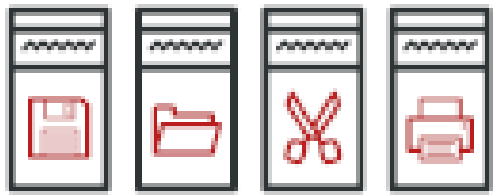
Design Mønstre

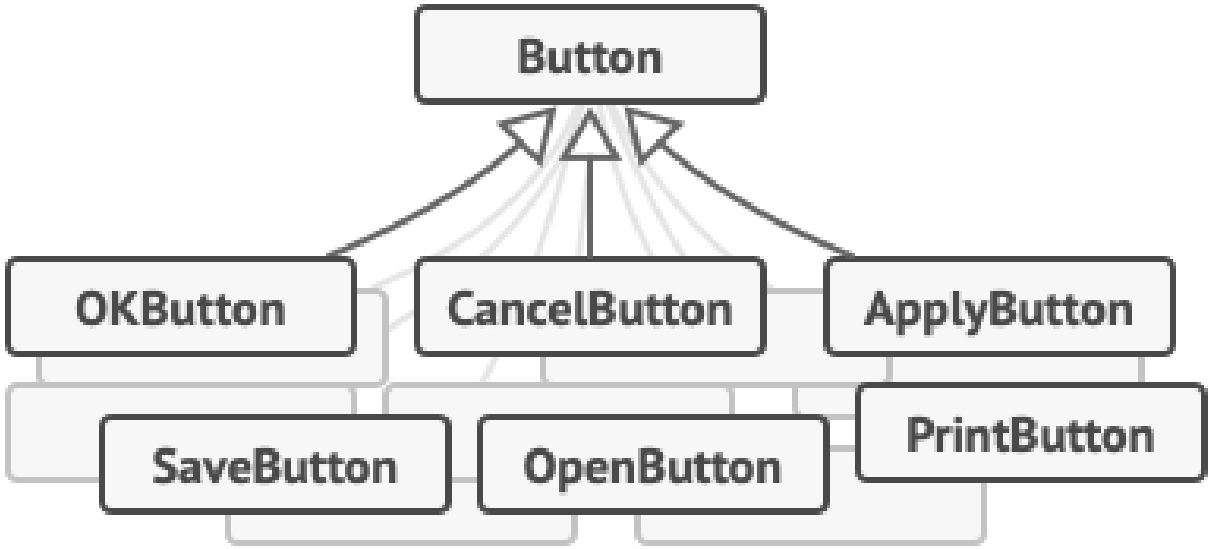
Command Mønstre



Command Mønstre

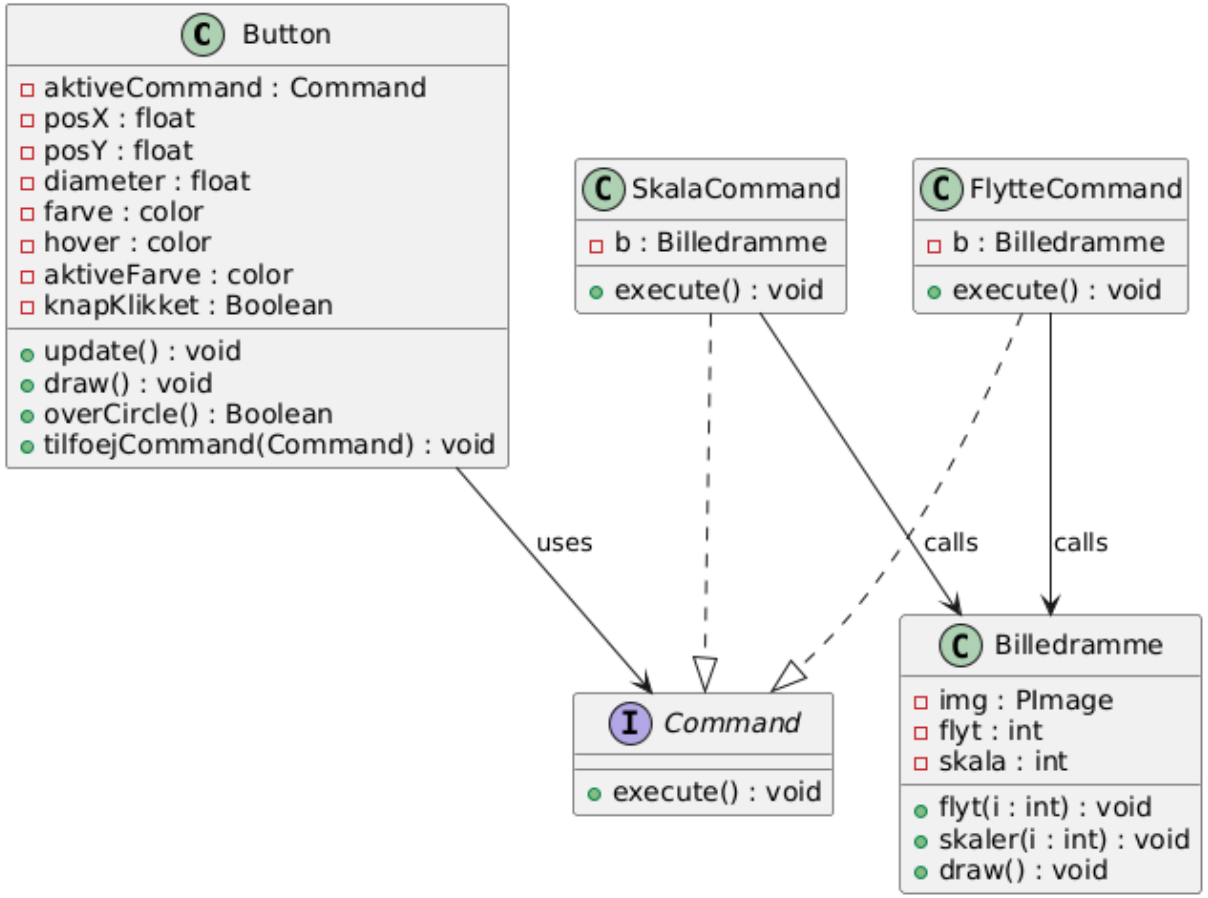
- En del af Behavioral Pattens
- Fokus på samhæng mellem objekter
- principle of separation
- Oprette vores udførelse som et objekt.

Knap med Subklasser



Lots of button subclasses. What can go wrong?

Command implementeret



fordel

- **Single Responsibility Principle**: Hver klasse skal have ét ansvar.
  Med Command Pattern adskilles klasser, der udløser en handling (f.eks. en knap), fra de klasser, der udfører handlingen (modtageren), hvilket gør systemet lettere at vedligeholde.
- **Open/Closed Principle**:
  Systemet kan udvides med nye kommandoer uden at ændre eksisterende kode.
  Nye funktioner kan tilføjes ved at skrive nye kommandoer, uden at ændre knapperne eller modtagerne.
- **Undo/Redo**:
  Du kan nemt tilføje funktionalitet til at fortryde (undo) og gendanne (redo) handlinger, da hver kommando kan gemmes og vendes.
- **Udsat udførelse**:
  Kommandoer kan udføres på et senere tidspunkt, fordi de er indkapslet i objekter. Det gør det muligt at planlægge handlinger til at blive udført senere.
- **Kompleks kommando**:
  Du kan kombinere flere simple kommandoer til en enkelt kompleks kommando, hvilket gør det nemt at udføre flere handlinger samtidigt.

opgave

- Se programeksemplet på lectio,
- - Forklar mønsteret med egne ord ud fra

  kode eksempel og supplerer eksemplet med endnu en
- Kommand.
- (ekstra Opgave) opret en måde hvor programmet kan redo eller undo fx at få billedet til at gå til den tidligere position.
- (ekstra Opgave) Opret en kommando som kan udføres senere eller udsættes.