

designmønstre

OOP-designmønstre: intro til...

Hvad menes der med et designmønster? Og hvilke forskellige findes der?

Designmønstre er typiske løsninger på problemer i software-design-De udgør skabeloner for løsninger, som man skal tilpasse den konkrete kode i et program:

Man kan ikke bare copy-paste koden; den skal tilpasses til ens program. Det er et mere **generelt løsningskoncept**.

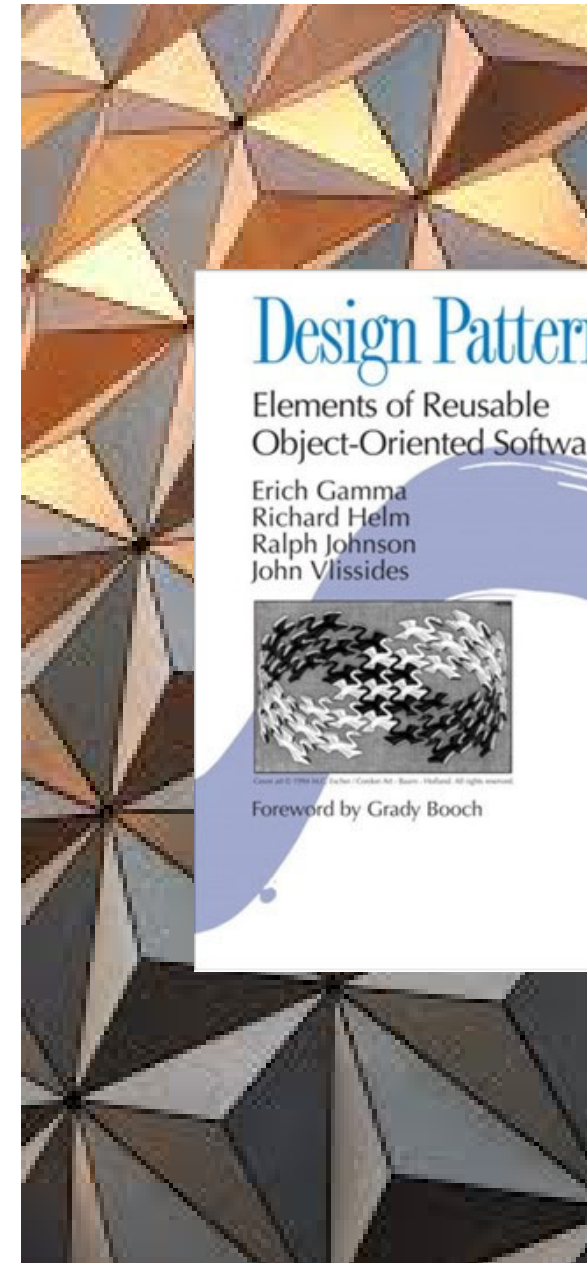
Begrebet er lånt fra arkitekturens verden, og overført til objekt-orienteret programmering via bogen Design Patterns, Elements of Reusable Object-Oriented Software, fra 1995:

Bogen beskriver 23 klassiske designmønstre - opdelt i:

Skabende mønstre - handler om at skabe nye objekter på en hensigtsmæssig måde

Strukturelle mønstre - handler om det at skabe gode relationer imellem klasser

Adfærdsmæssige mønstre - handler om at skabe kommunikation imellem klasser



OOP- designmønstre: intro til

Hvilke forskellige
mønstre findes der?

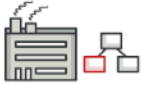


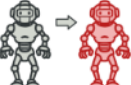


PS:

Der findes flere,
Men dette er nogle
af de mest kendte,
og de originale.

The Catalog of Design Patterns

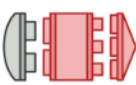


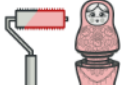


Creational patterns

These patterns provide various object creation mechanisms, which increase flexibility and reuse of existing code.

 Factory Method	 Abstract Factory
 Builder	 Prototype
 Singleton	 Proxy







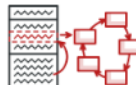
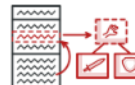


Structural patterns

These patterns explain how to assemble objects and classes into larger structures while keeping these structures flexible and efficient.

 Adapter	 Bridge
 Composite	 Decorator
 Facade	 Flyweight

Behavioral patterns

These patterns are concerned with algorithms and the assignment of responsibilities between objects.

 Chain of Responsibility	 Command	 Iterator	 Mediator
 Memento	 Observer	 State	 Strategy
 Template Method	 Visitor		

OOP-designmønstre: intro til

Et mønster har

- en standard-struktur
- en intention
- Og udgør en løsning på et typisk problem

Brugen af et designmønster kan komplicere koden, så man kan afveje, om det gør koden unødvendig kompliceret eller ej.

Visse mønstre ligner hinanden meget, men intentionen kan være forskellig!

SE FÆLLES EKSEMPEL 1: Om State-pattern

Hvad er strukturen i mønsteret, eller hvordan designer man koden ud fra dette mønster?



OOP-designmønstre

Lav Avatar-klasse i tilstande via state-mønstret!

Mønstret gør lav kobling og høj binding mulig, idet hver tilstands-klasse håndterer eller netop implementerer kernemetoderne på forskellig vis...

1: Læs først indledningsvist om designmønstret via linket -
Diagrammet på denne side er et eksempel på en afspiller i forskellige tilstande.

<https://refactoring.guru/design-patterns/state>

2: Gør rede for designmønstret og tegn **UML-diagram** over koden med:

Avatar-klassen & dens base-klasse (nedarvingspil med hvidt fyld)

Interfacet IState og dens implementerende klasser (stiplet implementeringspil)

Associer evt. **Animation-klassen** til State-klasserne (pil uden fyld)

3: Kør koden -

og gør rede for koden ud fra jeres smukke diagram!

4: Tilføj **ny tilstandsklasse** til koden.

5: Se også gerne hvordan Animation-klassen håndterer at bruge spriteSheet.

6: Overvej i hvilke andre situationer mønstret kan bruges?

