Mönster och datastrukturer, del 1

Data- och informationsvetenskap: Objektorienterad programmering och modellering för IA

Johan Holmberg 2019-10-17

MALMÖ UNIVERSITET

Dagens agenda

- Vad är ett mönster?
- Designmönster
  - Att arbeta med designmönster
  - Olika designmönstertyper

MALMÖ UNIVERSITET

Vad är ett mönster?

MALMÖ UNIVERSITET

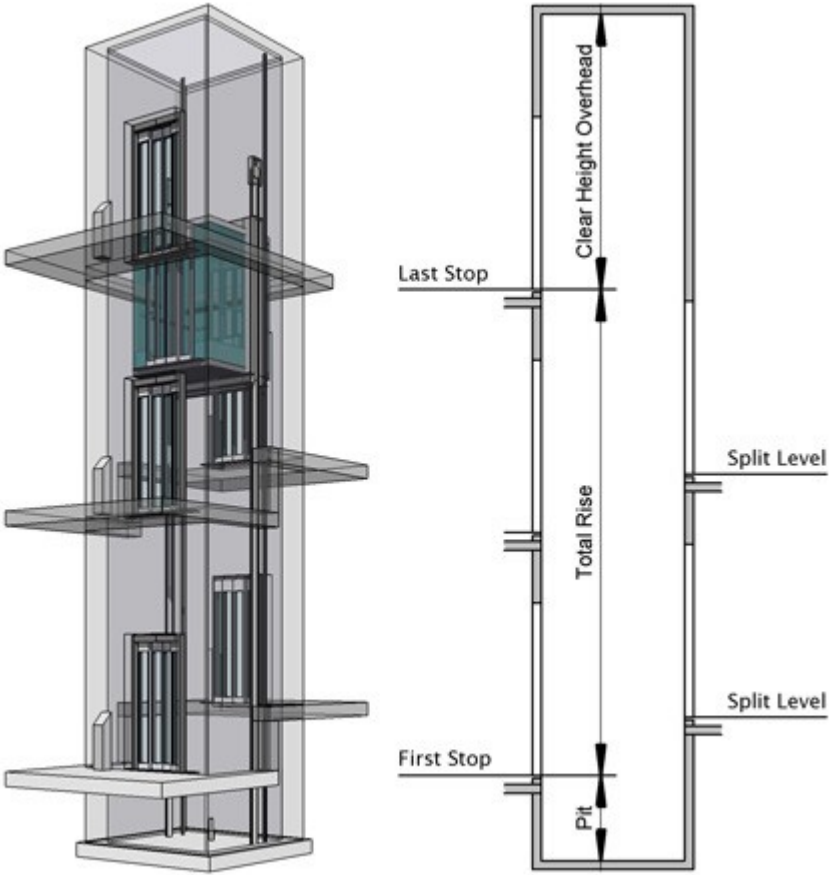
Vad är ett mönster?

Mönster är generella lösningar på vanligt förekommande problem.

För arkitektur: Alexander et al. 1977:

> ”Each pattern describes a problem which occurs over and over and over again in our environment, and then describes the core of a solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.”

MALMÖ UNIVERSITET

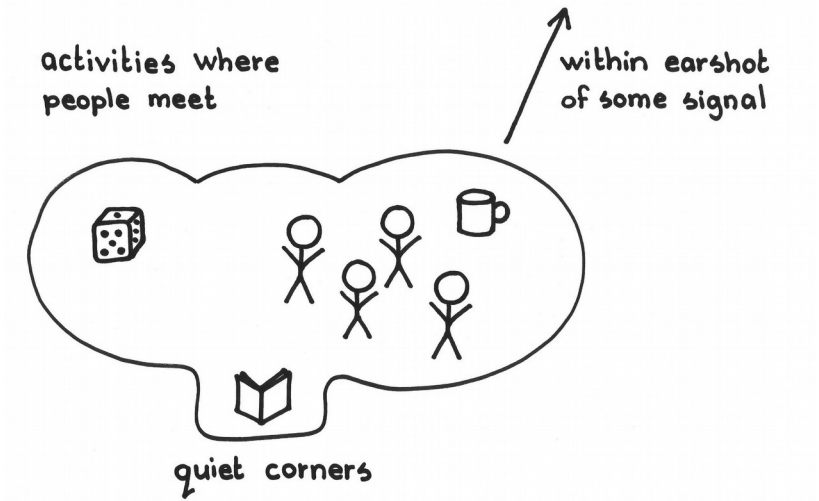


Exempel: hissen

Problem: I en *byggnad* med två eller flera *våningar* måste besökare och materiel effektivt och säkert kunna förflyttas mellan de olika våningarna.

Lösning: En hiss är ett motoriserat transportmedel för personer eller materiel mellan våningsplan i byggnader. Hissen är inte en *trappa*.

MALMÖ UNIVERSITET



A place to wait
(exempel från Alexander et al. (1977))

…in any office, or workshop, or public service, or station, or clinic, where people have to wait – I*nterchange (34)*, *Health Center (47)*, *Small Services without Red Tape (81)*, *Office Connections (82)*, it is essential to provide a special place for waiting, and doubly essential that this place not have the sordid, enclosed, time-slowed character of ordinary waiting rooms.

MALMÖ UNIVERSITET

Exempel från Malmö universitet: Game Design Patterns

Dahlskog (2014)

Table 21: Patterns for Super Mario Bros. grouped by theme part 1 [48].

Enemies	
Enemy	A single enemy
2-Horde	Two enemies together
3-Horde	Three enemies together
4-Horde	Four enemies together
Roof	Enemies underneath a hanging platform making Mario bounce in the ceiling
Gaps	
Gaps	Single gap in the ground/platform
Multiple gaps	More than one gap with fixed platforms in between
Variable gaps	Gap and platform width is variable
Gap enemy	Enemies in the air above gaps
Pillar gap	Pillar (pipes or blocks) are placed on platforms between gaps
Valleys	
Valley	A valley created by using vertically stacked blocks or pipes but without Piranha plant(s)
Pipe valley	A valley with pipes and Piranha plant(s)
Empty valley	A valley without enemies
Enemy valley	A valley with enemies
Roof valley	A valley with enemies and a roof making Mario bounce in the ceiling

Baldwin et al. (2017)

C. Pattern Detection

The evaluation of individuals is partly based on the detection of the occurrence of the following fundamental design patterns [2]: micro-patterns *corridor*, *connector* and *room*, though we refer to room as *chamber* to avoid confusion with our previous use of room. The pattern *space* is equivalent to what we refer to as passable tile.

Figure 3 shows several sample expressions of those patterns that may occur in the generated individuals. These are chambers (a), corridors (b), and two types of connectors: joints (c and d), and turns (e). Each detected pattern is assigned an associated *quality* value between 0 and 1 as a measure of how well it conforms to some desired control parameters. Detailed descriptions for these patterns and quality metrics follow.

(a) Chamber (b) Corridor (c) Joint (d) Joint (e) Turn

Fig. 3: Examples of each micro-pattern detected by the generator: (a) the minimal chamber, (b) a three tile long corridor, (c) a three-way joint (the central tile), (d) a four-way joint (the central tile), (e) and a turn (the central tile).

MALMÖ UNIVERSITET

Mönster i mjukvara

Mönster för mjukvara dyker upp i samband med att objekt-orienterad programmering växer fram, under 1980-talet och 1990-talet

- Design Patterns (the GoF book)
- Gamma et al. (Gang of Four – GoF) 1995

Vad är ett mjukvarumönster?

Gabriel 1996:

A three-part relation between:

- Context (ett sammanhang
- System of forces (ett problem som måste åtgärdas)
- Software configuration which resolves the forces

Coplien 1996:

- It solves a problem.
- It is a proven concept.
- The solution is not obvious.
- It describes a relationship.

MALMÖ UNIVERSITET

Olika typer av mönster

Analys-mönster
Beskriver koncept som är viktiga för att modellera krav.

Design-mönster
Beskriver struktur och interaktion mellan mindre komponenter i koden.

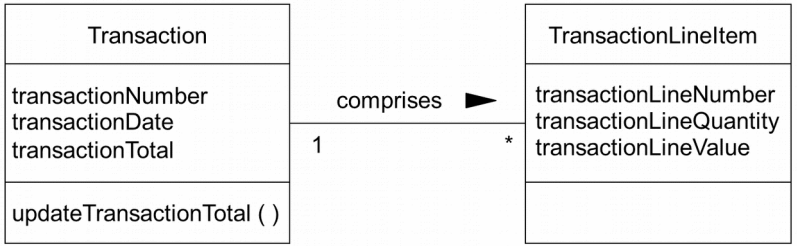
Arkitekturiella mönster
Beskriver hur de större komponenterna i ett system är strukturerade i förhållande till varandra.

Anti-mönster
Hur man inte bör göra – lösningar som visat sig vara olämpliga på olika sätt.

Analysmönster: Transaction

Försäljning:

SalesOrder	**SalesOrderlineItem**
orderNumber	orderLineNumber
orderDate	orderLineQuantity
orderTotalValue	orderLinevalue
updateOrderTotalValue()	



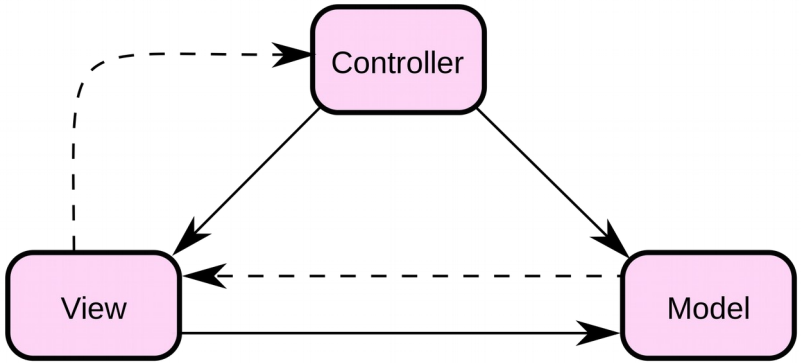
Frakt, Betalning, …

MALMÖ UNIVERSITET

Arkitekturiella mönster: MVC

Model – View – Controller

Ett klassiskt mönster för att separera data, presentation och affärslogik.

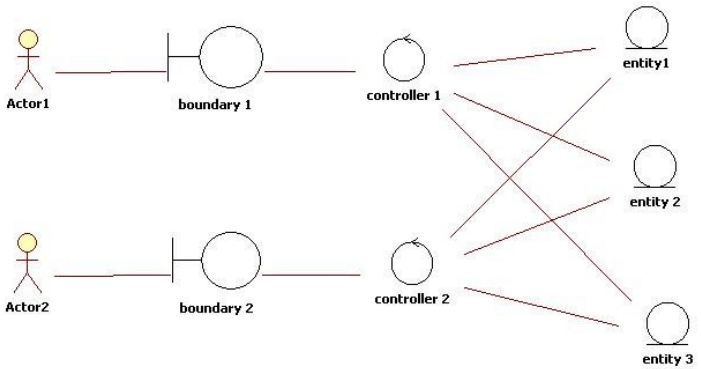


MALMÖ UNIVERSITET

Arkitekturiella mönster: ECB

Entity – Control – Boundary

Ännu ett mönster för att separera data, interaktionsobjekt och affärslogik.



MALMÖ UNIVERSITET

Designmönster

MALMÖ UNIVERSITET

Designmönster

Ett designmönster är en beskrivning av hur en viss typ av problem kan lösas på en mer abstrakt nivå.

Ett designmönster innehåller inte färdig kod. Klasserna som beskrivs i designmönstret måste implementeras i det aktuella systemet.

Ska inte förväxlas med ramverk (framework). Ett ramverk är ett implementerat system som i sig kan använda ett antal designmönster.

Designmönster

Viktiga principer för designmönster som vi redan stött på:

- **Abstraktion** (arv, polymorfism, etc)
- **Inkapsling** (metoder som påverkar det egna objektet)
- **Information hiding** (göm detaljer för omvärlden)
- **Cohesion** (gör få, men rätt, saker)
- **Coupling** (relationer mellan moduler/klasser)

Designmönster

Designmönster beskrivs strukturerat. Mallarna för dessa varierar lite beroende på varifrån de kommer. Generellt bör en beskrivning av ett mönster innehålla:

- Namn
- Problembeskrivning
- Kontext
- Krafter/Forces
- Lösning

Hur väljer man mönster eller vet om det finns ett lämpligt mönster?

- Känn till mönster eller vet var du hittar dem
- Erfarenhet av att känna igen problem
- Kunna överväga fördelar och nackdelar med ett mönster – finns det en enklare lösning?
- Vilka konsekvenser får valt mönster?
- Fungerar lösningen med det språk och omgivning du har?

MALMÖ UNIVERSITET

För- och nackdelar med designmönster

+ Stöder återanvändning
+ Definierar beprövade lösningar på vanliga problem
+ Återuppfinner inte hjulet och diverse buggar
+ Produktivt, underhållbart och utbyggbart
+ Lätt att förstå för andra utvecklare

- Ibland svårt att hitta ett lämpligt mönster
- Är ett mönster för restriktivt? För omfattande?
- Fungerar mönstret med valt programmeringsspråk?
- Finns det enklare lösningar?

MALMÖ UNIVERSITET

Olika typer av designmönster

Creational patterns – skapa instansobjekt

Structural patterns – design av klasser och relationer

Behavioural patterns – kommunikationsmönster mellan objekt

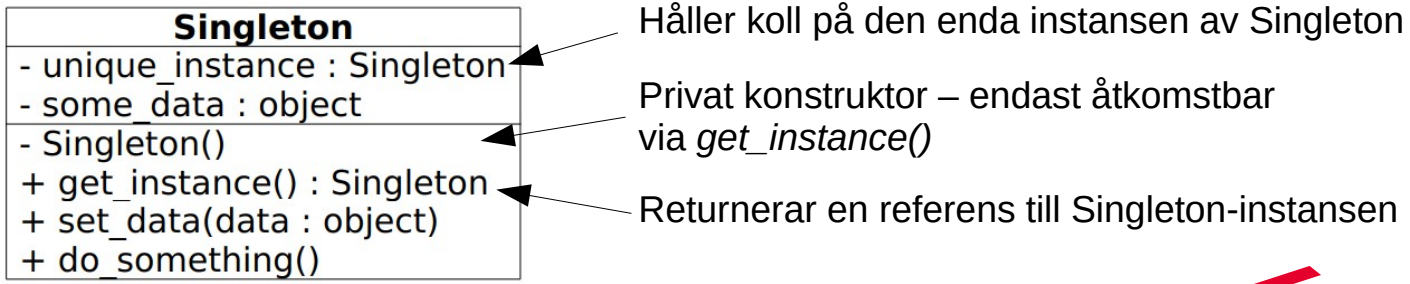
Ursprung i GoF:s *Software Patterns*

Creational Pattern: Singleton

Namn: Singleton

Problem: Säkerställa att endast en instans av en klass skapas i ett system.

Kontext: Då en viss typ av objekt endast ska instansieras en gång i systemet, men användas av olika delar av systemet.



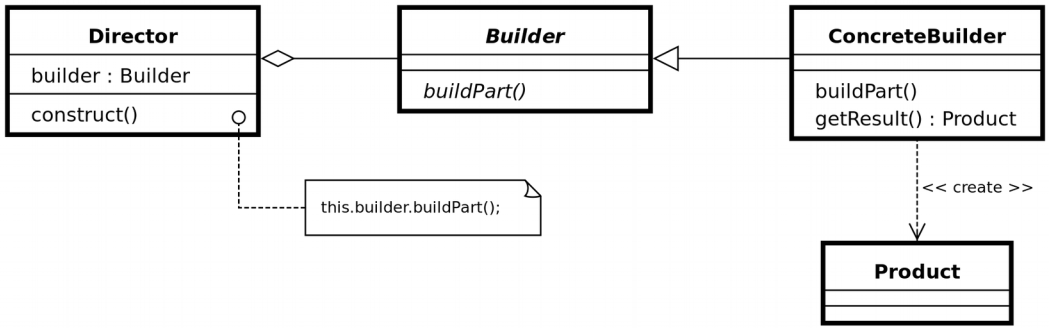
MALMÖ UNIVERSITET

Creational Pattern: Builder

Namn: Builder

Problem: Förenkla skapandet av komplexa objekt

Kontext: Då skapandet av en viss typ av objekt är särskilt komplicerat och vi vill separera skapandeprocessen från den faktiska representationen av objektet.



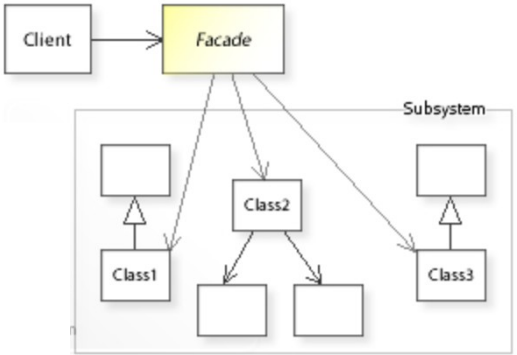
MALMÖ UNIVERSITET

Structural Pattern: Façade

Namn: Façade

Problem: Erbjuda ett gränssnitt till ett subsystem vilket förenklar användningen av subsystemet.

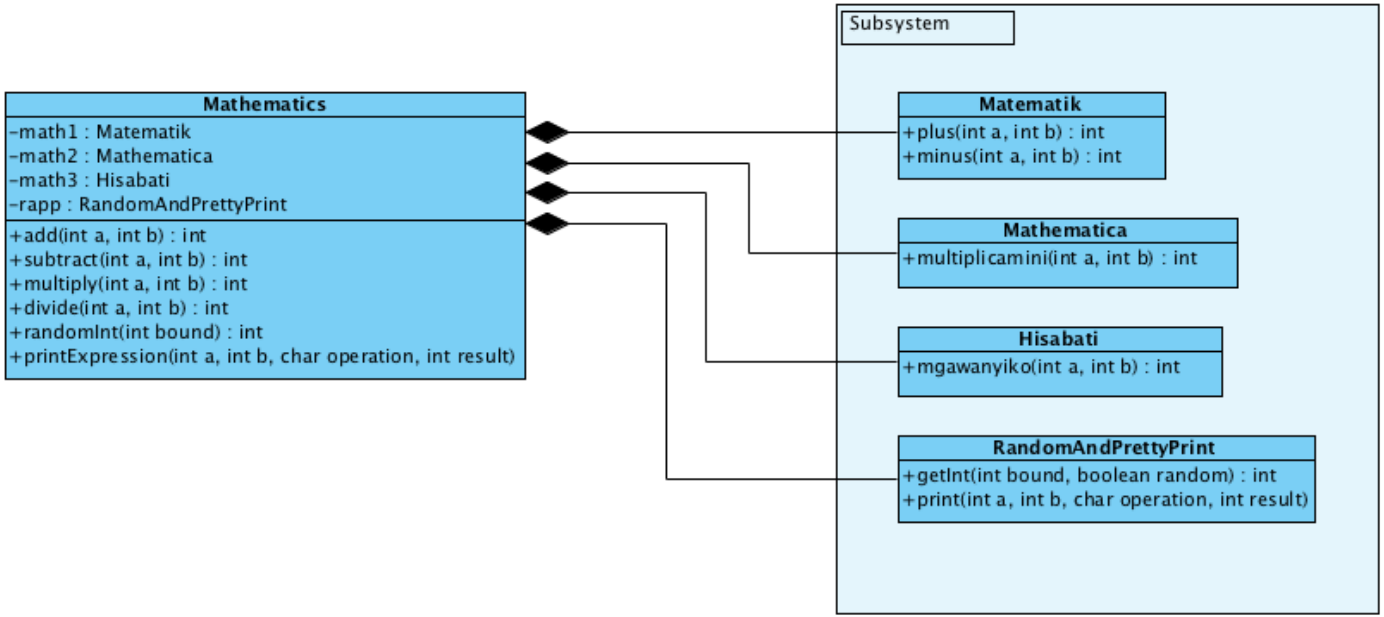
Kontext: Förenkla användningen av ett subsystem i andra system.



Facade
- class_1_obj : Class1 - class_2_obj : Class2 - class_3_obj : Class3
+ metod1() : string + metod2(some_data : int)

MALMÖ UNIVERSITET

Structural Pattern: Façade



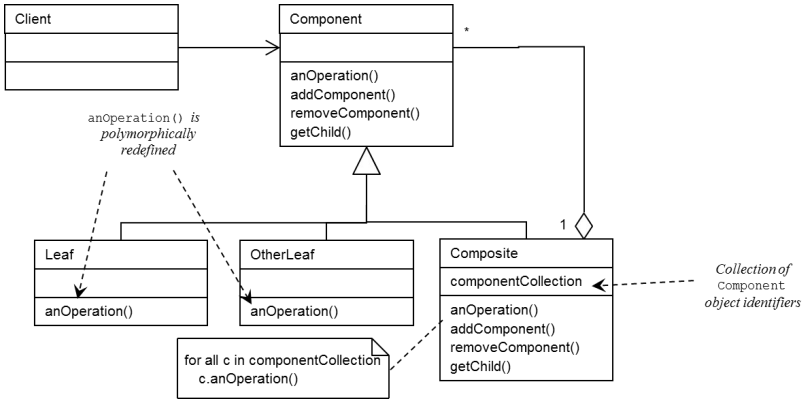
MALMÖ UNIVERSITET

Structural Pattern: Composite

Namn: Composite

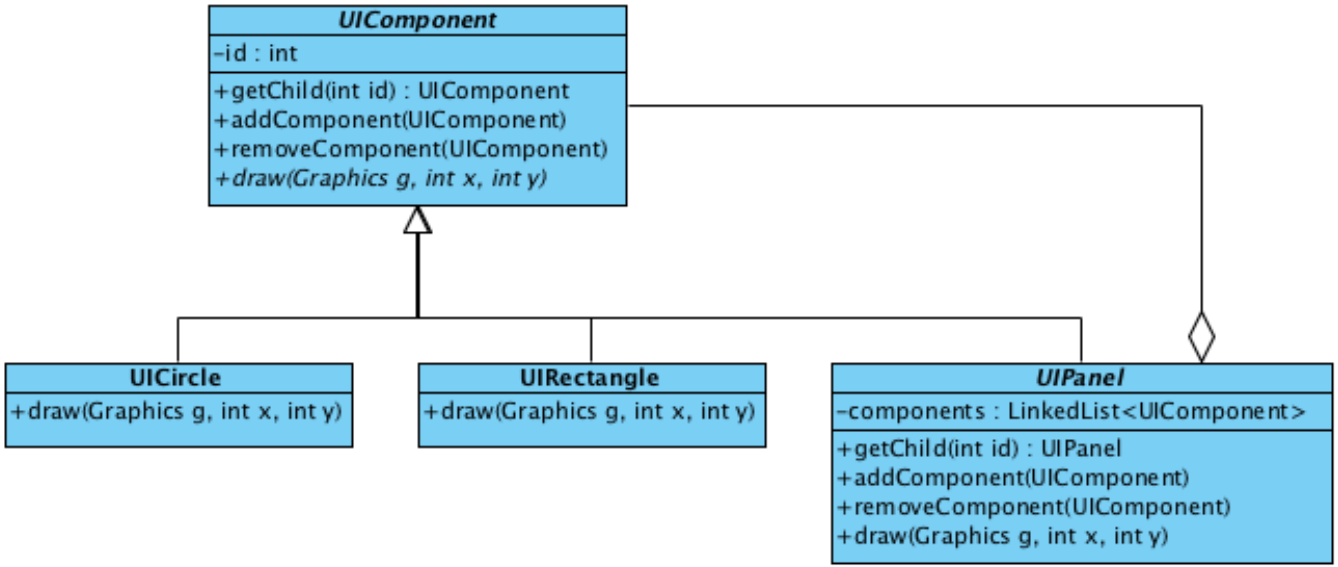
Problem: Erbjuda samma gränssnitt till individuella objekt och en samling av objekt.

Kontext: I ett system används enstaka objekt och samling av objekt i liknande sammanhang.



MALMÖ UNIVERSITET

Structural Pattern: Composite

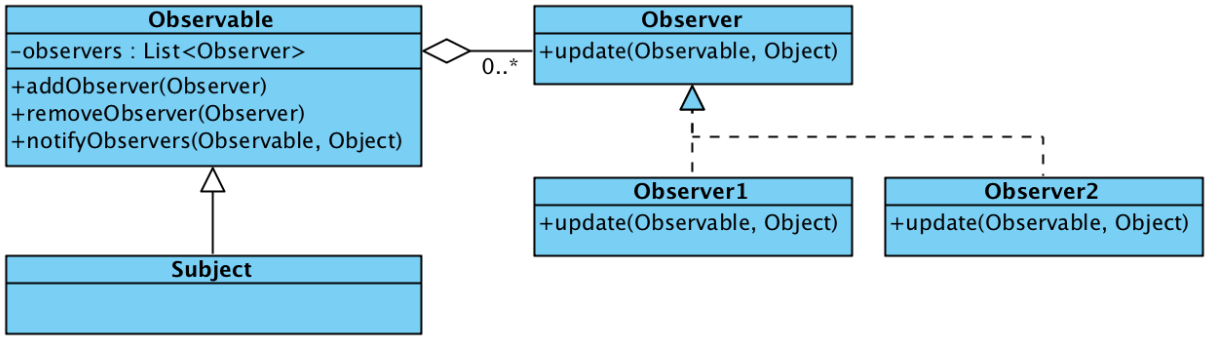


Behavioural Pattern: Observer

Namn: Observer

Problem: Förändring i ett objekts tillstånd ska meddelas till andra objekt.

Kontext: Då ett eller flera objekt ska notifieras då ett objekts tillstånd ändras.



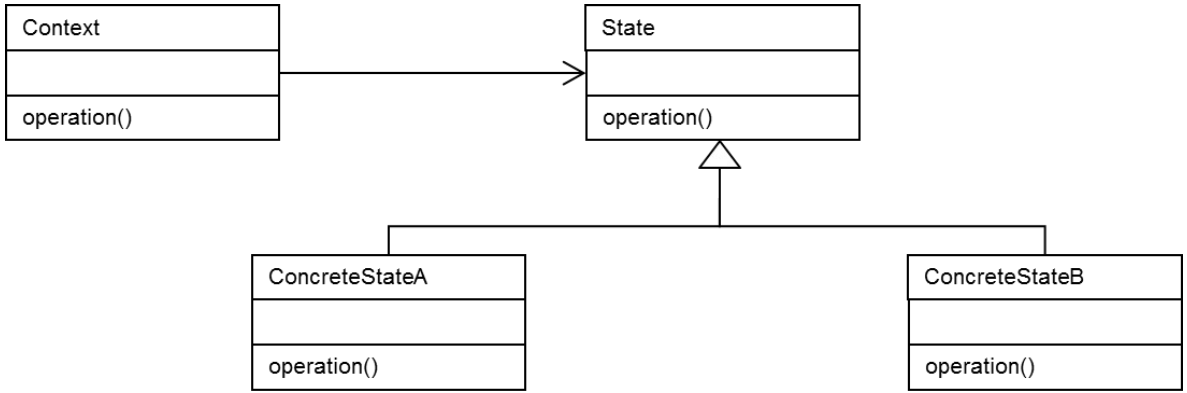
MALMÖ UNIVERSITET

Behavioural Pattern: State

Namn: State

Problem: Ett objekt ska ha olika beteende beroende på objektets tillstånd.

Kontext: Beroende på objektets tillstånd så ger ett meddelande till objektet olika beteende.



MALMÖ UNIVERSITET

Slutligen

MALMÖ UNIVERSITET

Läsanvisningar

Object-Oriented Systems Analysis and Design Using UML

- 15 Designmönster

MALMÖ UNIVERSITET

Referenser

Alexander et al. 1977

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., King, I. & Shlomo, A. (1977). *A pattern language : towns, buildings, construction.* New York: Oxford University Press.

Gamma et al. (Gang of Four – GoF) 1995

Gamma, E. (1995). *Design patterns : elements of reusable object-oriented software.* Reading, Mass: Addison-Wesley.

Gabriel 1996

Gabriel, RP. (1996). *Patterns of Software. Tales from the Software Community.* New York: Oxford University Press.

Coplien 1996

Coplien, J. (1996). *Software patterns.* New York London: SIGS.

Dahlskog 2014

Dahlskog, S (2014). *A multi-level level generator.* Proceedings of the 2014 IEEE Conference on Computational Intelligence and Games IEEE

Baldwin et al. 2017

Baldwin, A., Dahlskog, S., Font, J. M., Holmberg, J. (2017). *Mixed-Initiative Procedural Generation of Dungeons using Game Design Patterns.* Proceedings of the 2017 IEEE Conference on Computational Intelligence and Games

MALMÖ UNIVERSITET