

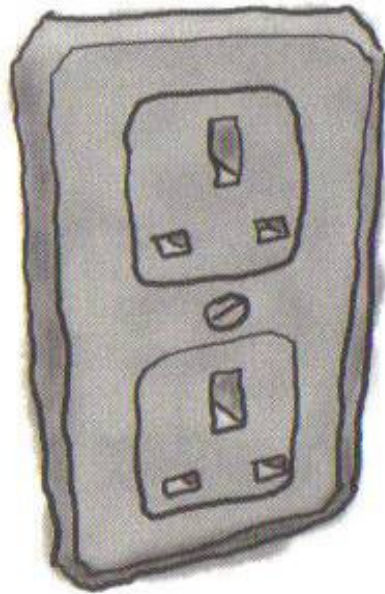
# Tecniche di Progettazione: Design Patterns

GoF: Adapter

# Adapters in real life

---

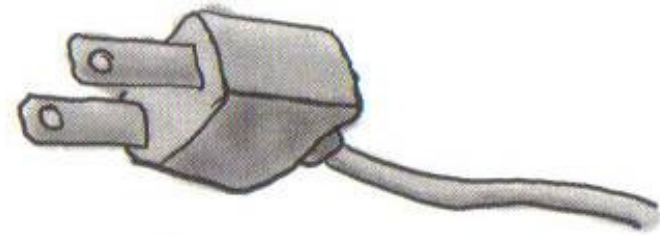
European Wall Outlet



AC Power Adapter



Standard AC Plug

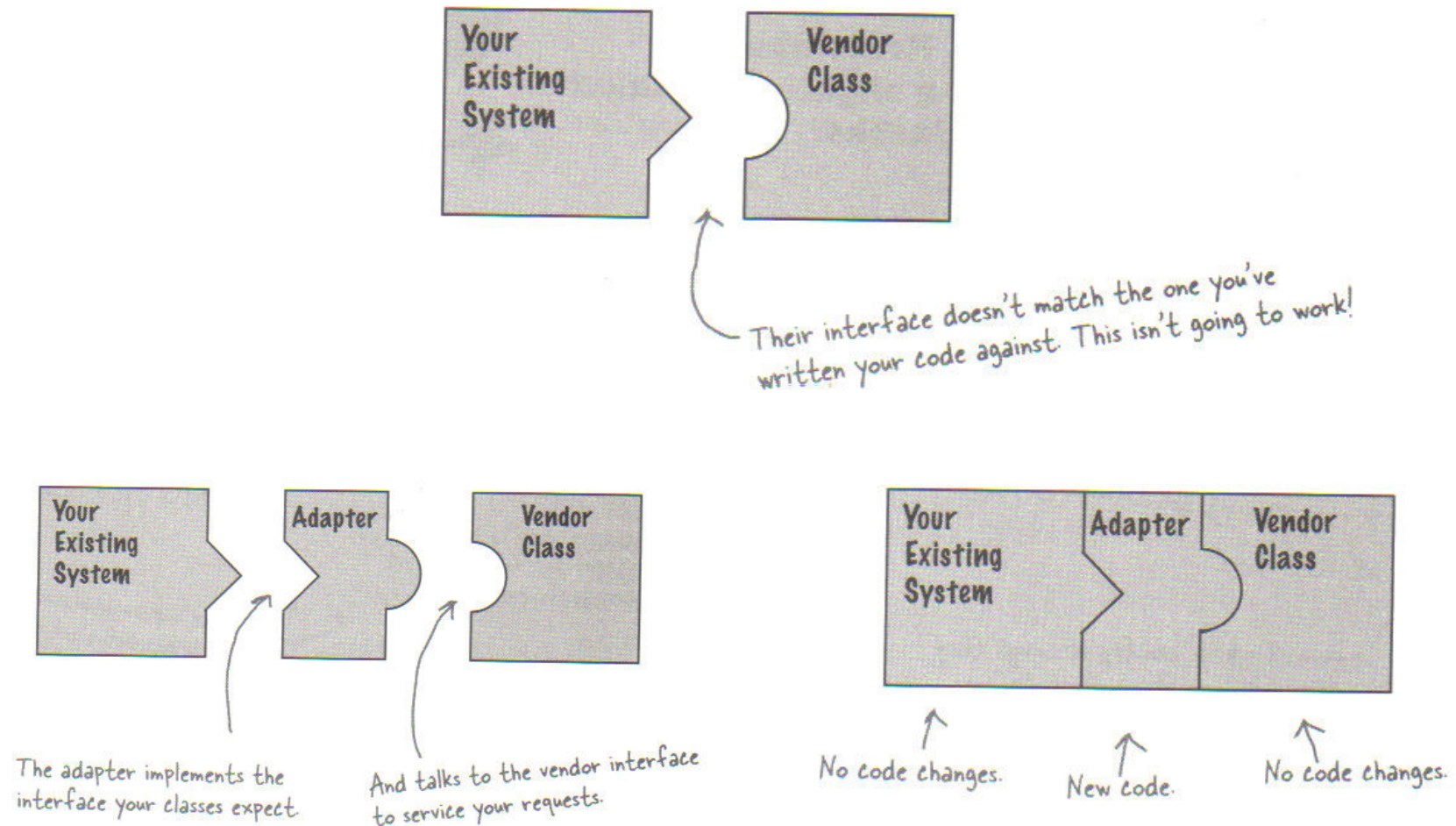


The European wall outlet exposes one interface for getting power.

The US laptop expects another interface.

The adapter converts one interface into another.

# Object-Oriented Adapters



# Hugly Duckling example

---

```
public interface Duck {  
    public void display();  
    public void swim();  
}
```

```
public interface Swan{  
    public void show();  
    public void swim();  
}
```

```
public class Duckling implements Duck {  
    public void display() {  
        System.out.println("I'm a pretty little  
        duckling");  
    }  
    public void swim() {  
        System.out.println("I'm learning...");  
    }  
}
```

```
public class HuglyDuckling implements Swan{  
    public void show() {  
        System.out.println("I'm large and hugly");  
    }  
    public void swim() {  
        System.out.println("I'm swimming!");  
    }  
}
```



# Turkey adapter – that makes a turkey look like a duck

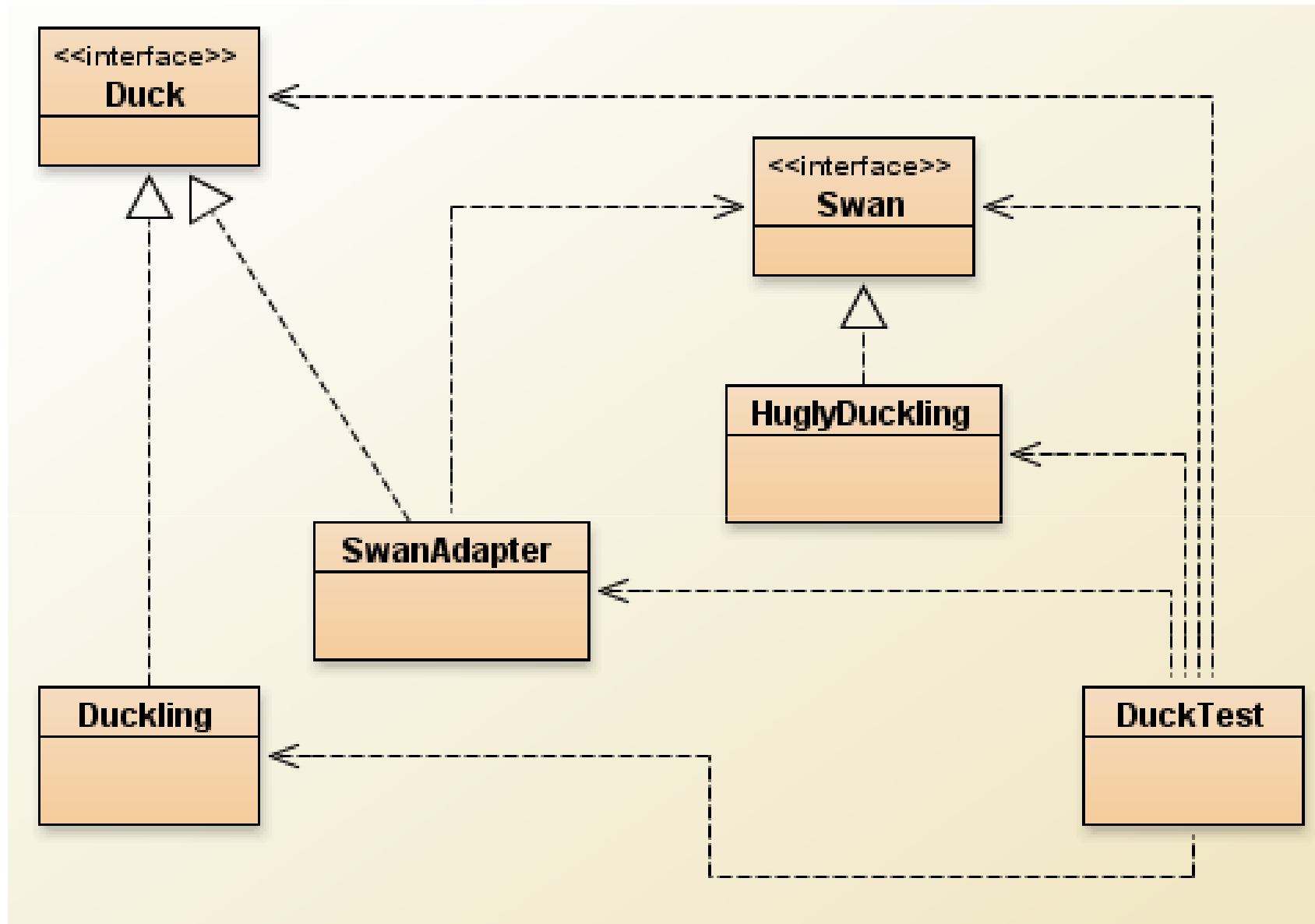
---

```
public class SwanAdapter implements Duck {
    Swan swan;
    public SwanAdapter(Swan swan) {
        this.swan = swan;
    }
    public void display() {
        swan.show ();
    }
    public void swim() {
        for(int i=0; i < 3; i++) {
            swan.swim();
        }
    }
}
```

# Duck test drive

---

```
public class DuckTest {
    public static void main(String[] args) {
        Duck duck = new Duckling();
        Swan hg= new HuglyDuckling();
        Duck swanAdapter = new SwanAdapter(hg);
        System.out.println("The HuglyDuckling says...");
        hg.show();
        hg.swim();
        System.out.println("\nThe Duck says...");
        testDuck(duck);
        System.out.println("\nThe SwanAdapter says...");
        testDuck(swanAdapter);
    }
    static void testDuck(Duck duck) {
        duck.display();
        duck.swim();
    }
}
```



# Test run – turkey that behaves like a duck

---

The HuglyDuckling says...

I'm large and hugly

I'm swimming!"

The Duck says...

I'm a pretty little duckling

I'm learning...

The SwanAdapter says...

I'm large and hugly

I'm swimming!"

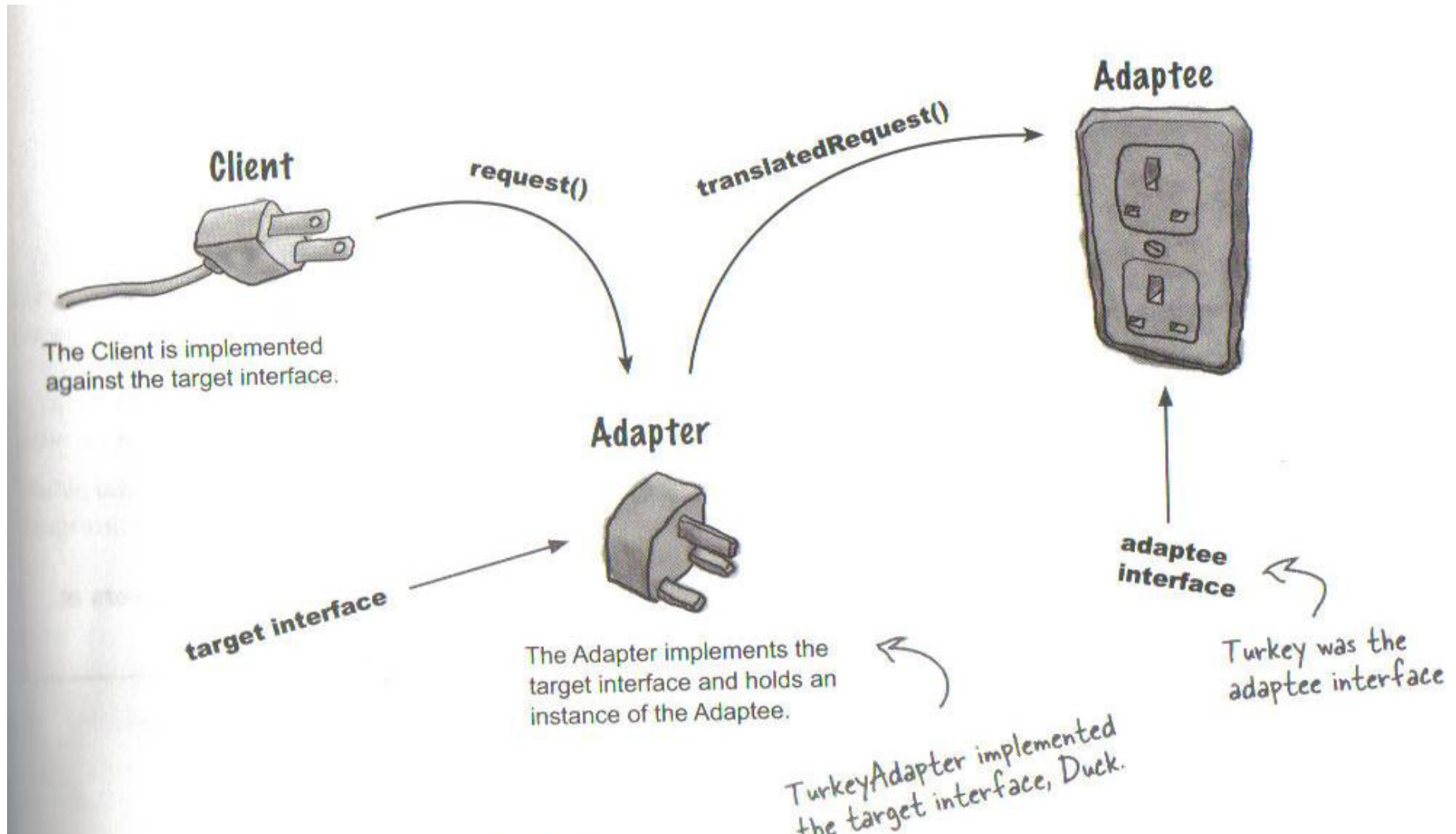
I'm swimming!"

I'm swimming!"





# Adapter Pattern explained



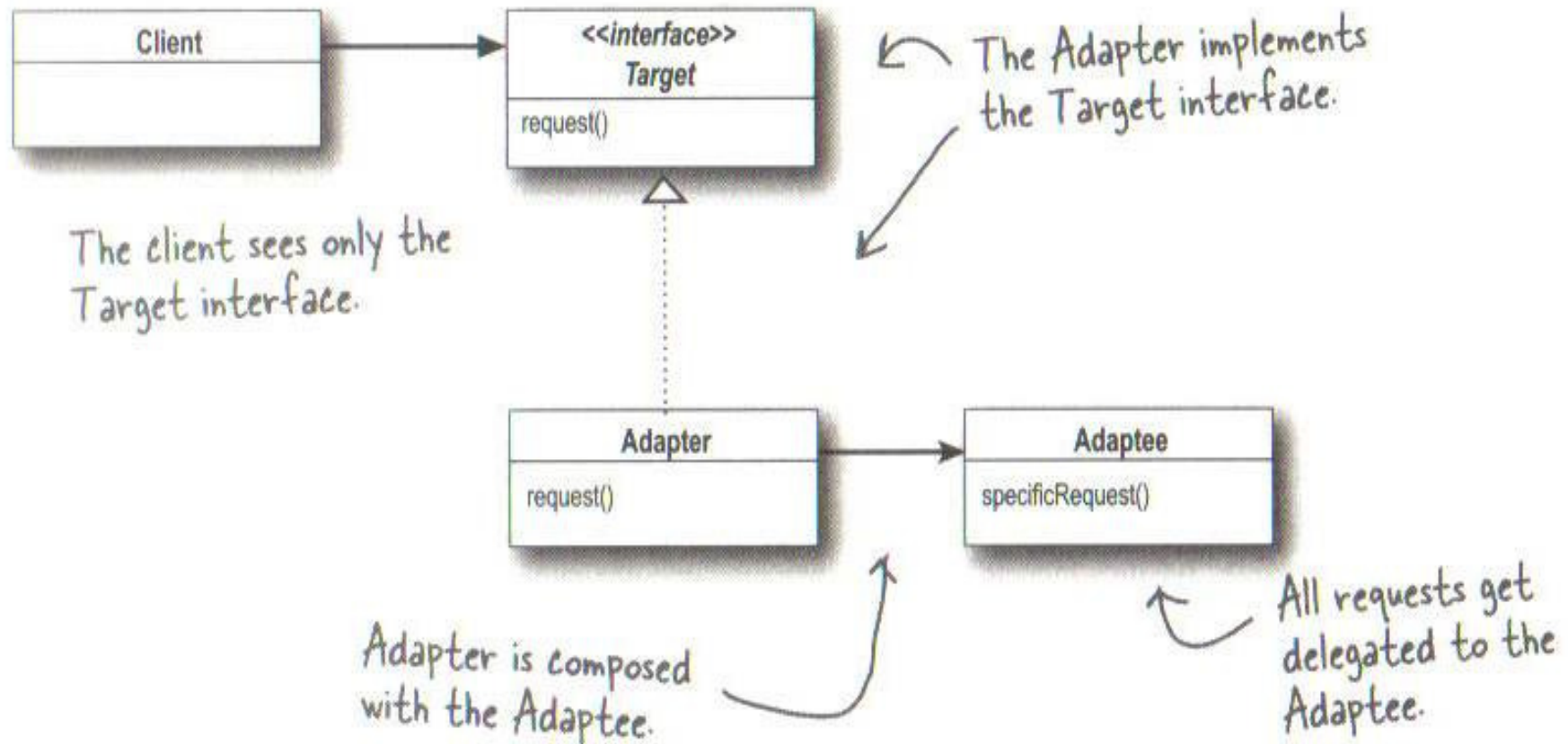
# Adapter Pattern defined

---

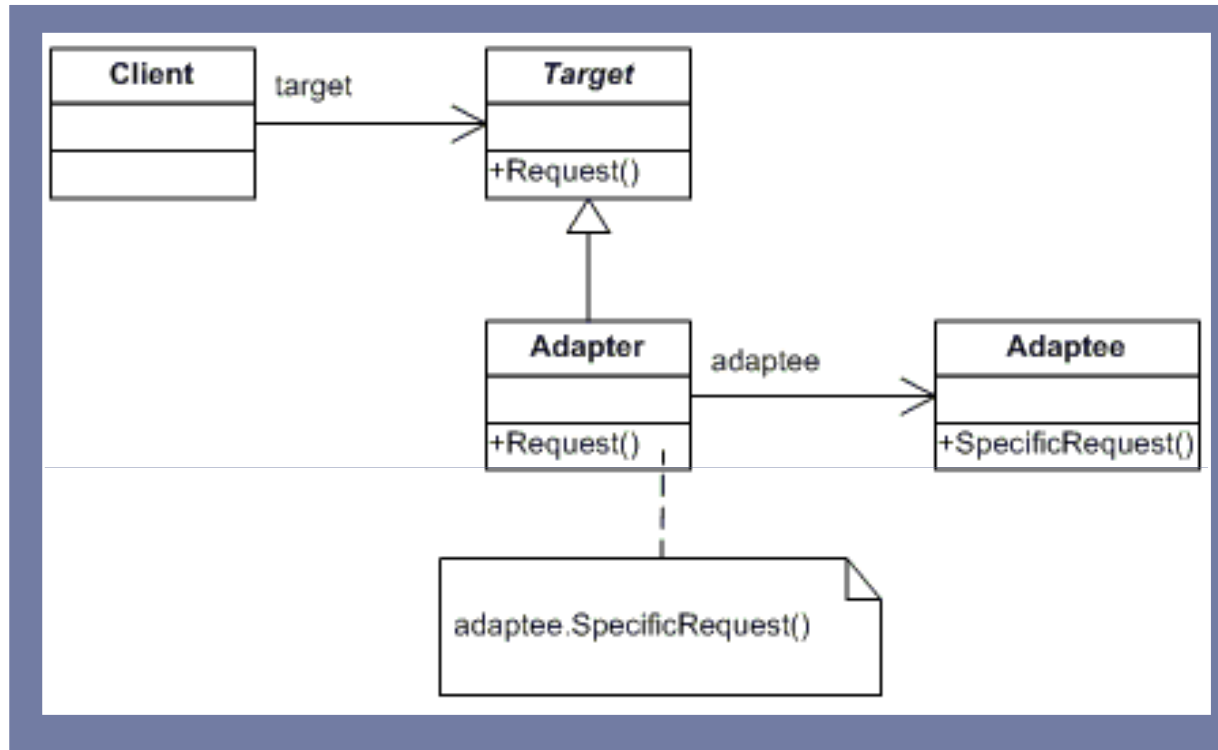
**The Adapter Pattern converts the interface of a class into another interface the clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.**



# Adapter Pattern



# Adapter pattern



- ▶ Delegation is used to bind an Adapter and an Adaptee
- ▶ Interface inheritance is used to specify the interface of the Adapter class.
- ▶ Target and Adaptee (usually called legacy system) pre-exist the Adapter.
- ▶ Target may be realized as an interface in Java.

# Participants

---

- ▶ **Target:** Defines the application-specific interface that clients use.
- ▶ **Client:** Collaborates with objects conforming to the target interface.
- ▶ **Adaptee:** Defines an existing interface that needs adapting.
- ▶ **Adapter:** Adapts the interface of the adaptee to the target interface.

# Adapter with multiple inheritance

