3. Patterns 1: GoF Strategy + GoF Template Method

# Redegør for, hvad et software design pattern er.

Design Pattern kan beskriver et problem der opstår igen og igen og derefter beskriver kernen i en løsning til det problem. Det der gør det til et mønster er så at man kan bruge løsningen om og om igen, uden at gøre samme.

Når vi anvender designregler skal det nemlig altid tilpasses. Et design pattern er derfor aldrig en færdig løsning på et problem, men en skabelon til løsning af et problem.

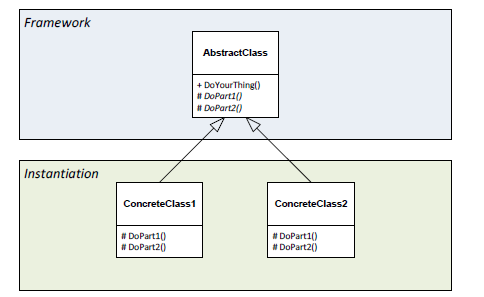
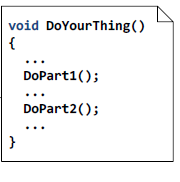
# Sammenlign de to design patterns GoF Strategy og GoF Template Method

Strategy og template forsøger begge at løse problemet at adskille en algoritme fra dens implementation.

## GoF Template Method

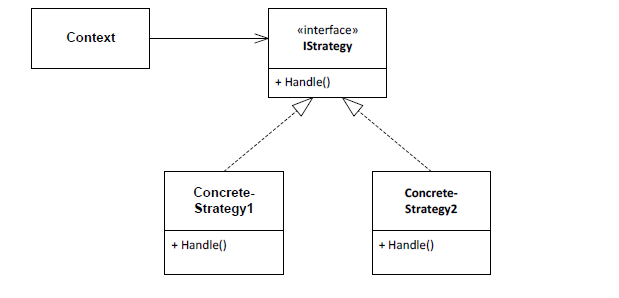
GoF template har til formål at strukturere generel adfærd i Software. Bruges de samme strategier kan man lave en template klasse som håndterer den mest generelle adfærd.

* Template Method er typisk brugt i frameworks, her bestemmer frameworked typisk hvad der skal ske og hvornår.
* Man opretter en instans af framworked ved at implementerer framwork metoderne i en afledt klasse.



## GoF Strategy

Formålet med Strategy er at definere en samling af algoritmer, indkapsle dem hver for sig og gøre det muligt at skifte i mellem dem. Her lader Strategy, algoritmen varierer uafhængigt af klienter, der bruger det.



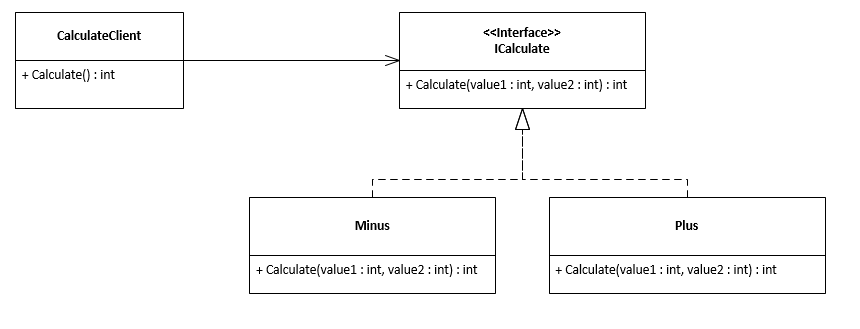
Strategy patteren gør det muligt for contexts opførsel til at blive defineret ved runtime ved at uddelegerer den til et andet objekt.

## Hvornår vil du anvende hvilket, og hvorfor? (Template Method vs. Strategy)

Hvis det er nødvendigt at kunne skifte den implementering det ønskes at bruges run time må vi bruge strategy da template ikke kan skiftes run time. Template er tilgengæld mere simpelt at implementere.

* Begge patterns/mønstre er adfærds mønstre
* Begge er anvendt til at lave adfærds af et system letter at udvide
* Arv VS Delegation
* Man skal anvende Strategy Pattern, nar det skal være muligt at ændre kontekstens adfærd runtime. Strategy pattern er derfor mere genanvendeligt.
* Mens hvis kontekstens adfærd ikke skal ændres under runtime, altså bestemt under compile time, så kan Template Method anvendes. Da den er lettere at implementere.

# Vis et designeksempel på anvendelsen af GoF Strategy



**protected** **void** Page\_Load(object sender, EventArgs e) {

CalculateClient minusClient = **new** CalculateClient(**new** Minus());

Response.Write("<br />Minus: " + minusClient.Calculate(7, 1).ToString());

CalculateClient plusClient = **new** CalculateClient(**new** Plus());

Response.Write("<br />Plus: " + plusClient.Calculate(7, 1).ToString());

}

På runtime vil vi gerne være i stand til både at kunne plus og minus, derfor er det strategy så det kan ændres under runtime

# Redegør for, hvordan anvendelsen af GoF Template fremmer godt software design.

GoF templete gør at kode kan genbruges da den gør brug af arv, dette er rigtigt godt hvis vi har noget vi ønsker at gøre med samme fremgangs måde men ændre lidt på selv udførslen.

## Redegør for, hvilke(t) SOLID-princip(per) du mener anvendelsen af GoF Strategy understøtter

* Open-Close (OPC)
  + Open close fordi det er lukket for ændringer med åbent for udvidelser
* Single Responsebility (SRP)
  + Konteksten har kun til opgave hvad der skal ske men ikke hvordan det skal ske
* Dependency Inversion (DIP)
  + Konteksten er ligeglad med klasserne, den bruger et interface.