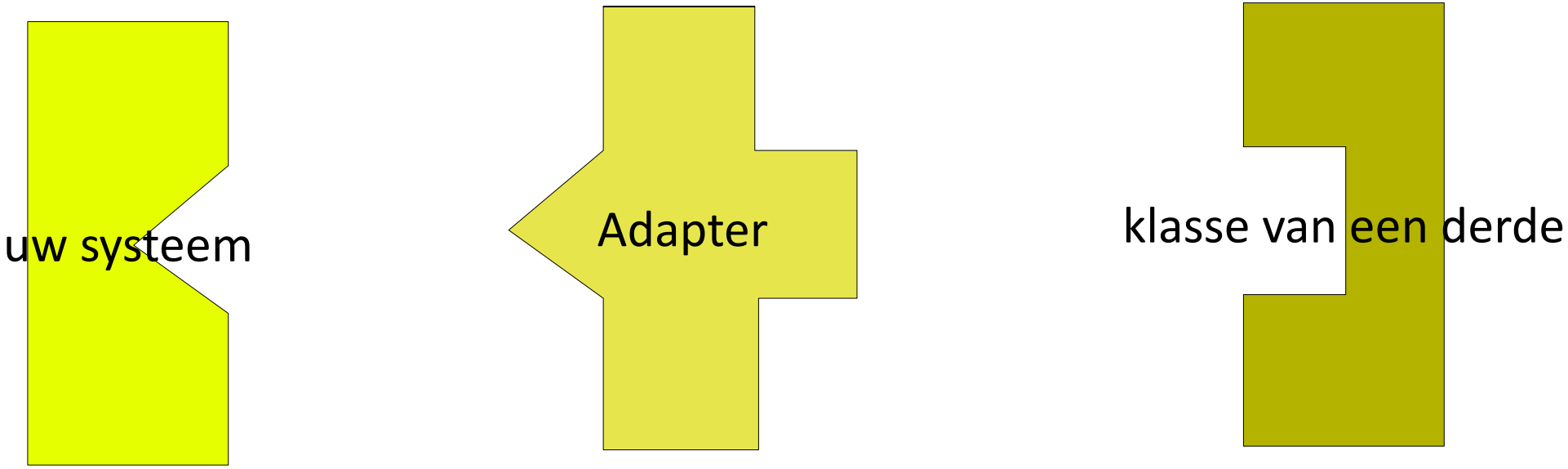
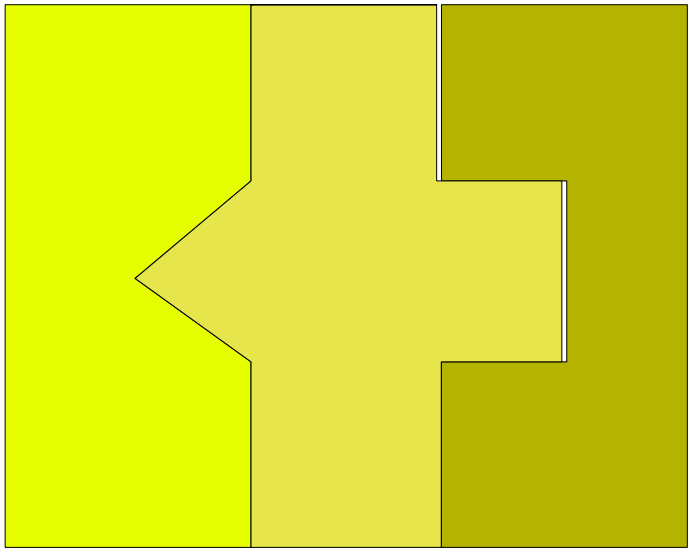
Het patroon Adapter

Een vierkante pin in een rond gat stoppen.

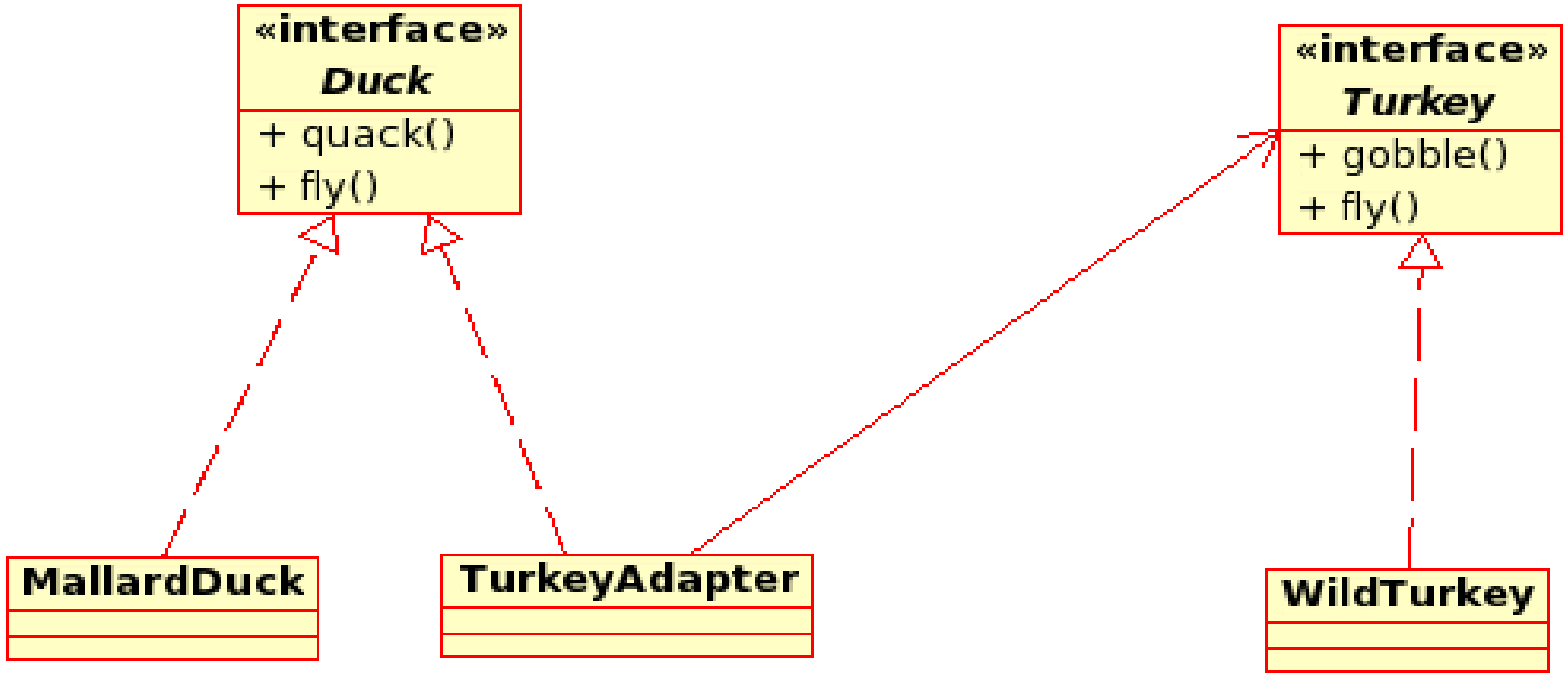
Onmogelijke combinaties maken



De adapter helpt u



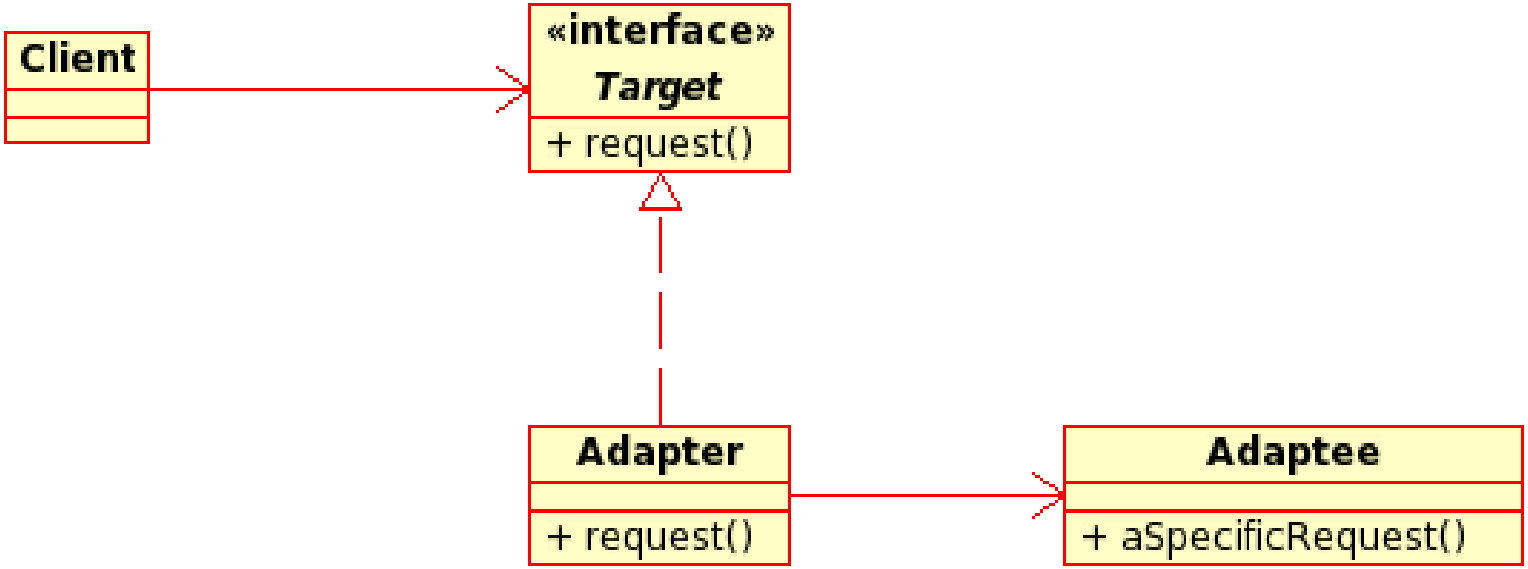
Kalkoen vermomt zich als eend



Objectgeoriënteerd ontwerp

Het **Adapter Pattern** converteert de interface van een klasse naar een andere interface die de client verwacht. Adapters zorgen ervoor dat klassen samenwerken. Zonder de adapters lukt dit niet vanwege incompatibele interfaces.

Het adapterpatroon



Het adapterpatroon

- Een client doet een verzoek door een operatie in een bepaalde interface aan te roepen.
- De adapter zet het verzoek om naar oproepen aan een adaptee.
- De client ontvangt het resultaat.

De vermomming van de kalkoen

```
public class TurkeyAdapter implements Duck {
      Turkey turkey;
      public TurkeyAdapter(Turkey turkey) {
            this.turkey = turkey; }
      public void quack() {
            turkey.gobble(); }
      public void fly() {
            for (int i=0; i<5; i++) {
                  turkey.fly(); }
}
```

Zwakke koppeling

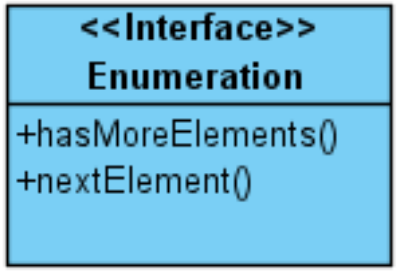
- De client weet niets van de adaptee af.
- De client is zich alleen bewust van de adapter.
- De adaptee weet niets van de client.

Wat met uitgebreide interfaces.

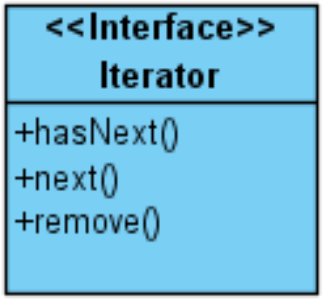
- Een adapter schrijven is veel werk.
- Maar dat is nog minder werk dan alle client-oproepen die verspreid liggen over je hele programma te gaan herschrijven.

Adapters in de praktijk

- Vroeger



- Nu



Probleemstelling: Binnen onze nieuwe code willen we nu altijd gebruik maken van de Iterator interface. Hoe lossen we dit op?

Klassen diagram