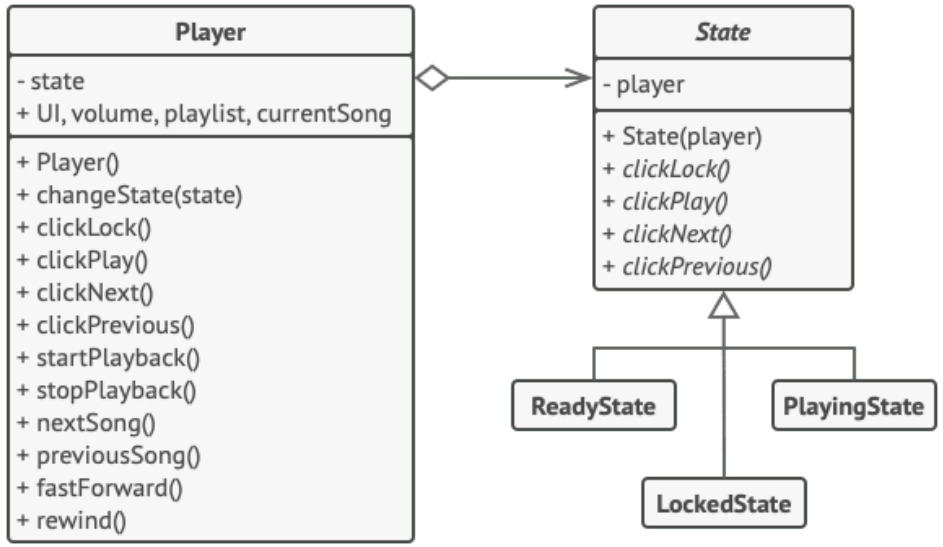
Design Pattens

State og Kommand Mønstre

State Design Mønstre

- Programmer har forskellige Tilstande.
- Et hvert tilstand er forskellige og har forskellige adfærd.
- Forslår fleksibilitet ved at oprette klasser til programmets forskellige tilstande.



Command Mønstre

- Indkapsle anmodninger som objekter
- Adskiller anmodninger fra udførelsen, hvilket gør det lettere at ændre kommandoer og dens implementeringer.
- man kan gemme anmodninger i en historik for senere udførelse eller fortrydelse.
- Det gør det muligt at implementere en Undo, da man kan gemme den tidligere tilstand.

State & Command

Et program med begge mønstre.

fordel

Adskillelse af tilstand og handling

- **State Pattern**: Håndterer, hvordan objektets opførsel ændrer sig afhængigt af dets interne tilstand (f.eks. om billedrammen er i flyttetilstand, skaleretilstand eller normal tilstand).
- **Command Pattern**: Håndterer brugerens handlinger (f.eks. når en knap trykkes for at flytte eller skalere billedrammen).

fordel

- En klar opdeling mellem **hvad** der skal ske (via kommandoer) og **hvordan** tilstanden skal opføre sig (via State Pattern).
- Command Pattern giver mulighed for at binde forskellige brugerinput (f.eks. knapper eller tastaturgenveje)
- State Pattern sørger for, at objektet reagerer forskelligt afhængigt af dets nuværende tilstand (f.eks. kan billedrammen flyttes på en bestemt måde, mens den er i flyttetilstand, men forstørres i en anden tilstand).
- Ved at kombinere begge mønstre holder du koden **simpel**, **modulær**, og **letforståelig**. (hvornår der skal ske noget og hvordan det skal ske)

Opgave

Ændre koden på kommandProjekt hvor i implementer State patten.

- Ændre de forskellige kommandoer så at de kan flytte billedet i forskellige states. Fx skalerstate, flytstate og forstør state.
- Ekstra opgave. Lav en kommand, hvor den går tilbage til netruelstate.

- Lav en klasse diagram af jeres færdig program.