setRestaurantData(String lat, String lng);
}  - Certain customers give us different geocoded addresses.
     - Allows us to save to our database for later retrieval
     -  Create a new object, and tailor the parameters to fit into existing structure

class CustomerAdapterImpl implements CustomerAdapter{
        private Customer customer = new Customer();
        public void setRestaurantData(String lat, String lng){
                // calculate latitude and longitude
                // return address, city, state, zip
                customer.setAddress(address, city, state, zip);
        }
}
```

Advantages and Disadvantages

- Advantage: Adapter can add functionality to many Adaptees. CustomerAdapter can be more abstract and adapter more than just customer object.
- Disadvantage: Harder to override Adaptee behavior. Customer object behavior can’t be changed without subclassing it.