S — Single Responsibility Principle

- Én klasse = ét ansvar
  - Gælder også metoder: Én metode = ét ansvar
- En klasse skal kun have én grund til at ændre sig.
- Løsning: Del store klasser op

O — Open / Closed Principle

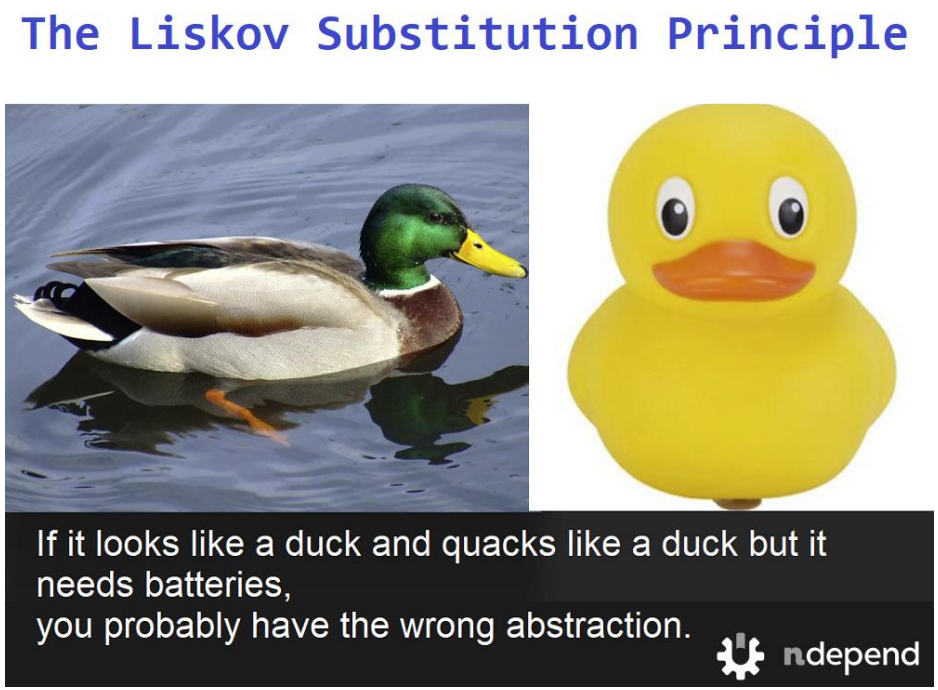
- Klasse skal være åben for udvidelse – lukket for ændring
- Ny funktionalitet → ny klasse , ikke ændre gl. kode
- Ikke flere if/else:

```
class DiscountCalculator {
    public double priceAfterDiscount(String type, double price) {
        return switch (type) {
            case "student" -> price * 0.9;
            case "senior"  -> price * 0.8;
            default        -> price;
        };
    }
}
```

- Løsning: Typisk interface eller abstrakt klasse

L — Liskov Substitution Principle

- Subklasser skal kunne erstatte superklasse uden overraskende adfærd



- Hvis det ikke passer, er designet forkert – brug interface i stedet for

I — Interface Segregation Principle

- Små interfaces er bedre end store
- Klasser skal ikke tvinges til metoder, de ikke bruger

- ✂️ Split store interfaces

D — Dependency Inversion Principle

- Afhæng af abstraktion – ikke konkret klasse
- Gør udskiftning lettere

- High-level kode må ikke kende detaljerne

 Udskift uden at ændre high-level koden